Fast Covariance Estimation Techniques

Kenny Huang

Advisor: Drs. Nicholas Marshall, Amit Singer

Submitted in partial fulfillment

of the requirements for the program of

Applied and Computational Mathematics

Princeton University

April 2023

Abstract

My independent work centers around a paper by Bhamre et. al that proposes a new method of Covariance Wiener Filtering (CWF) to extract the underlying signal from a large number of noisy 2D projections. This method requires the estimation of the covariance matrix of the underlying signal of these 2D images, for which the paper provides an adequate method. Under Dr. Nick Marshall's guidance, I sought to find a way to further improve the efficiency of this methodology. In the end, Nick and other collaborators completed a paper outlining their "Fast-CWF" solution, which results in a speedup of two magnitudes on large datasets. In this report, I will discuss my contributions to Nick and my work, specifically in regards to what we call the "lookup table" method.

Acknowledgements

I'd like to take this opportunity to thank Nick for your enthusiastic guidance. I truly enjoyed working with you and wish you the best at Oregon State University. In addition, thank you to Professor Singer, both for writing the paper that was the precursor to my work and your work as a second reader. Finally, thank you to Yunpeng Shi, who I worked with closely in conjunction with Nick.

Contents

Chapter 1

Fast Covariance Estimation Techniques

1.1 Motivation

My work is based on a paper called "Denoising and covariance estimation of single particle cryo-EM images" by Bhamre et. al. Essentially, this paper sought to take a large number of noisy 2D projections of a macromolecule, which can be seen in Figure 1.1, and denoise the images to unveil the underlying signal. Previous methods apply clustering algorithm to group the projections and take class averages of those groups to isolate the signal (Figure 1.1).

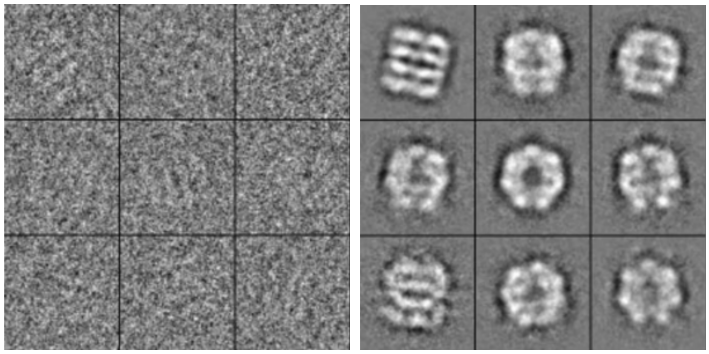

Figure 1.1: Noisy projections and results of class-average approach to extract signal

In contrast, this paper proposes a new method called Covariance Weiner Filtering, which is unique in that it is able to complete this denoising process on an image-by-image basis, as seen in Figure 1.2. To do so, this method requires the pre-estimation of the covariance matrix of the underlying signal in these images. The paper provides an efficient methodology for this computation using conjugate gradient, but the goal of our work is to improve this further.

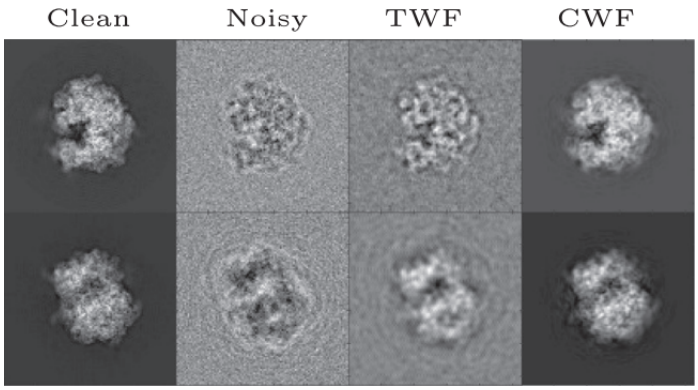

Figure 1.2: Results of Covariance Weiner Filtering approach to extract signal

1.2 Introduction

The images in the paper dataset are modeled with a convolutional operation $y_i = a_i \circ x_i + e_i$, where y_i are the observed images and a_i are known convolutional filters that are inherent properties of the lenses the images were taken with. Importantly, as matrices describing lenses, we assume that the a_i are radially symmetric. We can now apply a Fourier transform to arrive at the following equation:

$$Y_i = A_i X_i + \epsilon_i$$

This equation uses matrix operations instead, which allows for techniques like differentiation. From here, we can now seek to solve for the covariance matrix of X_i.

1.3 Solving for Covariance Matrix

We begin with the previous expression:

$$Y_i = A_i X_i + \epsilon_i$$

For this section of the paper, we assume that the noise is white noise, i.e. that $X_i \sim N(0, \sigma)$, so taking the expectation on both sides, we get that

$$\begin{aligned} E[Y_i] &= E[A_i X_i + \epsilon_i] \\ &= A_i E[X_i] + E[\epsilon_i] \\ &= A_i E[X_i] \end{aligned}$$

Computing the covariance of Y_i, we can get the following expression:

$$\begin{aligned} E[(Y_i - E[Y_i])(Y_i - E[Y_i])^T] &= E[(Y_i - A_i E[X_i])(Y_i - A_i E[X_i])^T] \\ &= E[A_i(X_i - E[X_i])(X_i - E[X_i])^T A_i^T] \\ &= A_i E[(X_i - E[X_i])(X_i - E[X_i])^T] A_i^T \\ &= A_i \Sigma A_i^T + \sigma^2 I \end{aligned}$$

Since noise exists here, we can introduce a loss function of the sum of the norms of the difference between the two sides. Our covariance matrix estimation is thus the solution to the corresponding optimization problem.

$$\begin{aligned} L(\Sigma) &= \sum_{i=1}^{n} \| (Y_i - E[Y_i])(Y_i - E[Y_i])^T - (A_i \Sigma A_i^T + \sigma^2 I) \|_F \\ \hat{\Sigma} &= \arg\min_{\Sigma} L(\Sigma) \end{aligned}$$

Differentiating and setting to zero, we get the following crucial equation:

$$\sum_{i=1}^{n} A_i^T A_i \hat{\Sigma} A_i^T A_i = \sum_{i=1}^{n} A_i^T C_i A_i - \sum_{i=1}^{n} \sigma^2 A_i^T A_i$$

where $\mu = E[X_i]$ and $C_i = (Y_i - A_i\mu)(Y_i - A_i\mu)^T$. This yields an equation that contains what we are trying to compute, X_i, and where everything else is known, i.e. A_i and Y_i. In theory, we could compute this directly, but when working with a large number of large images this method becomes intractable. To proceed, the paper uses a method called conjugate gradient, which treats the left-hand side as a linear operator and solves for an approximate solution in an iterative manner. This is more efficient than brute force, but our goal is to optimize this process even further.

1.4 Methodology

Our idea to do so is to estimate the elements in this equation quickly even if we lose some accuracy, so long as the relative error is below a threshold. One preliminary step to consider is to efficiently estimate $\sum A_i^T A_i$, which is the right-most term in the central equation. Brute-force computation has computational complexity of $O(nL^3)$, so this is the baseline that all of our results are compared against. Recall that A_i is radially symmetric; to take advantage of this property, we outline what we call the lookup table method, which is as follows:

We sample the points along the diagonal and store their values in a vector v as well as the distances from the center of the image. We then precompute the pairwise products of those values and store them in a matrix $B = vv^T$. We can now define a function h that takes a pixel location and outputs the sampled point from the diagonal with largest distance from the center less than that of our pixel. Once we've

done so, we can follow this equation to estimate each element of A^TA:

$$(A^TA)_{i,j} = \sum_{k=1}^{L} A_{k,i}A_{k,j} \approx \sum_{k=1}^{L} v_{h(k,i)}v_{h(k,j)} = \sum_{k=1}^{L} B_{h(k,i),h(k,j)}$$

In short, we rewrite each entry as a sum of products of values of A as the result of matrix multiplication and then use the h function to estimate each of those values of A by replacing it with the closest sampled point. Since there are limited combinations of sampled points, we can precompute their pairwise products to minimize recalculations. Essentially, instead of computing the sum of products of values from A, we are now computing the sum of values from B.

Once here, we can also introduce linear splines. The previous calculation replaces each value of A with only one of the two sampled pixels surrounding it; incorporating the values of both as well as the distance of the pixel from each significantly improves our estimates. With vectorizing, the linear spline calculation takes the following form:

$$A^TA \approx W_{0,0}C_{0,0} + W_{0,1}C_{0,1} + W_{1,0}C_{1,0} + W_{1,1}C_{1,1}$$

$$(C_{0,0})_{i,j} = \textstyle\sum_{k=1}^{L} B_{h(k,i),h(k,j)} \qquad (C_{0,1})_{i,j} = \textstyle\sum_{k=1}^{L} B_{h(k,i),h(k,j)+1}$$
$$(C_{1,0})_{i,j} = \textstyle\sum_{k=1}^{L} B_{h(k,i)+1,h(k,j)} \qquad (C_{1,1})_{i,j} = \textstyle\sum_{k=1}^{L} B_{h(k,i)+1,h(k,j)+1}$$

In other words, A^TA is approximately a weighted average of four slightly different calculations, where the weight matrices are functions of the distances from the pixels to the surrounding sample pixels. All operations in this calculation are element-wise. In terms of time complexity, this methodology is on the order of $O(L^3 + nL^2)$, which asymptotically beats the brute force computation.

1.5 Results

To empirically test our method, we ran each method 1000 times and took the average of the time elapsed and relative error. We can see in Table 1.1 that the weighted method is indeed less affected by an increase in image count than the direct computation, so even though the times were similar for the largest size we tested, we can assume that asymptotically the weighted method is faster.

Table 1.1: Time Elapsed Results

# of Images	Direct Comp	Unweighted	Weighted	fbpca_1	fbpca_2
1024	0.042	0.100	0.139	0.118	0.129
2048	0.077	0.116	0.144	0.162	0.170
4096	0.150	0.150	0.165	0.288	0.310

Direct Comp: direct computation.

Compared to the unweighted method, the weighted method also significantly reduced relative error to just 1.3% (Table 1.2). More tests could have been done, but there were technical issues with the computing cluster, so this is the extent of the empirical results available.

Table 1.2: Relative Error Results

# of Images	Direct Comp	Unweighted	Weighted	fbpca_1	fbpca_2
1024	-	0.117	0.013	0.340	0.339
2048	-	0.118	0.013	0.319	0.332
4096	-	0.118	0.013	0.328	0.330

Direct Comp: direct computation.

In addition to the unweighted and weighted methods, we also implemented a pair of estimation methods involving Facebook's proprietary fast PCA estimation package, denoted "fbpca_1" and "fbpca_2". These methods hold potential because PCA

decomposes A into some matrices that are diagonal matrices or cancel in the A^TA operation, greatly simplifying the overall calculation. However, the decomposition itself is expensive, and we can see that empirically the faster computation cannot make up for that inefficiency. The relative error also explodes for images past a certain size.

Finally, in addition to our efforts to efficiently compute different parts of the covariance equation, we also explored more high-level avenues of how to replace the conjugate gradient process, such as eigenvalue analysis and Fourier-Bessel series, which would incorporate a special form of Fourier transform using Bessel functions.

1.6 Conclusion

In all, my contributions were largely probing and did not lead to any major breakthroughs. Even so, I am very happy with the progress that we made, and I learned a lot about cryo-EM imaging, linear algebra intricacies, and the research process as a whole.

As I mentioned, I worked with Nick during the spring. As the semester came to end, I shifted my focus to my summer internship, but in the meantime, Nick and the rest of the team had a great summer of their own and were able to make significant progress and publish a paper. They ended up modifying and refining the Fourier-Bessel method that we had tried near the end of the semester. Naturally, I was a bit disappointed that I could not directly contribute to the contents of the final paper, but I believe that my work helped illuminate ideas and avenues for Nick and his collaborators to explore. In addition, they very graciously mentioned me in their acknowledgements, which I very much appreciated.