SPARSE CHOLESKY FACTORIZATION BY GREEDY CONDITIONAL SELECTION

STEPHEN HUAN*, JOE GUINNESS†, MATTHIAS KATZFUSS‡, HOUMAN OWHADI§, AND FLORIAN SCHÄFER¶

Abstract. Dense kernel matrices resulting from pairwise evaluations of a kernel function arise naturally in machine learning and statistics. Previous work in constructing sparse transport maps or sparse approximate inverse Cholesky factors for such matrices by minimizing Kullback-Leibler divergence recovers the Vecchia approximation for Gaussian processes. However, this method for Cholesky factorization usually relies only on geometry to construct the sparsity pattern, ignoring the conditional effect of adding an entry. In this work, we construct the sparsity pattern by leveraging a greedy selection algorithm which selects points that maximize mutual information with target points, conditional on all points selected previously. For selecting k points out of N, the naive time complexity is $\mathcal{O}(Nk^4)$, but by maintaining a partial Cholesky factor we reduce this to $\mathcal{O}(Nk^2)$. Furthermore, for multiple (m) targets we achieve a time complexity of $\mathcal{O}(Nk^2 + Nm^2 + m^3)$ which is maintained in the setting of Cholesky factorization where a selected point need not condition every target. We directly apply the selection algorithm to image classification and recovery of sparse Cholesky factors, improving upon k-th nearest neighbors in every case. By minimizing Kullback-Leibler divergence, we apply the algorithm to Cholesky factorization and Gaussian process regression, improving in high dimensional geometries as well as when preconditioning with the conjugate gradient.

Key words. to do

AMS subject classifications.

1. Introduction.

The problem. This work is concerned with Gaussian process regression in the setting of N points in D dimension space, whose covariance matrix $\Theta \in \mathbb{R}^{N \times N}$ is induced by pairwise evaluation of a prescribed kernel function. Gaussian processes enjoy widespread application in spatial statistics and geostatistics [20], machine learning through kernel machines [19], and optimal experimental design [18], e.g. in sensor placement [14]. However, statistical inference with Gaussian processes often requires computing quantities such as $\Theta \boldsymbol{v}$, $\Theta^{-1}\boldsymbol{v}$, $\operatorname{logdet}(\Theta)$, at a computational cost of $\mathcal{O}(N^2)$ or $\mathcal{O}(N^3)$ and a memory cost of $\mathcal{O}(N^2)$ to store the *dense* $N \times N$ covariance matrix. Overcoming these computational challenges for Gaussian process regression naturally generalizes to fast algorithms for numerical linear algebra on positive-definite matrices, since all positive-definite matrices can be viewed as covariance matrices and vice-versa.

Existing work. Computation with positive-definite matrices is often accelerated with sparse Cholesky factorization, either through incomplete Cholesky factorization of the covariance Θ [22, 3, 4], sparse approximate Cholesky factorization of the precision Θ^{-1} [21], or factorized sparse approximate inverse (FSAI) preconditioners [11, 32]. From the perspective of transport maps, the Cholesky factor L satisfying $\Theta^{-1} = L^{\top} L$ can be viewed as a linear transport map mapping an arbitrary multivariate Gaussian distribution $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, \Theta)$ to the standard Gaussian $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, I_N)$ through $\boldsymbol{z} = L\boldsymbol{x}$. Thus, one generalization of Cholesky factorization to nonlinear and

*Georgia Institute of Technology
†Joe
‡Matthias
§Houman
¶Georgia Institute of Technology, S1317 CODA, 756 W Peachtree St Atlanta, GA 30332, florian.schaefer@cc.gatech.edu,
Corresponding Author

non-Gaussian distributions is the Knothe-Rosenblatt rearrangement, which preserves lower triangularity, monotonicity, and sparsity, particularly in the inverse mapping [26]. These triangular transport maps have been applied to problems in spatial statistics, particularly for simulation and sampling problems [17, 13].

Vecchia approximation. Many of these existing approaches can be viewed through the lens of an early approach for fast Gaussian process regression, the Vecchia approximation [31, 12]. The key observation is a decomposition of the joint likelihood π.

$$\pi(\boldsymbol{x}) = \pi(x_1)\pi(x_2 \mid x_1) \dots \pi(x_N \mid x_1, x_2, \dots, x_{N-1}) \tag{1.1}$$

Assuming that there are many points, the key assumption is that many of the points contribute little additional information, conditional on a carefully chosen subset of the points. Letting s_i denote the indices of training points to include for the ith variable, the Vecchia approximation proposes to approximate (1.1) by the sparse approximation

$$\pi(\boldsymbol{x}) \approx \pi(x_1)\pi(x_2 \mid x_{s_2}) \dots \pi(x_N \mid x_{s_N}) \tag{1.2}$$

From the Vecchia approximation, the joint likelihood can be factored into N independent regression problems, where each regression problem is to approximate the conditional distribution of the ith variable. Independence allows for embarrassingly parallel algorithms, which is exploited both in Gaussian process regression [12] and Cholesky factorization [21]. The difference is that while the Gaussian process perspective emphasizes local regression problems and sparsity in point selection (or variable interaction), Cholesky factorization emphasizes minimizing global functionals and sparsity in the resulting matrix factor. If an appropriate functional is chosen and a sparsity pattern $\mathcal{S}$ specified, then it is an optimization problem to determine the entries of the matrix factor. Multiple functionals in the literature actually converge to the same closed-form expression for the entries of the resulting Cholesky factor L subject to the constraint that $L \in \mathcal{S}$: Minimizers of the Kaporin condition number $(\operatorname{trace}(L\Theta L^\top)/N)^N / \det(L\Theta L^\top)$ [11], minimizes of $\|\mathrm{Id} - L \operatorname{chol}(\Theta)\|_{\mathrm{FRO}}$ additionally subject to the constraint $\operatorname{diag}(L\Theta L^\top) = 1$ [32], and minimizers of the KL-divergence $\mathbb{D}_{\mathrm{KL}}(\mathcal{N}(\mathbf{0}, \Theta) \parallel \mathcal{N}(\mathbf{0}, (LL^\top)^{-1}))$ [21]. This shared closed-form expression for the entries of the resulting factor can be shown to be equivalent to the formula used to compute the Vecchia approximation for Gaussian processes [31]. In addition, the KL-divergence is used to compute sparse lower-triangular maps in [17, 13].

The primary difference between the numerical linear algebra viewpoint (Kaporin conditional number or Frobenius norm minimization) and the statistical viewpoint (KL-minimization) is whether the covariance is factored, $L \approx \operatorname{chol}(\Theta)^{-1}$, or the precision, $L \approx \operatorname{chol}(\Theta^{-1})$. As observed in [21, 26], factors of the precision are often much sparser than factors of the covariance, because the precision encodes conditional independence while the covariance encodes marginal independence. Covariance matrices arising from kernel functions are often fully dense, but the approximate factors of their precision can be sparse if the ordering and sparsity pattern are chosen carefully.

Sparsity selection by geometry. Assuming that a suitable elimination ordering is chosen, it remains to be decided the sparsity set for each point. Indices are often chosen to be the closest points to the current point by Euclidean distance [31, 22, 21, 13]. This choice can be justified by noting that popular kernel functions like the Matérn family of kernel functions decay exponentially with increasing distance. For more general kernel functions, geostatisticians have long observed the "screening effect", or the observation that conditional on points close to the point of interest, far away points are nearly conditionally independent [27, 28].

However, selecting by distance alone ignores the conditional effect of adding new points to the sparsity set. As an illustrative example, imagine the closest point to the point of interest has been duplicated multiple times. Once the duplicated point has been selected, conditional on the selected point, the duplicates provide no additional information. But they are still closest to the point of interest, so selecting by distance alone would still add these redundant points to the sparsity set. Instead, we propose greedily selecting based on conditional mutual information with the point of interest.

Conditional selection. The machine learning community has long developed algorithms that greedily select the next point to include by maximizing an information-theoretic objective [25, 9, 23]. Similar algorithms have been developed in the context of sensor placement [14, 5] and experimental design [18] where it is assumed the target phenomena is modeled by a Gaussian process or is otherwise linearly dependent on the selected measurements. More recent work exploits the empirical observation that the marginal log-likelihood is approximately linear, giving a principled way to determine the number of selected points without viewing the whole dataset [2].

Our proposed selection method can be viewed as the covariance equivalent of a popular algorithm in signal processing and compressive sensing, orthogonal matching pursuit (OMP) [29, 30], which seeks to approximate a target vector as the sparse linear combination from a given collection of vectors. OMP is a workhorse algorithm used in as diverse contexts as signal recovery [29, 30], polynomial chaos expansion [1], and the use of neural networks to solve partial differential equations [8, 24].

Nearly all of these selection methods leverage numerical linear algebra for efficient computation, particularly the Cholesky factorization [9, 23, 2] or the closely related QR factorization [5, 29, 1]. We wish to emphasize that algorithms for Cholesky factorization translate naturally to Gaussian processes regression and vice-versa.

Main results. For a single point of interest, a direct computation of the conditional mutual information criterion would have a computational time complexity of $\mathcal{O}(Nk^4)$ to greedily select k points out of N total, but by maintaining a partial Cholesky factor, we are able to reduce this complexity to $\mathcal{O}(Nk^2)$. We extend this basic selection algorithm to maximize mutual information with *multiple* points of interest, to naturally take advantage of the "two birds with one stone" effect. For m target points we achieve a time complexity of $\mathcal{O}(Nk^2 + Nm^2 + m^3)$, which for $m \approx k$ is essentially m times faster than the single-point algorithm. In the setting of aggregated (or supernodal) Cholesky factorization where the sparsity pattern for multiple columns is determined at once, a candidate entry may be between two columns — above one, but below another. Adding this entry only conditions a *partial* subset of the target points. By carefully applying rank-one downdating of Cholesky factors, we are able to capture this structure at the same time complexity for multiple points.

Greedy selection *locally* infers the posterior distribution for distinguished point(s) of interest. In order to get a *global* approximation for the entire Gaussian process, we compute a sparse approximate Cholesky factor of the precision by minimizing the KL-divergence between the centered multivariate normal distributions with covariance matrices Θ and $(LL^\top)^{-1}$ subject to the constraint $L \in \mathcal{S}$ as in [21]. We show that the resulting optimization problem reduces to independent regression problems for each column. In applying our greedy selection method, we are able to get more accurate Cholesky factors for the same number of nonzeros compared to if the sparsity was selected with geometry. Finally, we show how to adaptively determine the number of nonzeros per column in order to reduce the global KL-divergence by maintaining a global priority queue shared between each local greedy selection algorithm.

en: todo
more of the
is solidified

en: put im-
n the intro-
on?

Outline.
The remainder of the paper is organized as follows.

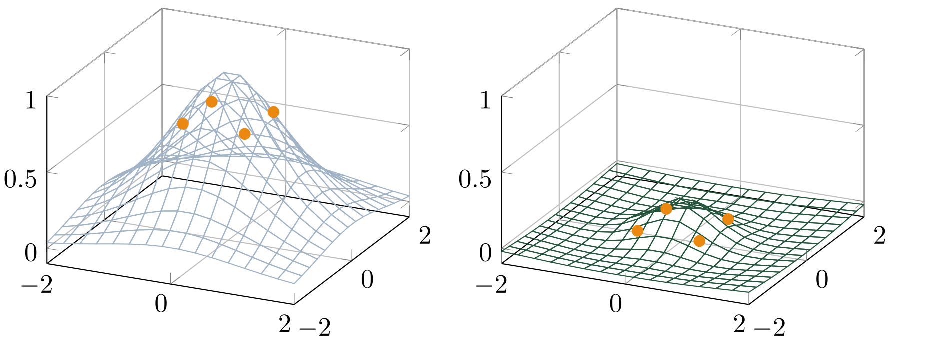

FIG. 1. *An illustration of the screening effect with the Matérn kernel with a length scale of 1 and smoothness $\nu = \frac{5}{2}$. The first figure shows the unconditional covariance with the point at (0, 0). The second figure shows the conditional covariance after conditioning on the four points in orange.*

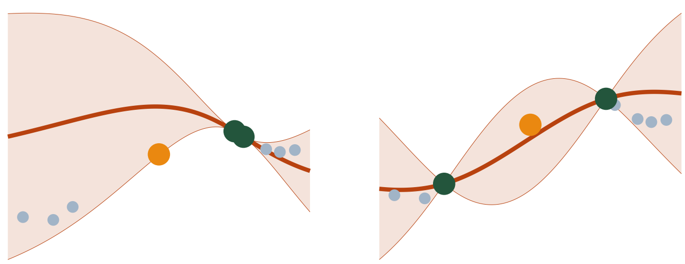

FIG. 2. *Here, the blue points are the candidates, or training points, the orange point is the unknown point, or the point to make a prediction at, and the green points are the k selected points. The red line is the conditional mean μ, conditional on the k selected points, and the $\mu - 2\sigma$ to $\mu + 2\sigma$ confidence interval is shaded for the conditional variance σ^2.*

n: "specific"
intuitively"
uff, meaning
hey don't
add infor-
n. In order
hieve crisp,
quality aca-
writing, it
portant to
those fluff
as much
ssible. It's
al to add
out of re-
itially, so
uires active
rocessing.

2. Greedy selection for directed inference. We motivate the greedy selection algorithm in the concrete setting of Gaussian process regression. We say the function $f(\boldsymbol{x})$ is distributed according to a Gaussian process prior with mean function $\mu(\boldsymbol{x})$ and covariance function or kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$, if for any finite set of points $X = \{\boldsymbol{x}_i\}_{i=1}^N$, $f(X) \sim \mathcal{N}(\boldsymbol{\mu}, \Theta)$, where $\mu_i = \mu(\boldsymbol{x}_i)$ and $\Theta_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, symbolized as $f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}'))$. By definition, the kernel function K yields positive-definite covariance matrices Θ for any set of points.

Given the training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ where the inputs $\boldsymbol{x}_i \in \mathbb{R}^D$ are collected in the matrix $X_{\text{Tr}} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N \times D}$ and the measurements at those points are collected in the vector $\boldsymbol{y}_{\text{Tr}} = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$, we wish to predict the values at M new points $X_{\text{Pr}} \in \mathbb{R}^{M \times D}$ for which $\boldsymbol{y}_{\text{Pr}} \in \mathbb{R}^M$ is unknown. We assume that there is a deterministic function $f(\boldsymbol{x})$ that maps the input points to observed output measurements and that this function is distributed according to a Gaussian process, $f \sim \mathcal{GP}(\mu(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}'))$ where we assume a zero mean function $\mu(\boldsymbol{x}) = \mathbf{0}$.

From the distribution on $f(\boldsymbol{x})$, the joint distribution of training and testing data

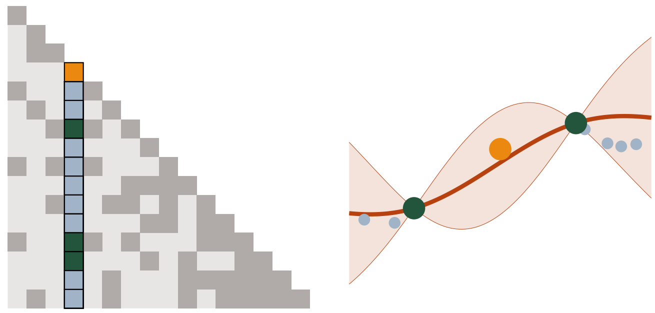

FIG. 3. *For a column in isolation, the unknown point is the diagonal entry, below it are candidates, and the selected entries are added to the sparsity pattern s_i. Thus, sparsity selection in Cholesky factorization is analogous to point selection in Gaussian processes.*

$\boldsymbol{y}$ has covariance $\Theta = \begin{pmatrix} \Theta_{\mathrm{Tr},\mathrm{Tr}} & \Theta_{\mathrm{Tr},\mathrm{Pr}} \\ \Theta_{\mathrm{Pr},\mathrm{Tr}} & \Theta_{\mathrm{Pr},\mathrm{Pr}} \end{pmatrix}$ where $\Theta_{I,J}$ denotes $K(X_I, X_J)$ for index sets I, J. In order to make predictions at the unknown points X_{Pr}, we condition the desired prediction $\boldsymbol{y}_{\mathrm{Pr}}$ on the observed measurements $\boldsymbol{y}_{\mathrm{Tr}}$. For Gaussian processes, the posterior distribution is of closed form given by

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

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

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

$$\Theta_{I,J \mid V} \coloneqq \Theta_{I,J} - \Theta_{I,V} \Theta_{V,V}^{-1} \Theta_{V,J} \tag{2.3}$$

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,

$$\Theta_{I,J \mid V_1, V_2, \ldots, V_n} \coloneqq \mathrm{Cov}[\boldsymbol{y}_I, \boldsymbol{y}_J \mid \boldsymbol{y}_{V_1 \cup V_2 \cup \cdots \cup V_n}] \tag{2.4}$$

denotes the covariance between the variables in index sets I and J, conditional on the variables in $V_1, V_2, \ldots, V_n$. Note that by the quotient rule of Schur complementation:

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

Calculating the posterior mean and variance requires inverting the training covariance matrix $\Theta_{\mathrm{Tr},\mathrm{Tr}}$, which has a computational time complexity of $\mathcal{O}(N^3)$ for N training points. This scaling is prohibitive for large datasets, so one natural approach is to carefully select a subset of s points out of the N, $s \ll N$, and pay a much smaller $\mathcal{O}(s^3)$ cost. Of course, throwing away $N - s$ points will necessarily lead to worse predictive accuracy. To maintain reasonable accuracy at a severely reduced computational cost requires a criterion to choose the most effective points.

2.1. Problem: optimal selection.

Stephen: elaborate more on mutual information information-theoretic criteria in the literature

One natural criterion is to maximize the *mutual information* between the selected points and the target point for prediction. The mutual information has been used by [14] to determine the best locations to place sensors. The mutual information, or *information gain* is defined as

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

Maximizing the mutual information is equivalent to minimizing the conditional entropy since the entropy of $\boldsymbol{y}_{\mathrm{Pr}}$ is constant. Because the differential entropy of a multivariate Gaussian is monotonically increasing with the log determinant of its covariance matrix, minimizing the conditional entropy is equivalent to minimizing the log determinant of the posterior covariance matrix. For a single predictive point, the log determinant reduces to its variance. Thus, maximizing mutual information minimizes the *conditional variance* of the target point. 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 outer 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 mutual information results from comparing the definition of mutual information (2.6) to the EV-VE identity,

$$\mathrm{H}[\boldsymbol{y}_{\mathrm{Pr}}] = \mathrm{H}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] + \mathrm{MI}[\boldsymbol{y}_{\mathrm{Pr}}; \boldsymbol{y}_{\mathrm{Tr}}] \tag{2.7}$$

$$\mathrm{Var}[\boldsymbol{y}_{\mathrm{Pr}}] = \mathrm{E}[\mathrm{Var}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]] + \mathrm{Var}[\mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]] \tag{2.8}$$

On the left hand side, entropy is monotone with variance. On the right hand side, the expectation of the conditional variance is monotone with conditional entropy and can be interpreted to be the fluctuation of the prediction point after conditioning. Because the sum of the expectation of conditional variance and variance of conditional expectation is constant, minimizing the expectation of the conditional variance is equivalent to maximizing the variance of conditional expectation, which corresponds to the mutual information. 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 is 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 $\boldsymbol{y}_{\mathrm{Tr}}$.

2.2. A greedy approach. In order to maximize conditional mutual information we greedily select the candidate with highest information, or the point which most reduces the conditional variance of the prediction point. Let $I = \{i_j\}_{j=1}^{s} \subseteq \mathrm{Tr}$ be the set of indices of selected training points. For a candidate index k, we condition the current covariance matrix on y_k according to (2.2):

$$\Theta_{:,:|I,k} = \Theta_{:,:|I} - \Theta_{:,k|I}\Theta_{k,k|I}^{-1}\Theta_{k,:|I} \tag{2.9}$$

$$= \Theta_{:,:|I} - \boldsymbol{u}\boldsymbol{u}^{\top} \tag{2.10}$$

$$\boldsymbol{u} = \frac{\Theta_{:,k|I}}{\sqrt{\Theta_{k,k|I}}} \tag{2.11}$$

From (2.10), conditioning on a new point is a rank-one downdate on the current covariance matrix. Thus, the amount that the variance of y_{Pr} will decrease after selecting k is given by u_{Pr}^2, or

$$u_{\mathrm{Pr}}^2 = \frac{\Theta_{\mathrm{Pr},k|I}^2}{\Theta_{k,k|I}} = \frac{\mathrm{Cov}[y_{\mathrm{Pr}}, \boldsymbol{y}_{\mathrm{Tr}}[k] \mid I]^2}{\mathrm{Var}[\boldsymbol{y}_{\mathrm{Tr}}[k] \mid I]} = \mathrm{Var}[y_{\mathrm{Pr}} \mid I]\,\mathrm{Corr}[y_{\mathrm{Pr}}, \boldsymbol{y}_{\mathrm{Tr}}[k] \mid I]^2 \tag{2.12}$$

We need to keep track of each candidate's variance and covariance with the prediction point after conditioning on the points already selected to compute (2.12). We start with the unconditional values given by $\Theta_{k,k}$ and $\Theta_{\mathrm{Pr},k}$ 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 $u_k u_{\mathrm{Pr}}$.

We have two strategies to efficiently compute $\boldsymbol{u}$. 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 computational time complexity $\mathcal{O}(s^2)$, see Appendix A.1. Once $\Theta_{I,I}^{-1}$ has been computed, $\boldsymbol{u}$ is computed directly 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 strategy is to take advantage of the quotient rule of Schur complementation. From a statistical perspective, 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 by two steps of block Gaussisan elimination,

$$\begin{pmatrix} \Theta_{1,1} & \Theta_{1,2} \\ \Theta_{2,1} & \Theta_{2,2} \end{pmatrix} = \begin{pmatrix} \mathrm{Id} & 0 \\ \Theta_{2,1}\Theta_{1,1}^{-1} & \mathrm{Id} \end{pmatrix} \begin{pmatrix} \Theta_{1,1} & 0 \\ 0 & \Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2} \end{pmatrix} \begin{pmatrix} \mathrm{Id} & \Theta_{1,1}^{-1}\Theta_{1,2} \\ 0 & \mathrm{Id} \end{pmatrix} \tag{2.13}$$

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

$$\begin{aligned} \mathrm{chol}(\Theta) &= \begin{pmatrix} \mathrm{Id} & 0 \\ \Theta_{2,1}\Theta_{1,1}^{-1} & \mathrm{Id} \end{pmatrix} \begin{pmatrix} \mathrm{chol}(\Theta_{1,1}) & 0 \\ 0 & \mathrm{chol}(\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}) \end{pmatrix} \\ &= \begin{pmatrix} \mathrm{chol}(\Theta_{1,1}) & 0 \\ \Theta_{2,1}\,\mathrm{chol}(\Theta_{1,1})^{-\top} & \mathrm{chol}(\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}) \end{pmatrix} \end{aligned} \tag{2.14}$$

Here the conditional expectation in (2.1) corresponds to $\Theta_{2,1}\Theta_{1,1}^{-1}$ and the conditional covariance in (2.2) corresponds to $\Theta_{2,2|1} = \Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}$. Thus, we see that Cholesky factorization is iteratively conditioning the Gaussian process. From the iterative conditioning perspective, the kth column of the Cholesky factor corresponds precisely to the corresponding $\boldsymbol{u}$ for k in (2.11) since a iterative sequence of conditioning on $i_1, i_2 \ldots, i_{k-1}$ is equivalent to conditioning on I by the quotient rule.

Adding a column to the current Cholesky factor can be efficiently computed without excess dependence on N with left-looking (see Algorithm 2.5), 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 time complexity of $\mathcal{O}(Ns^2)$, matching the time complexity of the explicit precision approach.

Stephen: should the algorithms be moved to the appendix?

While both approaches have the same time complexity, the precision algorithm uses $\mathcal{O}(s^2)$ space to store the precision $\Theta_{I,I}^{-1}$ while the Cholesky algorithm uses $\mathcal{O}(Ns)$ space 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. Although the precision algorithm uses less memory than the Cholesky algorithm, the Cholesky algorithm is preferred for better performance and ease of implementation.

Once the indices have been computed according to Algorithm 2.1 or Algorithm 2.2, 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 C.1.

Stephen: elaborate more on OMP in the literature

This algorithm is in fact the covariance equivalent of the signal recovery algorithm orthogonal matching pursuit (OMP) [29], a connection elaborated in Appendix A.11.

Algorithm 2.1 Point selection by explicit precision

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

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

Algorithm 2.2 Point selection by Cholesky factorization

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

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

FIG. 4. *Algorithms for single-point selection.*

2.3. Supernodes and blocked selection. We now consider efficiently dealing with multiple prediction points. The first question is how to generalize the objective for a single point (2.12) to multiple points. Following the same mutual information justification in subsection 2.1, a natural criterion is to minimize the log determinant of the prediction points' covariance matrix after conditioning on the selected points, or $\operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}|I})$. This objective, known as D-optimal design in the literature [14], has many intuitive interpretations: for example, as the volume of the region of uncertainty or as the scaling factor in the probability density function for multivariate Gaussians.

We need to efficiently compute the effect of selecting an index k on the log determinant. From (2.10), selecting an index is a rank-one update on the covariance matrix of the prediction points.

$$\operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I\cup\{k\}}\right) = \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I} - \frac{\Theta_{\mathrm{Pr},k|I}\Theta_{\mathrm{Pr},k|I}^\top}{\Theta_{k,k|I}}\right) \tag{2.15}$$

By application of the matrix determinant lemma (the details are in Appendix A.3),

$$= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I}\right) + \log\left(\frac{\Theta_{k,k\mid I,\mathrm{Pr}}}{\Theta_{k,k\mid I}}\right) \tag{2.16}$$

Equation (2.16) shows that to compute the updated log determinant, it suffices to only compute conditional variances of the candidate point. Intuitively this corresponds to *backwards* regression, where we imagine measuring the values of the *prediction points* instead of at the *candidates*. We then infer the posterior variance at a candidate, and pick the candidate whose conditional variance decreases the most (relative to its starting value). These candidates are likely to give information about the prediction points, because the prediction points give information about the candidate.

Stephen: image for "backwards" sensor placement

Re-writing the objective in this way motivates an efficient algorithm to compute the objective. We condition on a newly added point essentially the same as in subsection 2.2, but now maintaining two data structures instead of one: one for the variance after conditioning on the previously selected points, and the other for the 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 data 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 maintaining partial Cholesky factors.

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 the initial conditional variances $\Theta_{k,k\mid\mathrm{Pr}}$ for the N candidates. 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 (2.15) are given in Appendix A.4. Given the precisions, $\boldsymbol{u} = \frac{\Theta_{:,k\mid I}}{\sqrt{\Theta_{k,k\mid I}}}$ and $\boldsymbol{u}_{\mathrm{Pr}} = \frac{\Theta_{:,k\mid I,\mathrm{Pr}}}{\sqrt{\Theta_{k,k\mid I,\mathrm{Pr}}}}$ are computed as usual according to (2.2) in time Ns and Nm. Finally, for each candidate j, the conditional variance $\Theta_{j,j\mid I}$ is updated by subtracting u_j^2, the conditional covariance $\Theta_{\mathrm{Pr},k\mid I}$ is updated for each index c of a prediction point by subtracting $u_j u_c$, and the conditional variance $\Theta_{j,j\mid I,\mathrm{Pr}}$ 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 partial Cholesky factors 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 $\Theta_{j,j\mid I}$ and $\Theta_{j,j\mid I,\mathrm{Pr}}$ 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 $\mathcal{O}(Ns^2 + Nm^2 + m^3)$.

Like the single-point case, both approaches have the same time complexity but differ in space complexity. The precision algorithm 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 requires $\mathcal{O}((N+m)(m+s))$ memory to store the first $m+s$ columns of the Cholesky factor for the joint covariance matrix between training and prediction points, which simplifies to $\mathcal{O}(Ns + Nm + m^2)$. The memory usages are identical except for s^2 versus Ns, so the Cholesky algorithm again uses more memory than the precision algorithm.

Algorithm 2.3 Multiple prediction point selection by explicit precision

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

$N \leftarrow |\boldsymbol{x}_{\mathrm{Tr}}|$
$m \leftarrow |\boldsymbol{x}_{\mathrm{Pr}}|$
$\boldsymbol{x} \leftarrow \begin{pmatrix} \boldsymbol{x}_{\mathrm{Tr}} \\ \boldsymbol{x}_{\mathrm{Pr}} \end{pmatrix}$
$I \leftarrow \emptyset$
$-I \leftarrow \{1, \dots, N\}$
$\Theta^{-1}_{I,I} \leftarrow \mathbb{R}^{0\times 0}$
$\Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I} \leftarrow K(\boldsymbol{x}_{\mathrm{Pr}}, \boldsymbol{x}_{\mathrm{Pr}})$
$\Theta_{\mathrm{Tr},\mathrm{Pr}|I} \leftarrow K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Pr}})$
$\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I}) \leftarrow \mathrm{diag}(K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Tr}}))$
$\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}}) \leftarrow \mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I})$
  $- \mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Pr}|I} \Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I} \Theta_{\mathrm{Pr},\mathrm{Tr}|I})$
while $|-I| > 0$ and $|I| < s$ **do**
  $k \leftarrow \min_{j \in -I} \frac{\Theta_{j,j|I,\mathrm{Pr}}}{\Theta_{j,j|I}}$
  $I \leftarrow I \cup \{k\}$
  $-I \leftarrow -I - \{k\}$
  $\boldsymbol{v} \leftarrow \Theta^{-1}_{I,I} K(\boldsymbol{x}_{\mathrm{Tr}}[I - \{k\}], \boldsymbol{x}_{\mathrm{Tr}}[k])$
  $\Theta^{-1}_{I,I} \leftarrow \begin{pmatrix} \Theta^{-1}_{I,I} + \frac{\boldsymbol{v}\boldsymbol{v}^\top}{\Theta_{k,k|I}} & \frac{-\boldsymbol{v}}{\Theta_{k,k|I}} \\ \frac{-\boldsymbol{v}^\top}{\Theta_{k,k|I}} & \frac{1}{\Theta_{k,k|I}} \end{pmatrix}$
  $\boldsymbol{w} \leftarrow \Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I} \Theta^\top_{k,\mathrm{Pr}|I}$
  $\Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I} \leftarrow \Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I} + \frac{\boldsymbol{w}\boldsymbol{w}^\top}{\Theta_{k,k|I,\mathrm{Pr}}}$
  $\Theta_{:,k|I} \leftarrow K(\boldsymbol{x}, \boldsymbol{x}_k) - K(\boldsymbol{x}, \boldsymbol{x}_{I-\{k\}})\boldsymbol{v}$
  $\Theta_{:,k|I,\mathrm{Pr}} \leftarrow \Theta_{:,k|I} - \Theta_{:,\mathrm{Pr}|I}\boldsymbol{w}$
  $\boldsymbol{u} \leftarrow \frac{\Theta_{:,k|I}}{\sqrt{\Theta_{k,k|I}}}$
  $\boldsymbol{u}_{\mathrm{Pr}} \leftarrow \frac{\Theta_{:,k|I,\mathrm{Pr}}}{\sqrt{\Theta_{k,k|I,\mathrm{Pr}}}}$
  for $j \in -I$ **do**
    $\Theta_{j,j|I} \leftarrow \Theta_{j,j|I} - \boldsymbol{u}_j^2$
    $\Theta_{j,j|I,\mathrm{Pr}} \leftarrow \Theta_{j,j|I,\mathrm{Pr}} - (\boldsymbol{u}_{\mathrm{Pr}})_j^2$
    for $c \in \{1, \dots, m\}$ **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 2.4 Multiple prediction point selection by Cholesky factorization

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

$N \leftarrow |\boldsymbol{x}_{\mathrm{Tr}}|$
$m \leftarrow |\boldsymbol{x}_{\mathrm{Pr}}|$
$\boldsymbol{x} \leftarrow \begin{pmatrix} \boldsymbol{x}_{\mathrm{Tr}} \\ \boldsymbol{x}_{\mathrm{Pr}} \end{pmatrix}$
$I \leftarrow \emptyset$
$-I \leftarrow \{1, \dots, N\}$
$L \leftarrow \mathbf{0}^{(N+m)\times s}$
$L_{\mathrm{Pr}} \leftarrow \mathbf{0}^{(N+m)\times (s+m)}$
$\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I}) \leftarrow \mathrm{diag}(K(\boldsymbol{x}_{\mathrm{Tr}}, \boldsymbol{x}_{\mathrm{Tr}}))$
$\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}}) \leftarrow \mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I})$
for $i \in \{1, \dots, m\}$ **do**
  Update L_{Pr} and $\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}})$ with $k = N + i$ by Algorithm 2.5.
end for
while $|-I| > 0$ and $|I| < s$ **do**
  $k \leftarrow \min_{j \in -I} \frac{\Theta_{j,\mathrm{Pr}|I}}{\Theta_{j,j|I}}$
  $I \leftarrow I \cup \{k\}$
  $-I \leftarrow -I - \{k\}$
  $i \leftarrow |I|$
  Update L and $\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I})$ by Algorithm 2.5.
  Update L_{Pr} and $\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}})$ with $i = i + m$ by Algorithm 2.5.
end while
return I

Algorithm 2.5 Update Cholesky factor

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

$N \leftarrow |\mathrm{diag}(\Theta)|$
$L_{:,i} \leftarrow K(\boldsymbol{x}, \boldsymbol{x}_k) - L_{:,:i-1} L^\top_{k,:i-1}$
$L_{:,i} \leftarrow \frac{L_{:,i}}{\sqrt{L_{k,i}}}$
for $j \in \{1, \dots, N\}$ **do**
  $\Theta_{j,j} \leftarrow \Theta_{j,j} - L^2_{j,i}$
end for

FIG. 5. *Algorithms for multiple-point selection.*

2.4. Near optimality by empirical submodularity. If the objective satisfies the property of submodularity, then it is guaranteed that the greedy algorithm produces an objective within a a $1 - \frac{1}{e}$ of the optimal objective. Unfortunately the information gain objective is not submodular in general, see Appendix A.12. However, it is possible to empirically observe for a particular point set and choice of kernel function whether it is submodular. We note that one-dimensional points with a Matérn kernel function seems to be submodular, but not for any higher dimension.

[6, 10]

3. Greedy selection for *global* approximation by KL-minimization. reverse maximin ordering [7, 21, 22]

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 $\mathcal{O}(N^3)$ with standard algorithms, which is often prohibitive. Recall the problem of inference in Gaussian process regression as described in section 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 2.2 and 2.4 on each column of the Cholesky factorization.

Stephen: justify importance/downstream applications of 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 [21], or the expected difference in log-densities:

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

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

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

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

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

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

In order to maximize (3.4), we can ignore logdet(Θ) 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 $(\boldsymbol{e}_1^\top \Theta_{s_i,s_i}^{-1} \boldsymbol{e}_1)^{-1}$, 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 2.2 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.7.

We now show that the KL-minimization objective on the aggregated indices corresponds precisely to (2.15), 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:

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

We see that the objective (3.9) is equivalent to the objective (2.15), 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 2.4 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 $(LL^\top)^{-1} \approx K(\boldsymbol{x},\boldsymbol{x})$

1: $n \leftarrow |\boldsymbol{x}|$
2: **for** $\tilde{i} \in g$ **do**
3: $\quad J \leftarrow \{\max(\tilde{i}) + 1, \max(\tilde{i}) + 2, \ldots, n\}$
4: $\quad$ Compute I using Algorithm 2.3 or Algorithm 2.4 where $\boldsymbol{x}_{\mathrm{Tr}} = \boldsymbol{x}[J], \boldsymbol{x}_{\mathrm{Pr}} = \boldsymbol{x}[\tilde{i}], s = s - |\tilde{i}|$
5: $\quad \tilde{s} \leftarrow J[I]$
6: $\quad$ **for** $i \in \operatorname{reversed}(\operatorname{sorted}(\tilde{i}))$ **do**
7: $\quad\quad \tilde{s} \leftarrow \tilde{s} \cup \{i\}$
8: $\quad\quad s_i \leftarrow \operatorname{reversed}(\tilde{s})$
9: $\quad$ **end for**
10: **end for**
11: **return** L computed with Algorithm 3.3

Algorithm 3.2 Computing L without aggregation

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

1: $\Theta^{-1}_{s_i,s_i} \leftarrow K(\boldsymbol{x}[s_i], \boldsymbol{x}[s_i])^{-1}$
2: $L_{s_i,i} \leftarrow \dfrac{\Theta^{-1}_{s_i,s_i}\boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top \Theta^{-1}_{s_i,s_i}\boldsymbol{e}_1}}$
3: **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 $i \in \tilde{i}$

1: $s \leftarrow \tilde{i} \cup \tilde{s}$
2: $U \leftarrow P^{\updownarrow} \operatorname{chol}(P^{\updownarrow}\Theta_{s,s}P^{\updownarrow})P^{\updownarrow}$
3: **for** $i \in \tilde{i}$ **do**
4: $\quad k \leftarrow$ index of i in $\tilde{i}$
5: $\quad L_{s_i,i} \leftarrow U^{-\top}\boldsymbol{e}_k$
6: **end for**
7: **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:

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

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 (2.14), this can be directly read off the Cholesky factorization. Thus, we can simply compute:

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

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

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

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 2.3 or Algorithm 2.4 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 $\mathcal{O}(\frac{N^2s^2}{m})$ 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 $\mathcal{O}(N^2s^2)$ (using the non-aggregated algorithms Algorithms 2.2 and 3.2, but one can also use the aggregated versions Algorithms 2.4 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 [21, 22] 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.

4. Numerical experiments. All experiments were run on the Partnership for an Advanced Computing Environment (PACE) Phoenix cluster at the Georgia Institute of Technology, with 8 cores of a Intel(R) Xeon(R) Gold 6226 CPU @ 2.70GHz and 22 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

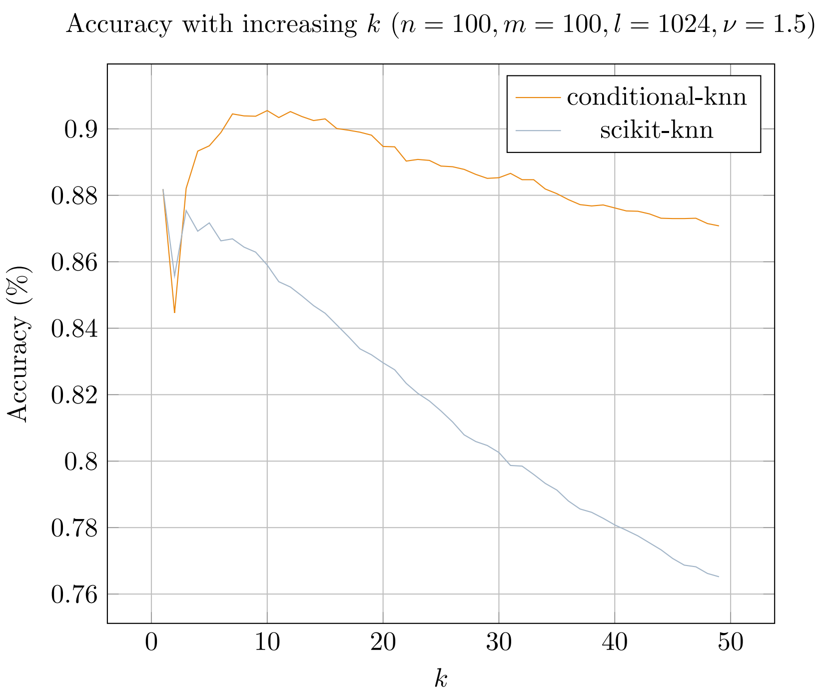

Fig. 6. *Accuracy with increasing k*

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. Recovery of sparse Cholesky factors. As noted in Appendix A.11, the selection algorithm can be viewed as orthogonal matching pursuit [29] in feature space. We experiment with the sparse recovery properties of the selection algorithm by randomly generating a sparse Cholesky factor L. We prescribe a fixed number of nonzeros per column s over N columns. For each column, we uniformly randomly pick s entry that satisfies lower triangularity to make nonzero. We randomly generate values according to i.i.d. standard normal $\mathcal{N}(0, 1)$. Finally, we fill the diagonal with a "large" positive value (10) to almost guarantee that the resulting matrix $\Theta = LL^\top$ is positive-definite. The selection algorithms are then given Θ and s and are asked to reconstruct L. The strategies are as follows: "cknn" uses conditional selection on each column to minimize the conditional variance of the diagonal entry, "knn" selects entries with the largest covariance with the diagonal entry, "corr" selects entries with the highest correlation objective (2.12) without accounting for conditioning, and "random" simply randomly samples entries uniformly. The strategies are given either the covariance Θ or the precision Θ^{-1} depending on which results in higher accuracy, in particular, the "cknn" strategy is given the precision while the rest of the methods are given the covariance. Accuracy is measured by taking the cardinality of the intersection of the recovered sparsity set with the ground truth sparsity set over the cardinality of their union, intersection over union (IOU).

As the number of nonzero entries per column is fixed and the number of rows and columns is increased, the "'cknn" retains high accuracy near perfect recovery, and the rest of the methods quickly degrade and asymptote to their final accuracies.

If the number of rows and columns is fixed while the number of nonzero entries per column is increased, all methods drop in accuracy with increasing density into a tipping point where the problem starts to become easier. Accuracy then increases until the Cholesky factor becomes fully dense, in which case perfect recovery is trivial. The "cknn" strategy exhibits the same behavior, but maintains much higher accuracy than the rest of the strategies.

Finally, we experiment with the addition of noise. Noise sampled i.i.d from $\mathcal{N}(0, \sigma^2)$ is added to each entry of Θ symmetrically (i.e. Θ_{ij} receives the same noise as Θ_{ji}) to preserve the symmetry of Θ. As expected, accuracy degrades with increasing noise, but the algorithm is fairly robust to low levels of noise. At higher levels of noise, Θ can lose positive-definiteness, which causes the algorithm to break down.

en: lapla- problems positive- teness and cholesky s

[16]

4.3. Cholesky factorization.

4.4. Gaussian process regression.

4.5. Preconditioning for conjugate gradient.

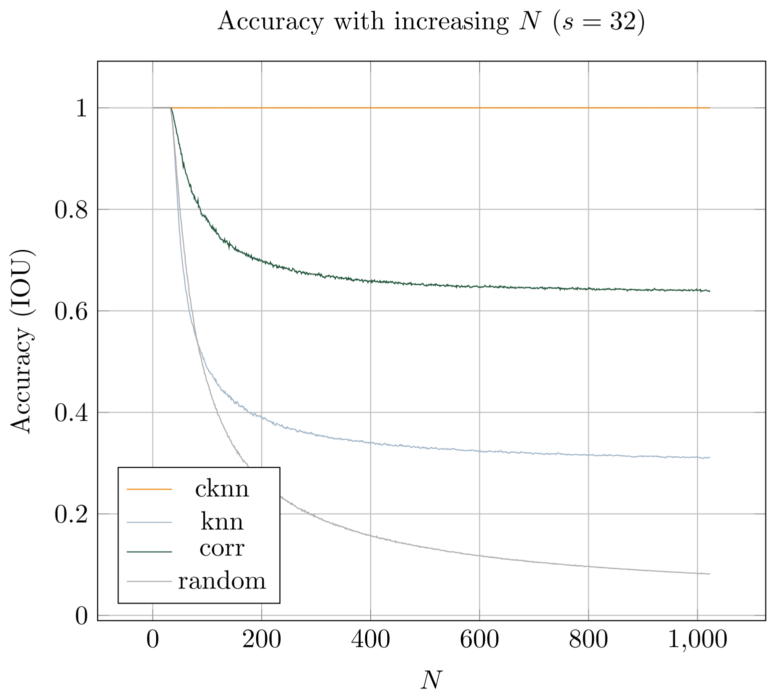

Fig. 7. *Accuracy with increasing N*

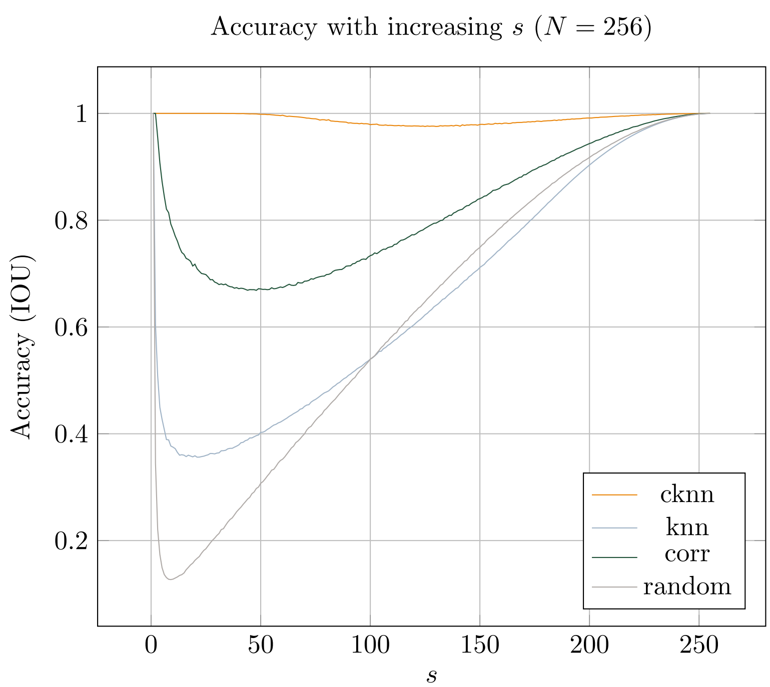

Fig. 8. *Accuracy with increasing s*

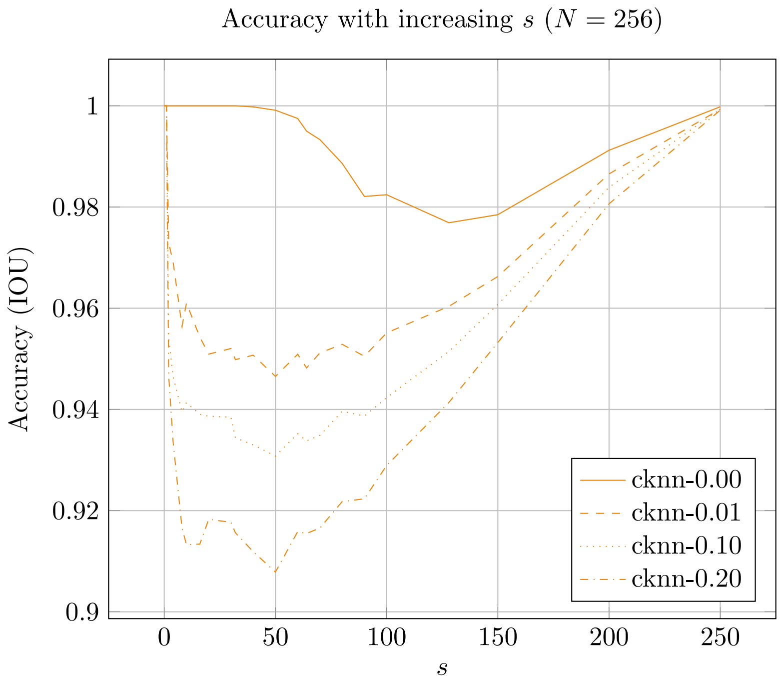

Fig. 9. *Accuracy with increasing s*

4.6. Comparison to other methods. against greedy gp information theoretic: most are trying to get some approximation for the entire process, *directed* greedy selection (towards a single point or group of points) (never mind, smola [25] is directed)

engineering for multiple points / nonadjacent case (conditioning structure)

omp: feature space versus covariance

previous Cholesky papers: conditional selection versus geometry

5. Conclusions.

nore funding
nation

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.

REFERENCES

[1] R. Baptista, V. Stolbunov, and P. B. Nair, *Some greedy algorithms for sparse polynomial chaos expansions*, Journal of Computational Physics, 387 (2019), pp. 303–325, https://doi.org/10.1016/j.jcp.2019.01.035.

[2] S. Bartels, K. Stensbo-Smidt, P. Moreno-Muñoz, W. Boomsma, J. Frellsen, and S. Hauberg, *Adaptive Cholesky Gaussian Processes*, arXiv:2202.10769 [cs], (2022), https://arxiv.org/abs/2202.10769.

[3] J. Chen, N. Cao, K. H. Low, R. Ouyang, C. K.-y. Tan, and P. Jaillet, *Parallel Gaussian Process Regression with Low-Rank Covariance Matrix Approximations*.

[4] E. Chow and A. Patel, *Fine-Grained Parallel Incomplete LU Factorization*, SIAM Journal on Scientific Computing, 37 (2015), pp. C169–C193, https://doi.org/10.1137/140968896.

[5] E. Clark, T. Askham, S. L. Brunton, and J. N. Kutz, *Greedy Sensor Placement with Cost Constraints*, arXiv:1805.03717 [math], (2018), https://arxiv.org/abs/1805.03717.

[6] A. Das and D. Kempe, *Submodular meets Spectral: Greedy Algorithms for Subset Selection, Sparse Approximation and Dictionary Selection*, arXiv:1102.3975 [cs, stat], (2011), https://arxiv.org/abs/1102.3975.

[7] J. Guinness, *Permutation and Grouping Methods for Sharpening Gaussian Process Approximations*, Technometrics, 60 (2018), pp. 415–429, https://doi.org/10.1080/00401706.2018.1437476, https://arxiv.org/abs/1609.05372.

[8] W. Hao, X. Jin, J. W. Siegel, and J. Xu, *An efficient greedy training algorithm for neural networks and applications in PDEs*, arXiv:2107.04466 [cs, math], (2021), https://arxiv.org/abs/2107.04466.

[9] R. Herbrich, N. Lawrence, and M. Seeger, *Fast Sparse Gaussian Process Methods: The Informative Vector Machine*, in Advances in Neural Information Processing Systems, vol. 15, MIT Press, 2002.

[10] J. Jagalur-Mohan and Y. Marzouk, *Batch greedy maximization of non-submodular functions: Guarantees and applications to experimental design*, arXiv:2006.04554 [cs, math], (2021), https://arxiv.org/abs/2006.04554.

[11] I. E. Kaporin, *An alternative approach to estimating the convergence rate of the CG method*, Numerical Methods and Software, Yu. A. Kuznetsov, ed., Dept. of Numerical Mathematics, USSR Academy of Sciences, Moscow, (1990), pp. 55–72.

[12] M. Katzfuss and J. Guinness, *A General Framework for Vecchia Approximations of Gaussian Processes*, Statistical Science, 36 (2021), pp. 124–141, https://doi.org/10.1214/19-STS755.

[13] M. Katzfuss and F. Schäfer, *Scalable Bayesian transport maps for high-dimensional non-Gaussian spatial fields*, Feb. 2022, https://arxiv.org/abs/2108.04211.

[14] 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.

[15] 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.

[16] R. Kyng and S. Sachdeva, *Approximate Gaussian Elimination for Laplacians: Fast, Sparse, and Simple*, arXiv:1605.02353 [cs], (2016), https://arxiv.org/abs/1605.02353.

[17] Y. Marzouk, T. Moselhy, M. Parno, and A. Spantini, *An introduction to sampling via measure transport*, arXiv:1602.05023 [math, stat], (2016), pp. 1–41, https://doi.org/10.1007/978-3-319-11259-6_23-1, https://arxiv.org/abs/1602.05023.

[18] M. Mutný and A. Krause, *Experimental Design for Linear Functionals in Reproducing Kernel Hilbert Spaces*, May 2022, https://arxiv.org/abs/2205.13627.

[19] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, Mass, 2006.

[20] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory and Applications*, Chapman and Hall/CRC, New York, Feb. 2005, https://doi.org/10.1201/9780203492024.

[21] 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.

[22] 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.

[23] M. Seeger and C. K. I. Williams, *Fast Forward Selection to Speed Up Sparse Gaussian Process Regression*, in In Workshop on AI and Statistics 9, 2003.

[24] J. W. Siegel and J. Xu, *Optimal Convergence Rates for the Orthogonal Greedy Algorithm*, arXiv:2106.15000 [cs, math, stat], (2022), https://arxiv.org/abs/2106.15000.

[25] A. Smola and P. Bartlett, *Sparse Greedy Gaussian Process Regression*, in Advances in Neural Information Processing Systems, vol. 13, MIT Press, 2000.

[26] A. Spantini, D. Bigoni, and Y. Marzouk, *Inference via low-dimensional couplings*, arXiv:1703.06131 [stat], (2018), https://arxiv.org/abs/1703.06131.

[27] M. L. Stein, *The screening effect in Kriging*, The Annals of Statistics, 30 (2002), pp. 298–323, https://doi.org/10.1214/aos/1015362194.

[28] M. L. Stein, *2010 Rietz lecture: When does the screening effect hold?*, The Annals of Statistics, 39 (2011), pp. 2795–2819, https://doi.org/10.1214/11-AOS909.

[29] 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.

[30] J. A. Tropp, A. C. Gilbert, and M. J. Strauss, *Algorithms for simultaneous sparse approximation. Part I: Greedy pursuit*, Signal Processing, 86 (2006), pp. 572–588, https:

//doi.org/10.1016/j.sigpro.2005.05.030.

[31] A. V. Vecchia, *Estimation and Model Identification for Continuous Spatial Processes*, Journal of the Royal Statistical Society: Series B (Methodological), 50 (1988), pp. 297–312, https://doi.org/10.1111/j.2517-6161.1988.tb01729.x.

[32] A. Y. Yeremin, L. Y. Kolotilina, and A. A. Nikishin, *Factorized sparse approximate inverse preconditionings. III. Iterative construction of preconditioners*, Journal of Mathematical Sciences, 101 (2000), pp. 3237–3254, https://doi.org/10.1007/BF02672769.

add proofs, if any, in appendix

Appendix A. Computation in sparse Gaussian process selection.

A.1. Updating precision after insertion. Assuming we have the precision of the selected entries, $\Theta_{I,I}^{-1}$, we wish to account for adding a new index k to I, that is, we wish to compute $\Theta_{I\cup\{k\},I\cup\{k\}}^{-1}$, adding a new row and column to $\Theta_{I,I}^{-1}$. In order to update efficiently, we block the matrix to separate new and old information.

Using the same block $LDL^\top$ factorization as (2.13),

(A.1)
$$\begin{pmatrix}\Theta_{1,1} & \Theta_{1,2}\\ \Theta_{2,1} & \Theta_{2,2}\end{pmatrix} = \begin{pmatrix}\mathrm{Id} & 0\\ \Theta_{2,1}\Theta_{1,1}^{-1} & \mathrm{Id}\end{pmatrix}\begin{pmatrix}\Theta_{1,1} & 0\\ 0 & \Theta_{2,2}-\Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}\end{pmatrix}\begin{pmatrix}\mathrm{Id} & \Theta_{1,1}^{-1}\Theta_{1,2}\\ 0 & \mathrm{Id}\end{pmatrix}$$

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

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

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

In the context of adding a new entry to the matrix, $\Theta_{1,1} = \Theta_{I,I}$, $\Theta_{1,2} = \Theta_{I,k}$, and $\Theta_{2,2} = \Theta_{k,k}$. Also note that $\Theta_{k,k|I}^{-1}$ is the precision of k conditional on the entries in I, which has already been computed in Algorithm 2.1. If we let $\boldsymbol{v} = \Theta_{I,I}^{-1}\Theta_{I,k}$, then

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

which is precisely the update in line 13 of Algorithm 2.1. Note that the bulk of the update is a rank-one update to $\Theta_{1,1}^{-1}$, which can be computed in $\mathcal{O}(|I|) = \mathcal{O}(s^2)$.

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 $\mathcal{O}(\#\ \mathrm{rows}(\Theta^{-1}))^2$.

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

A.3. Updating the log determinant after a rank-one downdate. Assuming we already have the log determinant of the covariance matrix of the prediction points conditional on the selected entries, $\operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I})$, we wish to compute the log determinant after we add an index k to I, that is, to compute $\operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I\cup\{k\}})$.

From (2.10), selecting a new point is a rank-one downdate on the covariance matrix.

$$\operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I\cup\{k\}}\right) = \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I} - \frac{\Theta_{\mathrm{Pr},k\mid I}\Theta_{\mathrm{Pr},k\mid I}^{\top}}{\Theta_{k,k\mid I}}\right) \tag{A.6}$$

Using the matrix determinant lemma,

$$= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I}\right) + \log\left(1 - \frac{\Theta_{\mathrm{Pr},k\mid I}^{\top}\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I}^{-1}\Theta_{\mathrm{Pr},k\mid I}}{\Theta_{k,k\mid I}}\right) \tag{A.7}$$

Focusing on the second term, we can turn the quadratic form into condtioning.

$$= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I}\right) + \log\left(\frac{\Theta_{k,k\mid I} - \Theta_{k,\mathrm{Pr}\mid I}\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I}^{-1}\Theta_{\mathrm{Pr},k\mid I}}{\Theta_{k,k\mid I}}\right) \tag{A.8}$$

By the quotient rule, we combine the conditioning.

$$= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I}\right) + \log\left(\frac{\Theta_{k,k\mid I,\mathrm{Pr}}}{\Theta_{k,k\mid I}}\right) \tag{A.9}$$

A.4. Updating precision after conditioning. Assuming we have the precision matrix of the prediction points conditional on the selected entries, $\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I}^{-1}$, we want to take into account selecting an index k, or to compute $\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I\cup\{k\}}^{-1}$, which is a rank-one update to the covariance (but not necessarily the precision) from (2.15). We can directly apply the Sherman–Morrison–Woodbury formula which states that:

$$\Theta_{1,1\mid 2}^{-1} = \Theta_{1,1}^{-1} + \left(\Theta_{1,1}^{-1}\Theta_{1,2}\right)\Theta_{2,2\mid 1}^{-1}\left(\Theta_{2,1}\Theta_{1,1}^{-1}\right) \tag{A.10}$$

Expanding the conditioning from (2.2),

$$\left(\Theta_{1,1} - \Theta_{1,2}\Theta_{2,2}^{-1}\Theta_{2,1}\right)^{-1} = \Theta_{1,1}^{-1} + \left(\Theta_{1,1}^{-1}\Theta_{1,2}\right)\Theta_{2,2\mid 1}^{-1}\left(\Theta_{2,1}\Theta_{1,1}^{-1}\right) \tag{A.11}$$

For brevity of notation, letting $\boldsymbol{u} = \Theta_{1,2}$ and $\boldsymbol{v} = \Theta_{1,1}^{-1}\Theta_{1,2} = \Theta_{1,1}^{-1}\boldsymbol{u}$,

$$(\Theta_{1,1} - \Theta_{2,2}^{-1}\boldsymbol{u}\boldsymbol{u}^{\top})^{-1} = \Theta_{1,1}^{-1} + \Theta_{2,2\mid 1}^{-1}\boldsymbol{v}\boldsymbol{v}^{\top} \tag{A.12}$$

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, $\Theta_{1,1} = \Theta_{\mathrm{Pr},\mathrm{Pr}\mid I}, \boldsymbol{u} = \Theta_{\mathrm{Pr},k\mid I}, \Theta_{2,2} = \Theta_{k,k\mid I}$ so $\Theta_{2,2\mid 1}^{-1} = \Theta_{k,k\mid \mathrm{Pr},I}^{-1}$ (this can be rigorously shown by expanding the Schur complement and taking advantage of the quotient rule as in (2.16)). $\boldsymbol{v}$ can be computed according to definition as $\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I}^{-1}\boldsymbol{u}$. Thus, we can write the update as

$$\left(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid I} - \frac{\Theta_{\mathrm{Pr},k\mid I}\Theta_{\mathrm{Pr},k\mid I}^{\top}}{\Theta_{kk\mid I}}\right)^{-1} = \Theta_{1,1}^{-1} + \Theta_{k,k\mid \mathrm{Pr},I}^{-1}\boldsymbol{v}\boldsymbol{v}^{\top} \tag{A.13}$$

which is the update in line 18 of Algorithm 2.3. Since the update is a rank-one update, it can be computed in $\mathcal{O}(|\mathrm{Pr}|^2) = \mathcal{O}(m^2)$.

en: remove
se unnec-
, describe
ion instead

A.5. Updating a Cholesky factor after a rank-one downdate. We use the approach from Lemma 1 of [15], 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, $L = \operatorname{chol}(\Theta)$. We wish to compute the updated Cholesky factor $L' = \operatorname{chol}(\Theta')$ where $\Theta' = \Theta - \boldsymbol{u}\boldsymbol{u}^\top$. To do so, assume L and L' are blocked according to the same block structure:

$$L = \begin{pmatrix} r_1 & \mathbf{0} \\ \boldsymbol{r}_2 & L_2 \end{pmatrix}, L' = \begin{pmatrix} r_1' & \mathbf{0} \\ \boldsymbol{r}_2' & L_2' \end{pmatrix} \tag{A.14}$$

Multiplying, we find

$$LL^\top = \Theta = \begin{pmatrix} r_1^2 & r_1\boldsymbol{r}_2^\top \\ r_1\boldsymbol{r}_2 & L_2L_2^\top + \boldsymbol{r}_2\boldsymbol{r}_2^\top \end{pmatrix} \tag{A.15}$$

$$L'L'^\top = \Theta' = \begin{pmatrix} r_1'^2 & r_1'\boldsymbol{r}_2'^\top \\ r_1'\boldsymbol{r}_2' & L_2'L_2'^\top + \boldsymbol{r}_2'\boldsymbol{r}_2'^\top \end{pmatrix} \tag{A.16}$$

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

$$r_1'^2 = \Theta_{11}' = \Theta_{11} - u_1^2 \tag{A.17}$$
$$= r_1^2 - u_1^2 \tag{A.18}$$
$$r_1' = \sqrt{r_1^2 - u_1^2} \tag{A.19}$$
$$r_1'\boldsymbol{r}_2' = \Theta_{2:,1}' = \Theta_{2:,1} - u_1\boldsymbol{u}_2 \tag{A.20}$$
$$= r_1\boldsymbol{r}_2 - u_1\boldsymbol{u}_2 \tag{A.21}$$
$$\boldsymbol{r}_2' = \frac{1}{r_1'}(r_1\boldsymbol{r}_2 - u_1\boldsymbol{u}_2) \tag{A.22}$$
$$L_2'L_2'^\top = L_2L_2^\top + \boldsymbol{r}_2\boldsymbol{r}_2^\top - \boldsymbol{u}_2\boldsymbol{u}_2^\top - \boldsymbol{r}_2'\boldsymbol{r}_2'^\top \tag{A.23}$$

Plugging in the expresion for $\boldsymbol{r}_2'$,

$$= L_2L_2^\top + \boldsymbol{r}_2\boldsymbol{r}_2^\top - \boldsymbol{u}_2\boldsymbol{u}_2^\top - \left(\frac{r_1}{r_1'}\boldsymbol{r_2} - \frac{u_1}{r_1'}\boldsymbol{u_2}\right)\left(\frac{r_1}{r_1'}\boldsymbol{r_2} - \frac{u_1}{r_1'}\boldsymbol{u_2}\right)^\top \tag{A.24}$$

$$= L_2L_2^\top + \left(1 - \frac{r_1^2}{r_1'^2}\right)\boldsymbol{r}_2\boldsymbol{r}_2^\top + \frac{r_1u_1}{r_1'^2}\boldsymbol{r_2}\boldsymbol{u_2}^\top + \frac{u_1r_1}{r_1'^2}\boldsymbol{u_2}\boldsymbol{r_2}^\top - \left(1 + \frac{u_1^2}{r_1'^2}\right)\boldsymbol{u}_2\boldsymbol{u}_2^\top \tag{A.25}$$

Using $r_1' = \sqrt{r_1^2 - u_1^2}$,

$$= L_2L_2^\top - \frac{u_1^2}{r_1'^2}\boldsymbol{r}_2\boldsymbol{r}_2^\top + \frac{r_1u_1}{r_1'^2}\boldsymbol{r_2}\boldsymbol{u_2}^\top + \frac{u_1r_1}{r_1'^2}\boldsymbol{u_2}\boldsymbol{r_2}^\top - \frac{r_1^2}{r_1'^2}\boldsymbol{u}_2\boldsymbol{u}_2^\top \tag{A.26}$$

After factoring we find

$$L_2'L_2'^\top = L_2L_2^\top - \left(\frac{r_1}{r_1'}\boldsymbol{u}_2 - \frac{u_1}{r_1'}\boldsymbol{r}_2\right)\left(\frac{r_1}{r_1'}\boldsymbol{u}_2 - \frac{u_1}{r_1'}\boldsymbol{r}_2\right)^\top \tag{A.27}$$

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:

$$r_1' = \sqrt{r_1^2 - u_1^2} \tag{A.28}$$

$$\boldsymbol{r}' = \frac{r_1}{r_1'}\boldsymbol{r} - \frac{u_1}{r_1'}\boldsymbol{u} \tag{A.29}$$

$$\boldsymbol{u}' = \frac{r_1}{r_1'}\boldsymbol{u} - \frac{u_1}{r_1'}\boldsymbol{r} \tag{A.30}$$

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 $\boldsymbol{r}'$,

$$\boldsymbol{r} = \frac{r_1'}{r_1}\boldsymbol{r}' + \frac{u_1}{r_1}\boldsymbol{u} \tag{A.31}$$

Plugging the expression for $\boldsymbol{r}$ into the update for $\boldsymbol{u}'$,

$$\boldsymbol{u}' = \frac{r_1}{r_1'}\boldsymbol{u} - \frac{u_1}{r_1'}\left(\frac{r_1'}{r_1}\boldsymbol{r}' + \frac{u_1}{r_1}\boldsymbol{u}\right) \tag{A.32}$$

$$= \frac{r_1^2 - u_1^2}{r_1 r_1'}\boldsymbol{u} - \frac{u_1}{r_1}\boldsymbol{r}' \tag{A.33}$$

$$= \frac{r_1'}{r_1}\boldsymbol{u} - \frac{u_1}{r_1}\boldsymbol{r}' \tag{A.34}$$

Thus, the updates proceed sequentially as follows:

$$\gamma \leftarrow \sqrt{r_1^2 - u_1^2} \tag{A.35}$$

$$\alpha \leftarrow \frac{r_1}{\gamma} \tag{A.36}$$

$$\beta \leftarrow \frac{u_1}{\gamma} \tag{A.37}$$

$$\boldsymbol{r} \leftarrow \alpha\boldsymbol{r} - \beta\boldsymbol{u} \tag{A.38}$$

$$\boldsymbol{u} \leftarrow \frac{1}{\alpha}\boldsymbol{u} - \frac{\beta}{\alpha}\boldsymbol{r} \tag{A.39}$$

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

A.6. Updating Cholesky factor after insertion. Suppose we have a cholesky factor L of Θ and we insert a new point into Θ. We wish to update the Cholesky L to account for this insertion. Using the recursive conditioning interpretation of Cholesky factorization in (2.14), we see that the columns of L before the insertion will remain unchanged, the column at the insertion point is a new column given by the conditional covariance of the new point with the rest of the points, conditional on the points before it, which can be computed with standard left-looking, and the columns of L after the insertion correspond to the Cholesky factor of the conditional covariance, conditional on the newly inserted point in addition to the previous points. From (2.10) we know that conditioning on an additional point is a rank-one update of the covariance, so we can use rank-one downdating from Appendix A.5 to update L for the columns after the insertion point, where the vector in the rank-one downdate is the newly inserted column. This update touches every value in the Cholesky factor exactly once, so its complexity is $\mathcal{O}(N^2)$ as opposed to the $\mathcal{O}(N^3)$ cost of completely regenerating the Cholesky factor from scratch.

A.7. 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,

$$\operatorname{Cov}[y_i, y_j] = \mathrm{E}[(y_i - \mathrm{E}[y_i])(y_j - \mathrm{E}[y_j])] \tag{A.40}$$

$$= \mathrm{E}[(L_i\boldsymbol{z})(L_{i|k}\boldsymbol{z})] \tag{A.41}$$

$$= \mathrm{E}[(L_{1,i}z_1 + \cdots + L_{N,i}z_N)(L_{1,j|k}z_1 + \cdots + L_{N,j|k}z_N)] \tag{A.42}$$

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

$$= \mathrm{E}[L_{1,i}L_{1,j|k}z_1^2 + \cdots + L_{N,i}L_{N,j|k}z_N^2] \tag{A.43}$$

$$= L_{1,i}L_{1,j|k}\,\mathrm{E}[z_1^2] + \cdots + L_{N,i}L_{N,j|k}\,\mathrm{E}[z_N^2] \tag{A.44}$$

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

$$= L_{1,i}L_{1,j|k} + \cdots + L_{N,i}L_{N,j|k} \tag{A.45}$$

$$= L_i \cdot L_{j|k} \tag{A.46}$$

Thus, the new covariance matrix can be written as:

$$\begin{pmatrix} L_{:j}L_{:j}^\top & L_{:j}L_{j:|k}^\top \\ L_{j:|k}L_{:j}^\top & L_{j:|k}L_{j:|k}^\top \end{pmatrix} = \begin{pmatrix} L_{:j} \\ L_{j:|k} \end{pmatrix} \begin{pmatrix} L_{:j} \\ L_{j:|k} \end{pmatrix}^\top \tag{A.47}$$

We will denote a partially conditioned matrix as

$$\Theta_{:,:,\shortmid k} \tag{A.48}$$

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.47) and make use of the fact that the determinant of a triangular matrix is the product of its diagonal entries:

$$\frac{1}{2}\operatorname{logdet}(\Theta_{:,:\shortmid k}) = \underbrace{\log(L_{11}) + \cdots + \log(L_{jj})}_{\text{the same}} + \underbrace{\log(L_{j+1,j+1|k}) + \cdots + \log(L_{M,M|k})}_{\text{conditioned}} \tag{A.49}$$

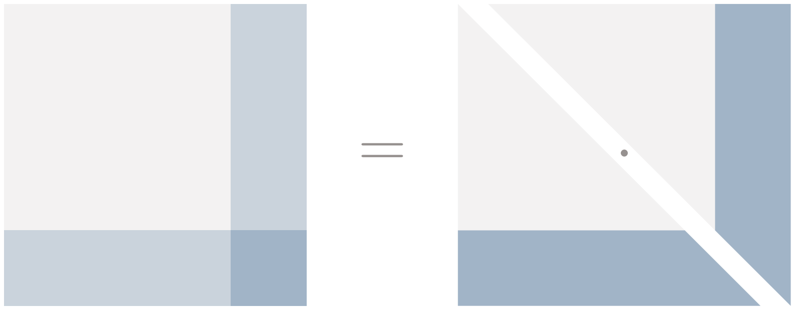

Fig. 10. *Illustration of the Cholesky factorization of a partially conditioned matrix.*

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

$$= \sum_{i=1}^{M} \log\left(\Theta_{ii|s_i-\{i\}}\right) \tag{A.50}$$

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

$$\begin{aligned} &= \underbrace{\log\left(\Theta_{11|s_1-\{1\}}\right) + \cdots + \log\left(\Theta_{jj|s_j-\{j\}}\right)}_{\text{the same}} + \\ &\underbrace{\log\left(\Theta_{j+1,j+1|s_{j+1}-\{j+1\}}\right) + \cdots + \log\left(\Theta_{MM|s_M-\{M\}}\right)}_{\text{conditioned}} \end{aligned} \tag{A.51}$$

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

$$2\log(L_{ii}) = \log(\Theta_{ii|s_i-\{i\}}) \tag{A.52}$$

So minimizing the log determinant of the partially conditioned covariance matrix (A.49) is the same as minimizing the KL divergence (3.4).

A.8. Algorithm for partial updates. We now need an efficient algorithm to keep track of partial updates. The key idea is to implicitly 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.

$$\Theta = \begin{pmatrix} \Theta_{xx} & \Theta_{xy} & \Theta_{x1} & \Theta_{x2} \\ \Theta_{yx} & \Theta_{yy} & \Theta_{y1} & \Theta_{y2} \\ \Theta_{1x} & \Theta_{1y} & \Theta_{11} & \Theta_{12} \\ \Theta_{2x} & \Theta_{2y} & \Theta_{21} & \Theta_{22} \end{pmatrix} \tag{A.53}$$

Computing the log determinant by chain rule,

$$\operatorname{logdet}(\Theta) = \log(\Theta_{xx}) + \log(\Theta_{yy|x}) + \log(\Theta_{11|x,y}) + \log(\Theta_{22|x,y,1}) \tag{A.54}$$

Isolating the objective — the variances of the prediction points

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

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

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

(A.57)
$$\Delta = \operatorname{logdet}(\Theta) - \log(\Theta_{yy|x}) - \operatorname{logdet}(\Theta_{-y,-y})$$

But from (2.16) we know

(A.58)
$$\Delta = \log\left(\frac{\Theta_{yy|x,1,2}}{\Theta_{yy|x}}\right)$$

Substituting,

(A.59)
$$\log(\Theta_{yy|x,1,2}) - \log(\Theta_{yy|x}) = \operatorname{logdet}(\Theta) - \log(\Theta_{yy|x}) - \operatorname{logdet}(\Theta_{-y,-y})$$

(A.60)
$$\log(\Theta_{yy|x,1,2}) = \operatorname{logdet}(\Theta) - \operatorname{logdet}(\Theta_{-y,-y})$$

In general,

(A.61)
$$\log(\Theta_{kk|I,\mathrm{Pr}}) = \operatorname{logdet}(\Theta) - \operatorname{logdet}(\Theta_{-k,-k})$$

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.

(A.62)
$$\Theta = \begin{pmatrix} \Theta_{11} & \Theta_{1y} & \Theta_{1x} & \Theta_{12} \\ \Theta_{y1} & \Theta_{yy} & \Theta_{yx} & \Theta_{y2} \\ \Theta_{x1} & \Theta_{xy} & \Theta_{xx} & \Theta_{x2} \\ \Theta_{21} & \Theta_{2y} & \Theta_{2x} & \Theta_{22} \end{pmatrix}$$

Computing the log determinant by chain rule,

(A.63)
$$\operatorname{logdet}(\Theta) = \log(\Theta_{11}) + \log(\Theta_{yy|1}) + \log(\Theta_{xx|1,y}) + \log(\Theta_{22|1,y,x})$$

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.

(A.64)
$$\log(\Theta_{11}) + \log(\Theta_{22|1,y,x}) = \operatorname{logdet}(\Theta) - \log(\Theta_{yy|1}) - \log(\Theta_{xx|1,y})$$

We can substitute $\log(\Theta_{yy|1,x,2})$ for $\operatorname{logdet}(\Theta)$ by (A.61). Athough it differs by a constant, this does not change the objective.

(A.65)
$$= \log(\Theta_{yy|1,x,2}) - \log(\Theta_{yy|1}) - \log(\Theta_{xx|1,y})$$

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.6 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. In addition, the algorithm can be considerably simplified by simply adding the conditional variances of the prediction points, instead of starting with a proxy for the log determinant of the entire matrix and subtracting out the spurious interactions from the training points.

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:

$$\Theta_{ii|:i-1} = (L_{ii})^2 \tag{A.66}$$

$$\Theta_{ij|:i-1} = L_{ij} \cdot L_{ii} \tag{A.67}$$

$$\Theta_{ii|:i-1,j} = \Theta_{ii|:i-1} - \frac{\Theta_{ij|:i-1}^2}{\Theta_{jj|:i-1}} \tag{A.68}$$

$$\Theta_{jj|:i-1,i} = \Theta_{jj|:i-1} - \frac{\Theta_{ij|:i-1}^2}{\Theta_{ii|:i-1}} \tag{A.69}$$

$$= \Theta_{jj|:i} \tag{A.70}$$

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.9. Global greedy selection. Although each column is essentially independent from the perspective of selection, if there is a prescribed budget for the number of nonzeros then there is the problem of distributing the nonzeros over the columns. A natural method is to distribute as evenly as possible, this is efficient and practically useful. However, one principled way of allocating nonzeros is to maintain a *global* priority queue over all columns, and selecting from this queue determines not only which entry out of the candidate set is added as a nonzero, but also which column to select from. This allows the algorithm to greedily select the next entry which will decrease the global objective (3.4) as much as possible. The main change is that within a column, any monotonic transformation of the objective will preserve the relative ranking of candidates, for example adding a constant, multiplying by a constant, taking the log, etc. However, from the global perspective, if one column adds a different constant to their objectives than another column, the relative ranking of candidates between columns is skewed. Thus, each column must maintain an objective that corresponds directly to minimizing the global objective (3.4). Here we describe the modifications that must be made to the selection algorithms to support global comparison.

A.9.1. Single column selection. For a single prediction point, the objective is $\frac{\Theta_{k,\mathrm{Pr}|I}^2}{\Theta_{kk|I}}$ (2.12) which is exactly the amount the variance of the prediction point is decreased if the kth candidate is selected, that is, $\Theta_{\mathrm{Pr},\mathrm{Pr}|I} - \Theta_{\mathrm{Pr},\mathrm{Pr}|I,k} = \frac{\Theta_{k,\mathrm{Pr}|I}^2}{\Theta_{kk|I}}$. From the global perspective, all other prediction points are untouched, so the amount the sum of the log variances of all the prediction points changes is

$$\min \Delta = \min \left[\log(\Theta_{\mathrm{Pr},\mathrm{Pr}|I,k}) - \log(\Theta_{\mathrm{Pr},\mathrm{Pr}|I})\right] \tag{A.71}$$

$$= \min \frac{\Theta_{\mathrm{Pr},\mathrm{Pr}|I,k}}{\Theta_{\mathrm{Pr},\mathrm{Pr}|I}} \tag{A.72}$$

$$= \min \frac{\Theta_{\mathrm{Pr},\mathrm{Pr}|I} - \frac{\Theta_{k,\mathrm{Pr}|I}^2}{\Theta_{kk|I}}}{\Theta_{\mathrm{Pr},\mathrm{Pr}|I}} \tag{A.73}$$

$$= \min \left[1 - \frac{\Theta_{k,\mathrm{Pr}|I}^2}{\Theta_{kk|I}\Theta_{\mathrm{Pr},\mathrm{Pr}|I}}\right] \tag{A.74}$$

$$= \max \frac{\Theta_{k,\mathrm{Pr}|I}^2}{\Theta_{kk|I}\Theta_{\mathrm{Pr},\mathrm{Pr}|I}} \tag{A.75}$$

where (A.75) can be interpreted as the *percentage* of the decrease in variance from selecting the kth point to the variance before selecting the point.

A.9.2. Aggregated selection. Since the nonadjacent algorithm directly computes the sum of the log of the conditional variances of the prediction points, few modifications have to be made. One heuristical improvement is to take into account for "bang-for-buck", that is, to account for the fact that different candidates cost a different amount of nonzero entries. Selecting a candidate can add between 1 and the number of columns in its aggregated group, depending on its relative index. Thus, candidates with larger groups will appear to decrease the global variance more, even if they are not as efficient as a candidate with a single group. In practice, it is better to use the objective $\frac{\Delta}{n}$ where Δ is the change in variance after selecting the candidate and n is the number of nonzero entries selecting the candidate adds.

A.10. Equivalence of Cholesky and QR factorization. We show a well-known fact that QR factorization can be viewed as the feature-space equivalent of Cholesky factorization, which can be viewed as operating in covariance-space.

Let Θ be a symmetric positive-definite matrix such that

$$\Theta = F^\top F \tag{A.76}$$

for some matrix F whose columns can be viewed as vectors in feature space:

$$\Theta_{ij} = \langle F_i, F_j \rangle \tag{A.77}$$

where F_i is the ith column of F. Now suppose F has the QR factorization

$$F = QR \tag{A.78}$$

where Q is a $N \times N$ orthonormal matrix and R is a $N \times N$ upper triangular matrix.

$$\Theta = F^\top F = (QR)^\top (QR) \tag{A.79}$$

$$= R^\top Q^\top QR \tag{A.80}$$

$$= R^\top R \tag{A.81}$$

from the orthogonality of Q. But note that R is an upper triangular matrix, so $L = R^\top$ is an lower triangular matrix. So we have $\Theta = LL^\top$ for lower triangular L. By the uniqueness of Cholesky factorization, $R^\top$ is precisely the Cholesky factor of Θ. In addition, the columns of Q are formed from Gram-Schmitt orthogonalization on the columns of F (in feature-space), and R the coefficients resulting from the Gram-Schmitt procedure. From $R^\top = \operatorname{chol}(\Theta)$ and the statistical interpretation of Cholesky factorization (2.14), this iterative orthogonalization in feature-space is equivalent to iterative conditioning in covariance.

A.11. Equivalence of selection and orthogonal matching pursuit. We show that the single-point selection algorithm described in Algorithm 2.2 is the covariance space equivalent to the feature space orthogonal matching pursuit (OMP) algorithm described in [29]. 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

$$\Theta = F^\top F \tag{A.82}$$

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

$$F = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \dots & \boldsymbol{x}_N \end{pmatrix} \tag{A.83}$$

Immediately we have

$$\Theta_{ij} = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \tag{A.84}$$

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 $\boldsymbol{x}_1$,

$$\Theta' = \Theta - \frac{\boldsymbol{x}_1 \boldsymbol{x}_1^\top}{\Theta_{11}} \tag{A.85}$$

$$\Theta'_{ij} = \Theta_{ij} - \frac{\Theta_{i1}\Theta_{j1}}{\Theta_{ii}} \tag{A.86}$$

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

$$\boldsymbol{x}'_i = \boldsymbol{x}_i - \frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_1 \rangle}{\langle \boldsymbol{x}_1, \boldsymbol{x}_1 \rangle} \boldsymbol{x}_1 \tag{A.87}$$

$$\langle \boldsymbol{x}'_i, \boldsymbol{x}'_j \rangle = \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle - \frac{\langle \boldsymbol{x}_i, \boldsymbol{x}_1 \rangle \langle \boldsymbol{x}_j, \boldsymbol{x}_1 \rangle}{\langle \boldsymbol{x}_1, \boldsymbol{x}_1 \rangle} \tag{A.88}$$

Comparing (A.86) and (A.88), 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

$$\|\boldsymbol{x}_{\mathrm{Pr}} - \operatorname{proj}_{\boldsymbol{x}_k} \boldsymbol{x}_{\mathrm{Pr}}\| = \langle \boldsymbol{x}_{\mathrm{Pr}}, \boldsymbol{x}_{\mathrm{Pr}} \rangle - \frac{\langle \boldsymbol{x}_{\mathrm{Pr}}, \boldsymbol{x}_k \rangle^2}{\langle \boldsymbol{x}_k, \boldsymbol{x}_k \rangle} \tag{A.89}$$

which is precisely the squared covariance of the candidate with the prediction over the variance of the candidate, as in (2.12). This shows the equivalence as the objective is the same.

A.12. Checking submodularity. Our objective is the mutual information between the prediction and training points (2.6). A natural question is whether this objective is submodular with respect to the training set. The answer is no in general, see [14], section 8.3 for a simple counterexample. However, we can empirically check submodularity for particular geometries and choices of kernel function. If Pr is the set of prediction points, I is a set of indexes, and x_1, x_2 are additional indices not in I, then the set function is submodular if and only if

$$\mathrm{MI}(\mathrm{Pr}, I \cup \{x_2\}) - \mathrm{MI}(I) \overset{?}{\geq} \mathrm{MI}(\mathrm{Pr}, I \cup \{x_1, x_2\}) - \mathrm{MI}(\mathrm{Pr}, I \cup \{x_1\})$$

Expanding from the definition of mutual information (2.6),

$$\mathrm{H}[\mathrm{Pr} \mid I] - \mathrm{H}[\mathrm{Pr} \mid I \cup \{x_2\}] \overset{?}{\geq} \mathrm{H}[\mathrm{Pr} \mid I \cup \{x_1\}] - \mathrm{H}[\mathrm{Pr} \mid I \cup \{x_1, x_2\}]$$

Since this is the objective (2.15) (with additional log),

$$\frac{\Theta_{x_2,x_2|I}}{\Theta_{x_2,x_2|I,\mathrm{Pr}}} \overset{?}{\geq} \frac{\Theta_{x_2,x_2|I} - \frac{\Theta^2_{x_1,x_2|I}}{\Theta_{x_1,x_1|I}}}{\Theta_{x_2,x_2|I,\mathrm{Pr}} - \frac{\Theta^2_{x_1,x_2|I,\mathrm{Pr}}}{\Theta_{x_1,x_1|I,\mathrm{Pr}}}}$$

From the fact that $\frac{a}{b} \geq \frac{a-c}{b-d}$ if and only if $\frac{a}{b} \leq \frac{c}{d}$,

$$\frac{\Theta_{x_2,x_2|I}}{\Theta_{x_2,x_2|I,\mathrm{Pr}}} \overset{?}{\leq} \frac{\frac{\Theta^2_{x_1,x_2|I}}{\Theta_{x_1,x_1|I}}}{\frac{\Theta^2_{x_1,x_2|I,\mathrm{Pr}}}{\Theta_{x_1,x_1|I,\mathrm{Pr}}}}$$

Multiplying by $\frac{\Theta_{x_1,x_1|I}}{\Theta_{x_1,x_1|I,\mathrm{Pr}}}$ on both sides,

$$\frac{\Theta_{x_1,x_1|I}\Theta_{x_2,x_2|I}}{\Theta_{x_1,x_1|I,\mathrm{Pr}}\Theta_{x_2,x_2|I,\mathrm{Pr}}} \overset{?}{\leq} \frac{\Theta^2_{x_1,x_2|I}}{\Theta^2_{x_1,x_2|I,\mathrm{Pr}}}$$

Multiplying by $\frac{\Theta^2_{x_1,x_2|I,\mathrm{Pr}}}{\Theta_{x_1,x_1|I}\Theta_{x_2,x_2|I}}$ on both sides,

$$\frac{\Theta^2_{x_1,x_2|I,\mathrm{Pr}}}{\Theta_{x_1,x_1|I,\mathrm{Pr}}\Theta_{x_2,x_2|I,\mathrm{Pr}}} \overset{?}{\leq} \frac{\Theta^2_{x_1,x_2|I}}{\Theta_{x_1,x_1|I}\Theta_{x_2,x_2|I}}$$

By definition, this is

$$\mathrm{Corr}[x_1, x_2 \mid I, \mathrm{Pr}] \overset{?}{\leq} \mathrm{Corr}[x_1, x_2 \mid I]$$

so the mutual information objective is submodular if and only if conditioning on additional point(s) decreases the correlation between every pair of points. Intuitively, this corresponds to the screening effect observed in spatial statistics literature — conditioning on a nearby point decreases the correlation for far away points.

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

(B.1)

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

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

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

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.

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

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

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

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

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

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),

$$2\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0}, \Theta) \,\middle\|\, \mathcal{N}(\mathbf{0}, (LL^\top)^{-1})\right) = \operatorname{trace}(LL^\top\Theta) - \operatorname{logdet}(LL^\top) - \operatorname{logdet}(\Theta) - N \tag{B.16}$$

Ignoring terms not depending on L,

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

By the cyclic property of trace,

$$= \operatorname{trace}(L\Theta L^\top) - \operatorname{logdet}(LL^\top) \tag{B.18}$$

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

$$\operatorname{trace}(L\Theta L^{\top}) = \sum_{i=1}^{N} L_{s_i,i}^{\top}\Theta_{s_i,s_i}L_{s_i,i} \tag{B.19}$$

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

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

$$= \sum_{i=1}^{N} \frac{\boldsymbol{e}_1^{\top}\Theta_{s_i,s_i}^{-1}\Theta_{s_i,s_i}\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}{\boldsymbol{e}_1^{\top}\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1} \tag{B.21}$$

$$= \sum_{i=1}^{N} 1 = N \tag{B.22}$$

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

$$2\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0},\Theta)\,\middle\|\,\mathcal{N}(\mathbf{0},(LL^{\top})^{-1})\right) = -\operatorname{logdet}(LL^{\top}) - \operatorname{logdet}(\Theta) \tag{B.23}$$

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

$$= -2\sum_{i=1}^{N}\left[\log(L_{ii})\right] - \operatorname{logdet}(\Theta) \tag{B.24}$$

Plugging (3.3) for the diagonal entries,

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

Bringing the negative inside,

$$= \sum_{i=1}^{N}\left[\log\left((\boldsymbol{e}_1^{\top}\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1)^{-1}\right)\right] - \operatorname{logdet}(\Theta) \tag{B.26}$$

So minimizing the KL divergence (given optimal L) corresponds to minimizing the sum of the inverse of the diagonal entries. We can give an intuitive view of this result by making use of (3.7) and expanding the log determinant by the chain rule (3.8).

$$\sum_{i=1}^{N}\left[\log\left((\boldsymbol{e}_1^{\top}\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1)^{-1}\right)\right] - \operatorname{logdet}(\Theta) = \sum_{i=1}^{N}\left[\log\left(\Theta_{ii\mid s_i-\{i\}}\right)\right] - \operatorname{logdet}(\Theta) \tag{B.27}$$

$$= \sum_{i=1}^{N}\log\left(\Theta_{ii\mid s_i-\{i\}}\right) - \sum_{i=1}^{N}\log\left(\Theta_{ii\mid i+1:}\right) \tag{B.28}$$

$$= \sum_{i=1}^{N}\left[\log\left(\Theta_{ii\mid s_i-\{i\}}\right) - \log\left(\Theta_{ii\mid i+1:}\right)\right] \tag{B.29}$$

We can view this sum as the accumulated *difference* in prediction error for a series of prediction problems, where each prediction problem is to predict the value of the ith variable given variables $i+1, i+2, \ldots, N$. The left term $\log\left(\Theta_{ii\mid s_i-\{i\}}\right)$ is restricted to only using those variables in the sparsity pattern s_i, while the right term $\log\left(\Theta_{ii|i+1:}\right)$ is able to use every variable after i. The left term will necessarily have greater variance than the right, and the goal is to minimize the accumulated deviation. Thus, the KL divergence 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. This interpretation is also given in [13].

Another interpretation is from [2], where they view the full prediction problems as the log likelihood of the variables. Under this interpretation, conditional independence (through the screening effect) corresponds to a near-constant value of $\log(\Theta_{ii|i+1:})$, which results in a linear plot of log-likelihood with increasing N.

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

$$\mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0},\Theta)\,\middle\|\,\mathcal{N}(\mathbf{0},(LL^\top)^{-1})\right) = \mathbb{D}_{\mathrm{KL}}\left(\mathcal{N}(\mathbf{0},LL^\top)\,\middle\|\,\mathcal{N}(\mathbf{0},\Theta^{-1})\right) \tag{B.30}$$

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.

Appendix C. Algorithms.

Algorithm C.1 Direct Gaussian process regression 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 2.1 or Algorithm 2.2
2: $\Theta_{\mathrm{Tr,Tr}} \leftarrow K(\boldsymbol{x}_{\mathrm{Tr}}[I], \boldsymbol{x}_{\mathrm{Tr}}[I])$
3: $\Theta_{\mathrm{Pr,Pr}} \leftarrow K(\boldsymbol{x}_{\mathrm{Pr}}, \boldsymbol{x}_{\mathrm{Pr}})$
4: $\Theta_{\mathrm{Tr,Pr}} \leftarrow K(\boldsymbol{x}_{\mathrm{Tr}}[I], \boldsymbol{x}_{\mathrm{Pr}})$
5: $\mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \leftarrow \Theta_{\mathrm{Pr,Tr}}\Theta_{\mathrm{Tr,Tr}}^{-1}\boldsymbol{y}_{\mathrm{Tr}}[I]$
6: $\mathrm{Cov}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \leftarrow \Theta_{\mathrm{Pr,Pr}} - \Theta_{\mathrm{Tr,Pr}}^\top\Theta_{\mathrm{Tr,Tr}}^{-1}\Theta_{\mathrm{Tr,Pr}}$
7: **return** $\mathrm{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}], \mathrm{Cov}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]$