Dynamic Covariances in Time Sequence Data

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Master of Science

in

Computer Science

by

Hao Chang

DARTMOUTH COLLEGE

Hanover, New Hampshire

May, 2017

1 Abstract

High dimensional time sequence data that seem highly random at first glance can intrinsically be structured processes with covariance that changes over time (dynamic covariance). Discovery and analysis of the dynamic covariances behind time sequence data can not only help us understand underlying patterns, but also give us insight into the inherent connections between the events represented by the data. In order to accurately recover dynamic covariances from time sequence data, we present the Time Sequence Covariance Recovery (TimeCorr) method. TimeCorr first approximates covariance fragments at each time point using a fixed range of neighboring time points, then calculates the covariance at each time point t by finding the sum of the Gaussian average over the covariance fragments centering at time point t. To illustrate the effectiveness of TimeCorr, we compared its performance with that of the more probabilistic Multivariate Gaussian Process approach and other deterministic covariance recovery techniques. We show that TimeCorr produces more accurate and more stable results across all of our test cases.

2 Acknowledgements

I would first like to thank my mentors Professor Qiang Liu of the Computer Science Department at Dartmouth College and Professor Jeremy Manning of the Psychological and Brain Sciences Department at Dartmouth College. Professor Liu and Professor Manning were great inspirations for me throughout this project. Their deep understanding of their respective fields and constant flow of bright ideas really expanded my understanding of what it means to love and excel at what I want to do. Professor Liu and Professor Manning provided timely guidance whenever I was in need and constantly encouraged me to try out new ideas and to reach for higher standards.

I would also like to thank Professor Hany Farid of the Computer Science Department at Dartmouth College. Professor Farid kindled my first spark of interest in Computer Science and pushed me to pursue graduate education at Dartmouth. He has been a role model for me throughout my five years at Dartmouth and will always be an inspiration for me in my future pursuits in life.

I would also like to acknowledge Dilin Wang, Jun Han and Yihao Feng in the DartML laboratory of the Computer Science Department at Dartmouth College. Their door were always open whenever I ran into technical difficulties or had questions about my research or writing.

Finally, I must express my very profound gratitude to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

3 List of Figures

4 Introduction

High dimensional time sequence data that seem very random on the surface can have underlying dynamic covariances that intricately weaves structure and connection into variable behaviors. Finding these dynamic covariances can give us insight into patterns in the data, which has practical values in many different areas including finance, economics, brain science, mathematics, etc. However, two major problems inhibit the traditional covariance formula from effectively recovering dynamic covariances:

Problems:

1. Finding covariance using data over long time ranges decreases focus on the time point of interest
2. Finding covariance using data within short time ranges introduces instability in estimation and are often inaccurate due to lack of data

To address these problems, we employed the Multivariate Gaussian Process (MGP) approach—a very traditional Bayesian Machine Learning model. However, the probabilistic model was overly computationally intensive and gave very poor performance in recovering dynamic covariances from our synthetic time sequence datasets. Shifting our focus, we decided to explore deterministic methods for the solution. Finding inspiration directly from the covariance formula

$$cov(x, y) = \frac{1}{N} \sum_{i=1}^{N} (x_i - E(X)) \cdot (y_i - E(Y)) \tag{1}$$

and partially incorporating the intrinsic structure of the MGP model, we designed the Naive TimeCorr method by applying Gaussian averaging in our covariance formula. This method addresses the lack of locality in the covariance formula by increasing the influence of time points that are closer to our center of interest. Although the results were slightly better than that of our MGP approach, the recovered dynamic covariances were highly unstable with underwhelming accuracy. Finally, we decided to modify the structure of our formula to put even more emphasis on locality. To achieve this, we estimated covariance fragments at each time point using a tight neighborhood of adjacent points, then applied Gaussian averaging over these covariance fragments to integrate information from the rest of the dataset. This is our TimeCorr method. To illustrate our approach, we generated synthetic datasets with underlying dynamic covariances. We demonstrate that TimeCorr is able to accurately recover the dynamic covariances under our synthetic dataset and show that the results significantly improved compared our previous efforts.

5 Model Descriptions

5.1 The Problem

Given a time sequence dataset of dimensions $T \times V$, we put it into the framework of a Multivariate Gaussian Process with number of variables V, number of time points T, and underlying dynamic covariance C_t at time t. At every time point t, the covariance C_t is a square matrix of dimensions $V \times V$, with the element C_T^{ij} at the i^{th} row and j^{th} column representing the joint variability of variable i and variable j at time t. Our mission is to recover the dynamic covariance C_t at each time point.

5.2 Multivariate Gaussian Process (MGP)

The MGP narrative formulates dynamic covariance recovery as an optimization problem. At each time point t, we proposed that the covariance features of each variable v_t^i can be represented as a feature vector f_t^i of arbitrary length. Furthermore, given the general pattern of the data, we also proposed a kernel function that can use the feature vectors to capture the covariance between each variable. The setting can be illustrated using the following equations:

$$f_t \sim \mathcal{N}(f_{t-1}, s_0)$$
$$C_t^{ij} = K(f_t^i, f_t^j) = 2 * \exp(-\frac{||f_t^i - f_t^j||^2}{2\sigma_k^2}) - 1$$
$$v_t \sim \mathcal{N}(v_{t-1}, C_t)$$

where f_t, C_t and v_t represent the feature vectors, covariance matrices and variable values, respectively. Finally, we used the Adagrad algorithm to optimize our feature vectors at each time point until the likelihood function converges.

The general optimization process is delineated below:

1. **Multivariate Gaussian Process Parameter Initialization**

   (a) Given the dataset described in 5.1, first calculate variable differences between adjacent time points to form matrix A, where the element at the t^{th} row and i^{th} column $A_t^i = v_t^i - v_{t-1}^i$

   (b) Use the Spacial Distance method from the Scipy library to calculate the covariance between different variables using the entire time series

   (c) Use the Multidimensional Scaling algorithm to extract feature vectors from the covariance matrices in step 2. These features will be heretofore referred to as average features f_a, and will later be used to help initialize features f_t for each time point t

2. **General Optimization Process**

   (a) For each time point t, the feature map f_t is formed by adding random Gaussian noise to the average feature vector f_a:

   $$f_t = f_a + \mathcal{N}(0, \sigma_n)$$

   (b) The covariance matrix C_t for each time point is then calculated using the kernel function:

   $$C_t^{ij} = K(f_t^i, f_t^j) = 2 * \exp(-\frac{||f_t^i - f_t^j||^2}{2\sigma_k^2}) - 1$$

(c) Optimize the likelihood function:

$$
\begin{aligned}
L(a|f) &= \sum_t P(f_t)P(a_t|f_t) \\
&= \sum_t [-\frac{1}{2}(a_t^T C_t^{-1} a_t) - \frac{1}{2}\log ||C_t|| - \frac{f_0^2}{2\sigma_0^2} - \sum_t \frac{(f_t - f_{t-1})^2}{2\sigma_f^2}]
\end{aligned}
$$

using the Adagrad algorithm:

$$f_{new} = f_{prev} + \frac{\eta}{\sqrt{G+\epsilon}} \nabla_f L(a|f)$$

Where $G = \sum_{0:prev} \nabla_f L(a|f)$, a sum of all previous gradients

(d) Derivation for the likelihood function and the derivative is included in the Appendix

5.3 Naive TimeCorr

The Naive TimeCorr approach was our initial attempt at using a deterministic approach to recover dynamic covariances from time sequence data. Seeking to combine the simplicity of the covariance formula and the structure of MGP, the Naive TimeCorr method intuitively incorporates Gaussian coefficients into the covariance formula.

The algorithm is described below:

1. Given the dataset described in 5.1, calculate and normalize a list of Gaussian coefficients centering around t with variance σ for each time point t

2. Calculate covariance between variables v^i and v^j at time point t using the following equation:

$$C_T^{ij} = \frac{1}{Z}\sum_{s=0}^{T}(v_s^i - \bar{v_t^i}) \cdot (v_s^j - \bar{v_t^j}) \cdot \mathcal{N}(s|t,\sigma) \tag{2}$$

Where

$$Z = \sum_{s=0}^{T} \mathcal{N}(s|t,\sigma) = 1 \tag{3}$$

$$\bar{v_t^i} = \sum_{s=0}^{T} v_s^i \cdot \mathcal{N}(s|t,\sigma) \tag{4}$$

3. Repeat above steps for all variable pairs and for all time points

5.4 TimeCorr

The TimeCorr method was developed to further improve local accuracy and to address the instability issues that manifested from the Naive TimeCorr method. Derived from the same idea behind the Naive Timecorr method—incorporating Gaussian averaging into covariance calculation—the TimeCorr method seeks to achieve stability and accuracy by first estimating covariance fragments at each time point using a small neighborhood of adjacent points, then applying Gaussian averaging over the covariance fragments. Like the Naive TimeCorr method, the TimeCorr approach employs a deterministic approach that improves accuracy by using Gaussian averaging; and stability, incorporating information from all available time points. However, TimeCorr is able to achieve a higher level of performance by averaging over covariance fragments—which is better at capturing local patterns—rather than single point covariances $(x_t^i - E(X))(y_t^i - E(Y))$.

The algorithm is described below:

1. Covariance fragments estimation emphasizing locality
   (a) Given the dataset described in 5.1, choose a neighborhood of R time points for local estimation
   (b) Estimate covariance fragment between variables v^i and v^j at time point t

using the following equation:

$$c_t^{ij} = \sum_{s=t-R}^{t+R} (v_s^i - \bar{v^i})(v_s^j - \bar{v^j}) \tag{5}$$

Where

$$\bar{v^i} = \frac{1}{2R+1} \sum_{s=t-R}^{t+R} v_s^i \tag{6}$$

(c) Repeat above steps for all variable pairs and for all time points

2. Apply Gaussian coefficients over covariance fragments of all time points to guarantee stability and consistency

   (a) For each time point t, calculate and normalize a list of Gaussian coefficients centering around t with variance σ: $\mathcal{N}(t, \sigma)$

   (b) Calculate covariance between variables v^i and v^j by applying the Gaussian coefficients to the covariance fragments calculated before:

$$C_t^{ij} = \frac{1}{Z} \sum_{s=0}^{T} c_s^{ij} * \mathcal{N}(s|t, \sigma) \tag{7}$$

   Where, given normalization step from before,

$$Z = \sum_{s=0}^{T} \mathcal{N}(s|t, \sigma) = 1 \tag{8}$$

   (c) Repeat above steps for all variable pairs and for all time points

6 Results

6.1 Multivariate Gaussian Process (MGP)

To accurately assess the covariance recovery performance of the MGP Process, we need to take into consideration recovery of both the dynamic covariance and the feature vectors. To do this, we took special steps to generate the synthetic data.

Synthetic Data Initialization

1. The synthetic dataset is designed to contain 5 time points, 5 variables and 10 features for each variable.

2. The feature vector f_1 containing the covariance features for the first time point is initialized using a normal distribution with mean $\mu = 0$ and standard deviation s_0:

$$f_1 \sim \mathcal{N}(0, \sigma_0)$$

3. Then, the covariance between different variables v_1^i and v_1^j is calculated using their corresponding feature vectors f_1^i and f_1^j and the following kernel function:

$$C_1^{ij} = K(f_1^i, f_1^j) = 2 * \exp(-\frac{||f_1^i - f_1^j||^2}{2\sigma_k^2}) - 1$$

4. After forming the covariance matrix C_1 for the first time point by calculating the covariance between all variable pairs, we can generate variable values v_1 for the first time point using multivariate Gaussian distribution with mean 0 and covariance C_1

$$v_1 \sim \mathcal{N}(\mathbf{0}, C_1)$$

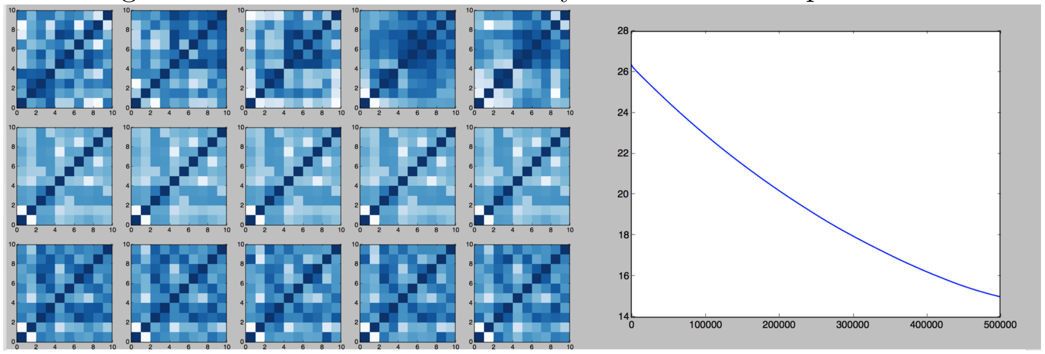
5. For all subsequent time points t, the features f_t, covariance matrices C_t and voxel activations v_t are generated using information from the previous time point $t-1$ with the following procedures:

$$\begin{aligned} f_t &\sim \mathcal{N}(f_{t-1}, s_0) \\ C_t^{ij} &= K(f_t^i, f_t^j) = 2 * \exp(-\frac{||f_t^i - f_t^j||^2}{2\sigma_k^2}) - 1 \\ v_t &\sim \mathcal{N}(v_{t-1}, C_t) \end{aligned}$$

Optimization and Results

After experimenting with many setups, we found that the best parameters for optimization is 100,000 iterations of adagrad with step size of 0.01. The results of our optimization is shown in Figure 1

Figure 1: MGP covariance recovery and likelihood optimization



The rows in the left graph, from top to bottom, represent the ground truth covariances. the initializations and the recovered covariances, respectively; the columns, time points from 1 to 5. The graph on the right shows the value of the negative log-likelihood function over number of iterations.

From the right graph, we can see that MGP, through the process of optimizing the feature vectors and in extension the dynamic covariances, increases the likelihood (decreasing the negative log-likelihood) of the model. This is further confirmed from the results in the left graph. Comparing with the initializations, the recovered covariances objectively show more resemblance to the ground truth. However, when looking across all the time points, the recovered covariances looks very similar throughout the time sequence, without any distinctive features that could differentiate one time point from another. This is due to the Bayesian nature of MGP, by which the optimization process seeks to find the most probable covariance at each time point. However, as the ground truth is not always the point of highest probability on the distribution, MGP falls short of TimeCorr with respect to covariance recovery accuracy.

In addition, throughout the hundreds of test cases we went through, we discovered that MGP is prone to being stuck in local minimums. This problem is further exacerbated when we increase the number of variables, time points and feature length. As the dimensionality of the dataset increases, the probability space becomes increasingly more complex and more difficult to optimize on, and thus the likelihood sometimes sees minimal improvement over thousands of iterations.

Framing dynamic covariance recovery as a MGP optimization problem also increases the demand for computing resources. Even with only 5 time points, the MGP model takes 10-100 times longer to converge to a decent result than a TimeCorr model with 1000 time points. Furthermore, like most other Machine Learning models, it is very difficult to find the optimal configuration for optimization of the MGP model.

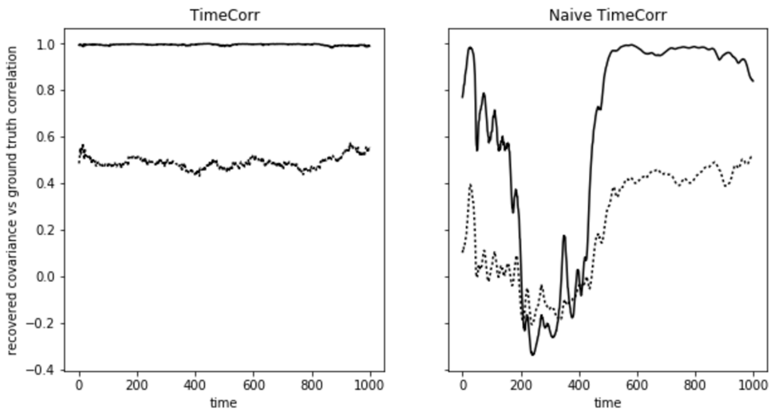
Lastly, selection of the kernel function is a very important factor when applying the MGP Model. Different data patterns correspond to very different kernel functions. Thus, correct application of the MGP Model also requires prior knowledge of the structure of the dataset.

6.2 TimeCorr and Naive TimeCorr

To compare the performance of TimeCorr and Naive TimeCorr, we generated three different datasets, all of which has Gaussian Variance=1000, Number of Time points=1000 and Number of Variables=5, to highlight TimeCorr's superiority in consistency and local accuracy:

1. A validation dataset with a constant covariance matrix C for all time points, and a random unrelated covariance matrix for comparison. The results are displayed in Figure 2

Figure 2: Validation Correlation: Ground Truth (solid); Random (dotted)



From Figure 2, we can see that recovery from TimeCorr is highly consistent and robust across all time points. In addition, TimeCorr also creates greater distinction between dissimilar covariances. Furthermore, TimeCorr appears to be immune to the irregularities that is manifested by the Naive TimeCorr method

2. A fading dataset with dynamic covariance C_t as a combination of one covariance C_1 linearly fading into another covariance C_2, where $C_t = \frac{1000-t}{1000} * C_1 + \frac{t}{1000} * C_2$ at time point t. The results are displayed in Figure 3

Figure 3: Fading Correlation: C_1 (solid), C_2 (dotted)

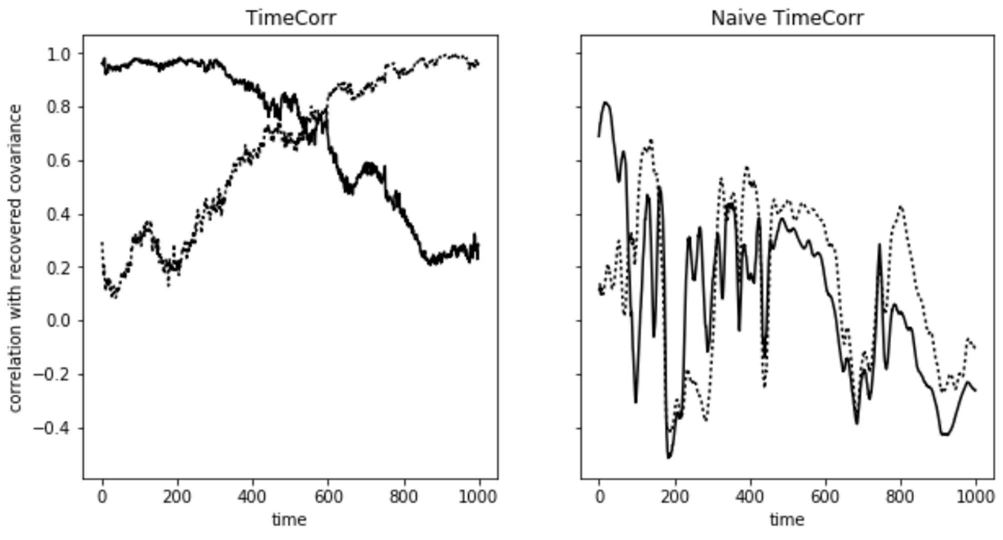


Figure 3 shows the correlation between recovered dynamic covariances and ground truth. The TimeCorr approach is able to clearly recover the X shape that is to be expected of the correlations, whereas the Naive TimeCorr method is barely able to retrieve this pattern. From this comparison, we can see that TimeCorr is able to consistently recover dynamically changing covariances with very high accuracy, while results from the Naive TimeCorr method appear to be very inconsistent.

3. A cross dataset generated using C_1 as covariance for the first half, then abruptly changing to another covariance C_2 for the second half. The results are displayed in Figure 4

Figure 4: Cross Correlation: C_1 (solid), C_2 (dotted)

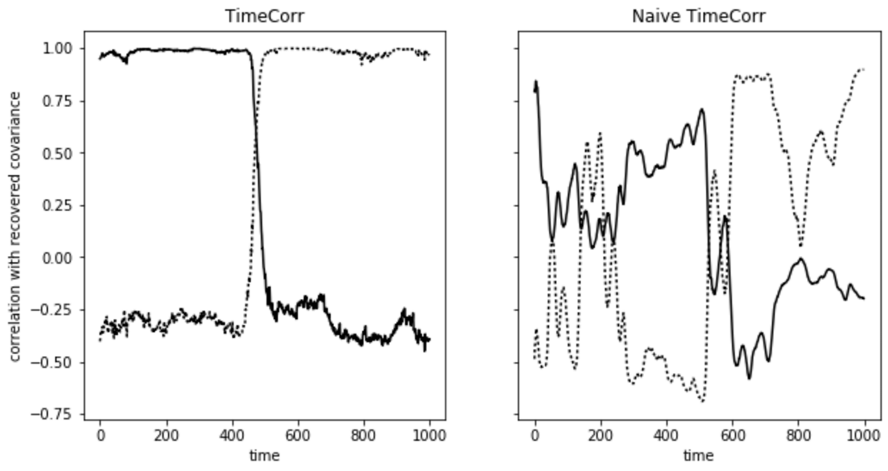


Figure 4 displays the correlation between the recovered dynamic covariances and C_1 and C_2. As expected, the TimeCorr results remain consistently horizontal on the two ends of the graph, and distinctly and abruptly crosses at the midpoint. This behavior shows that the TimeCorr approach is robust to sudden changes in the covariance. In comparison, although the Naive TimeCorr method roughly displays the cross pattern, it is very inaccurate in estimation and generally very inconsistent.

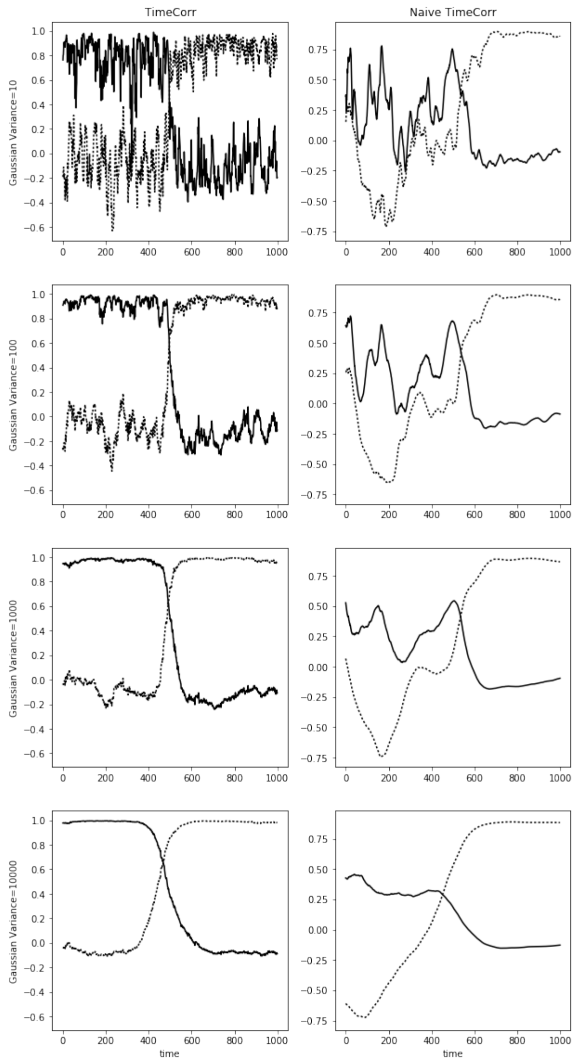
6.3 More on TimeCorr

To understand the full scope of capabilities of TimeCorr, we ran further tests using multiple configurations of varying Gaussian variances and Covariance Fragment Estimation Ranges (CFER)—the number of neighboring time points we use to calculate each covariance fragment. Through increasing the Gaussian variance of the Gaussian distribution behind the coefficients, the influence of a wider range of time points on the recovery process is increased. We expected that this will stabilize the results of our recovery even more. Following the same logic, we also explored different ranges for the CFER to see if higher values will also have stabilizing effects on our recovery.

In addition, to understand the limitations of TimeCorr, we attempted to use TimeCorr on multiple synthetic datasets that had dynamic covariances changing at increasingly rapid rates. As TimeCorr depends heavily on neighboring time points in the recovery of dynamic covariances, we suspected that TimeCorr will break down if the covariances change too fast.

1. Varying Gaussian Variance and comparing between TimeCorr and Naive TimeCorr

Figure 5: Varying Gaussian Variance: TimeCorr vs Naive TimeCorr

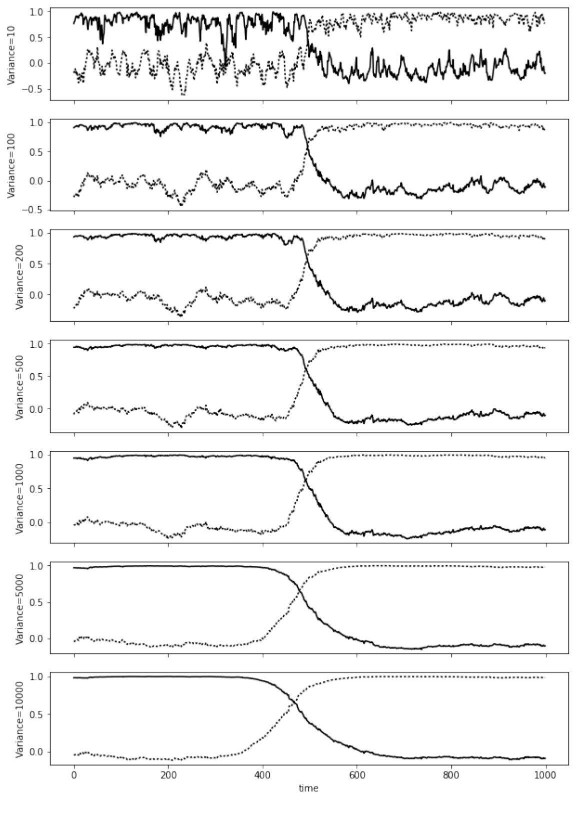


As the variance of the Gaussian distribution behind the coefficients increases, neighboring time points gain more influence. As a result, irregular fluctuations in the correlation are dampened and flattened. This effect is evident in results from both the TimeCorr and the Naive TimeCorr method. In Figure 5, we can see that the expected X shape becomes more prominent as Gaussian Variance increases. At the same time, abrupt changes in the underlying dynamic covariance are dragged out into longer time frames in the recovered covariance from TimeCorr models with higher Gaussian Variance. For dynamic covariances that shifts slowly, this effect will stabilize the recovery process. But for dynamic covariances that changes rapidly, higher Gaussian variance may have adverse effects on the recovery.

2. Varying Gaussian Variance over a greater range, TimeCorr only

   To further emphasize the effects of increasing Gaussian variance, we implemented TimeCorr on a cross dataset using Gaussian variances of 10, 100, 200, 500, 1000 and 10,000.

Figure 6: Varying Gaussian Variance: focusing TimeCorr

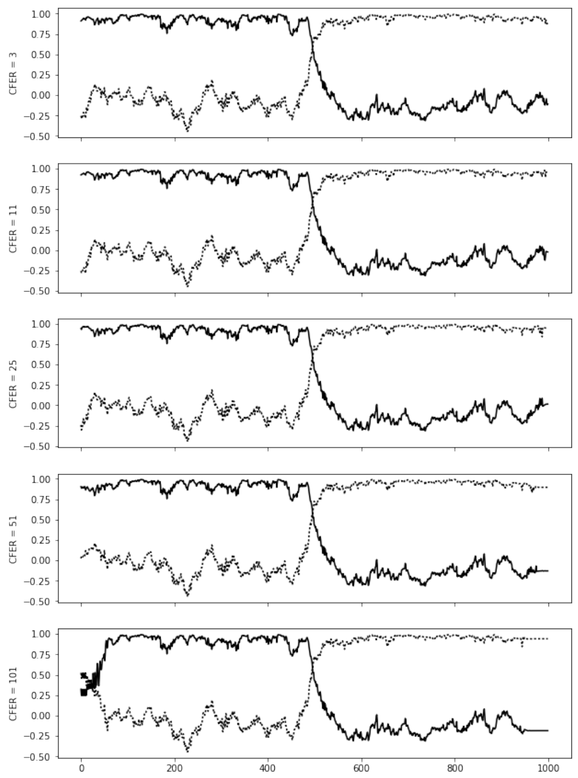


Consistent with our previous results, we see in Figure 6 that, as the Gaussian Variance increases, the curve becomes more smooth. However, the transition between the two covariances increases from 100 time points to almost 300 time points as Gaussian Variance increases from 10 to 10,000.

3. Varying Covariance Fragment Estimation Range (CFER), TimeCorr only

   Another aspect of the TimeCorr that we explored is the length of the neighborhood that we use to calculate the covariance fragments—Covariance Fragment Estimation (CFER). We implemented TimeCorr on a cross dataset using CFER of 3, 11, 25, 51 and 101.

   Figure 7: Varying CFER

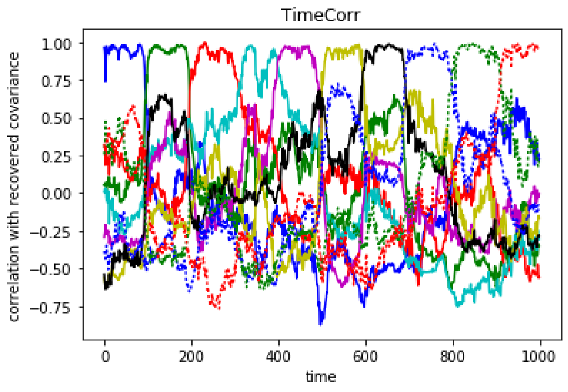
   

   From Figure 7, we see that there is not a huge difference in TimeCorr performance when using CFER of 3 to 25. However, as CFER increases beyond 51, we start to see abnormalities on the edges of the graphs. In addition, we suspect that TimeCorr models with large CFER values do not perform well on time sequence datasets with rapidly changing dynamic covariances. This is verified in the following section.

4. TimeCorr performance on time sequence data with rapidly changing covariance

Our first synthetic dataset was generated using 10 random covariances, each persisting for 100 time points before abruptly changing into another. For covariance recovery, we choose to use a TimeCorr model with Gaussian Variance=100 and CFER=3. The correlation between the recovered covariance and the true covariances are displayed in Figure 8, with each line representing correlation with a different true covariance.

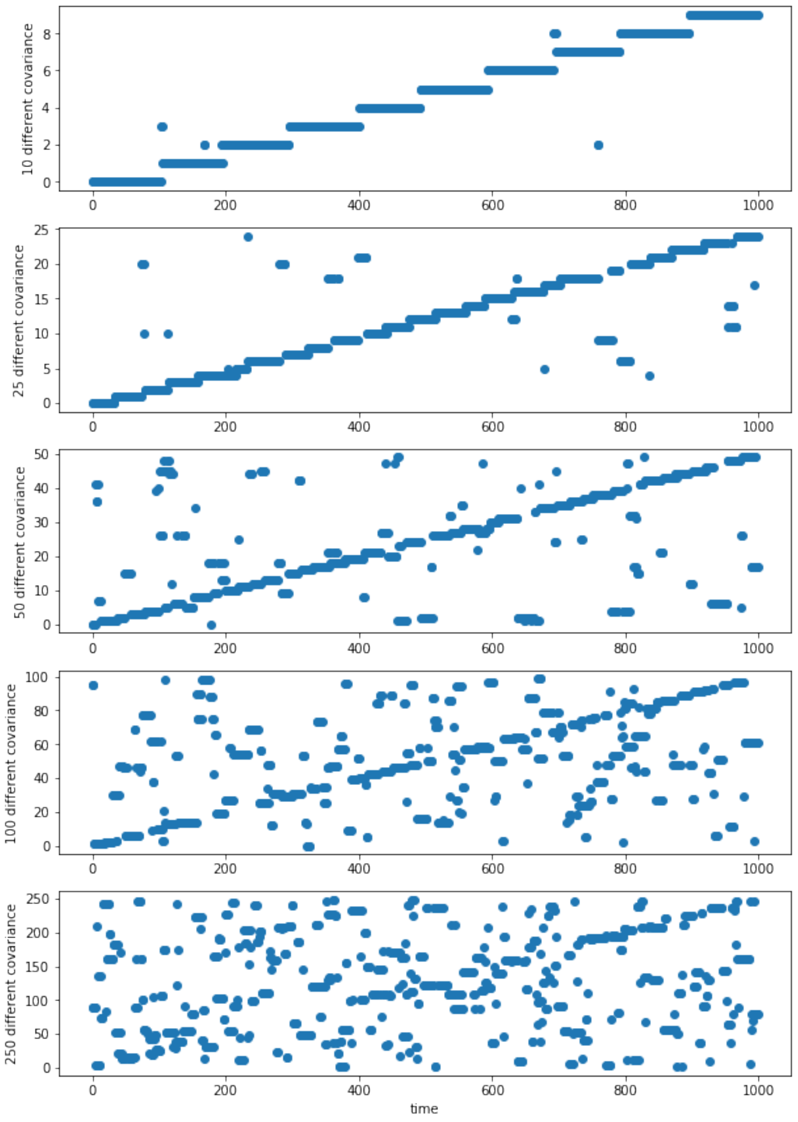
Figure 8: TimeCorr performance on dataset with 10 distinct covariances



From Figure 8, we can see that the correlation between recovered covariance and the true covariances at each time point is distinctively higher than the correlation between recovered covariance and other covariances. For further testing and for better visualization, we used TimeCorr on 1000 time point synthetic datasets with 10, 25, 50, 100 and 250 covariances. The true covariances are indexed from 0 to [number of covariances-1], sequentially. The scatter plot shows the index of the true covariance that has the highest correlation with the recovered covariance at each time point. In the ideal scenario, the points should be distributed along the diagonal.

The average percentage of time points that the TimeCorr model with CFER of 3 was able to correctly recover from 10 repetitions per scenario was 94.3%, 79.4%, 52.2%, 19.9% and 5.2% for datasets with 10, 25, 50, 100 and 250 distinct covariances, respectively. This is consistent from the results shown in Figure 9, where the recovered covariances is consistently correct for datasets with slower changing dynamic covariances, but become more random for datasets

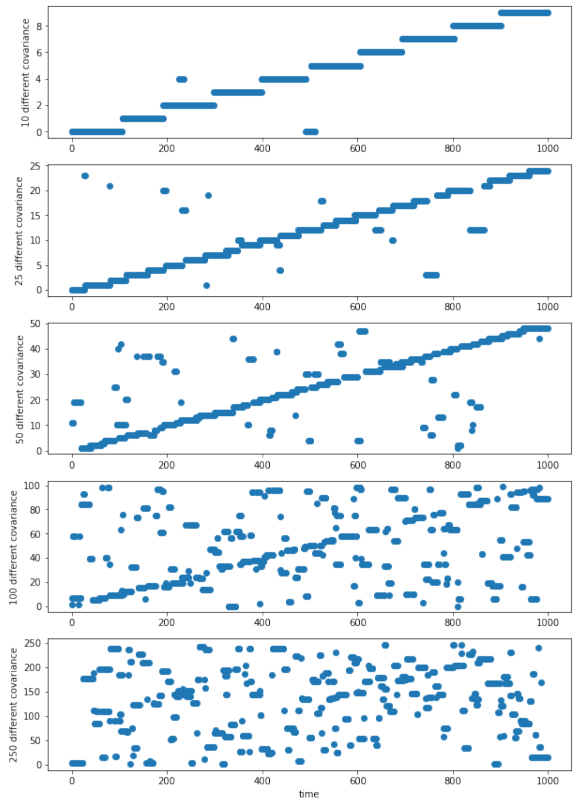
Figure 9: TimeCorr performance with Variance=100 and CFER=3



whose underlying dynamic covariances change more rapidly. As the time spread of each distinct covariance approaches the Covariance Fragment Estimation Range, TimeCorr become increasingly less accurate. To further verify the inverse relationship between recovery accuracy and the difference between CFER and dataset covariance time interval, we changed CFER to 11 and received the results in Figure 10.

The average percentage of time points that the TimeCorr model with CFER of 11 was able to correctly recover from 10 repetitions per scenario was 94.6%,

Figure 10: TimeCorr performance with Variance=100 and CFER=11

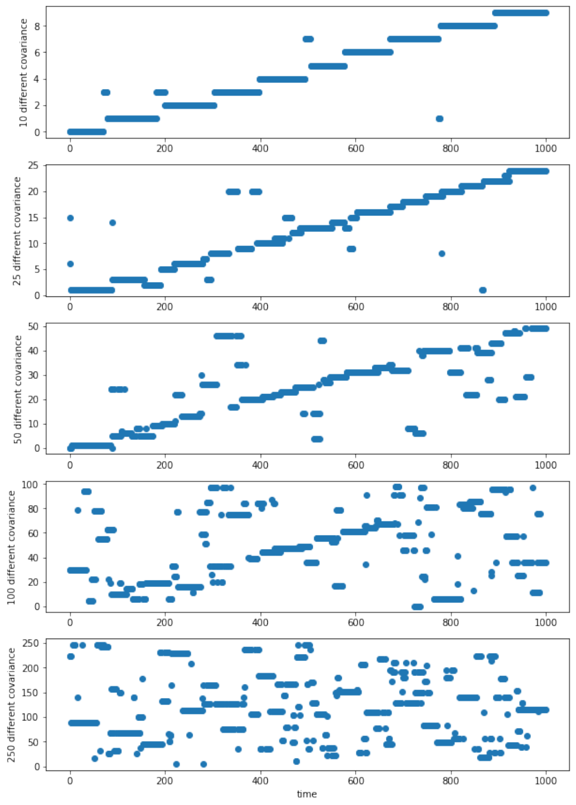


81.2%, 58.3%, 14.5% and 3.1% for datasets with 10, 25, 50, 100 and 250 distinct covariances, respectively. When the covariance time interval is greater than the CFER, the accuracy for the model with variance=100 and CFER=11 is consistently higher than that of the model with variance=100 and CFER=3—94.6% vs 94.3%, 81.2% vs 79.4%, 58.3% vs 52.2%—which means longer CFER produces higher recovery accuracy. However, when the covariance time interval = $\frac{1000}{100} = 10$, the accuracy for the model with CFER=3<10 becomes significantly higher than that of the model with CFER=11>10, which proves that the recovery accuracy of TimeCorr is highly dependent on the CFER being shorter than the time interval each distinct covariance occupies.

Finally, to analyze the effect of Gaussian Variance on recovery accuracy, we conducted testing using a TimeCorr model with Variance=1000 and CFER=3. The results are show in 11

The average percentage of time points that TimeCorr was able to correctly

Figure 11: TimeCorr performance with Variance=1000 and CFER=3



recover from 10 repetitions per scenario was 84.7%, 49%, 19.9%, 7.8% and 1% for datasets with 10, 25, 50, 100 and 250 distinct covariances, respectively. The results were consistently worse that that of the TimeCorr model with Gaussian Variance=100. This is also evident qualitatively as the results in Figure 11 appears to be much more random than that of Figure 9. Therefore, we conclude that smaller Gaussian Variance produces higher recovery accuracy on datasets with rapidly changing dynamic covariances

7 Conclusion

In conclusion, we demonstrated that the TimeCorr model performs significantly better than both the Naive TimeCorr model and the Multivariate Gaussian model across all of our test cases. Specifically, the TimeCorr method

1. takes up significantly less computing resources when comparing with the Multivariate Gaussian Model
2. achieves more accurate and consistent recovery of the dynamic covariances at every time point
3. is more sensitive to both gradual and abrupt changes in the dynamic covariances
4. is very simple and intuitive, and very easy to configure

In addition, to maximize dynamic covariance recovery accuracy of time sequence data, it is important to:

1. select proper Covariance Fragment Estimation Range (CFER) that is short enough to capture the rate of change of the dynamic covariance, but also large enough to maximize accuracy
2. select proper Gaussian Variance that is proportional to the size of the dataset

Next Steps

As recovery of dynamic covariances in time sequence data is a very general problem that exists in many different fields—Brain Science, Finance, Economics, Biology, Mathematics, etc—we hope to organize this method into a toolbox and make it available for the general public.

In addition, the development of the TimeCorr model was conceived as an answer to the demand from the Contextual Dynamics Laboratory to recover the dynamic covariances within the fRMI data. We are excited to see what kind of interesting patterns we can obtain when we incorporate the TimeCorr model in analysis of the human brain.

8 Appendix

1. **Likelihood Function**

$$\begin{aligned}
L(a|f) &= \sum_t P(f_t)P(a_t|f_t) \\
&\sim \sum_t \log[P(f_t)P(a_t|f_t)] \\
&= \sum_t [\log P(f_t) + \log P(a_t|f_t)] \\
&= \sum_t [-\frac{1}{2}(a_t^T C_t^{-1} a_t) - \frac{1}{2}\log\det(C_t) + \log P(f_t)] \\
&= \sum_t [-\frac{1}{2}(a_t^T C_t^{-1} a_t) - \frac{1}{2}\log ||C^t|| + \log[\exp(-\frac{f_0^2}{2\sigma_0^2})\prod_t \exp(-\frac{(f_t - f_{t-1})^2}{2\sigma_f^2})] \\
&= \sum_t [-\frac{1}{2}(a_t^T C_t^{-1} a_t) - \frac{1}{2}\log ||C_t|| - \frac{f_0^2}{2\sigma_0^2} - \sum_t \frac{(f_t - f_{t-1})^2}{2\sigma_f^2}]
\end{aligned}$$

2. **Multivariate Gaussian Process Gradient Derivation**

Feature k of variable i at time t is represented by f_t^{ik}. The gradient of the likelihood function with respect to f_t^{ik} is:

$$\begin{aligned}
\nabla_{f_t^{ik}} L(a|f) &= \underset{f_t^{ik}}{\operatorname{argmax}}\, P(f_t)P(a_t|f_t) \\
&= \underset{f_t^{ik}}{\operatorname{argmax}} [\log P(f_t) + \log P(a_t|f_t)] \\
&= \underset{f_t^{ik}}{\operatorname{argmax}} [-\frac{1}{2}(a_t^T C_t^{-1} a_t) - \frac{1}{2}\log ||C_t|| - \frac{(f_0^{ik})^2}{2\sigma_0^2} - \sum_t \frac{(f_t^{ik} - f_{t-1}^{ik})^2}{2\sigma_f^2}]
\end{aligned}$$

To find the optimal features that maximizes this probability, we take the derivative:

$$\frac{\delta[P(f_t)P(a_t|f_t)]}{\delta f_t^{ik}} = \begin{cases} \frac{1}{2}a_t^T C_t^{-1}\frac{\delta C_t}{\delta f_t^{ik}}C_t^{-1}a_t - \frac{1}{2}tr(C_t^{-1}\frac{\delta C_t}{\delta f_t^{ik}}) - \frac{f_t^{ik}}{\sigma_0^2} & \text{for } t = 0 \\ \frac{1}{2}a_t^T C_t^{-1}\frac{\delta C_t}{\delta f_t^{ik}}C_t^{-1}a_t - \frac{1}{2}tr(C_t^{-1}\frac{\delta C_t}{\delta f_t^{ik}}) - \frac{f_t^{ik} - f_{t-1}^{ik}}{\sigma_f^2} & \text{for } t \neq 0 \end{cases}$$

Setting $\alpha_t = C_t^{-1} a_t$ and simplifying:

$$\frac{\delta[P(f_t)P(a_t|f_t)]}{\delta f_t^{ik}} = \begin{cases} \frac{1}{2} tr[(\alpha_t^T \alpha_t - C_t^{-1}) \frac{\delta C_t}{\delta f_t^{ik}}] - \frac{f_t^{ik}}{\sigma_0^2} \text{ for } t = 0 \\ \\ \frac{1}{2} tr[(\alpha_t^T \alpha_t - C_t^{-1}) \frac{\delta C_t}{\delta f_t^{ik}}] - \frac{f_t^{ik} - f_{t-1}^{ik}}{\sigma_f^2} \text{ for } t \neq 0 \end{cases}$$

Given kernel function for the covariance matrix:

$$C_t^{ij} = K(f_t^i, f_t^j) = 2\exp(-\frac{||f_t^i - f_t^j||^2}{2\sigma_k^2}) - 1$$

The derivative of covariance matrix C_t with respect to feature f_t^{ik} becomes zero except for row j and column j, where it takes the value of:

$$\frac{\delta C_t^{ij}}{\delta f_t^{ik}} = -\frac{2(f_t^{ik} - f_t^{jk})}{\sigma_k^2} \exp(-\frac{||f_t^i - f_t^j||^2}{2\sigma_k^2})$$

*Part of this derivation comes from Murphy's "Machine Learning: A Probabilistic Perspective" [1]

9 References

[1] Murphy, Kevin P ”Gaussian Process.” Machine Learning: A Probabilistic Perspective. 2012: MIT Press.