MAT 271: Applied & Computational Harmonic Analysis Comments on Homework 3

Problem 2: Almost everyone got Part (a) correct. However, some people did not give enough explanations. You need to *explain* and *justify* your derivations!

For Part (b), what you really need to show is:

1. Compute the eigenvalues of the covariance matrix Γ, which is 0 and $1/n$ with geometric multiplicity 1 and $n - 1$, respectively, which can be done by explicitly deriving the characteristic equation, $\det(\Gamma - \lambda I) = \lambda(1/n - \lambda)^{n-1} = 0$. Not all of you did this way, but in my opinion, this should be the easiest way to conclude this. Please review your linear algebra textbooks, in particular, how to compute the determinant of a given matrix, and how to simplify the computation!

2. Γ is a real symmetric matrix; thus it can be *unitarily* diagonalizable, i.e., there exists an orthonormal basis diagonalizing Γ. This is very important to state explicitly!

3. The above two also means that the eigenspace corresponding to the eigenvalue 0 and the one corresponding to $1/n$ are orthogonal.

For more information and the origin of the interest of this process, please read my own paper [1], [8], [7], and references therein.

Problem 3: Part (a) should be done in the expectation in the integration w.r.t. the uniform probability density function, i.e., 1 over the unit interval $[0, 1]$. So,

$$E[X(t)] = \int_0^1 (t - H(t - \tau)) \cdot 1 \, d\tau = t - \int_0^1 H(t - \tau) \, d\tau = t - \int_0^t 1 \, d\tau = 0.$$

So, the mean is always 0. Using the same strategy, you should be able to derive $\Gamma(s, t) = \min(s, t) - st$.

As for Part (b), several people derived the conclusion unsatisfactorily, i.e., those of you simply substituted $\phi_k(t) = \sqrt{2}\sin(k\pi t)$ into the integral equation, computing the eigenvalues, and claimed they are the eigenfunctions. With this argument, you cannot be sure whether there exists other eigenfunctions. The correct argument is to derive the eigenvalue problem in the ordinary differential equation from the integral equation, that is:

$$
\begin{aligned}
\lambda\phi(t) &= \int_0^1 \Gamma(t, s)\phi(s) \, ds \\
&= \int_0^1 (\min(t, s) - ts)\phi(s) \, ds \\
&= \int_0^t (s - ts)\phi(s) \, ds + \int_t^1 (t - ts)\phi(s) \, ds \\
&= \int_0^t s\phi(s) \, ds - t\int_0^t s\phi(s) \, ds + t\int_t^1 (1 - s)\phi(s) \, ds
\end{aligned}
$$

Now, differentiating both sides with respect to t leads to the following ODE:

$$\phi''(t) = -\frac{1}{\lambda}\phi(t)$$

The boundary condition can be derived by setting $t = 0$ and $t = 1$ in the above integral equation. It turns out to be the *Dirichlet* boundary condition:

$$\phi(0) = \phi(1) = 0.$$

Several of you derived this boundary condition by saying that $X(0) = X(1) = 0$, which is not true! $X(0)$ could be nonzero because $\tau \sim \mathrm{unif}[0, 1)$ and consequently a jump could happen at $\tau = 0$. From these, we can derive the desired solution. Also, note that you need to justify that $\lambda > 0$ is of our only interest, and the $\lambda \leq 0$ case does not provide us the eigenfunctions. By the way, if you took MAT 207B, then you may have noticed that this covariance function is exactly the same as the *Green's function* for this Dirichlet boundary value problem!

For Part (c), some people used `svd` instead of `eigen` and claimed that `svd` gave them better or closer eigenvectors to the analytical ones compared to `eigen`. OK, why does this happen? It's a good exercise to think about it!

Also, note that the eigenvectors are not always uniquely determined for a given matrix. There is always an ambiguity about its sign. In other words, if ϕ is an eigenvector, you may get $-\phi$ depending on what software package you use.

Finally, some of you used `cov` function in Julia erroneously. This is very important: *Julia's* `cov` *function assumes the row vectors of a data matrix* `X` *are the realizations (or observations), which follows the convention in Statistics. We have been following the convention of the Signal Processing literature, i.e., the column vectors are the realizations/observations.* Hence, if you followed Problem 3's setup, you need to run `cov(X')` or `cov(X, dims=2)` instead of just `cov(X)`. But the best approach is not to compute the covariance matrix followed by `eigen`; as I explained in the class, you should compute the SVD of `X`. Please see my Lecture 10 slides.

Also, to compute the error of given two matrices `A, B`, it is usually best to use *the relative Frobenius norm error*, i.e., `norm(A-B)/norm(A)`. Note that this `norm` function in Julia behaves differently from that of MATLAB. So, you have to be careful. See any numerical linear algebra textbook such as [10] for more about the matrix/operator norms.

Finally, if you compute the error between the theoretical eigenvectors (sinusoids in this case) and the computed eigenvectors, you need to realize one important thing: *there is a sign ambiguity in each computed eigenvector.* In other words, we do not know a priori if an eigenvector ϕ_k is computed or $-\phi_k$ is computed. Hence, you need to compute the error between the kth theoretical vector and ϕ_k as well as the error between that theoretical vector and $-\phi_k$, then pick the smaller one as the real error; otherwise the true error could not be computed.

For more information about this stochastic process, please read the following papers: [2], [3], [5, p. 19], [9].

Problem 4: Here, I would like to point out some major mistakes several people made.

- Some of you treated each face as a matrix, and a set of faces as 3D array, which make many procedures and computations more cumbersome than necessary. It is much easier on Julia to treat each face as a vector of length 128^2 and a training dataset as a matrix of size $128^2 \times 72$. Then you can always convert a resulting vector after processing (e.g., reconstruction or approximation) as a matrix by the `reshape` command, which you can use to display as a face.

- Suppose X is a training dataset (matrix) of size $128^2 \times 72$. `mean(X, dims=2)`, which immediately gives you the average face vector. No need to compute the mean using a `for` loop.

- It is good to note that the function `dct` in the `FFTW.jl` package computes the 2D version of DCT if you just give a matrix input without any option. If you want to compute the 1D DCT of each column vector of an input matrix, then you should run `dct(X, dims=1)`.

- Suppose $\widetilde{X}$ is the data matrix after subtracting the mean column vector (i.e., the mean face) from each column vector. $\widetilde{X}$ can be quickly computed by the following Julia construct: `X-mean(X, dims=2)*ones(1,N)` or `X-repeat(mean(X, dims=2),1,N)`. Now, the sample covariance matrix is $\widehat{\Gamma} := (1/N)\widetilde{X}\widetilde{X}^\mathsf{T}$. (Note that several people forgot the factor $1/N = 1/72$ here.) You can compute the eigenvectors of $\widehat{\Gamma}$ using MATLAB's `eigen` function. Alternatively, you can use the SVD of $\widetilde{X}$, via `U, S, V = svd(Xtilde)`; where `Xtilde` represents $\widetilde{X}$ in Julia of course, and use `Xtilde * V` for the KLB basis vectors. However, you should be careful! The column vectors of `Xtilde * V` are orthogonal but clearly not orthonormal. In order to make them orthonormal, you need to divide its jth column vector by either σ_j or the norm of that column, as I discussed in Lecture 10. The best way to compute the top k KLB, however, is to use the `tsvd` in the `TSVD.jl` package as follows: `U, S, V = tsvd(Xtilde, k)`; Then, the column vectors of `U` is the KLB, which are already orthonormal, as I showed in my Lecture 10.

- It is very important to know that the `eigen` function in Julia sorts the eigenvalues and eigenvectors in the *nondecreasing order*, i.e., from the smallest to the largest. Thus, to use the top k KLB vectors means that you need to use the last k KLB vectors in the KLB matrix if you do not reorder it immediately after getting it from `eigen`. That is why the relative ℓ^2 curves for KLT were worse than those of DCT for some of you. For those of you made that mistake, I would strongly suggest that you recompute the error curve and plot against those of the DCT! Note that if you use `svd` or `tsvd` to compute the top k KLB via the column vectors of `U` or `Xtilde * V`, then you can use the first k column vectors. This is because `svd` and `tsvd` sort the singular values and singular vectors in the *nonincreasing order*, i.e., from the largest to the smallest!

- Suppose $\widetilde{X}_{\text{recon}}$ be the reconstructed or approximated version of $\widetilde{X}$. Then, the relative ℓ^2 error of the kth face is defined as

$$Relerr[k] = \frac{\|\widetilde{X}[:,k] - \widetilde{X}_{\text{recon}}[:,k]\|}{\|\widetilde{X}[:,k]\|}, \quad k = 1, \ldots, N,$$

  from which you can compute the average error easily by `mean(Relerr)`.

- The other thing I want to point out is that *one should use the inverse transform routines to compute the basis functions.* Note that if the input signal is one of the basis functions/vectors, then the output is one of the standard basis vector. This means that if you apply the inverse transform to the identity matrix, you get all the basis functions. Thus, use `idct` to compute the DCT basis vectors! That's much faster and nicer than the code segments some of you wrote.

- Finally, the most important thing I wanted to convey to you by this problem is the following: The KLB is an excellent tool for compressing the training dataset, but not necessarily for the test dataset unless the covariance matrix of the test dataset is the same as or very close to that of the training dataset. With a relatively small number of the signals in the training and test datasets, this usually won't happen. On the contrary, the DCT performs on both the training and test datasets in the same way. There should be no essential difference between its performance on the training dataset and that on the test dataset.

For more information about this dataset, please read the following papers [4], [6].

Problem 5:

References

[1] B. BÉNICHOU AND N. SAITO, *Sparsity vs. statistical independence in adaptive signal representations: A case study of the spike process*, in Beyond Wavelets, G. V. Welland, ed., vol. 10 of Studies in Computational Mathematics, Academic Press, San Diego, CA, 2003, ch. 9, pp. 225–257.

[2] J. B. BUCKHEIT AND D. L. DONOHO, *Time-frequency tilings which best expose the non-Gaussian behavior of a stochastic process*, in Proc. International Symposium on Time-Frequency and Time-Scale Analysis, IEEE, 1996, pp. 1–4. Jun. 18–21, 1996, Paris, France.

[3] D. L. DONOHO, M. VETTERLI, R. A. DEVORE, AND I. DAUBECHIES, *Data compression and harmonic analysis*, IEEE Trans. Inform. Theory, 44 (1998), pp. 2435–2476. Invited paper.

[4] M. KIRBY AND L. SIROVICH, *Application of the Karhunen-Loève procedure for the characterization of human faces*, IEEE Trans. Pattern Anal. Machine Intell., 12 (1990), pp. 103–108.

[5] Y. MEYER, *Oscillating Patterns in Image Processing and Nonlinear Evolution Equations*, vol. 22 of University Lecture Series, Amer. Math. Soc., Providence, RI, 2001.

[6] N. SAITO, *Image approximation and modeling via least statistically dependent bases*, Pattern Recognition, 34 (2001), pp. 1765–1784.

[7] ——, *The generalized spike process, sparsity, and statistical independence*, in Modern Signal Processing, D. Rockmore and J. D. Healy, eds., vol. 46 of MSRI Publications, Cambridge University Press, 2004, pp. 317–340.

[8] N. SAITO AND B. BÉNICHOU, *The spike process: a simple test case for independent or sparse component analysis*, in Proc. 3rd International Conference on Independent Component Analysis and Signal Separation, T.-W. Lee, T.-P. Jung, S. Makeig, and T. J. Sejnowski, eds., IEEE, 2001, pp. 698–703. Dec. 10–12, 2001, San Diego, CA.

[9] N. SAITO, B. M. LARSON, AND B. BÉNICHOU, *Sparsity and statistical independence from a best-basis viewpoint*, in Wavelet Applications in Signal and Image Processing VIII, A. Aldroubi, A. F. Laine, and M. A. Unser, eds., vol. Proc. SPIE 4119, 2000, pp. 474–486. Invited paper.

[10] L. N. TREFETHEN AND D. BAU, III, *Numerical Linear Algebra*, SIAM, Philadelphia, 1997.