EXPERIMENTAL DESIGN FOR FAST LINEAR ALGEBRA

STEPHEN HUAN* AND FLORIAN SCHÄFER†

Abstract. experimental design for linear algebra

Key words.

to do

AMS subject classifications.

1. Introduction.

[4] test citation

2. Greedy selection for directed inference.

2.1. Conditional k-th nearest neighbors.

Consider the simple regression algorithm kth-nearest neighbors (k-NN). Given a training set $X_{\text{Tr}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ and corresponding labels $\boldsymbol{y}_{\text{Tr}} = \{y_1, \ldots, y_n\}$, the goal is to estimate the unknown label y_{Pr} of some unseen prediction point $\boldsymbol{x}_{\text{Pr}}$ Stated informally, the k-NN approach is to select the k points in X_{Tr} *most informative* about $\boldsymbol{x}_{\text{Pr}}$ and combine their results.

Algorithm 2.1 Idealized k-NN regression

Given $(X_{\text{Tr}}, \boldsymbol{y}_{\text{Tr}}) = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ and $\boldsymbol{x}_{\text{Pr}}$

1. Select the k points in X_{Tr} most informative about $\boldsymbol{x}_{\text{Pr}}$
2. Combine the labels of the selected points to generate a prediction

One specific approach is intuitively, points close to $\boldsymbol{x}_{\text{Pr}}$ should be similar to it. So we select the k closest points in X_{Tr} to $\boldsymbol{x}_{\text{Pr}}$ and pool their labels (e.g., by averaging).

Florian: "specific" and "intuitively" are fluff, meaning that they don't really add information. In order to achieve crisp, high-quality academic writing, it is important to try to those fluff words as much as possible. It's normal to add them out of reflex initially, so it requires active postprocessing.

Algorithm 2.2 k-NN regression

1. Select the k points $\{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}\} \subseteq X_{\text{Tr}}$ closest to $\boldsymbol{x}_{\text{Pr}}$
2. Compute $\boldsymbol{y}_{\text{Pr}}$ by $\frac{1}{k}\sum_{j=1}^{k} y_{i_j}$

However, we can generalize the notion of "closest" with the *kernel trick*, by using an arbitrary kernel function to measure similarity. For example, commonly used kernels like the Gaussian kernel and Matérn family of covariance functions are *isotropic*; they depend only on the distance between the two vectors. If such isotropic kernels monotonically decrease with distance, then selecting points based on the largest kernel similarity recovers k-NN. However, kernels need not be isotropic in general — they just need to capture some sense of "similarity", motivating kernel k-NN.

Stephen: not sure whether "stationary" or "isotropic" are the right word(s) to use here

Algorithm 2.3 Kernel k-NN regression

Given kernel function $K(\boldsymbol{x}, \boldsymbol{y})$

1. Select the k points $\{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}\} \subseteq X_{\text{Tr}}$ most similar to $\boldsymbol{x}_{\text{Pr}}$
2. Compute $\boldsymbol{y}_{\text{Pr}}$ by an average weighted by similarity

Although the kernel k-NN approach is more general than its normed counterpart, it still suffers from a fundamental issue. Suppose the closest point to $\boldsymbol{x}_{\text{Pr}}$ has many

*Georgia Institute of Technology

†Georgia Institute of Technology, S1317 CODA, 756 W Peachtree St Atlanta, GA 30332, florian.schaefer@cc.gatech.edu, Corresponding Author

duplicates in the training set. Then the algorithm will select the same point multiple times, even though in some sense the duplicate point has stopped giving additional information about the prediction point. In order to fix this issue, we should be selecting new points *conditional* on the points we've already selected. This preserves the idealized algorithm of selecting points based on the information they tell us about the prediction point — once we've selected a point, conditioning on it reduces the information similar points tells us, encouraging diverse point selection.

Algorithm 2.4 Conditional kernel k-NN regression

1. Select the k points $\{\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}\} \subseteq X_{\mathrm{Tr}}$ most informative to $\boldsymbol{x}_{\mathrm{Pr}}$ after conditioning on all points selected beforehand
2. Compute $\boldsymbol{y}_{\mathrm{Pr}}$ by an average weighted by information

In order to make the notions of conditioning and information precise, we need a specific framework. Kernel methods lead naturally to Gaussian processes, whose covariance matrices naturally result from kernel functions and allows us to use the rigorous statistical and information-theoretic notions of conditioning and information.

Stephen: mention sensor placement/spatial statistics perspective/literature

2.2. Sparse Gaussian process regression. A *Gaussian process* is a prior distribution over functions, such that for any finite set of points, the corresponding function over the points is distributed according to a multivariate Gaussian. In order to generate such a distribution over an uncountable number of points consistently, a Gaussian process is specified by a *mean function* $\mu(\boldsymbol{x})$ and *covariance function* or *kernel function* $K(\boldsymbol{x}, \boldsymbol{y})$. For any finite set of points $X = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$,$f(X) \sim \mathcal{N}(\boldsymbol{\mu}, \Theta)$, where $\boldsymbol{\mu}_i = \mu(\boldsymbol{x}_i)$ and $\Theta_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

In order to compute a prediction at $\boldsymbol{x}_{\mathrm{Pr}}$, we can simply condition the desired prediction $\boldsymbol{y}_{\mathrm{Pr}}$ on the observed outputs and compute the conditional expectation. We can also find the conditional variance, which will quantify the uncertainty of our prediction. If we block our covariance matrix $\Theta = \begin{pmatrix} \Theta_{\mathrm{Tr},\mathrm{Tr}} & \Theta_{\mathrm{Tr},\mathrm{Pr}} \\ \Theta_{\mathrm{Pr},\mathrm{Tr}} & \Theta_{\mathrm{Pr},\mathrm{Pr}} \end{pmatrix}$ where $\Theta_{\mathrm{Tr},\mathrm{Tr}}, \Theta_{\mathrm{Pr},\mathrm{Pr}}, \Theta_{\mathrm{Tr},\mathrm{Pr}}, \Theta_{\mathrm{Pr},\mathrm{Tr}}$ are the covariance matrices of the training data, testing data, and training and test data respectively, then the conditional expectation and covariance are:

$$\mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] = \boldsymbol{\mu}_{\mathrm{Pr}} + \Theta_{\mathrm{Pr},\mathrm{Tr}} \Theta_{\mathrm{Tr},\mathrm{Tr}}^{-1} (\boldsymbol{y}_{\mathrm{Tr}} - \boldsymbol{\mu}_{\mathrm{Tr}}) \tag{2.1}$$

$$\mathrm{Cov}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] = \Theta_{\mathrm{Pr},\mathrm{Pr}} - \Theta_{\mathrm{Pr},\mathrm{Tr}} \Theta_{\mathrm{Tr},\mathrm{Tr}}^{-1} \Theta_{\mathrm{Tr},\mathrm{Pr}} \tag{2.2}$$

For brevity of notation, we will often denote the conditional covariance matrix as

$$\Theta_{\mathrm{Pr},\mathrm{Pr} \mid \mathrm{Tr}} := \Theta_{\mathrm{Pr},\mathrm{Pr}} - \Theta_{\mathrm{Pr},\mathrm{Tr}} \Theta_{\mathrm{Tr},\mathrm{Tr}}^{-1} \Theta_{\mathrm{Tr},\mathrm{Pr}} \tag{2.3}$$

When conditioning on multiple sets, the sets are given in order of computation. Although the resulting covariance matrix is the same, a different order of conditioning means different intermediate results in repeated application of (2.2). In general,

$$\Theta_{I,J \mid K_1, K_2, \ldots, K_n} := \mathrm{Cov}[\boldsymbol{y}_I, \boldsymbol{y}_J \mid \boldsymbol{y}_{K_1 \cup K_2 \cup \cdots \cup K_n}] \tag{2.4}$$

denotes the covariance between the variables in index sets I and J, conditional on the variables in $K_1, K_2, \ldots, K_n$. Note that by the quotient rule of Schur complementation:

$$\Theta_{I,J \mid K_{1\ldots n}} = \Theta_{I,J \mid K_{1\ldots n-1}} - \Theta_{I,K_n \mid K_{1\ldots n-1}} \Theta_{K_n,K_n \mid K_{1\ldots n-1}}^{-1} \Theta_{K_n,J \mid K_{1\ldots n-1}} \tag{2.5}$$

Note that calculating the posterior mean and variance requires inverting the training covariance matrix $\Theta_{\mathrm{Tr},\mathrm{Tr}}$, which costs $\mathcal{O}(N^3)$, where N is the number of training points. This scaling is prohibitive for large datasets, so many *sparse* Gaussian process regression techniques have been proposed. These methods often focus on selecting a subset of the training data that is most informative about the prediction points, which naturally aligns with our k-NN perspective. If s points are selected out of the N, then the inversion will cost $\mathcal{O}(s^3)$, which could be substantially cheaper if s is significantly smaller than N. The question is then how to select as few points as possible while maintaining predictive accuracy.

Stephen: cite sparse Gaussian regression papers

2.3. Problem: optimal selection. The natural criterion justified from the k-NN perspective is to maximize the *mutual information* between the selected points and the target point for prediction. Such information-theoretic objectives have seen success in the spatial statistics community [1], who use such criteria to determine the best locations to place sensors in a Gaussian process regression context. The mutual information, or *information gain* is defined as

$$\mathrm{I}[\boldsymbol{y}_{\mathrm{Pr}}; \boldsymbol{y}_{\mathrm{Tr}}] = \mathrm{H}[\boldsymbol{y}_{\mathrm{Pr}}] - \mathrm{H}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \tag{2.6}$$

We can use the fact that the entropy of a multivariate Gaussian is monotonically increasing with the log determinant of its covariance matrix to efficiently compute these entropies. Because the entropy of $\boldsymbol{y}_{\mathrm{Pr}}$ is constant, maximizing the mutual information is equivalent to minimizing the conditional entropy. From (2.2) we see that minimizing the conditional entropy is equivalent to minimizing the log determinant of the posterior covariance matrix. Note that for a single predictive point, this is monotonic with its variance. So another justification is that we are reducing the *conditional variance* of the desired point as much as possible. In particular, because our estimator is the conditional expectation (2.1), it is unbiased because $\mathrm{E}[\mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]] = \mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}}]$. Because it is unbiased, its expected mean squared error is simply the conditional variance since $\mathrm{E}[(\boldsymbol{y}_{\mathrm{Pr}} - \mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}])^2 \mid \boldsymbol{y}_{\mathrm{Tr}}] = \mathrm{Var}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]$ where the expectation is taken under conditioning because of the assumption that $\boldsymbol{y}_{\mathrm{Pr}}$ is distributed according to the Gaussian process. So maximizing the mutual information is equivalent to minimizing the conditional variance which is in turn equivalent to minimizing the expected mean squared error of the prediction. Another perspective on the objective can be derived from comparing the mutual information to the EV-VE identity, which states

$$\begin{aligned} \mathrm{H}[\boldsymbol{y}_{\mathrm{Pr}}] &= \mathrm{H}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] + \mathrm{I}[\boldsymbol{y}_{\mathrm{Pr}}; \boldsymbol{y}_{\mathrm{Tr}}] \\ \mathrm{Var}[\boldsymbol{y}_{\mathrm{Pr}}] &= \mathrm{E}[\mathrm{Var}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]] + \mathrm{Var}[\mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]] \end{aligned}$$

On the left hand side, entropy is monotone with variance. On the right hand side, the expectation of the conditional variance can be interpreted to be the fluctuation of the prediction point after conditioning, and is monotone with the conditional entropy. Because the expectation of conditional variance and variance of conditional expectation add to a constant, minimizing the expectation of the conditional variance is equivalent to maximizing the variance of conditional expectation, which we see corresponds to the mutual information term. Supposing $\boldsymbol{y}_{\mathrm{Pr}}$ was independent of $\boldsymbol{y}_{\mathrm{Tr}}$, then the conditional expectation becomes simply the expectation, whose variance is 0. Thus, the variance of the conditional expectation can be interpreted to be the "information" shared between $\boldsymbol{y}_{\mathrm{Pr}}$ and $\boldsymbol{y}_{\mathrm{Tr}}$, as the larger it is, the more the prediction for $\boldsymbol{y}_{\mathrm{Pr}}$ (the conditional expectation) depends on the observed results of $\boldsymbol{y}_{\mathrm{Tr}}$.

2.4. A greedy approach. We now consider how to efficiently minimize the conditional variance objective using a greedy approach. At each iteration, we pick the training point which most reduces the conditional variance of the prediction point.

Let $I = \{i_1, i_2, \dots, i_j\}$ be the set of indexes of training points selected already. Let the prediction point have index $n+1$, the last index. For a candidate index k, we update the covariance matrix after conditioning on y_k, in addition to the indices already selected according to (2.2):

$$\begin{aligned}\Theta_{:,:|I,k} &= \Theta_{:,:|I} - \Theta_{:,k|I}\Theta_{k,k|I}^{-1}\Theta_{k,:|I} \\ &= \Theta_{:,:|I} - \frac{\Theta_{:,k|I}\Theta_{:,k|I}^{\top}}{\Theta_{kk|I}}\end{aligned} \tag{2.7}$$

We see that conditioning on a new entry is a rank-one update on the current covariance matrix $\Theta_{|I}$, given by the vector $\boldsymbol{u} = \frac{\Theta_{:,k|I}}{\sqrt{\Theta_{kk|I}}}$. Thus, the amount that the variance of $\boldsymbol{y}_{\mathrm{Pr}}$ will decrease after selecting k is given by u_{n+1}^2, or

$$\frac{\mathrm{Cov}[\boldsymbol{y}_{\mathrm{Tr}}[k], \boldsymbol{y}_{\mathrm{Pr}}]^2}{\mathrm{Var}[\boldsymbol{y}_{\mathrm{Tr}}[k], \boldsymbol{y}_{\mathrm{Tr}}[k]]} = \frac{\Theta_{k,n+1|I}^2}{\Theta_{kk|I}} = \mathrm{Var}[\boldsymbol{y}_{\mathrm{Pr}}]\mathrm{Corr}[\boldsymbol{y}_{\mathrm{Tr}}[k], \boldsymbol{y}_{\mathrm{Pr}}]^2 \tag{2.8}$$

For each candidate k, we need to keep track of its conditional variance and conditional covariance with the prediction point after conditioning on the points already selected to compute (2.8). We then simply choose the candidate with the largest decrease in predictive variance. To keep track of the conditional variance and covariance, we can simply start with the initial values given by Θ_{kk} and Θ_{nk} and update after selecting an index j. We compute $\boldsymbol{u}$ for j directly according to (2.2) and update k's conditional variance by subtracting u_k^2 and update its conditional covariance by subtracting $u_k u_{n+1}$.

In order to efficiently compute $\boldsymbol{u}$, we rely on two main strategies. The direct method is to keep track of $\Theta_{I,I}^{-1}$, or the precision of the selected entries, and update the precision every time a new index is added to I. This can be done efficiently in $\mathcal{O}(s^2)$, see Appendix A.1. Once $\Theta_{I,I}^{-1}$ has been computed, $\boldsymbol{u}$ is computed trivially according to (2.2). For each of the s rounds of selection, it takes s^2 to update the precision, and costs Ns to compute $\boldsymbol{u}$, costing $\mathcal{O}(Ns^2 + s^3) = \mathcal{O}(Ns^2)$ overall.

The second approach is to take advantage of the quotient rule of Schur complementation. Stated statistically, the quotient rule states that conditioning on I and then conditioning on J is the same as conditioning on $I \cup J$. We then remind ourselves that Cholesky factorization can be viewed as iterative conditioning:

Re-writing the joint covariance matrix,

$$\begin{pmatrix}\Theta_{1,1} & \Theta_{1,2} \\ \Theta_{2,1} & \Theta_{2,2}\end{pmatrix} = \begin{pmatrix} I & 0 \\ \Theta_{2,1}\Theta_{1,1}^{-1} & I\end{pmatrix}\begin{pmatrix}\Theta_{1,1} & 0 \\ 0 & \Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}\end{pmatrix}\begin{pmatrix} I & \Theta_{1,1}^{-1}\Theta_{1,2} \\ 0 & I\end{pmatrix} \tag{2.9}$$

so we see that the Cholesky factorization of the joint covariance Θ is

$$\begin{aligned}\mathrm{chol}(\Theta) &= \begin{pmatrix} I & 0 \\ \Theta_{2,1}\Theta_{1,1}^{-1} & I\end{pmatrix}\begin{pmatrix}\mathrm{chol}(\Theta_{1,1}) & 0 \\ 0 & \mathrm{chol}(\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2})\end{pmatrix} \\ &= \begin{pmatrix}\mathrm{chol}(\Theta_{1,1}) & 0 \\ \Theta_{2,1}\,\mathrm{chol}(\Theta_{1,1})^{-\top} & \mathrm{chol}(\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2})\end{pmatrix}\end{aligned} \tag{2.10}$$

Here $\Theta_{2,1}\Theta_{1,1}^{-1}$ corresponds to the conditional expectation in (2.1) and $\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}$ corresponds to the conditional covariance in (2.2). Thus, we see that Cholesky factorization is iteratively conditioning the Gaussian process. From the iterative conditioning perspective, the ith column of the Cholesky factor corresponds precisely to the corresponding $\boldsymbol{u}$ for i since a iterative sequence of conditioning on $i_1, i_2 \ldots$ is equivalent to conditioning on I by the quotient rule.

The Cholesky factorization can be efficiently computed without excess dependence on N with left-looking, so the conditioning only happens when we need it. For each of the s rounds of selection, it costs $\mathcal{O}(Ns)$ to compute the next column of the Cholesky factorization, for a total cost of $\mathcal{O}(Ns^2)$, matching the time complexity of the explicit precision approach.

Algorithm 2.5 Point selection by explicit precision

Input: $\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Pr}}, K(\cdot,\cdot), s$
Output: I

$n \leftarrow |\boldsymbol{x}_{\mathrm{Tr}}|$
$\boldsymbol{x} \leftarrow \begin{pmatrix} \boldsymbol{x}_{\mathrm{Tr}} \\ \boldsymbol{x}_{\mathrm{Pr}} \end{pmatrix}$
$I \leftarrow \emptyset$
$-I \leftarrow \{1, 2, \ldots, n\}$
$\Theta_{I,I}^{-1} \leftarrow \mathbb{R}^{0\times 0}$
$\Theta_{\mathrm{Tr},\mathrm{Pr}|I} \leftarrow K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Pr}})$
$\operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I}) \leftarrow \operatorname{diag}(K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Tr}}))$
while $|-I| > 0$ and $|I| < s$ **do**
  $k \leftarrow \max_{j\in -I} \frac{\Theta_{j,\mathrm{Pr}|I}}{\Theta_{jj|I}}$
  $I \leftarrow I \cup \{k\}$
  $-I \leftarrow -I - \{k\}$
  $\boldsymbol{v} \leftarrow \Theta_{I,I}^{-1} K(\boldsymbol{x}_{\mathrm{Tr}}[I - \{k\}], \boldsymbol{x}_{\mathrm{Tr}}[k])$
  $\Theta_{I,I}^{-1} \leftarrow \begin{pmatrix} \Theta_{I,I}^{-1} + \frac{\boldsymbol{v}\boldsymbol{v}^\top}{\Theta_{kk|I}} & \frac{-\boldsymbol{v}}{\Theta_{kk|I}} \\ \frac{-\boldsymbol{v}^\top}{\Theta_{kk|I}} & \frac{1}{\Theta_{kk|I}} \end{pmatrix}$
  $\Theta_{:,k|I} \leftarrow K(\boldsymbol{x}, \boldsymbol{x}_k) - K(\boldsymbol{x}, \boldsymbol{x}_{I-\{k\}})\boldsymbol{v}$
  $\boldsymbol{u} \leftarrow \frac{\Theta_{:,k|I}}{\sqrt{\Theta_{kk|I}}}$
  for $j \in -I$ **do**
    $\Theta_{jj|I} \leftarrow \Theta_{jj|I} - \boldsymbol{u}_j^2$
    $\Theta_{j,\mathrm{Pr}|I} \leftarrow \Theta_{j,\mathrm{Pr}|I} - \boldsymbol{u}_j \boldsymbol{u}_{n+1}$
  end for
end while
return I

Algorithm 2.6 Point selection by Cholesky factorization

Input: $\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Pr}}, K(\cdot,\cdot), s$
Output: I

$n \leftarrow |\boldsymbol{x}_{\mathrm{Tr}}|$
$\boldsymbol{x} \leftarrow \begin{pmatrix} \boldsymbol{x}_{\mathrm{Tr}} \\ \boldsymbol{x}_{\mathrm{Pr}} \end{pmatrix}$
$I \leftarrow \emptyset$
$-I \leftarrow \{1, 2, \ldots, n\}$
$L \leftarrow \mathbf{0}^{(n+1)\times s}$
$\Theta_{\mathrm{Tr},\mathrm{Pr}|I} \leftarrow K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Pr}})$
$\operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I}) \leftarrow \operatorname{diag}(K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Tr}}))$
while $|-I| > 0$ and $|I| < s$ **do**
  $k \leftarrow \max_{j\in -I} \frac{\Theta_{j,\mathrm{Pr}|I}}{\Theta_{jj|I}}$
  $I \leftarrow I \cup \{k\}$
  $-I \leftarrow -I - \{k\}$
  $i \leftarrow |I|$
  $L_{:,i} \leftarrow K(\boldsymbol{x}, \boldsymbol{x}_k) - L_{:,:i-1} L_{k,:i-1}^\top$
  $L_{:,i} \leftarrow \frac{L_{:,i}}{\sqrt{L_{k,i}}}$
  for $j \in -I$ **do**
    $\Theta_{jj|I} \leftarrow \Theta_{jj|I} - L_{j,i}^2$
    $\Theta_{j,\mathrm{Pr}|I} \leftarrow \Theta_{j,\mathrm{Pr}|I} - L_{j,i} L_{n,i}$
  end for
end while
return I

While both approaches have the same time complexity, the explicit precision algorithm uses $\mathcal{O}(s^2)$ space to store the precision while the Cholesky factorization uses $\mathcal{O}(Ns)$ to store the first s columns of the Cholesky factorization of Θ, which is always more memory than the precision ($N > s$). Both algorithms use an additional $\mathcal{O}(N)$ space to store the conditional variances and covariances.

Once the indices have been computed according to Algorithm 2.5 or Algorithm 2.6, inferring the conditional mean and covariance of the unknown data can be done directly according to (2.1) and (2.2) in time $\mathcal{O}(s^3)$ using Algorithm 2.7.

This algorithm is in fact the covariance equivalent of the signal recovery algorithm orthogonal matching pursuit (OMP) [5], a connection elaborated in Appendix A.7.

Algorithm 2.7 Gaussian process inference by selection

Input: $\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{y}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Pr}}, K(\cdot, \cdot), s$

Output: $\mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}], \mathrm{Cov}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]$

1: Compute I using Algorithm 2.5 or Algorithm 2.6
2: $\Theta_{\mathrm{Tr},\mathrm{Tr}} \leftarrow K(\boldsymbol{x}_{\mathrm{Tr}}[I], \boldsymbol{x}_{\mathrm{Tr}}[I])$
3: $\Theta_{\mathrm{Pr},\mathrm{Pr}} \leftarrow K(\boldsymbol{x}_{\mathrm{Pr}}, \boldsymbol{x}_{\mathrm{Pr}})$
4: $\Theta_{\mathrm{Tr},\mathrm{Pr}} \leftarrow K(\boldsymbol{x}_{\mathrm{Tr}}[I], \boldsymbol{x}_{\mathrm{Pr}})$
5: $\mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \leftarrow \Theta_{\mathrm{Pr},\mathrm{Tr}} \Theta_{\mathrm{Tr},\mathrm{Tr}}^{-1} \boldsymbol{y}_{\mathrm{Tr}}[I]$
6: $\mathrm{Cov}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \leftarrow \Theta_{\mathrm{Pr},\mathrm{Pr}} - \Theta_{\mathrm{Tr},\mathrm{Pr}}^{\top} \Theta_{\mathrm{Tr},\mathrm{Tr}}^{-1} \Theta_{\mathrm{Tr},\mathrm{Pr}}$
7: **return** $\mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}], \mathrm{Cov}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]$

2.5. Supernodes and blocked selection. We now consider how to efficiently deal with multiple prediction points. The first question is how to generalize the previous objective for a single point (2.8) to multiple points. Following the same mutual information justification as before, a natural criterion is to minimize the log determinant of the prediction points' covariance matrix after conditioning on the selected points, or $\operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr} \mid I})$. This objective, known as D-optimal [1], has many intuitive interpretations — for example, as the volume of the region of uncertainty or as the scaling factor in the density function for the Gaussian process.

We now need to be able to efficiently compute the effect of selecting an index k on the log determinant. From (2.7), we know that selecting an index is a rank-one update on the prediction covariance. Using the matrix determinant lemma,

(2.11)
$$\begin{aligned}\operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr} \mid I \cup \{k\}}\right) &= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr} \mid I} - \frac{\Theta_{\mathrm{Pr},k \mid I} \Theta_{\mathrm{Pr},k \mid I}^{\top}}{\Theta_{kk \mid I}}\right) \\ &= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr} \mid I}\right) + \log\left(1 - \frac{\Theta_{\mathrm{Pr},k \mid I}^{\top} \Theta_{\mathrm{Pr},\mathrm{Pr} \mid I}^{-1} \Theta_{\mathrm{Pr},k \mid I}}{\Theta_{kk \mid I}}\right)\end{aligned}$$

Focusing on the second term, we can turn the quadratic form into conditioning:

$$= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr} \mid I}\right) + \log\left(\frac{\Theta_{kk \mid I} - \Theta_{k,\mathrm{Pr} \mid I} \Theta_{\mathrm{Pr},\mathrm{Pr} \mid I}^{-1} \Theta_{\mathrm{Pr},k \mid I}}{\Theta_{kk \mid I}}\right)$$

By the quotient rule, we combine the conditioning:

$$= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr} \mid I}\right) + \log\left(\frac{\Theta_{kk \mid I,\mathrm{Pr}}}{\Theta_{kk \mid I}}\right) \tag{2.12}$$

The greedy objective (2.12) tells us that to minimize the log determinant, we can simply select the index k with the smallest ratio between the conditional variance after conditioning on the previously selected points as well as the prediction points, and the conditional variance after just conditioning on the selected points. Intuitively this tells us that we can place sensors *backwards*, where we imagine placing sensors at the *prediction points* instead of the candidates. We then measure the conditional variance at a candidate, and pick the candidate whose conditional variance decreases the most (relative from what it started out as). Intuitively, these candidates are likely to give information about the prediction points, because the prediction points give information about the candidate.

Re-writing the objective in this way also gives an efficient algorithm to compute the necessary quantities. We condition on the prediction points essentially the same as described in subsection 2.4, by simply maintaining two structures instead of one, one for the conditional variance after conditioning on the previously selected points, and the other for the conditional variance after also conditioning on the prediction points. By the quotient rule, the order of conditioning does not matter as long as the order is consistent. For the second structure, we therefore condition on the prediction points *first* before any points have been selected. We again have two strategies, one which explicitly maintains precisions and the other which relies on Cholesky factorization.

For the precision algorithm, using (2.2) directly, for m prediction points it costs $\mathcal{O}(m^3)$ to compute $\Theta_{\mathrm{Pr},\mathrm{Pr}}^{-1}$ and then $\mathcal{O}(Nm^2)$ to compute $\Theta_{kk|\mathrm{Pr}}$ for the N candidates k. For each of the s rounds of selecting candidates, it costs s^2 and m^2 to update the precisions $\Theta_{I,I}^{-1}$ and $\Theta_{\mathrm{Pr},\mathrm{Pr}}^{-1}$ respectively, where the details of efficiently updating $\Theta_{\mathrm{Pr},\mathrm{Pr}}^{-1}$ after the rank-one update in (2.11) are given in Appendix A.3. Given the precisions, $\boldsymbol{u} = \frac{\Theta_{:,k|I}}{\sqrt{\Theta_{kk|I}}}$ and $\boldsymbol{u}_{\mathrm{Pr}} = \frac{\Theta_{:,k|I,\mathrm{Pr}}}{\sqrt{\Theta_{kk|I,\mathrm{Pr}}}}$ are computed as usual according to (2.2) in time Ns and Nm. Finally, for each candidate j, the conditional variance $\Theta_{jj|I}$ is updated by subtracting u_j^2, the conditional covariance $\Theta_{\mathrm{Pr},k|I}$ is updated for each prediction point index c each by subtracting $u_j u_c$, and the conditional variance $\Theta_{jj|I,\mathrm{Pr}}$ is updated by subtracting ${u_{\mathrm{Pr}}}_j^2$. The total time complexity after simplification is $\mathcal{O}(Ns^2 + Nm^2 + m^3)$.

For the Cholesky algorithm, two Cholesky factorizations are stored. We first compute the Cholesky factorization after selecting each prediction point, for a cost of $(n+m)m$ for each of the m columns. We then begin selecting candidates, which requires updating both Cholesky factors in time $(n+m)(m+s)$ which is dominated by updating the preconditioned Cholesky factor. The columns of the Cholesky factors correspond precisely to $\boldsymbol{u}$ and $\boldsymbol{u}_{\mathrm{Pr}}$ and both conditional variances $\Theta_{jj|I}$ and $\Theta_{jj|I,\mathrm{Pr}}$ can be computed as above. The conditional covariances do not need to be computed. Over s rounds the total time complexity is $\mathcal{O}((N+m)m^2 + s(N+m)(m+s))$ which simplifies to $\mathcal{O}(Ns^2 + Nm^2 + m^3)$.

Although both approaches have the same time complexity, like the single point case they differ in memory usage. The explicit precision requires $\mathcal{O}(s^2 + m^2)$ memory to store both precisions, as well as $\mathcal{O}(Nm)$ memory to store the conditional covariances. The Cholesky algorithm, on the othe rhand, requires $\mathcal{O}((n+m)(m+s))$ to store the first $m+s$ columns of the Cholesky factorization of the joint covariance matrix between training and prediction points, which simplifies to $\mathcal{O}(Ns + Nm + m^2)$. Comparing the memory usage, they are the same except for s^2 versus Ns, so the Cholesky algorithm again uses more memory than the explicit precision algorithm.

Algorithm 2.8 Multiple prediction point selection by explicit precision

Input: $\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Pr}}, K(\cdot,\cdot), s$
Output: I

$n \leftarrow |\boldsymbol{x}_{\mathrm{Tr}}|$
$m \leftarrow |\boldsymbol{x}_{\mathrm{Pr}}|$
$\boldsymbol{x} \leftarrow \begin{pmatrix} \boldsymbol{x}_{\mathrm{Tr}} \\ \boldsymbol{x}_{\mathrm{Pr}} \end{pmatrix}$
$I \leftarrow \emptyset$
$-I \leftarrow \{1, 2, \ldots, n\}$
$\Theta_{I,I}^{-1} \leftarrow \mathbb{R}^{0 \times 0}$
$\Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1} \leftarrow K(\boldsymbol{x}_{\mathrm{Pr}}, \boldsymbol{x}_{\mathrm{Pr}})$
$\Theta_{\mathrm{Tr},\mathrm{Pr}|I} \leftarrow K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Pr}})$
$\operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I}) \leftarrow \operatorname{diag}(K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Tr}}))$
$\operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}}) \leftarrow \operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I}) - \operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Pr}|I}\Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1}\Theta_{\mathrm{Pr},\mathrm{Tr}|I})$
while $|-I| > 0$ and $|I| < s$ **do**
  $k \leftarrow \min_{j \in -I} \frac{\Theta_{jj|I,\mathrm{Pr}}}{\Theta_{jj|I}}$
  $I \leftarrow I \cup \{k\}$
  $-I \leftarrow -I - \{k\}$
  $\boldsymbol{v} \leftarrow \Theta_{I,I}^{-1} K(\boldsymbol{x}_{\mathrm{Tr}}[I - \{k\}], \boldsymbol{x}_{\mathrm{Tr}}[k])$
  $\Theta_{I,I}^{-1} \leftarrow \begin{pmatrix} \Theta_{I,I}^{-1} + \frac{\boldsymbol{v}\boldsymbol{v}^{\top}}{\Theta_{kk|I}} & \frac{-\boldsymbol{v}}{\Theta_{kk|I}} \\ \frac{-\boldsymbol{v}^{\top}}{\Theta_{kk|I}} & \frac{1}{\Theta_{kk|I}} \end{pmatrix}$
  $\boldsymbol{w} \leftarrow \Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1}\Theta_{k,\mathrm{Pr}|I}^{\top}$
  $\Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1} \leftarrow \Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1} + \frac{\boldsymbol{w}\boldsymbol{w}^{\top}}{\Theta_{kk|I,\mathrm{Pr}}}$
  $\Theta_{:,k|I} \leftarrow K(\boldsymbol{x}, \boldsymbol{x}_k) - K(\boldsymbol{x}, \boldsymbol{x}_{I-\{k\}})\boldsymbol{v}$
  $\Theta_{:,k|I,\mathrm{Pr}} \leftarrow \Theta_{:,k|I} - \Theta_{:,\mathrm{Pr}|I}\boldsymbol{w}$
  $\boldsymbol{u} \leftarrow \frac{\Theta_{:,k|I}}{\sqrt{\Theta_{kk|I}}}$
  $\boldsymbol{u}_{\mathrm{Pr}} \leftarrow \frac{\Theta_{:,k|I,\mathrm{Pr}}}{\sqrt{\Theta_{kk|I,\mathrm{Pr}}}}$
  for $j \in -I$ **do**
    $\Theta_{jj|I} \leftarrow \Theta_{jj|I} - \boldsymbol{u}_j^2$
    $\Theta_{jj|I,\mathrm{Pr}} \leftarrow \Theta_{jj|I,\mathrm{Pr}} - (\boldsymbol{u}_{\mathrm{Pr}})_j^2$
    for $c \in \{1, 2, \ldots, m\}$ **do**
      $\Theta_{j,\mathrm{Pr}[c]|I} \leftarrow \Theta_{j,\mathrm{Pr}[c]|I} - \boldsymbol{u}_j \boldsymbol{u}_{n+c}$
    end for
  end for
end while
return I

Algorithm 2.9 Multiple prediction point selection by Cholesky factorization

Input: $\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Pr}}, K(\cdot,\cdot), s$
Output: I

$n \leftarrow |\boldsymbol{x}_{\mathrm{Tr}}|$
$m \leftarrow |\boldsymbol{x}_{\mathrm{Pr}}|$
$\boldsymbol{x} \leftarrow \begin{pmatrix} \boldsymbol{x}_{\mathrm{Tr}} \\ \boldsymbol{x}_{\mathrm{Pr}} \end{pmatrix}$
$I \leftarrow \emptyset$
$-I \leftarrow \{1, 2, \ldots, n\}$
$L \leftarrow \mathbf{0}^{(n+m) \times s}$
$L_{\mathrm{Pr}} \leftarrow \mathbf{0}^{(n+m) \times (s+m)}$
$\operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I}) \leftarrow \operatorname{diag}(K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Tr}}))$
$\operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}}) \leftarrow \operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I})$
for $i \in \{1, 2, \ldots, m\}$ **do**
  Update L_{Pr} and $\operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}})$ with $k = n + i$ by Algorithm 2.10.
end for
while $|-I| > 0$ and $|I| < s$ **do**
  $k \leftarrow \max_{j \in -I} \frac{\Theta_{j,\mathrm{Pr}|I}}{\Theta_{jj|I}}$
  $I \leftarrow I \cup \{k\}$
  $-I \leftarrow -I - \{k\}$
  $i \leftarrow |I|$
  Update L and $\operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I})$ by Algorithm 2.10.
  Update L_{Pr} and $\operatorname{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}})$ with $i = i + m$ by Algorithm 2.10.
end while
return I

Algorithm 2.10 Update Cholesky factor

Input: $\boldsymbol{x}, K(\cdot,\cdot), i, k, L, \operatorname{diag}(\Theta)$
Output: $L_{:,i}, \operatorname{diag}(\Theta_{|k})$

$n \leftarrow |\operatorname{diag}(\Theta)|$
$L_{:,i} \leftarrow K(\boldsymbol{x}, \boldsymbol{x}_k) - L_{:,:i-1}L_{k,:i-1}^{\top}$
$L_{:,i} \leftarrow \frac{L_{:,i}}{\sqrt{L_{k,i}}}$
for $j \in \{1, 2, \ldots, n\}$ **do**
  $\Theta_{jj} \leftarrow \Theta_{jj} - L_{j,i}^2$
end for

2.6. Near optimality by submodularity.

3. Greedy selection for *global* approximation by KL-minimization. We have a covariance matrix Θ and wish to compute the Cholesky factorization of Θ into a lower triangular factor L such that $\Theta = LL^{\top}$. . This can be done in $\mathcal{O}(N^3)$

Stephen: justify importance/downstream applications of Cholesky factorization with standard algorithms, which is often prohibitive. Recall the problem of inference in Gaussian process regression as described in subsection 2.2 also took $\mathcal{O}(N^3)$ to invert the covariance matrix Θ. Thus, similar to Guassian process regression, we will use *sparsity* to mitigate the computational cost. In fact, we will be able to re-use our previous algorithms Algorithms 2.5 and 2.8 on each column of the Cholesky factorization.

We will first compute the Cholesky factorization of Θ^{-1}, also known as the *precision matrix*, and use the resulting sparse factorization to efficiently compute an approximation for Θ. Because the precision matrix encodes the distribution of the full conditionals, the (i, j)th entry of the precision matrix is 0 if and only if the variables x_i and x_j are conditionally independent, conditional on the rest of the variables. Thus, the precision matrix Θ^{-1} can be sparse as a result of conditional independence even if the original covariance matrix Θ is dense. It therefore makes sense to attempt to approximately "sparsify" Θ^{-1} instead of Θ with iterated conditioning.

Because of sparsity, we can only get an approximate Cholesky factor L, $\hat{L}$ belonging to a pre-specified sparsity pattern S — a set of (row, column) indices that are allowed to be nonzero. In order to measure the performance of the estimator, we treat the matrices as covariance matrices of centered Gaussian processes (mean $\mathbf{0}$). In order to compare the resulting distributions, we use the *KL-divergence* according to [3], or the expected difference in log-densities:

$$L \coloneqq \operatorname*{argmin}_{\hat{L} \in S} \mathbb{D}_{\mathrm{KL}} \left(\mathcal{N}(\mathbf{0}, \Theta) \,\middle\|\, \mathcal{N}(\mathbf{0}, (\hat{L}\hat{L}^\top)^{-1}) \right) \tag{3.1}$$

Note that here we are computing the Cholesky factorization of Θ^{-1}. Surprisingly enough, it is possible to exactly compute L. First, we re-write the KL-divergence:

$$2\mathbb{D}_{\mathrm{KL}} \left(\mathcal{N}(\mathbf{0}, \Theta_1) \,\middle\|\, \mathcal{N}(\mathbf{0}, \Theta_2) \right) = \operatorname{trace}(\Theta_2^{-1}\Theta_1) + \operatorname{logdet}(\Theta_2) - \operatorname{logdet}(\Theta_1) - N \tag{3.2}$$

where Θ_1 and Θ_2 are both of size $N \times N$. See Appendix B.1 for details.

THEOREM 3.1. *[3]. The non-zero entries of the ith column of L in (3.1) are:*

$$L_{s_i,i} = \frac{\Theta^{-1}_{s_i,s_i} \boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top \Theta^{-1}_{s_i,s_i} \boldsymbol{e}_1}} \tag{3.3}$$

Plugging the optimal L (3.3) back into the KL-divergence (3.2), we obtain:

$$2\mathbb{D}_{\mathrm{KL}} \left(\mathcal{N}(\mathbf{0}, \Theta) \,\middle\|\, \mathcal{N}(\mathbf{0}, (LL^\top)^{-1}) \right) = \sum_{i=1}^{N} \left[\log \left((\boldsymbol{e}_1^\top \Theta^{-1}_{s_i,s_i} \boldsymbol{e}_1)^{-1} \right) \right] - \operatorname{logdet}(\Theta) \tag{3.4}$$

See Appendix B.2 for details. In particular, it is important which direction the KL-divergence is or else cancellation of the $\operatorname{trace}(\Theta_2^{-1}\Theta_1)$ term may not occur.

In order to maximize (3.4), we can ignore $\operatorname{logdet}(\Theta)$ since it does not depend on L and maximize over each column independently, since each term in the sum only depends on a single column. We want to minimize $(\boldsymbol{e}_1^\top \Theta^{-1}_{s_i,s_i} \boldsymbol{e}_1)^{-1}$, the term corresponding to the diagonal entry in the inverse of the submatrix of Θ corresponding to the entries we've taken. We can give this value statistical interpretation by using the fact that marginalization in covariance is conditioning in precision.

$$\Theta_{1,1|2} = ((\Theta^{-1})_{1,1})^{-1} \tag{3.5}$$

where Θ is blocked according to

$$\Theta = \begin{pmatrix} \Theta_{1,1} & \Theta_{1,2} \\ \Theta_{2,1} & \Theta_{2,2} \end{pmatrix} \tag{3.6}$$

Thus, we see that

$$\begin{aligned} (\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1} \boldsymbol{e}_1)^{-1} &= ((\Theta_{s_i,s_i}^{-1})_{11})^{-1} \\ &= \Theta_{ii|s_i - \{i\}} \end{aligned} \tag{3.7}$$

So our objective on each column is to minimize the conditional variance of the ith variable, conditional on the entries we've selected — s_i contains i to begin with, so $s_i - \{i\}$ is the selected entries. We can therefore use algorithm Algorithm 2.5 directly on each column, where the prediction point is the i variable and the number of points selected is the number of nonzero entries per column. The only difference is that the candidates is limited to indices lower than i, that is, candidate indices k such that $k > i$ to maintain the lower triangularity of L. Once s_i has been computed for each i, L can be constructed according to Theorem 3.1. Each column costs $\mathcal{O}(s^3)$ to compute Θ_{s_i,s_i}^{-1} for a total cost of $\mathcal{O}(Ns^3)$ for the N columns of L.

3.1. Aggregated sparsity pattern. We can also use the Gaussian process regression viewpoint to efficiently aggregate multiple columns, that is, to use the same sparsity pattern for multiple columns. We denote aggregating the column indices $i_1, \dots, i_m$ into the same group as $\tilde{i} = \{i_1, i_2, \dots i_m\}$, letting $s_{\tilde{i}} = \bigcup_{i \in \tilde{i}} s_i$ be the aggregated sparsity pattern, and letting $\tilde{s} = s_{\tilde{i}} - \tilde{i}$ be the set of selected entries excluding the diagonal entries. Each $s_i = \tilde{s} \cup \{j \in \tilde{i} \mid j \geq i\}$, that is, the sparsity pattern of the i column is the selected entries plus all the diagonal entries lower than it. We will enforce that all the selected entries, excluding the indices of the diagonals of the columns themselves, are below the lowest index so that indices are not selected "partially" — that is, an index could be above some indices in the aggregated columns, and therefore invalid to add to their column, but below others. That is, we restrict the candidate indices $k > \max \tilde{i}$ so that the selected index can be added to each column in $\tilde{i}$ without violating the lower triangularity of L. It is in fact possible to properly account for these partial updates, but the reasoning and eventual algorithm becomes more complicated. We defer a detailed discussion of the partial update case to Appendix A.5.

We now show that the KL-minimization objective on the aggregated indices corresponds precisely to (2.11), the objective multiple point Gaussian regression with the chain rule of log determinant through conditioning.

$$\operatorname{logdet}(\Theta) = \operatorname{logdet}(\Theta_{1,1|2}) + \operatorname{logdet}(\Theta_{2,2}) \tag{3.8}$$

where Θ is blocked according to (3.6). The KL-divergence objective for $\tilde{i}$ is:

$$\begin{aligned} \sum_{i \in \tilde{i}} \log(\Theta_{ii|s_i - \{i\}}) &= \log(\Theta_{i_m i_m|\tilde{s}}) + \log(\Theta_{i_{m-1} i_{m-1}|\tilde{s} \cup \{i_m\}}) + \cdots \\ &= \operatorname{logdet}(\Theta_{\{i_m, i_{m-1}\}, \{i_m, i_{m-1}\}|\tilde{s}}) + \log(\Theta_{i_{m-2} i_{m-2}|\tilde{s} \cup \{i_m, i_{m-1}\}}) + \cdots \\ &= \operatorname{logdet}(\Theta_{\tilde{i}, \tilde{i}|\tilde{s}}) \end{aligned} \tag{3.9}$$

We see that the objective (3.9) is equivalent to the objective (2.11), that is, to minimize the log determinant of the conditional covariance matrix corresponding to a set of prediction points, conditional on the selected entries. We can therefore directly use Algorithm 2.8 on the aggregated columns, where the prediction points correspond to indices in the aggregation and where we restrict the candidates k to those below each column in the aggregation, $k > \max \tilde{i}$.

Hence the sparse Cholesky factorization motivated by KL-divergence can be viewed as sparse Gaussian process selection over each column, where entries are selected to maximize mutual information with the entry on the diagonal of the current column. In the aggregated case, the multiple columns in the aggregated group correspond directly to predicting for multiple prediction points, where entries are again selected to maximize mutual information with each diagonal entry in the aggregation. This viewpoint leads directly to Algorithm 3.1.

Algorithm 3.1 Cholesky factorization by selection

Input: $\boldsymbol{x}, K(\cdot,\cdot), s, g = \{\tilde{i}_1, \ldots, \tilde{i}_{N/m}\}$
Output: L such that $(LL^\top)^{-1} \approx K(\boldsymbol{x}, \boldsymbol{x})$

- $n \leftarrow |\boldsymbol{x}|$
- **for** $\tilde{i} \in g$ **do**
  - $J \leftarrow \{\max(\tilde{i}) + 1, \max(\tilde{i}) + 2, \ldots, n\}$
  - Compute I using Algorithm 2.8 or Algorithm 2.9 where $\boldsymbol{x}_{\mathrm{Tr}} = \boldsymbol{x}[J], \boldsymbol{x}_{\mathrm{Pr}} = \boldsymbol{x}[\tilde{i}], s = s - |\tilde{i}|$
  - $\tilde{s} \leftarrow J[I]$
  - **for** $i \in \mathrm{reversed}(\mathrm{sorted}(\tilde{i}))$ **do**
    - $\tilde{s} \leftarrow \tilde{s} \cup \{i\}$
    - $s_i \leftarrow \mathrm{reversed}(\tilde{s})$
  - **end for**
- **end for**
- **return** L computed with Algorithm 3.3

Algorithm 3.2 Computing L without aggregation

Input: $\boldsymbol{x}, K(\cdot,\cdot), s_i$
Output: $L_{s_i,i}$

- $\Theta_{s_i,s_i}^{-1} \leftarrow K(\boldsymbol{x}[s_i], \boldsymbol{x}[s_i])^{-1}$
- $L_{s_i,i} \leftarrow \frac{\Theta_{s_i,s_i}^{-1} \boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1} \boldsymbol{e}_1}}$
- **return** $L_{s_i,i}$

Algorithm 3.3 Computing L with aggregation

Input: $\boldsymbol{x}, K(\cdot,\cdot), \tilde{s}, \tilde{i}$
Output: $L_{s_i,i}$ for all $i \in \tilde{i}$

- $s \leftarrow \tilde{i} \cup \tilde{s}$
- $U \leftarrow P^{\updownarrow} \operatorname{chol}(P^{\updownarrow} \Theta_{s,s} P^{\updownarrow}) P^{\updownarrow}$
- **for** $i \in \tilde{i}$ **do**
  - $k \leftarrow$ index of i in $\tilde{i}$
  - $L_{s_i,i} \leftarrow U^{-\top} \boldsymbol{e}_k$
- **end for**
- **return** L

Once the sparsity pattern has been determined, we need to compute each column of L according to Theorem 3.1. Because the sparsity pattern for each column in the same group are subsets of each other, we can efficiently compute all their columns at once. The observation is that the smallest index in the group (corresponding to the entry highest in the matrix) will have the largest sparsity pattern, the next index will have one less entry (lacking the entry above it, which would violate lower triangularity), and so on. We need to compute $\Theta_{s_i,s_i}^{-1} \boldsymbol{e}_1$ for each $i \in \tilde{i}$, or the precision of the marginalized covariance corresponding to the selected entries. By (3.5), we can turn marginalization in covariance into conditioning in precision:

$$
\begin{aligned}
L_{s_i,i} &= \frac{\Theta_{s_i,s_i}^{-1} \boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1} \boldsymbol{e}_1}} \\
&= \frac{(\Theta_{s,s})_{k:,k:|:k-1}^{-1} \boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top (\Theta_{s,s})_{k:,k:|:k-1}^{-1} \boldsymbol{e}_1}} \qquad (3.10)
\end{aligned}
$$

where $s = \tilde{i} \cup \tilde{s}$ and k is i's index in $\tilde{i}$. So we want the kth column of the precision of the marginalized covariance, conditional on all the entries before it. From (2.10), this can be directly read off the Cholesky factorization. Thus, we can simply compute:

$$L = \operatorname{chol}\left(\Theta_{s,s}^{-1}\right) \qquad (3.11)$$

and read off the kth column to compute (3.10) for each $i \in \tilde{i}$. However, instead of computing a lower triangular factor for the precision, we can compute an *upper* triangular factor the covariance whose inverse transpose will be a *lower* triangular factor for the original matrix. In particular, we see that

$$U = P^{\updownarrow} \operatorname{chol}(P^{\updownarrow} \Theta_{s,s} P^{\updownarrow}) P^{\updownarrow} \qquad (3.12)$$

satisfies $UU^\top = \Theta_{s,s}$ where $P^{\updownarrow}$ is the order-reversing permutation. Thus,

$$\Theta_{s,s}^{-1} = U^{-\top} U^{-1}$$

where $U^{-\top}$ is an *lower* triangular factor for $\Theta_{s,s}^{-1}$ equal to (3.3) because the Cholesky factorization is unique. Computing $U^{-\top}$ leads directly to Algorithm 3.3.

Recall that the complexity of selecting s out of N total training points for m prediction points using Algorithm 2.8 or Algorithm 2.9 was $\mathcal{O}(Ns^2 + Nm^2 + m^3)$. In the context of Cholesky factorization, N is the size of the matrix, m is the number of columns to aggregate, and s is the number of nonzero entries in each column of L. We therefore need to do $\frac{N}{m}$ selections, one for each aggregated group, where we only need to select $s - m$ entries (since the m prediction points are automatically added). We then need to actually construct each column of L after determining the sparsity pattern, with Algorithm 3.3. This costs $\mathcal{O}(s^3)$ for each aggregated group to compute the Cholesky factor of the submatrix, which dominates the time to compute each column of L for the m columns in the group, $\mathcal{O}(ms^2)$ $(N > s > m)$. Thus, the overall complexity is $\mathcal{O}(\frac{N}{m}(N(s-m)^2 + Nm^2 + m^3 + s^3))$, which simplifies to $\mathcal{O}(\frac{N^2 s^2}{m})$ by making use of the bound that $(s-m)^2 = \mathcal{O}(s^2 + m^2)$.

Note that the non-aggregated factorization is equivalent to $m = 1$, which yields $\mathcal{O}(N^2 s^2)$ (using the non-aggregated algorithms Algorithms 2.5 and 3.2, but one can also use the aggregated versions Algorithms 2.8 and 3.3 with $m = 1$ and achieve equivalent complexity). Thus, we see that the aggregated version is m times faster than its non-aggregated counterpart, at the cost that the resulting sparsity pattern will be lower quality (since the algorithm is forced to select the same entry for *all* columns in the group).

Unlike the geometric algorithms of [3, 4] which rely on the pairwise distance between points, and whose covariance matrix is implicitly determined by a list of points and kernel function, this algorithm relies only on the entries of the covariance matrix Θ. Thus, it can factor arbitrary symmetric positive definite matrices without access to points or an explicit kernel function.

3.2. Review of KL approximation.

4. Numerical experiments. All experiments were run on the PACE Phoenix cluster, with 8 cores of a Intel(R) Xeon(R) Gold 6226 CPU @ 2.70GHz and 6 GB of RAM per core. Python code for all numerical experiments can be found at https://github.com/stephen-huan/conditional-knn.

Stephen: cite python libraries that want citations (e.g. scikit-learn)

4.1. *k*th-nearest neighbors selection. We justify that diverse point selection based on conditional information can lead to better performance than simply selecting the nearest neighbor in a toy example on the MNIST dataset. We compare kth-nearest neighbors (KNN) directly to conditional kth-nearest neighbors (CKNN) in the following experiment. We randomly select 1000 images to form the training set and 100 to form the testing set. For each image in the testing set, we select the k "closest" training points with either KNN or CKNN. For KNN we use the standard Euclidean distance and for CKNN we use Matérn kernel with smoothness $\nu = 1.5$ and length scale $l = 2^{10}$. Finally, we predict the label of the test point by taking the most frequently occurring label in the k selected points.

Stephen: cite mnist dataset

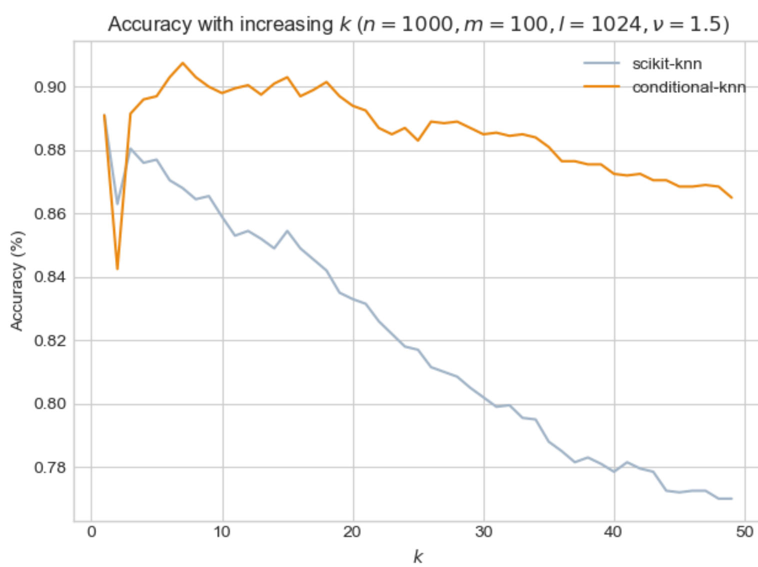

FIG. 1. *Accuracy of classification with increasing k.*

Stephen: replace images with pgfplots/tikz

As k increases, KNN degrades near-linearly in accuracy. We hypothesize that nearby images are more likely to have the same label as a given test image. By forcing the algorithm to select more points, it increases the likelihood that the algorithm becomes confused by differently labeled images. However, CKNN is more accurate than KNN for nearly every k, suggesting that conditional selection is able to take advantage of selecting more points. We emphasize that the difference in accuracy is solely a result of conditional selection — because the Matérn kernel degrades monotonically with distance, sorting by covariance is identical to sorting by distance. In addition, we use the mode to aggregate the labels of the selected points, rather than performing Gaussian process classification. The difference in accuracy can therefore be attributed to precisely the difference in which points were selected.

4.2. Gaussian process regression.

4.3. Recovery of sparse Cholesky factors.

4.4. Cholesky factorization.

4.5. Preconditioning for conjugate gradient.

Acknowledgments. This research was supported in part through research cyberinfrastructure resources and services provided by the Partnership for an Advanced Computing Environment (PACE) at the Georgia Institute of Technology, Atlanta, Georgia, USA.

add more funding information

REFERENCES

[1] A. Krause, A. Singh, and C. Guestrin, *Near-Optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies*, The Journal of Machine Learning Research, 9 (2008), pp. 235–284.

[2] O. Krause and C. Igel, *A More Efficient Rank-one Covariance Matrix Update for Evolution Strategies*, in Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII, FOGA '15, New York, NY, USA, Jan. 2015, Association for Computing Machinery, pp. 129–136, https://doi.org/10.1145/2725494.2725496.

[3] F. Schäfer, M. Katzfuss, and H. Owhadi, *Sparse Cholesky factorization by Kullback-Leibler minimization*, arXiv:2004.14455 [cs, math, stat], (2021), https://arxiv.org/abs/2004.14455.

[4] F. Schäfer, T. J. Sullivan, and H. Owhadi, *Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity*, arXiv:1706.02205 [cs, math], (2020), https://arxiv.org/abs/1706.02205.

[5] J. A. Tropp and A. C. Gilbert, *Signal Recovery From Random Measurements Via Orthogonal Matching Pursuit*, IEEE Transactions on Information Theory, 53 (2007), pp. 4655–4666, https://doi.org/10.1109/TIT.2007.909108.

add proofs, if any, in appendix

Appendix A. Computation in sparse Gaussian process selection.

A.1. Updating precision after insertion. We have the matrix $\Theta_{I,I}^{-1}$ corresponding to the precision of the selected entries, and wish to take into account the addition of a new entry k into I. That is, we wish to compute $\Theta_{I',I'}^{-1}$ for $I' = I \cup \{k\}$, which in effect adds a new row and column to $\Theta_{I,I}^{-1}$. In order to invert the new matrix efficiently, we can block the matrix to separate the new and old information.

$$\begin{pmatrix} \Theta_{1,1} & \Theta_{1,2} \\ \Theta_{2,1} & \Theta_{2,2} \end{pmatrix} = \begin{pmatrix} I & 0 \\ \Theta_{2,1}\Theta_{1,1}^{-1} & I \end{pmatrix} \begin{pmatrix} \Theta_{1,1} & 0 \\ 0 & \Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2} \end{pmatrix} \begin{pmatrix} I & \Theta_{1,1}^{-1}\Theta_{1,2} \\ 0 & I \end{pmatrix} \tag{A.1}$$

For notational convenience, we denote the Schur complement $\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}$ as $\Theta_{2,2|1}$. Inverting both sides of the equation,

$$\Theta^{-1} = \begin{pmatrix} I & -\Theta_{1,1}^{-1}\Theta_{1,2} \\ 0 & I \end{pmatrix} \begin{pmatrix} \Theta_{1,1}^{-1} & 0 \\ 0 & \Theta_{2,2|1}^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -\Theta_{2,1}\Theta_{1,1}^{-1} & I \end{pmatrix} \tag{A.2}$$

$$= \begin{pmatrix} \Theta_{1,1}^{-1} + (\Theta_{1,1}^{-1}\Theta_{1,2})\Theta_{2,2|1}^{-1}(\Theta_{2,1}\Theta_{1,1}^{-1}) & -(\Theta_{1,1}^{-1}\Theta_{1,2})\Theta_{2,2|1}^{-1} \\ -\Theta_{2,2|1}^{-1}(\Theta_{2,1}\Theta_{1,1}^{-1}) & \Theta_{2,2|1}^{-1} \end{pmatrix} \tag{A.3}$$

In the context of adding a new entry to the matrix, $\Theta_{1,1} = \Theta_{I,I}$, $\Theta_{1,2} = \Theta_{I,k}$, and $\Theta_{2,2} = \Theta_{kk}$. Also note that $\Theta_{kk|I}^{-1}$ is the inverse of the variance of k conditional on the entries in I, which has already been computed in Algorithm 2.5. If we let $\boldsymbol{v} = \Theta_{I,I}^{-1}\Theta_{I,k}$, then we can write the update as:

$$= \begin{pmatrix} \Theta_{I,I}^{-1} + \Theta_{kk|I}^{-1}\boldsymbol{v}\boldsymbol{v}^T & -\Theta_{kk|I}^{-1}\boldsymbol{v} \\ -\Theta_{kk|I}^{-1}\boldsymbol{v}^\top & \Theta_{kk|I}^{-1} \end{pmatrix} \tag{A.4}$$

which is precisely the update in line 13 of Algorithm 2.5. Note that the update is a rank-one update to $\Theta_{1,1}^{-1}$, which can be computed in $\mathcal{O}(|I|^2) = \mathcal{O}(s^2)$.

A.2. Updating precision after marginalization. Suppose we have the precision Θ^{-1} and wish to compute the precision of the marginalized covariance after ignoring an index k. That is, we wish to compute the inverse of a matrix after deleting a row and column, given the inverse of the original matrix. We could use the result in Appendix A.1 by "reading" the update backwards. That is, we could identify $\Theta_{2,2|1}^{-1}$ from $(\Theta^{-1})_{kk}$ and $\boldsymbol{v} = \Theta_{1,1}^{-1}\Theta_{1,2}$ from $-\frac{(\Theta^{-1})_{-k,k}}{\Theta_{2,2|1}^{-1}}$ where $-k$ denotes all rows excluding the kth row. We can then revert the rank-one update by subtracting out the update, computing $\Theta_{-k,-k}^{-1} = (\Theta^{-1})_{-k,-k} - \Theta_{kk|I}^{-1}\boldsymbol{v}\boldsymbol{v}^\top$. However, a more intuitive derivation relies on the fact that marginalization in covariance is conditioning in precision. Using (3.5), we see that $\Theta_{-k,-k}^{-1} = (\Theta^{-1})_{-k,-k|k}$, or the precision conditional on the deleted entry. By (2.2), we immediately obtain the equivalent update

$$(\Theta^{-1})_{-k,-k|k} = \Theta_{-k,-k}^{-1} - \frac{(\Theta^{-1})_{-k,k}(\Theta^{-1})_{-k,k}^\top}{(\Theta^{-1})_{kk}} \tag{A.5}$$

Since this is a rank-one update to the precision Θ^{-1}, this can be computed in $\mathcal{O}(\#\text{ rows}(\Theta^{-1}))^2$.

Stephen: this is not used in the paper but is nice to know + used in the sensor placement

A.3. Updating precision after conditioning. We have the matrix $\Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I}$, or the precision of the prediction points, conditional on the selected entries. We want to take into account selecting an entry k, or to compute $\Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I\cup\{k\}}$ which is a rank-one update to the original matrix from (2.11). We can directly apply the Sherman–Morrison–Woodbury formula which states that:

$$\Theta^{-1}_{1,1|2} = \Theta^{-1}_{1,1} + \left(\Theta^{-1}_{1,1}\Theta_{1,2}\right)\Theta^{-1}_{2,2|1}\left(\Theta_{2,1}\Theta^{-1}_{1,1}\right) \tag{A.6}$$

Expanding the conditioning by definition,

$$\left(\Theta_{1,1} - \Theta_{1,2}\Theta^{-1}_{2,2}\Theta_{2,1}\right)^{-1} = \Theta^{-1}_{1,1} + \left(\Theta^{-1}_{1,1}\Theta_{1,2}\right)\Theta^{-1}_{2,2|1}\left(\Theta_{2,1}\Theta^{-1}_{1,1}\right) \tag{A.7}$$

Letting $\boldsymbol{u} = \Theta_{1,2}$ and $\boldsymbol{v} = \Theta^{-1}_{1,1}\Theta_{1,2} = \Theta^{-1}_{1,1}\boldsymbol{u}$,

$$(\Theta_{1,1} - \Theta^{-1}_{2,2}\boldsymbol{u}\boldsymbol{u}^\top)^{-1} = \Theta^{-1}_{1,1} + \Theta^{-1}_{2,2|1}\boldsymbol{v}\boldsymbol{v}^\top \tag{A.8}$$

So we see that a rank-one update to $\Theta_{1,1}$ then inverting is a rank-one update to $\Theta^{-1}_{1,1}$. In our context, $\Theta_{1,1} = \Theta_{\mathrm{Pr},\mathrm{Pr}|I}, \boldsymbol{u} = \Theta_{\mathrm{Pr},k|I}, \Theta_{2,2} = \Theta_{kk|I}$ so $\Theta^{-1}_{2,2|1} = \Theta^{-1}_{kk|\mathrm{Pr},I}$ (this can be rigorously shown by expanding the Schur complement and taking advantage of the quotient rule as in (2.12)). $\boldsymbol{v}$ can be computed according to definition as $\Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I}\boldsymbol{u}$. Thus, we can write the update as

$$\left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I} - \frac{\Theta_{\mathrm{Pr},k|I}\Theta^\top_{\mathrm{Pr},k|I}}{\Theta_{kk|I}}\right)^{-1} = \Theta^{-1}_{1,1} + \Theta^{-1}_{kk|\mathrm{Pr},I}\boldsymbol{v}\boldsymbol{v}^\top \tag{A.9}$$

which is the update in line 18 of Algorithm 2.8. Since the update is a rank-one update, it can be computed in $\mathcal{O}(|\mathrm{Pr}|^2) = \mathcal{O}(m^2)$.

A.4. Updating Cholesky Factorization after Rank-one Downdate.

Stephen: remove because unnecessary, describe insertion insertead

We use the approach from Lemma 1 of [2], slightly adapted to use in-place operations and to make no assumption on the particular row ordering of the Cholesky factor. Let L be a Cholesky factorization of Θ, that is, $L = \mathrm{chol}(\Theta)$. We wish to compute the updated Cholesky factor $L' = \mathrm{chol}(\Theta')$ where $\Theta' = \Theta - \boldsymbol{u}\boldsymbol{u}^\top$. To do so, assume L and L' are blocked according to the same block structure:

$$L = \begin{pmatrix} r_1 & \mathbf{0} \\ \boldsymbol{r}_2 & L_2 \end{pmatrix} \qquad L' = \begin{pmatrix} r'_1 & \mathbf{0} \\ \boldsymbol{r}'_2 & L'_2 \end{pmatrix} \tag{A.10}$$

Multiplying, we find

$$LL^\top = \Theta = \begin{pmatrix} r_1^2 & r_1\boldsymbol{r}_2^\top \\ r_1\boldsymbol{r}_2 & L_2L_2^\top + \boldsymbol{r}_2\boldsymbol{r}_2^\top \end{pmatrix} \tag{A.11}$$

$$L'L'^\top = \Theta' = \begin{pmatrix} r_1'^2 & r'_1\boldsymbol{r}_2'^\top \\ r'_1\boldsymbol{r}'_2 & L'_2L_2'^\top + \boldsymbol{r}'_2\boldsymbol{r}_2'^\top \end{pmatrix} \tag{A.12}$$

From here, we solve for r_1', $\boldsymbol{r}'$, and L_2'

$$r_1'^2 = \Theta_{11}' = \Theta_{11} - u_1^2 \tag{A.13}$$
$$= r_1^2 - u_1^2 \tag{A.14}$$
$$r_1' = \sqrt{r_1^2 - u_1^2} \tag{A.15}$$
$$r_1' \boldsymbol{r}_2' = \Theta_{2:,1}' = \Theta_{2:,1} - u_1 \boldsymbol{u}_2 \tag{A.16}$$
$$= r_1 \boldsymbol{r}_2 - u_1 \boldsymbol{u}_2 \tag{A.17}$$
$$\boldsymbol{r}_2' = \frac{1}{r_1'}(r_1 \boldsymbol{r}_2 - u_1 \boldsymbol{u}_2) \tag{A.18}$$
$$L_2' L_2'^\top + \boldsymbol{r}_2' \boldsymbol{r}_2'^\top = \Theta_{22}' = \Theta_{22} - \boldsymbol{u}_2 \boldsymbol{u}_2^\top \tag{A.19}$$
$$L_2' L_2'^\top = L_2 L_2^\top + \boldsymbol{r}_2 \boldsymbol{r}_2^\top - \boldsymbol{u}_2 \boldsymbol{u}_2^\top - \boldsymbol{r}_2' \boldsymbol{r}_2'^\top \tag{A.20}$$

After some simplification we find

$$L_2' L_2'^\top = L_2 L_2^\top - \left(\frac{u_1}{r_1'} \boldsymbol{r}_2 - \frac{r_1}{r_1'} \boldsymbol{u}_2\right)\left(\frac{u_1}{r_1'} \boldsymbol{r}_2 - \frac{r_1}{r_1'} \boldsymbol{u}_2\right)^\top \tag{A.21}$$

which is a rank-one downdate to the subfactor L_2. Recursively updating L_2 yields a $\mathcal{O}(N^2)$ algorithm. We now re-write the algorithm to be in-place to take advantage of BLAS routines. The updates can be summarized as:

$$r_1' = \sqrt{r_1^2 - u_1^2} \tag{A.22}$$
$$\boldsymbol{r}' = \frac{r_1}{r_1'} \boldsymbol{r} - \frac{u_1}{r_1'} \boldsymbol{u} \tag{A.23}$$
$$\boldsymbol{u}' = \frac{u_1}{r_1'} \boldsymbol{r} - \frac{r_1}{r_1'} \boldsymbol{u} \tag{A.24}$$

Note that we drop the subscripting on $\boldsymbol{r}$ and $\boldsymbol{u}$. By updating the entire vector on each iteration, we can avoid keeping track of the lower triangular structure of L. We will first update $\boldsymbol{r}'$ and then use it to update $\boldsymbol{u}$. Solving for $\boldsymbol{r}$ in terms of $\boldsymbol{r}'$,

$$\boldsymbol{r} = \frac{r_1'}{r_1} \boldsymbol{r}' + \frac{u_1}{r_1} \boldsymbol{u} \tag{A.25}$$
$$\boldsymbol{u}' = -\frac{u_1}{r_1} \boldsymbol{r}' + \frac{r_1'}{u_1} \boldsymbol{u} \tag{A.26}$$

Thus, the updates proceed sequentially as follows:

$$\gamma \leftarrow \sqrt{r_1^2 - u_1^2} \tag{A.27}$$
$$\alpha \leftarrow \frac{r_1}{\gamma} \tag{A.28}$$
$$\beta \leftarrow \frac{u_1}{\gamma} \tag{A.29}$$
$$\boldsymbol{r} \leftarrow \alpha \boldsymbol{r} - \beta \boldsymbol{u} \tag{A.30}$$
$$\boldsymbol{u} \leftarrow -\frac{\beta}{\alpha} \boldsymbol{r} + \frac{1}{\alpha} \boldsymbol{u} \tag{A.31}$$

These can be efficiently performed in-place by BLAS as level-one `daxpy` operations.

A.5. Partial Updates in the Selection Algorithm. In the context of the selection algorithm, we have M prediction points and wish to minimize the log determinant of the resulting covariance matrix of the prediction points, conditional on the points we've selected from the training data. In the specific context of Cholesky factorization, it is possible to add a training point and have it apply *partially* on the prediction points. If nonadjacent columns indices are aggregated, a entry selected between two indices can be higher than one column, but lower than another. Adding the entry to the sparsity pattern would therefore only add to some, but not all, columns in the aggregation. We will model this as partially conditioning the variables of interest. In particular, if we have prediction variables $y_1, y_2, \dots, y_M$, a partial condition ignoring the first j variables on the selected index k would result in $y_1, y_2, \dots, y_j, y_{j+1|k}, \dots, y_{M|k}$.

The first question is to compute the resulting covariance matrix. We know $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \Theta)$ and $\boldsymbol{y}_{|k}$ has conditional distribution according to (2.2), $\boldsymbol{y}_{|k} \sim \mathcal{N}(\mu, \Theta - \Theta_{:,k}\Theta_{k,k}^{-1}\Theta_{k,:})$. Taking the Cholesky factorization of both covariance matrices, let $L = \operatorname{chol}(\Theta)$ and $L_{|k} = \operatorname{chol}(\Theta_{|k})$. We can then view $\boldsymbol{y}$ as $L\boldsymbol{z}$, where $\boldsymbol{z}$ is distributed according to $\mathcal{N}(\mathbf{0}, I)$. Similarly, $\boldsymbol{y}_{|k} = L_{|k}\boldsymbol{z} + \boldsymbol{\mu}$. For unconditioned y_i and y_j, the covariance between them is defined to be Θ_{ij}. Similarly, for conditioned y_i and y_j, the covariance is $\Theta_{ij|k}$. The only question is what the covariance between unconditioned y_i and conditioned y_j is. By definition,

$$\operatorname{Cov}[y_i, y_j] = \operatorname{E}[(y_i - \operatorname{E}[y_i])(y_j - \operatorname{E}[y_j])] \tag{A.32}$$

$$= \operatorname{E}[(L_i\boldsymbol{z})(L_{i|k}\boldsymbol{z})] \tag{A.33}$$

$$= \operatorname{E}[(L_{1,i}z_1 + \cdots + L_{N,i}z_N)(L_{1,j|k}z_1 + \cdots + L_{N,j|k}z_N)] \tag{A.34}$$

For $i \neq j$, $E[z_i z_j] = \operatorname{E}[z_i]\operatorname{E}[z_j] = 0$ since z_i is independent of z_j and has mean 0.

$$= \operatorname{E}[L_{1,i}L_{1,j|k}z_1^2 + \cdots + L_{N,i}L_{N,j|k}z_N^2] \tag{A.35}$$

$$= L_{1,i}L_{1,j|k}\operatorname{E}[z_1^2] + \cdots + L_{N,i}L_{N,j|k}\operatorname{E}[z_N^2] \tag{A.36}$$

For any i, $\operatorname{E}[z_i^2] = \operatorname{Var}[z_i] + \operatorname{E}[z_i]^2 = 1 + 0 = 1$

$$= L_{1,i}L_{1,j|k} + \cdots + L_{N,i}L_{N,j|k} \tag{A.37}$$

$$= L_i \cdot L_{j|k} \tag{A.38}$$

Thus, the new covariance matrix can be written as:

$$\begin{pmatrix} L_{:j}L_{:j}^\top & L_{:j}L_{j:|k}^\top \\ L_{j:|k}L_{:j}^\top & L_{j:|k}L_{j:|k}^\top \end{pmatrix} = \begin{pmatrix} L_{:j} \\ L_{j:|k} \end{pmatrix} \begin{pmatrix} L_{:j} \\ L_{j:|k} \end{pmatrix}^\top \tag{A.39}$$

We will denote a partially conditioned matrix as

$$\Theta_{:,:,\shortmid k} \tag{A.40}$$

We can now connect minimization of the log determinant of the partially updated covariance matrix to the KL-divergence objective of Cholesky factorization. Computing the log determinant of the partially updated covariance matrix, we make use of (A.39) and make use of the fact that the determinant of a triangular matrix is the product of its diagonal entries:

$$\frac{1}{2}\operatorname{logdet}(\Theta_{:,:\shortmid k}) = \underbrace{\log(L_{11}) + \cdots + \log(L_{jj})}_{\text{the same}} + \underbrace{\log(L_{j+1,j+1|k}) + \cdots + \log(L_{M,M|k})}_{\text{conditioned}} \tag{A.41}$$

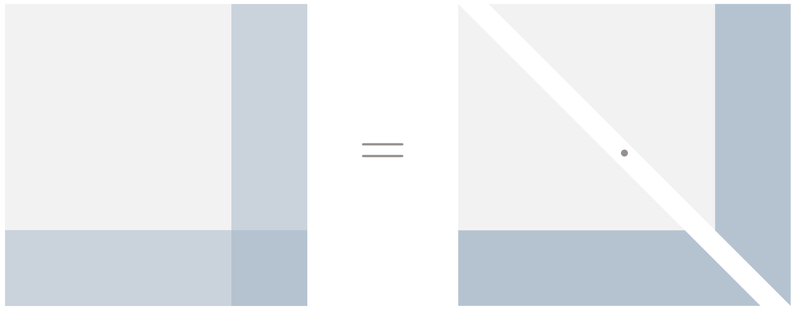

FIG. 2. *Illustration of the Cholesky factorization of a partially conditioned matrix.*

Comparing to the KL-divergence (3.4), $\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0},\Theta)\,\middle\|\,\mathcal{N}(\mathbf{0},(LL^\top)^{-1})\right)$ which is equivalent to maximizing

$$= \sum_{i=1}^{M} \log\left(\Theta_{ii|s_i-\{i\}}\right) \tag{A.42}$$

Recalling that k is added partially to some s_i, only those $i > j$

$$\begin{aligned} &= \underbrace{\log\left(\Theta_{11|s_1-\{1\}}\right) + \cdots + \log\left(\Theta_{jj|s_j-\{j\}}\right)}_{\text{the same}} + \\ &\underbrace{\log\left(\Theta_{j+1,j+1|s_{j+1}-\{j+1\}}\right) + \cdots + \log\left(\Theta_{MM|s_M-\{M\}}\right)}_{\text{conditioned}} \end{aligned} \tag{A.43}$$

Since L_{ii} is the square root of the variance of the ith variable conditional on each entry before it in the ordering, we have

$$2\log(L_{ii}) = \log(\Theta_{ii|s_i-\{i\}}) \tag{A.44}$$

So minimizing the log determinant of the partially conditioned covariance matrix (A.41) is the same as minimizing the KL-divergence (3.4).

A.6. Algorithm for Partial Updates. We now need an efficient algorithm to keep track of partial updates. The key idea is to maintain the prediction matrix with selected points inserted to maintain proper ordering, and keep track of the log determinant throughout selection. We first give how this different perspective affects the interpretation of the multiple point selection algorithm. In the example, let x and y be selected points and 1 and 2 be prediction points.

$$\Theta = \begin{pmatrix} \Theta_{xx} & \Theta_{xy} & \Theta_{x1} & \Theta_{x2} \\ \Theta_{yx} & \Theta_{yy} & \Theta_{y1} & \Theta_{y2} \\ \Theta_{1x} & \Theta_{1y} & \Theta_{11} & \Theta_{12} \\ \Theta_{2x} & \Theta_{2y} & \Theta_{21} & \Theta_{22} \end{pmatrix} \tag{A.45}$$

Computing the log determinant by chain rule,

$$\operatorname{logdet}(\Theta) = \log(\Theta_{xx}) + \log(\Theta_{yy|x}) + \log(\Theta_{11|x,y}) + \log(\Theta_{22|x,y,1}) \tag{A.46}$$

Isolating the objective — the variances of the prediction points

$$\log(\Theta_{11|x,y}) + \log(\Theta_{22|x,y,1}) = \operatorname{logdet}(\Theta) - \log(\Theta_{xx}) - \log(\Theta_{yy|x}) \tag{A.47}$$

Now consider how inserting y changed the objective from when it was just x.

$$\log(\Theta_{11|x}) + \log(\Theta_{22|x,1}) = \operatorname{logdet}(\Theta_{-y,-y}) - \log(\Theta_{xx}) \tag{A.48}$$

$$\Delta = \operatorname{logdet}(\Theta) - \log(\Theta_{yy|x}) - \operatorname{logdet}(\Theta_{-y,-y}) \tag{A.49}$$

But from (2.12) we know

$$\Delta = \log\left(\frac{\Theta_{yy|x,1,2}}{\Theta_{yy|x}}\right) \tag{A.50}$$

Substituting,

$$\log(\Theta_{yy|x,1,2}) - \log(\Theta_{yy|x}) = \operatorname{logdet}(\Theta) - \log(\Theta_{yy|x}) - \operatorname{logdet}(\Theta_{-y,-y}) \tag{A.51}$$

$$\log(\Theta_{yy|x,1,2}) = \operatorname{logdet}(\Theta) - \operatorname{logdet}(\Theta_{-y,-y}) \tag{A.52}$$

In general,

$$\log(\Theta_{kk|I,\mathrm{Pr}}) = \operatorname{logdet}(\Theta) - \operatorname{logdet}(\Theta_{-k,-k}) \tag{A.53}$$

Another way to arrive at the same result is to note that if we inserted y at the *end* of $\Theta_{-y,-y}$, to compute the log determinant of the new, bigger matrix Θ we would add the variance of y conditional on every entry in the matrix to the old determinant by chain rule. Since the determinant is invariant to symmetric permutation, the matrix inserting y at the end has the same determinant as inserting y where it should be.

So we see that the conditional variance of a candidate point conditional on everything else in the matrix is the difference in log determinant between the matrix with the candidate inserted and the original matrix. The multiple prediction point algorithm can therefore be interpreted as we insert the candidate *after* all the previously selected points (so it is conditional on all the previous points) and *before* the prediction points (which conditions all of them). We then compute $\log(\Theta_{kk|I,\mathrm{Pr}})$ for some candidate k which represents the difference in log determinant and then subtract $\log(\Theta_{kk|I})$ which is the spurious variance introduced by inserting k into the matrix. We do not need to subtract the spurious variances from the previously selected points because k does not affect them, and we select candidates by *relative* score.

We now apply this result to partial selection. In the example, let 1 and 2 be prediction points while x and y are both a selected points below 2 but above 1, where x has already been selected and y is a candidate.

$$\Theta = \begin{pmatrix} \Theta_{11} & \Theta_{1y} & \Theta_{1x} & \Theta_{12} \\ \Theta_{y1} & \Theta_{yy} & \Theta_{yx} & \Theta_{y2} \\ \Theta_{x1} & \Theta_{xy} & \Theta_{xx} & \Theta_{x2} \\ \Theta_{21} & \Theta_{2y} & \Theta_{2x} & \Theta_{22} \end{pmatrix} \tag{A.54}$$

Computing the log determinant by chain rule,

$$\operatorname{logdet}(\Theta) = \log(\Theta_{11}) + \log(\Theta_{yy|1}) + \log(\Theta_{xx|1,y}) + \log(\Theta_{22|1,y,x}) \tag{A.55}$$

We see that y conditions 2 but not 1, precisely what we want to encode. However, we have introduced a spurious term $\log(\Theta_{yy|1})$ and changed the variance of x, both of which must be subtracted out.

$$\log(\Theta_{11}) + \log(\Theta_{22|1,y,x}) = \operatorname{logdet}(\Theta) - \log(\Theta_{yy|1}) - \log(\Theta_{xx|1,y}) \tag{A.56}$$

We can substitute $\log(\Theta_{yy|1,x,2})$ for $\operatorname{logdet}(\Theta)$ by (A.53). Athough it differs by a constant, this does not change the objective.

$$= \log(\Theta_{yy|1,x,2}) - \log(\Theta_{yy|1}) - \log(\Theta_{xx|1,y}) \tag{A.57}$$

As long as we can compute conditional variances of our candidate on each *prefix* of the current ordering of prediction points interspersed with selected points, we can use the conditional variances to compute the updated conditional variances of the selected points by using their conditional covariances with the candidate. We are then able to compute every term in the objective. To do so, we maintain a partial Cholesky factor whose ordering is given by the current ordering. When we select a new point, we insert it in its appropriate place in the Cholesky factor. To update the Cholesky factor after an insertion efficiently, we left-look to get the column of its insertion position, and then update all columns right of the column by a rank-one downdate as described in Appendix A.4 which touches every entry in the Cholesky factor, $\mathcal{O}(N(m+s)$ per update for a total cost of $\mathcal{O}(N(m+s)(s))$ over s selections.

By inspecting the Cholesky factor, we get the covariance of a selected point with a candidate, conditional on all the points prior to the selected point in the ordering. The conditional variance of the selected point is the diagonal entry. We can then compute the new conditional variance given the variance of the candidate, conditional on all points prior to the selected point. Suppose we are at index i and the candidate is index j, the updates are as follows:

$$\Theta_{ii|:i-1} = (L_{ii})^2 \tag{A.58}$$

$$\Theta_{ij|:i-1} = L_{ij} \cdot L_{ii} \tag{A.59}$$

$$\Theta_{ii|:i-1,j} = \Theta_{ii|:i-1} - \frac{\Theta_{ij|:i-1}^2}{\Theta_{jj|:i-1}} \tag{A.60}$$

$$\Theta_{jj|:i-1,i} = \Theta_{jj|:i-1} - \frac{\Theta_{ij|:i-1}^2}{\Theta_{ii|:i-1}} \tag{A.61}$$

$$= \Theta_{jj|:i} \tag{A.62}$$

Of course, the base case Θ_{jj} is simply $K(\boldsymbol{x}_j, \boldsymbol{x}_j)$, the variance of the jth point.

For each of the N candidates, it requires $m+s$ operations from the above updates to compute the objective. Over s selections, the total time is the same as the cost to update the Cholesky factor, matching the complexity of the non-partial multiple point algorithm. However, the asymptotic work in the non-partial algorithm can be implemented as BLAS level-2 calls, while the partial algorithm relies heavily on vector (level-1) calls, affecting the constant-factor performance of the algorithm.

A.7. Equivalence of Selection and Orthogonal Matching Pursuit. We show that the single-point selection algorithm described in Algorithm 2.5 is the covariance space equivalent to the feature space orthogonal matching pursuit (OMP) algorithm described in [5]. The equivalence comes from the fact that Cholesky factorization is Gram-Schmitt in feature space.

Let Θ be a symmetric positive definite matrix such that

$$\Theta = F^\top F \tag{A.63}$$

For some matrix F whose columns are vectors in feature space,

$$F = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N \end{pmatrix} \tag{A.64}$$

Immediately we have

$$\Theta_{ij} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{A.65}$$

where $\langle \cdot, \cdot \rangle$ denotes the ordinary inner product on $\mathbb{R}^N$.

It suffices to see a single step of Cholesky factorization. Selecting $\boldsymbol{x}_1$,

$$\Theta' = \Theta - \frac{\boldsymbol{x}_1 \boldsymbol{x}_1^\top}{\Theta_{11}} \tag{A.66}$$

$$\Theta'_{ij} = \Theta_{ij} - \frac{\Theta_{i1}\Theta_{j1}}{\Theta_{ii}} \tag{A.67}$$

Switching to the feature space perspective, if we select $\boldsymbol{x}_1$ we force the rest of the feature vectors to be orthogonal to $\boldsymbol{x}_1$,

$$\boldsymbol{x}'_i = \boldsymbol{x}_i - \frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_1 \rangle}{\langle \boldsymbol{x}_1, \boldsymbol{x}_1 \rangle} \boldsymbol{x}_1 \tag{A.68}$$

$$\langle \boldsymbol{x}'_i, \boldsymbol{x}'_j \rangle = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_1 \rangle \langle \boldsymbol{x}_j, \boldsymbol{x}_1 \rangle}{\langle \boldsymbol{x}_1, \boldsymbol{x}_1 \rangle} \tag{A.69}$$

Comparing (A.67) and (A.69), we see that they are the same as expected. As a corollary, the objective of selecting the point $\boldsymbol{x}_k$ that minimizes the residual of some target point $\boldsymbol{x}_{\mathrm{Pr}}$ can be written as

$$\|\boldsymbol{x}_{\mathrm{Pr}} - \mathrm{proj}_{\boldsymbol{x}_k} \boldsymbol{x}_{\mathrm{Pr}}\| = \langle \boldsymbol{x}_{\mathrm{Pr}}, \boldsymbol{x}_{\mathrm{Pr}} \rangle - \frac{\langle \boldsymbol{x}_{\mathrm{Pr}}, \boldsymbol{x}_k \rangle^2}{\langle \boldsymbol{x}_k, \boldsymbol{x}_k \rangle} \tag{A.70}$$

Which is precisely the squared covariance of the candidate with the prediction over the variance of the candidate, as in (2.8).

(A.71)

This shows the equivalence as the objective is the same.

Appendix B. Derivations in KL-minimization.

B.1. Linear-algebraic formulation of objective. We want to show that the KL-divergence between two multivariate Gaussians centered at $\mathbf{0}$ with covariance matrices Θ_1 and Θ_2 can be written as

$$2\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0}, \Theta_1) \,\middle\|\, \mathcal{N}(\mathbf{0}, \Theta_2)\right) = \mathrm{trace}(\Theta_2^{-1}\Theta_1) + \mathrm{logdet}(\Theta_2) - \mathrm{logdet}(\Theta_1) - N \tag{B.1}$$

where Θ_1 and Θ_2 are both of size $N \times N$. Recall that the log density $\log \pi(\boldsymbol{x})$ for $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \Theta)$ is

$$\log \pi(\boldsymbol{x}) = -\frac{1}{2}(N \log(2\pi) + \mathrm{logdet}(\Theta) + \boldsymbol{x}^\top \Theta^{-1} \boldsymbol{x}) \tag{B.2}$$

By the definition of KL-divergence,

$$2\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0}, \Theta_1) \,\middle\|\, \mathcal{N}(\mathbf{0}, \Theta_2)\right) = 2\,\mathrm{E}_P[\log P - \log Q] \tag{B.3}$$

where P and Q are the corresponding densities for Θ_1 and Θ_2 respectively, and E_P denotes expectation under P.

$$\begin{aligned} &= 2\,\mathrm{E}_P[-\frac{1}{2}(N\log(2\pi) + \mathrm{logdet}(\Theta_1) + \boldsymbol{x}^\top \Theta_1^{-1} \boldsymbol{x}) \\ &\qquad + \frac{1}{2}(N\log(2\pi) + \mathrm{logdet}(\Theta_2) + \boldsymbol{x}^\top \Theta_2^{-1} \boldsymbol{x})] \end{aligned} \tag{B.4}$$

$$= \mathrm{E}_P[\boldsymbol{x}^\top \Theta_2^{-1} \boldsymbol{x} - \boldsymbol{x}^\top \Theta_1^{-1} \boldsymbol{x}] + \mathrm{logdet}(\Theta_2) - \mathrm{logdet}(\Theta_1) \tag{B.5}$$

$$\mathrm{E}_P[\boldsymbol{x}^\top \Theta_2^{-1} \boldsymbol{x} - \boldsymbol{x}^\top \Theta_1^{-1} \boldsymbol{x}] = \mathrm{E}_P[\mathrm{trace}(\boldsymbol{x}^\top \Theta_2^{-1} \boldsymbol{x}) - \mathrm{trace}(\boldsymbol{x}^\top \Theta_1^{-1} \boldsymbol{x})] \tag{B.6}$$

because the trace of a scalar is a scalar, and the linearity of trace.

$$
\begin{aligned}
&(\text{B.7}) && = \mathrm{E}_P[\operatorname{trace}(\Theta_2^{-1}\boldsymbol{x}\boldsymbol{x}^\top) - \operatorname{trace}(\Theta_1^{-1}\boldsymbol{x}\boldsymbol{x}^\top)] && \text{cyclic property of trace} \\
&(\text{B.8}) && = \mathrm{E}_P[\operatorname{trace}(\Theta_2^{-1}\boldsymbol{x}\boldsymbol{x}^\top - \Theta_1^{-1}\boldsymbol{x}\boldsymbol{x}^\top)] && \text{linearity of trace} \\
&(\text{B.9}) && = \mathrm{E}_P[\operatorname{trace}\left((\Theta_2^{-1} - \Theta_1^{-1})\boldsymbol{x}\boldsymbol{x}^\top\right)] && \text{factoring} \\
&(\text{B.10}) && = \operatorname{trace}(\mathrm{E}_P\left[(\Theta_2^{-1} - \Theta_1^{-1})\boldsymbol{x}\boldsymbol{x}^\top\right]) && \text{swapping trace and expectation} \\
&(\text{B.11}) && = \operatorname{trace}((\Theta_2^{-1} - \Theta_1^{-1})\,\mathrm{E}_P\left[\boldsymbol{x}\boldsymbol{x}^\top\right]) && \text{linearity of expectation} \\
&(\text{B.12}) && = \operatorname{trace}((\Theta_2^{-1} - \Theta_1^{-1})\Theta_1) && \Theta_1 = \mathrm{E}_P[\boldsymbol{x}\boldsymbol{x}^\top] \\
&(\text{B.13}) && = \operatorname{trace}(\Theta_2^{-1}\Theta_1 - I) && \text{multiplying} \\
&(\text{B.14}) && = \operatorname{trace}(\Theta_2^{-1}\Theta_1) - \operatorname{trace}(I) && \text{linearity of trace} \\
&(\text{B.15}) && = \operatorname{trace}(\Theta_2^{-1}\Theta_1) - N && \text{trace of } N \times N \text{ identity } N
\end{aligned}
$$

Combining (B.15) with (B.5), we obtain

$$
2\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0}, \Theta_1)\,\middle\|\,\mathcal{N}(\mathbf{0}, \Theta_2)\right) = \operatorname{trace}(\Theta_2^{-1}\Theta_1) + \operatorname{logdet}(\Theta_2) - \operatorname{logdet}(\Theta_1) - N
$$

as desired.

B.2. Reduction for optimal factor. We wish to compute the KL-divergence between Θ and the Cholesky factor L computed according to Theorem 3.1. From (3.2),

$$
2\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0}, \Theta)\,\middle\|\,\mathcal{N}(\mathbf{0}, (LL^\top)^{-1})\right) = \operatorname{trace}(LL^\top\Theta) - \operatorname{logdet}(LL^\top) - \operatorname{logdet}(\Theta) - N
$$

Ignoring terms not depending on L,

$$
= \operatorname{trace}(LL^\top\Theta) - \operatorname{logdet}(LL^\top) \tag{B.16}
$$

By the cyclic property of trace,

$$
= \operatorname{trace}(L\Theta L^\top) - \operatorname{logdet}(LL^\top) \tag{B.17}
$$

Focusing on $\operatorname{trace}(L\Theta L^\top)$ and expanding on the columns of L,

$$
\operatorname{trace}(L\Theta L^\top) = \sum_{i=1}^{N} \left(L_{s_i,i}^\top \Theta_{s_i,s_i} L_{s_i,i}\right) \tag{B.18}
$$

Plugging in $L_{s_i,i}$ from Theorem 3.1,

$$
= \sum_{i=1}^{N} \left[\left(\frac{\left(\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1\right)^\top}{\sqrt{\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}}\right) \Theta_{s_i,s_i} \left(\frac{\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}}\right)\right] \tag{B.19}
$$

$$
= \sum_{i=1}^{N} \left[\frac{\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1} \Theta_{s_i,s_i} \Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}{\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}\right] \tag{B.20}
$$

$$
= \sum_{i=1}^{N} 1 = N \tag{B.21}
$$

Using N for $\operatorname{trace}(LL^\top\Theta)$ in (B.16),

$$
2\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0}, \Theta)\,\middle\|\,\mathcal{N}(\mathbf{0}, (LL^\top)^{-1})\right) = -\operatorname{logdet}(LL^\top) - \operatorname{logdet}(\Theta) \tag{B.22}
$$

$L^\top$ has the same log determinant as L, and because L is lower triangular, its log determinant is just the sum of its diagonal entries:

$$= -2\sum_{i=1}^{N}\left[\log(L_{ii})\right] - \operatorname{logdet}(\Theta) \tag{B.23}$$

Plugging (3.3) for the diagonal entries,

$$= -\sum_{i=1}^{N}\left[\log(\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1)\right] - \operatorname{logdet}(\Theta) \tag{B.24}$$

Bringing the negative inside,

$$= \sum_{i=1}^{N}\left[\log\left((\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1)^{-1}\right)\right] - \operatorname{logdet}(\Theta) \tag{B.25}$$

∎

So minimizing the KL-divergence (given optimal L) corresponds to minimizing the sum of the (inverse) of the diagonal entries. Intuitively, because the Cholesky factorization is iterative conditioning (2.10), we can view the sum as the accumulated prediction error for a series of prediction problems, where each prediction problem is to predict the value of the ith variable given variables $1, 2, \ldots, i-1$. This gives a natural way to measure the quality of a sparsity pattern as a good sparsity pattern should maintain predictive accuracy while subject to the constraint that some variables have no interaction with others.

Because the KL-divergence is not symmetric, it matters which way the KL-divergence is taken as well as whether both matrices have been inverted or not. This seems to imply that there are four possible ways to compare two covariance matrices. However, note that

$$\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0}, \Theta)\,\middle\|\,\mathcal{N}(\mathbf{0}, (LL^\top)^{-1})\right) = \mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0}, LL^\top)\,\middle\|\,\mathcal{N}(\mathbf{0}, \Theta^{-1})\right) \tag{B.26}$$

from (3.2) and the cyclic property of trace, so inverting both matrices implicitly reverses the order of the KL-divergence. There are therefore only two possible ways to compare the two, which depends on the order of the arguments. A statistical interpretation comes from the fact that the KL-divergence can be interpreted as the likelihood-ratio test, so the non-symmetry of the order of the arguments corresponds to the asymmetry between the null and alternative hypotheses.