January 9, 2024

Comments received for *On the use of the Gram matrix for multivariate functional principal components analysis*

Note: Ed's comments in blue.

Reviewer 1

Comment #1 :

The paper is lacking any application besides the simulation study. I would find it illustrative to see a real data setting where the alternative estimation strategy improves on existing methods in terms of computation and/or results, as well as a discussion of any potential differences in results.

Reply: Ed proposed to use the same application as in [Shi et al., 2022], the MNIST dataset. We may have some problems with the smoothness of the surfaces.

That was an initial thought, since I know first-hand that their iterative approach was the only way they could compute FPCs for image-valued data but it is still slow. It would have been nice to show you can get the same FPCs, but faster! But if you go down to comment 4 from this reviewer (comment 4 and bullet point 4), MNIST is only really *univariate* functional data (observed on a 2-d domain). Are there examples of open-source datasets with curves and images?

Comment #2 :

Your assumption of no error hardly seems realistic in practice. What are your recommendations in practice? Pre-smoothing can be an option, but also induces error that is later not taken into account, and can have problems in not so dense case. It seems that in a setting with error, covariance smoothing based options may have an advantage and it would be good to discuss this more thoroughly and include a corresponding setting in the simulations.

Reply:

We can propose the 'smoothing-first-then-estimation' settings to handle the case of densely sampled curves with errors. The sparse or irreguarlarly sampled case could be handled using the methodology of [Benko et al., 2009] for univariate curve. They fit a step function to each curve and consider some adjustments for the noise. This can be adapted for higher dimensional data.

In my opinion, showing that the method is applicable to different situations is sufficient. We cannot expect the method to be faster or more accurate in all situations, and we have been clear in where we think it is. Does a brief simulation for noisy/ sparse data go in supplementary material somewhere maybe, or just note that the implementation is available?

Comment #3 :

As the scalar product was introduced without weights, the difference and correspondence between weights in the scalar product and rescaling of the data does not become very clear in Section 3.4. I would suggest to briefly introduce the weighted version of the scalar product and the correspondence of rescaling with the square-root of the weight (if not rescaling point-wise). E.g. your $w_p(t_p)$ in the second formula really corresponds to the point-wise version of the square root of the proposal in the first formula, as the first formula uses weighting and the second formula rescaling. Not sure how your own proposal fits in there – with the square-root it should be a rescaling rather than a weighting as stated, surely? Why do you propose the integrated squared norm of C_p rather than the variance and how do they relate? You could also mention the option of weights already on p. 3, as it seems relevant in many applications.

Reply:

I am actually not sure to understand this comment. The weights introduced in Section 2 are not the same as the ones that we talked about in Section 3.4. The first ones weight the rows of the matrix, given more or less weight to the observations (that would be a choice in some study), while the other ones weight the columns, making sure that all the features have roughly the same variance. Our proposal is derived from the total inertia of the clouds, so I would suggest to modify this part much.

What do you think about it?

Yes, observation weights are denoted by π_n and these weight observations, similar to using weights in e.g., weigthed averages or weighted least squares. However, I think what the reviewer is asking is in the formula

$$\langle\langle f, g \rangle\rangle := \sum_{p=1}^{P} \left\langle f^{(p)}, g^{(p)} \right\rangle = \sum_{p=1}^{P} \int_{\mathcal{T}_p} f^{(p)}(t_p) g^{(p)}(t_p) \mathrm{d}t_p, \quad f, g \in \mathcal{H},$$

could be generalised to show how the *variable weights* introduced in later sections factor into the scalar product. I think they just require something like the way Happ and Greven introduce their Equation (4) (s a generalisation of Equation (3), i.e.,

$$\langle\langle f, g \rangle\rangle_w := \sum_{p=1}^{P} w_p \left\langle f^{(p)}, g^{(p)} \right\rangle = \sum_{p=1}^{P} w_p \int_{\mathcal{T}_p} f^{(p)}(t_p) g^{(p)}(t_p) \mathrm{d}t_p, \quad f, g \in \mathcal{H},$$

Therefore, when we use the phrase "Happ and Greven (2018) propose to weight each component p…", it's clear how these weights are incoporated – that's my understanding… Now, the comment regarding weights vs. rescaling is valid:

1. Happ and Greven propose weights that **enter their scalar product** as above (hence they are exponentiated to $^{-1}$ rather than $^{-1/2}$).
2. My understanding is that in practice the weights in Chiou et al. (and Jacques and Preda, which maybe we should reference), are applied to **normalise the data, pointwise** before running mFPCA (hence they are exponentiated to $^{-1/2}$ rather than $^{-1}$).
3. This reviewer is assuming since our proposed weights are exponentiated to $^{-1/2}$ rather than $^{-1}$, that we propose to reweight observations by that constant rather than use those weights in the scalar product (i.e., similar to 1. rather than 2.).

I'll need pen and paper to double check all this, but it really shouldnt be hard to verify and provide definite clarification in the paper – I'm happy to deal with this part of revision if you wish?

Comment #4 :

In the simulation study, I have several questions regarding the choices made.

- *You do not seem to give sufficient detail to judge whether your comparison to Happ & Greven (2018) is fair. Not sure whether it is sufficient to use the FCP-TPA (Allen, 2013) option, as this seems to assume some separability (?) that may or may not be warranted, it would be better to also include another sensible basis choice such as tensor product B-splines. Right now it is hard to judge whether worse results in the second scenario are due to the image setting or this particular choice, given similarly good results in the first scenario. Also, how did you chose the smoothing parameter and how many univariate basis functions did you retain, as truncating too early will lead to a loss of information?*

- *The log-AE in (12) can become negative if the absolute difference is below 1. This does not seem like a sensible error measure and the relative versions below non-sensical. Please check.*
- *Which are the orthonormal basis functions chosen in Simulation scenario 1?*
- *Why do you only consider a univariate functional (image-valued) setting in Scenario 2 if your method targets multivariate functional data, not say a second one-dimensional function as in Happ & Greven?*
- *Figures have small axis labels etc. and are hard to read. Figures 7 and 9 are particularly bad. Figure 9 seems to be empty besides results for $p = 2$ – is this due to incomplete results or choice of a bad depiction? You state that in the other setting besides $p = 2$ the results of the two methods are 'identical' (which would indeed yield a boxplot not distinguishable from the 1-line), which is hardly exactly possible, please check. Sometimes the text does not seem to match the (later) figures, please check. There are likely too many figures and some parts may need to be delegated to an appendix.*

Reply:

- The FCP-TPA is used because I did not want to use a pre-specify basis, and it is the only method Happ and Greven propose in that case. The other ones expands the surfaces in a basis first (probably tensor product B-splines). Of course, it will result to better results if we expand the data first in this basis, as it is how the data have been generated. The idea was to do as-if we did not know the data generating process. The smoothing parameters for the FCP-TPA are the default ones. Otherwise, no smoothing parameters is involved. See the other paper for the comment on the early truncating resulting in a loss of information. Data are generated by Foruier but not measured with error. . . should it make that much of a difference to expand on a B-spline tensor product? I think I had made a note of the separability assumption being made in the initial comments, can revisit and look closer if needs be.
- I propose to just remove the log part. I do not remember why I used the log here.
- The Fourier basis. To be add in the text.
- To show the benefit of using the Gram matrix in the case of multidimensional data, even if the data are univariate. We can add another settings with multivariate data with surfaces and curves. Again, it would be really great to get some open-source data of curves and images.
- I'll check but the results are really identical. We may choose some settings to be in the main text, and let the complete figure for the appendix.

Smaller comment:

- *Please introduce any notation and names you use, even if also used by one of the papers you cite. For example, 'feature' middle of p. 3. What is $\mathcal{Z}_{n,\cdot}$ in 4.1?*
- *Your notation should distinguish between theoretical and empirical quantities, i.e. their estimators. For example $\mu^{(p)}$ or $C_{p,q}$ on p. 3 are used for both.*
- *distance should be similarity, last line of p. 3. diag would be clearer as blockdiag, p. 4 ff.*
- *The distinction between f and M_f could be clearer (p. 6 ff). Can it be that $M_f = M_g$ for $f \neq g$?*

- *The discussion of (9) and (10) could more clearly point out where quantities in (10) are the p-th element of the corresponding quantities in (9) (μ and c_k) and where they aren't (ϕ_k vs. φ_k) as the relationship can otherwise be confusing.*
- *In 4.1, I believe it should be "For a feature $X(p)$, the eigenfunctions and eigenvectors are computed using [not as] a matrix decomposition of the estimated covariance ..." as you will need to do a rescaling of eigenvectors to ensure orthonormality with respect to the functional not vector inner product, correct?*
- *On p. 9, v_k is used for the eigenvectors of both Z and M, please change one notation.*
- *You seem to go a bit back and forth between the weighted and equally weighted observation case, could you make this a bit more consistent?*
- *For the statements of computation complexity in Section 4.3, I could easily see some, while others seemed less obvious. It would be good to have brief derivations in an appendix. Could you also state the resulting complexities when using SVD in Remark 5?*
- *In the discussion, you say "... whereas the decomposition of the covariance operator necessitates the specification of the percentage of variance accounted for by each individual univariate feature. Specifying the percentage of variance explained for each feature does not guarantee that we recover the nominal percentage of variance explained for the multivariate data." An option that should be mentioned here as it is usually advisable is to retain* 100% *of the possible univariate components and to only truncate the multivariate expansion, which does not cause this particular problem.*
- *In the appendix, please make clearer where one proof ends and another starts and which formula in the paper you are showing at any given time.*

Reply:

- I'll add something for 'feature'. $\mathcal{Z}_{n,.}$ is the nth row of the matrix $\mathcal{Z}$.
- I do agree with that. I'll talk about it with Valentin at CASI, I choose to do it like that to not overloaded the paper. I'll change that, and eventually add a discussion about the convergence of the estimator.
- OK (oops – one of us should have spotted that similarity vs. distance one!)
- The question that two points in the space could be equal if the function are not equal is actually a good one. And I don't know how to answer to it. Some calculations may be necessary. I'll levae these calculations to you first!
- OK
- OK
- Notation are already different $\boldsymbol{v}$ and v. It is very subtle, though. Are there alternatives?
- I'll probably remove the equally weighted part in the diagonalization of the covariance operator / inner product matrix sections.
- OK, I'll add in the appendix.
- We can add a comment on that, pointing to the other paper. In practice, it is not a great idea to retain 100% of the variance for the univariate components as it will result to a large (even infinite) number of components.

- OK

Reviewer 2

Comment #1 :

The computational complexity in Section 4.3 does not seem to be accurate. The role of the dimension P was not correctly reflected in the results. The authors concluded that the number of features P did not have much impact on the computational complexity. This sounds counter intuitive. More discussion and careful examinations are needed for this section.

Reply:

I'll check the computation of the complexity in Section 4.3, but it should be correct. The influence of the number of feature P is here "only" because there are more sampling points when P increased. That's why I would say that P does not have much impact on "the ratio of the computational complexity". Of course, if we only look at the computational complexity for the methods separetely, it will increase with P. I'll make that clear in the text.

Comment #2 :

The simulation results are somewhat confusing. For the computational time, the authors reported the ratio of the computation time between the covariance method and the Gram matrix method. Figure 5 seems to indicate that when P is small, the ratio has a large variance. Please provide explanations for this.

Reply:

I don't think that the variance of the ratio is really larger when P is small. It might be because of some "strange" simulations or some artifacts in the ICHEC clusters.

Comment #3 :

The covariance decomposition method is faster for most of the cases in Figure 5. The authors need to explain the empirical results and check if they match the results in Section 4.3.

Reply:

I'll do the calculation to check that. But it may also mean that, in practice, some part of the algorithm is better implemented than others. For example, diagonalisation of a matrix is highly optimized in Python, while I implemented my own version of the inner-product matrix...

Comment #4 :

Figure 6 shows that the the Gram matrix method is faster in scenario 2. The authors then concluded that the Gram matrix method is faster for two-dimensional functional data without a clear explanation. It seems to me that this is primarily due to the fact that the number of sampling points is M^2, which is much larger than the sample size N.

Reply:

I also think it is because M^2 is much larger than N. I'll make that clear in the text.

Comment #5 :

In Figure 7 and Figure 9, the performance of the two methods are pretty much the same when the functions are defined on a one-dimensional domain. However, Figure 8 and Figure 10 indicate that when the domain is two-dimensional, the covariance decomposition method is more accurate for the estimation of eigenvalues; while the Gram matrix method is more accurate for the estimation of eigen-functions. This result needs further examination and explanation. The authors simply concluded that the Gram matrix method should be used for multi-dimensional functional data (image), which is pre-mature.

Reply:

The conclusion that the Gram matrix is better suited for surfaces is mostly due to Figure 12, not Figure 8 and 10. The reconstruction is better in that case. But I agree that other results, eigenfunctions and eigenvalues, may need more explanation.

References

[Benko et al., 2009] Benko, M., Härdle, W., and Kneip, A. (2009). Common functional principal components. *The Annals of Statistics*, 37(1):1–34.

[Shi et al., 2022] Shi, H., Yang, Y., Wang, L., Ma, D., Beg, M. F., Pei, J., and Cao, J. (2022). Two-Dimensional Functional Principal Component Analysis for Image Feature Extraction. *Journal of Computational and Graphical Statistics*, 31(4):1127–1140.