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 of such matrices by minimizing Kullback-Leibler divergence recovers the Vecchia approximation for Gaussian processes. However, these methods often rely 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 that maximizes mutual information with target points, conditional on all points previously selected. 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 aggregated 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. By minimizing Kullback-Leibler divergence, we apply the algorithm to Cholesky factorization, Gaussian process regression, and preconditioning with the conjugate gradient, improving over k-nearest neighbors particularly in high dimensional, unusual, or otherwise messy geometries.

Key words.

AMS subject classifications.

to do

1. Introduction.

The problem. Gaussian processes enjoy widespread application in spatial statistics and geostatistics [26], machine learning through kernel methods [25], optimal experimental design [21], and sensor placement [15]. However, Gaussian process statistics from N data points requires computing with the covariance matrix $\Theta \in \mathbb{R}^{N\times N}$ to obtain quantities such as $\Theta \boldsymbol{v}$, $\Theta^{-1}\boldsymbol{v}$, $\operatorname{logdet}(\Theta)$. For dense Θ, directly computing these quantities has a computational cost of $\mathcal{O}(N^3)$ and a memory cost of $\mathcal{O}(N^2)$, which is prohibitively expensive for large N. Beyond Gaussian processes, computations with large positive-definite matrices are required across computational mathematics, motivating the search for faster, approximate algorithms.

Existing work. TODO

Vecchia approximation. The Vecchia approximation [37] decomposes the joint distribution into the product of univariate densities, each density conditional only on a subset of those prior in the ordering. If an ordering and sparsity pattern are fixed, then the entries of the resulting sparse approximate Cholesky factor can be computed by optimizing a chosen objective, for example, minimizing the Kullback-Leibler (KL) divergence [27] is a natural objective for Gaussian process regression. Independently of the Vecchia approximation, within the context of factorized sparse approximate inverse (FSAI) preconditioners, the Kaporin condition number [12] and the Frobenius norm with an additional Jacobi scaling constraint [41] have also been proposed. These

I wonder if we should change the title to sparse inverse-cholesky factors, or something on that line?

*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

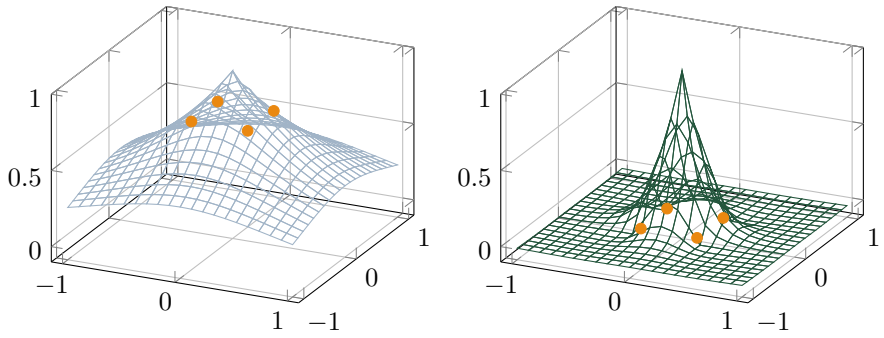

FIG. 1. *An illustration of the screening effect with the Matérn kernel with length scale $\ell = 1$ and smoothness $\nu = 1/2$. The first panel shows the unconditional correlation with the point at (0, 0). The second panel shows the conditional correlation after conditioning on the four points in orange.*

three objectives actually converge to the same closed-form expression for the entries of the resulting Cholesky factor, equivalent to one used to compute the Vecchia approximation [27]. In addition, the KL divergence is used to compute Knothe-Rosenblatt transport maps [20], generalizing Cholesky factors to non-Gaussian distributions while preserving triangularity and sparsity [31] and often applied to simulation and sampling problems [20, 14]. Finally, exploiting the independence of the Vecchia approximation allows for embarrassingly parallel algorithms for regression and factorization.

Ordering and sparsity selection by geometry. The chosen ordering and sparsity pattern significantly affect the quality of the resulting factor. Vecchia originally proposed ordering points lexicographically [37], but more recent work exploits space-covering orderings like the maximum-minimum ordering [7] which has become popular [28, 27, 13, 11, 14] and has rigorous approximation guarantees from results in numerical homogenization and operator-adapted wavelets [22, 28]. After choosing an ordering, following Vecchia's recommendation the sparsity set is often formed by selecting the closest points by Euclidean distance [37, 28, 27, 14]. This choice is often justified by the *screening effect*, or the observation that conditional on points near the point of interest, far away points are almost conditionally independent for many popular kernel functions [32, 33] (see Figure 1). Methods exploiting the screening effect achieve state of the art ϵ-accuracy in time complexity $\mathcal{O}(N \log^{2d}(N/\epsilon))$ using $\mathcal{O}(N \log^{d}(N/\epsilon))$ nonzero entries of the covariance matrix [27]. However, selecting by distance ignores the conditional effect of adding points to the sparsity set. Imagine that the closest point to the point of interest is duplicated multiple times. Conditional on the original point, the duplicates provide no additional information. But these redundant points are still just as close to the target, so they are still added.

Conditional selection. Instead of adding points to the sparsity pattern by distance, we propose greedily selecting points that maximize mutual information with the point of interest, conditional on all points previously selected. The machine learning community has long developed similar algorithms that greedily optimize information-theoretic objectives in the context of sparse Gaussian process inference [30, 9, 29]. Similar algorithms have also been developed in the context of sensor placement [15, 3] and experimental design [21] where it is assumed the target phenomena is modeled by a Gaussian process or is otherwise linearly dependent on the selected measurements. However, these works often focus on global approximation of the entire process, e.g. through sparse approximation of the likelihood or covariance matrix [19, 2, 24]. In contrast [40] uses inference *directed* towards a point of interest, selecting the active

ld add a
k here hat
ents on the
onship to
scale analysis
vavelets

l mention the
us results, as

sentence
d contain
on of the
tioning

hould start
something
In this work
opose a con-
al..." or so.
is, first make
r that ap-
g conditional
ion is the
contribu-
f the present
and then re-
to all the
approaches
niques.

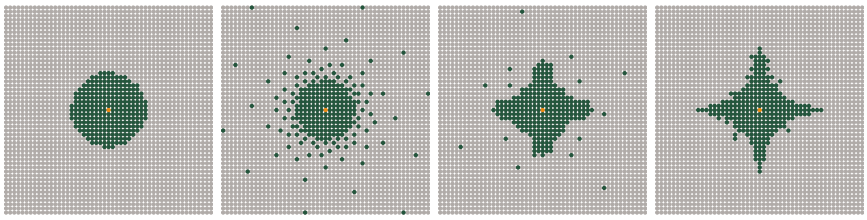

FIG. 2. *The first panel shows selecting the k-nearest neighbors to the center point out of a dense grid of candidates. The next panels show selection by greedily maximizing conditional mutual information with the center point for a Matérn kernel with length scale $\ell = 1$ and increasing smoothness ν, from right to left: $\nu = 1/2, 3/2, 5/2$. The full selections are found at https://youtu.be/lyJf3S5ThjQ.*

(sparsity) set by the kernel function itself like the later work [11]; [5] and the follow-up work [6] use the more sophisticated active learning Cohn (ALC) objective, yielding an algorithm equivalent to ours for a single point of interest. Our proposed algorithm can also be viewed as a variant of orthogonal matching pursuit (OMP) [35, 36], a workhorse algorithm in compressive sensing which seeks to approximate a target signal as the sparse linear combination from a given collection of signals.

Main results. Our main contribution is a selection algorithm that greedily maximizes mutual information with point(s) of interest, conditional on all points previously selected. We use this algorithm to select the sparsity pattern of sparse approximate Cholesky factors of precision matrices in the KL-minimization framework of [27], yielding more accurate factors at the same density compared to nearest neighbors. The final algorithm extends kernel-based selection [40, 11] to account for conditioning and extends directed Gaussian process regression [5, 6] to multiple points as well as global approximation. For a single target point, direct computation of the mutual information criterion would have time complexity $\mathcal{O}(Nk^4)$ to select k points out of N, but by maintaining a partial Cholesky factor we reduce the complexity to $\mathcal{O}(Nk^2)$. We extend the algorithm to maximize mutual information with *multiple* targets, naturally taking 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-target algorithm. In the setting of aggregated (or supernodal) Cholesky factorization where the sparsity patterns of multiple columns are determined simultaneously, a candidate entry may only condition a *subset* of the targets. By efficient rank-one downdating of Cholesky factors, we capture this structure at the same time complexity for multiple targets. Finally, we show how to adaptively determine the number of nonzeros per column in order to minimize the overall KL divergence by maintaining a global priority queue shared between all columns.

Want to start a one-sentence description of our contributions, something like "We propose...". We should first fully describe what we are doing (greedy selection, integration into the KL framework etc.). We want to start talking about technical things like the computational costs and the improvement only afterward, once we have established what it is that we are accelerating.

Outline. This paper is organized as follows. In section 2, we show how minimizing KL divergence to compute sparse Cholesky factors reduces to solving independent regression problems. In section 3, we develop greedy algorithms to select the sparsity pattern independently for each regression problem. In section 4, we combine the greedy selection algorithm with KL minimization to yield algorithms for sparse Cholesky factorization. In section 5 we extend these results to adjacent and non-adjacent aggregated factorization. In section 6, we present numerical experiments applying our method to image classification, recovery of *a priori* sparse Cholesky factors, Cholesky factorization, Gaussian process regression, and preconditioning with the conjugate gradient. In section 7, we summarize our results. Proofs and algorithmic details are provided in the appendix and supplementary material.

2. Sparse Cholesky factorization by KL-minimization. Let $\Theta \in \mathbb{R}^{N\times N}$ be a symmetric positive-definite matrix; we view Θ as the covariance matrix of a Gaussian process. We say a function $f(\boldsymbol{x})$ is distributed according to a Gaussian process prior with mean function $\mu(\boldsymbol{x})$ and covariance function or kernel function $K(\boldsymbol{x},\boldsymbol{x}')$, which we will denote as $f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), K(\boldsymbol{x},\boldsymbol{x}'))$, 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 $\Theta_{i,j} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

In many applications of Gaussian processes, we wish to infer unknown data given known data. 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 $X_{\mathrm{Tr}} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N]^\top \in \mathbb{R}^{N\times D}$ and the measurements at those points are collected in the vector $\boldsymbol{y}_{\mathrm{Tr}} = [y_1, \ldots, y_N]^\top \in \mathbb{R}^N$, we wish to predict the values at m new points $X_{\mathrm{Pr}} \in \mathbb{R}^{m\times D}$ for which $\boldsymbol{y}_{\mathrm{Pr}} \in \mathbb{R}^m$ is unknown. We assume the function $f(\boldsymbol{x})$ that maps input points to their outputs is distributed according to a Gaussian process with zero mean function, $f(\boldsymbol{x}) \sim \mathcal{GP}(\boldsymbol{0}, K(\boldsymbol{x},\boldsymbol{x}'))$. From the distribution of $f(\boldsymbol{x})$, the joint distribution of training and testing data $\boldsymbol{y}$ has covariance $\Theta = \begin{pmatrix} \Theta_{\mathrm{Tr,Tr}} & \Theta_{\mathrm{Tr,Pr}} \\ \Theta_{\mathrm{Pr,Tr}} & \Theta_{\mathrm{Pr,Pr}} \end{pmatrix}$ where $\Theta_{I,J} \coloneqq K(X_I, X_J)$ for index sets I, J. In order to make predictions at X_{Pr}, we condition the desired prediction $\boldsymbol{y}_{\mathrm{Pr}}$ on the known data $\boldsymbol{y}_{\mathrm{Tr}}$. For Gaussian processes, the closed-form posterior distribution is

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

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

where $\Theta_{\mathrm{Tr,Tr}}^{-1} \coloneqq (\Theta_{\mathrm{Tr,Tr}})^{-1}$. To denote the covariance between the variables in index sets I and J, conditional on the variables in the index sets $V_1, V_2, \ldots, V_n$ we write

$$\Theta_{I,J|V_1,V_2,\ldots,V_n} \coloneqq \mathbb{C}\mathrm{ov}[\boldsymbol{y}_I, \boldsymbol{y}_J \mid \boldsymbol{y}_{V_1\cup V_2\cup\cdots\cup V_n}] \tag{2.3}$$

Although the final covariance matrix is invariant to the written order of $V_1, \ldots, V_n$, a different order of conditioning means different intermediate results in repeated application of (2.2). Letting $W = \bigcup_{i=1}^{n-1} V_i$, by the quotient rule of Schur complements:

$$\Theta_{I,J|V_1,\ldots,V_n} = \Theta_{I,J|W} - \Theta_{I,V_n|W}\Theta_{V_n,V_n|W}^{-1}\Theta_{V_n,J|W} \tag{2.4}$$

Calculating the posterior mean (2.1) and covariance (2.2) requires inverting the training covariance matrix. Often the Cholesky factorization is used instead of the direct inverse for improved numerical stability and performance. However, the time complexity of computing the Cholesky factorization is $\mathcal{O}(N^3)$, which is prohibitive for large N. Instead, we will enforce that L is *sparse*, yielding an approximate factor.

2.1. Vecchia approximation. Sparse factors naturally arise from the Vecchia approximation for Gaussian processes [37]. Decomposing the joint likelihood π,

$$\pi(\boldsymbol{y}) = \pi(y_1)\,\pi(y_2 \mid y_1)\,\pi(y_3 \mid y_1, y_2)\cdots\pi(y_N \mid y_1, y_2, \ldots, y_{N-1}) \tag{2.5}$$

The key assumption is that many of the points are redundant after conditioning on a carefully chosen subset of the points. Letting $i_1, \ldots, i_N$ denote an ordering of the points and s_k the indices of points that condition the kth point in the ordering, the Vecchia approximation proposes to approximate (2.5) by the sparse approximation

$$\pi(\boldsymbol{y}) \approx \pi(y_{i_1})\,\pi(y_{i_2} \mid y_{s_2})\,\pi(y_{i_3} \mid y_{s_3})\cdots\pi(y_{i_N} \mid y_{s_N}) \tag{2.6}$$

Following (2.6) if an elimination ordering $\prec$ is fixed (a permutation of $\{1, \ldots, N\}$) and a lower triangular sparsity pattern $S \coloneqq \{(i,j) : i \in s_j, i \succeq j\}$ is specified then all is needed is a functional criterion $\mathcal{L} : \mathbb{R}^{N\times N} \to \mathbb{R}$ to specify the optimization problem

$$L \coloneqq \operatorname*{argmin}_{\hat{L}\in\mathcal{S}} \mathcal{L}(\hat{L}) \tag{2.7}$$

where $\mathcal{S} \coloneqq \{M \in \mathbb{R}^{N\times N} : M_{ij} \neq 0 \Rightarrow (i,j) \in S\}$ is the space of matrices satisfying the sparsity pattern S. Proposed functionals which compute inverse Cholesky factors L of the covariance matrix, i.e. $L\,\mathrm{chol}(\Theta) \approx \mathrm{Id}$, include the Kaporin condition number $(\mathrm{trace}(L\Theta L^\top)/N)^N/\det(L\Theta L^\top)$ [12] and the Frobenius norm $\|\mathrm{Id} - L\,\mathrm{chol}(\Theta)\|_{\mathrm{FRO}}$ additionally subject to the constraint $\mathrm{diag}(L\Theta L^\top) = 1$ [41], while the KL divergence $\mathbb{D}_{\mathrm{KL}}(\mathcal{N}(\mathbf{0},\Theta) \,\|\, \mathcal{N}(\mathbf{0},(LL^\top)^{-1}))$ [27] factors the precision matrix, i.e. $LL^\top \approx \Theta^{-1}$.

As observed in [27], factors of the precision matrix are often much sparser than factors of the covariance matrix, because the precision encodes conditional independence while the covariance encodes marginal independence. The same phenomenon is observed by [31] working with the more general transport maps. Covariance matrices arising from kernel functions are often fully dense, but the approximate factors of their precisions can be sparse if their ordering and sparsity pattern are chosen carefully.

2.2. Ordering and sparsity pattern. Although in this work we primarily focus on constructing sparsity patterns, the chosen ordering critically affects the accuracy of lower triangular sparsity patterns. Vecchia originally proposed ordering points lexicographically, which is most natural in a one-dimensional setting [37]. More recent work finds that in higher dimensions, exploiting space-covering orderings leads to significantly better approximation quality [7]. Specifically, we use the maximum-minimum (maximin) ordering [7], which has become popular for the Vecchia approximation [13] and Cholesky factorization [28, 27, 11, 14]. The reverse-maximin ordering $i_1,\ldots,i_N$ on a set of N points $\{\boldsymbol{x}_i\}_{i=1}^N$ is defined by first selecting the last index i_N arbitrarily and then choosing for $k = N-1, N-2, \ldots, 1$ the index

$$i_k = \operatorname*{argmax}_{i \in -\mathcal{I}_{k+1}} \min_{j \in \mathcal{I}_{k+1}} \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \tag{2.8}$$

where $-\mathcal{I} \coloneqq \{1,\ldots,N\} \setminus \mathcal{I}$ and $\mathcal{I}_n \coloneqq \{i_n, i_{n+1}, \ldots, i_N\}$, i.e. select the point farthest from previously selected points. The ordering is reversed for factorizing the precision. We write $i \prec j$ if i precedes j in the ordering and define $\ell_{i_k} \coloneqq \min_{j\in\mathcal{I}_{k+1}} \|\boldsymbol{x}_{i_k} - \boldsymbol{x}_j\|$, a length scale monotonically shrinking with decreasing position in the ordering.

Vecchia also originally proposed to select the sparsity set by Euclidean distance [37], which, unlike the lexicographic ordering, still remains widely used [28, 27, 14]. Instead, we will select the sparsity pattern to directly optimize the accuracy $\mathcal{L}$ (2.7).

2.3. Review of KL-minimization. The Kullback-Leibler (KL) divergence between two probability distributions P and Q is defined as $\mathbb{D}_{\mathrm{KL}}(P \,\|\, Q) \coloneqq \mathbb{E}_P[\log(\frac{P}{Q})]$. As the expected difference between true and approximate log-densities, the KL divergence naturally judges the quality of an approximating distribution. We identify the positive-definite matrix $\Theta \in \mathbb{R}^{N\times N}$ as the covariance matrix of a centered Gaussian process $\mathcal{N}(\mathbf{0},\Theta)$ which we seek to approximate by a sparse approximate Cholesky factor $L \in \mathcal{S}$ of its precision, $\mathcal{N}(\mathbf{0},(LL^\top)^{-1})$. We compare these distributions by the KL divergence as [27] does, specializing the generic optimization problem (2.7) to

$$L \coloneqq \operatorname*{argmin}_{\hat{L}\in\mathcal{S}} \mathbb{D}_{\mathrm{KL}}\Big(\mathcal{N}(\mathbf{0},\Theta) \,\Big\|\, \mathcal{N}(\mathbf{0},(\hat{L}\hat{L}^\top)^{-1})\Big) \tag{2.9}$$

For multivariate Gaussians, the KL divergence has a closed-form expression given by

$$2\mathbb{D}_{\mathrm{KL}}(\mathcal{N}(\mathbf{0},\Theta_1) \,\|\, \mathcal{N}(\mathbf{0},\Theta_2)) = \mathrm{trace}(\Theta_2^{-1}\Theta_1) + \mathrm{logdet}(\Theta_2) - \mathrm{logdet}(\Theta_1) - N \tag{2.10}$$

where $\Theta_1, \Theta_2 \in \mathbb{R}^{N\times N}$. Using this expression for the KL divergence and optimizing for L yields the following closed-form expression for the nonzero entries in the ith

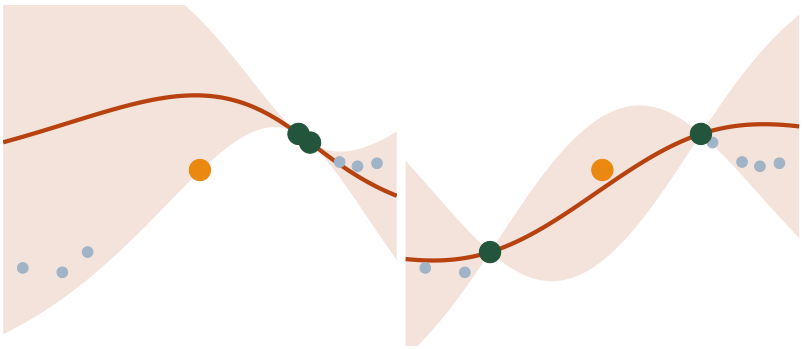

FIG. 3. *Here, the blue points are the candidates, the orange point is the target point to predict at, and the green points are the selected points. The red line is the conditional mean μ, conditional on the selected points, and the $\pm 2\sigma$ confidence interval is shaded for the conditional variance σ^2. Each method has a budget of two points; the left panel shows selection by Euclidean distance and the right by conditional variance. Euclidean distance prefers the two points right of the target. However, a more balanced view of the situation is obtained when picking the slightly further but ultimately more informative point to the left, reducing variance at the target and thereby reducing predictive error.*

column of L with sparsity pattern s_i, reproduced from Theorem 2.1 of [27]:

$$L_{s_i,i} = \frac{\Theta^{-1}_{s_i,s_i} \boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top \Theta^{-1}_{s_i,s_i} \boldsymbol{e}_1}} \tag{2.11}$$

where the notation for $\Theta^{-1}_{s_i,s_i}$ is from (2.3) and $\boldsymbol{e}_1 \in \mathbb{R}^{|s_i|\times 1}$ denotes the vector with first entry one and the rest zero. We enforce the convention that i is the first entry of s_i, also implying that L is of full rank. Plugging the optimal L (2.11) back into the KL divergence (2.10), we obtain the objective as a function of the sparsity pattern. See Appendix A.1 for details; importantly, the order of the KL divergence matters.

$$2\mathbb{D}_{\mathrm{KL}}\big(\mathcal{N}(\mathbf{0},\Theta) \,\big\|\, \mathcal{N}(\mathbf{0},(LL^\top)^{-1})\big) = \sum_{i=1}^{N} \left[\log\left(\Theta_{i,i|s_i\setminus\{i\}}\right) - \log\left(\Theta_{i,i|i+1:}\right)\right] \tag{2.12}$$

This sum is the accumulated *difference* in posterior variance for a series of independent regression problems: each to predict the ith variable given a subset of the variables after it in the ordering. The error $\log\left(\Theta_{i,i|s_i\setminus\{i\}}\right)$ made when restricted to the variables in the sparsity pattern s_i is compared to the ground truth $\log\left(\Theta_{i,i|i+1:}\right)$. A similar decomposition of the KL divergence into independent regression problems was observed in Equation (5) of [14] for lower triangular transport maps.

Thus, picking the right sparsity pattern to minimize KL divergence reduces to selecting the points s_i out of the possible candidates $i+1,\ldots,N$ that most reduce predictive error at point(s) of interest. In the next section, we develop such a selection algorithm for directed inference in Gaussian processes. We apply this algorithm for sparsity selection of sparse Cholesky factors in section 4 and extend it in section 5.

3. Greedy selection for directed inference. In directed Gaussian process regression we are given N points of training data and predict at a target point with unknown value by selecting the s points most "informative" to the target, $s \ll N$. From KL-minimization, the criterion for informativity should be to minimize the variance of the target point, conditional on the selected points (2.12). The variance

objective was first described by [4] for optimal experimental design and later applied to directed Gaussian process inference by [5] who refer to it as the active learning Cohn (ALC) technique in honor of [4]. In addition, the variance objective is equivalent to maximizing the *mutual information* or *information gain* with the target point as well as minimizing the expected mean squared error (see Appendix B.1). The mutual information (in a slightly different context) is also used by [15] for sensor placement.

In contrast to Euclidean distance [37] or unconditional correlation [40, 11], conditional variance incentives the often contradictory demands of being near the target point (nearby points have higher covariance), but away from previously selected points (to avoid redundancy); the resulting spread-out selections are illustrated in Figure 3. Even for isotropic kernels like the Matérn family, the resulting conditional selections can be unusual and seemly non-isotropic which is demonstrated in Figure 2.

3.1. A greedy approach. Minimizing the conditional variance over all possible $\binom{N}{s}$ subsets is intractable, so we greedily select the next point which most reduces the conditional variance. Let $I = \{i_j\}_{j=1}^{t} \subseteq \mathrm{Tr}$ be the indices of previously selected training points. For a newly selected index k, we condition the current covariance matrix on y_k according to the posterior (2.4), which we see is a rank-one downdate:

$$\Theta_{:,:|I,k} = \Theta_{:,:|I} - \boldsymbol{u}\boldsymbol{u}^\top \qquad \boldsymbol{u} = \frac{\Theta_{:,k|I}}{\sqrt{\Theta_{k,k|I}}} \tag{3.1}$$

The decrease in the variance of y_{Pr} after selecting k is given by u_{Pr}^2, or

$$u_{\mathrm{Pr}}^2 = \frac{\Theta_{\mathrm{Pr},k|I}^2}{\Theta_{k,k|I}} = \frac{\mathbb{C}\mathrm{ov}[y_{\mathrm{Pr}}, y_k \mid I]^2}{\mathbb{V}\mathrm{ar}[y_k \mid I]} = \mathbb{C}\mathrm{orr}[y_{\mathrm{Pr}}, y_j \mid I]^2\, \mathbb{V}\mathrm{ar}[y_{\mathrm{Pr}} \mid I] \tag{3.2}$$

To compute the objective (3.2) for each candidate index j, we start with the unconditional variance $\Theta_{j,j}$ and covariance $\Theta_{\mathrm{Pr},j}$, updating these quantities when an index k is selected by Equation (3.1). We have two efficient strategies to compute $\boldsymbol{u}$: either by maintaining the precision of selected entries $\Theta_{I,I}^{-1}$ (Algorithm C.1) or by storing only the $|I|$ columns corresponding to selected points from the Cholesky factor of the joint covariance matrix Θ (Algorithm C.2); both methods are detailed in Appendix C.1.

Both approaches have a time complexity of $\mathcal{O}(Ns^2)$ to select s points out of N candidates, differing in space complexity. The precision takes $\mathcal{O}(s^2)$ space while the first s columns of the Cholesky factor of Θ uses $\mathcal{O}(Ns)$ space, always more memory ($N > s$). Both algorithms use $\mathcal{O}(N)$ space to store the conditional (co)variances. Although the precision algorithm uses less memory than the Cholesky algorithm, the Cholesky algorithm is preferred for ease of implementation and better performance.

4. Greedy selection for global approximation by KL-minimization. Directed Gaussian process regression infers the *local* distribution at points of interest. We turn our attention to *global* approximation of the entire Gaussian process; given a kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$ and a set of N points $\{\boldsymbol{x}_i\}_{i=1}^{N}$ we have the covariance matrix $\Theta_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for which we seek a sparse approximate Cholesky factor L of the precision, $LL^\top \approx \Theta^{-1}$. We first order the points by the reverse-maximin ordering described in subsection 2.2 and then straightforwardly apply the selection algorithm developed in section 3 to form the sparsity pattern. For the ith column of L, the target point is the ith point in the ordering, the candidate points are those satisfying lower triangularity (after the target in the ordering), and running the selection algorithm for the desired number of nonzeros entries picks out indices which we add to the

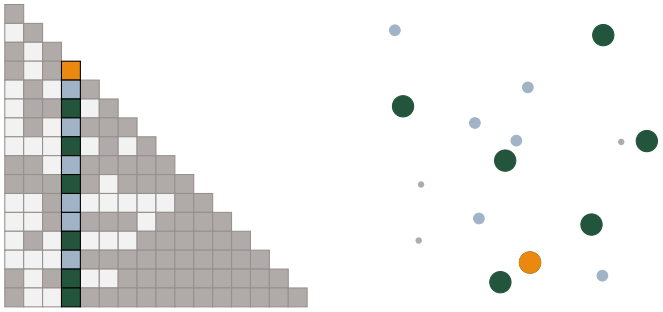

FIG. 4. *For a column of a Cholesky factor in isolation, the target point is the diagonal entry, candidates are below it, and the selected entries are added to the sparsity pattern. Points violating lower triangularity are shown in grey. Thus, sparsity selection in Cholesky factorization (left panel) is analogous to training point selection in directed Gaussian process regression (right panel).*

sparsity set s_i; this process is illustrated in Figure 4. Finally, we compute the values of the selected nonzero indices by the closed-form expression (2.11). Both sparsity selection and computing entries are embarrassingly parallel over L's columns.

Using the single-target selection algorithm from subsection 3.1 has time complexity $\mathcal{O}(NCs^2)$ to select s nonzero entries out of C candidates for N columns. Computing the corresponding values $\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1$ has time complexity $\mathcal{O}(Ns^3)$. Sparsity selection has the same complexity as entry computation if the number of candidates C is $\mathcal{O}(s)$, suggesting the need to limit the number of candidates considered. In practice, we pick the candidate set to be the nearest neighbors of the point of interest as [5] does; specifically, we use the framework of [27] which considers all points within a radius proportional to the length scale from the reverse-maximin ordering (2.8).

5. Extensions. Here we consider extensions of the basic single-column method to aggregated (or supernodal) Cholesky factorization and to determining the number of nonzeros per column. Numerical experiments are presented in section 6.

5.1. Aggregated sparsity pattern. We now derive a similar decomposition of the KL divergence if the same sparsity pattern is reused for multiple columns, known as aggregated or supernodal Cholesky factorization. Aggregation can lead to substantial time and space savings [27]. For this section we focus on a single group $\tilde{i} = \{i_1, \ldots, i_m\}$ composed from aggregating the column indices $i_1 \succ i_2 \succ \cdots \succ i_m$. Let $\tilde{i}$ have aggregated selected entries $s_{\tilde{i}}$ satisfying $s_{\tilde{i}} \supseteq \tilde{i}$ to guarantee that the Cholesky factor has full rank. Let $\tilde{s} \coloneqq s_{\tilde{i}} \setminus \tilde{i}$ be the selected entries excluding the columns in the group. The sparsity pattern for the kth column in the group is then the aggregated selected entries excluding the entries that violate lower triangularity, $s_k \coloneqq \{j \in s_{\tilde{i}} : j \succeq k\}$. Assuming every entry of $\tilde{s}$ is after every index in $\tilde{i}$, then $s_k = \tilde{s} \cup \{j \in \tilde{i} : j \succeq k\}$. This condition is guaranteed if the aggregated columns are adjacent in the ordering, for example; we defer handling the general case to the next section. The KL divergence (2.12) restricted to the contribution from the group $\tilde{i}$ is

$$\sum_{i \in \tilde{i}} \log\left(\Theta_{i,i|s_i \setminus \{i\}}\right) = \operatorname{logdet}(\Theta_{\tilde{i},\tilde{i}|\tilde{s}}) \tag{5.1}$$

from Appendix A.2. Thus, the generalization of the posterior variance (2.12) to aggregated columns is their log determinant conditional on (well-behaved) selected entries. We briefly discuss what happens when selected entries are *between* columns.

5.1.1. Nonadjacent or partial aggregation. Let the random variables corresponding to the indices $\tilde{i} = \{i_1, \ldots, i_m\}$ be collected in a vector $\boldsymbol{y} = [y_1, \ldots, y_m]^\top$ with joint density $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \Theta)$. We select an index k that conditions all but the first p variables (recall that the indices are sorted w.r.t $\succ$, so if k conditions a variable, it conditions all those afterwards). After this partial conditioning, the variables become $\boldsymbol{y}_{\|k} \coloneqq [y_1, \ldots, y_p, y_{p+1|k}, \ldots, y_{m|k}]^\top$ and the covariance matrix $\mathbb{C}\mathrm{ov}[\boldsymbol{y}_{\|k}]$ becomes

$$\Theta_{\tilde{i},\tilde{i}\|k} \coloneqq \mathbb{C}\mathrm{ov}[\boldsymbol{y}_{\|k}] = \begin{pmatrix} L_{:p}L_{:p}^\top & L_{:p}L'^\top_{p+1:} \\ L'_{p+1:}L_{:p}^\top & L'_{p+1:}L'^\top_{p+1:} \end{pmatrix} = \begin{pmatrix} L_{:p} \\ L'_{p+1:} \end{pmatrix} \begin{pmatrix} L_{:p} \\ L'_{p+1:} \end{pmatrix}^\top \tag{5.2}$$

where $L = \mathrm{chol}(\Theta)$ and $L' = \mathrm{chol}(\Theta_{:,:|k})$. See Figure 14 for an illustration and Appendix A.3 for details. Armed with this representation, we compute $\mathrm{logdet}(\Theta_{\tilde{i},\tilde{i}\|k})$:

$$\sum_{i \in \tilde{i}} \log\left(\Theta_{i,i|s_i \setminus \{i\}}\right) = \mathrm{logdet}(\Theta_{\tilde{i},\tilde{i}\|k}) \tag{5.3}$$

Like the aggregated case (5.1), minimizing the log determinant of the *partially* conditioned covariance matrix (5.2) is the same as minimizing the KL divergence (2.12).

5.2. Supernodes and blocked selection. For multiple ($m > 1$) target points, [5] suggests simply independently applying the single-target algorithm to each target. Instead, we will use *the same* selected points for *all* the target points, essentially speeding up selection by a factor of m, furthermore, the cost of computing the entries of the resulting Cholesky factor is reduced by this aggregation. The primary downside is reduced accuracy per sparsity entry since each target no longer receives individual attention. We mitigate this by paying heed to the "two birds with one stone" maxim, or by considering a candidate's simultaneous effect on *all* prediction points. In practice, this approach yields better accuracy per unit time than single-target selection.

The first question is generalizing the objective for a single target (3.2) to multiple targets. Continuing with KL-minimization (5.1), the criterion should be to minimize the log determinant of the posterior covariance matrix, $\mathrm{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}|I})$. This objective, known as D-optimal design in the literature [15], can be intuitively interpreted as a volume of uncertainty or as a scaling factor in the density of multivariate Gaussians. In addition, it is equivalent to maximizing mutual information since the differential entropy of a Gaussian strictly increases with its log determinant (see Appendix B.1).

We want to quickly compute how selecting an index k affects the log determinant. By application of the matrix determinant lemma (the details are in Appendix B.5),

$$\mathrm{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}|I,k}) - \mathrm{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}|I}) = \log(\Theta_{k,k|I,\mathrm{Pr}}) - \log(\Theta_{k,k|I}) \tag{5.4}$$

Equation (5.4) swaps the roles of the targets and the candidate: the *candidate* is now conditioned by the *targets*, reducing to single-target selection. Using the recipes from subsection 3.1 to compute conditional variances, we can compute the objective by maintaining a data structure for each term: one for $\Theta_{k,k|I,\mathrm{Pr}}$ and the other for $\Theta_{k,k|I}$. By the quotient rule $\Theta_{k,k|I,\mathrm{Pr}} = \Theta_{k,k|\mathrm{Pr},I}$ so we can condition on the prediction points *before* any points have been selected. After this initialization, we repeatedly update both data structures after selecting the best candidate by the objective (5.4).

We have two strategies from the two approaches of the single-target algorithm: one maintaining the precision of the selected entries $\Theta_{I,I}^{-1}$ as well as of the target points $\Theta_{\mathrm{Pr},\mathrm{Pr}}^{-1}$ (Algorithm C.4) and the other simply storing two Cholesky factors of the joint covariance matrix Θ (Algorithm C.5); both methods are detailed in Appendix C.2.

Both approaches have a time complexity of $\mathcal{O}(Ns^2+Nm^2+m^3)$ to select s points out of N candidates for m targets, again differing in space complexity: although using more memory, the Cholesky approach is preferred for simplicity and performance.

Partial selection. The multiple-target algorithm implicitly assumes a candidate conditions *every* target point, however, candidates can condition only a subset of the targets in the aggregated Cholesky factorization setting of subsection 5.1.1. Proper "partial" selection accounting for this structure is able to match the asymptotic time and space complexities of the multiple-target algorithm by also storing a partial Cholesky factor, the details are provided in Appendix C.3 and Algorithm C.7.

5.3. Aggregated Cholesky. In an aggregated sparsity pattern, columns are partitioned into groups and selecting an index for a group $\tilde{i}$ adds it to the sparsity pattern of every column in $\tilde{i}$ satisfying lower triangularity. We group columns by the framework of [27] which aggregates points that are close both geometrically as well as in the ordering. To select sparsity entries, the targets are all points in $\tilde{i}$ and the candidates are the union of the nearest neighbors to each target. If every candidate k satisfies $k \succ \max \tilde{i}$ which occurs if the group is contiguous in the ordering e.g., then every candidate conditions every target and so the multiple-target selection algorithm (subsection 5.2) can be directly applied. However, we empirically observe that forcing this condition irreparably damages the accuracy of the resulting factor: forming groups contiguous in the ordering no longer guarantees that grouped points are spatially close, and removing candidates between targets filters many of them out. If the condition is not forced, then selecting candidates can condition subsets of the group; the multiple-target algorithm now systematically overestimates the effect on targets. Using the partial selection algorithm (Appendix C.3) instead on unmodified grouping and candidate sets significantly improves the approximation quality.

Because the sparsity patterns for columns in the same group are subsets of each other, we can efficiently compute the group's entries in L (2.11) together in the time complexity for a single column (see Appendix A.4 or Algorithm 3.2 of [27]). If each group has m points, both the multiple-target and partial algorithms have time complexity $\mathcal{O}(Cs^2 + Cm^2 + m^3)$ to select s points out of C candidates. Over N/m groups the time complexity for both selection and entry computation is $\mathcal{O}(\frac{N}{m}(Cs^2 + Cm^2 + m^3 + s^3))$, simplifying to $\mathcal{O}(\frac{NCs^2}{m})$ assuming $m = \mathcal{O}(s)$, a m times improvement over non-aggregated factorization. Better time complexity yields denser and thereby more accurate factors in the same amount of time. Inheriting the primary downside of the multiple-target selection algorithm, the aggregated factor is less efficient at reducing the KL divergence per nonzero since the sparsity pattern is shared between all columns in the group, not tailored to any particular column.

5.4. Allocating nonzeros by global greedy selection. With a budget on the total number of nonzeros, one inevitably decides how many nonzeros each column gets. We recommend the simple strategy of distributing nonzeros as evenly as possible, maximizing computational efficiency since denser columns have an outsized impact on the computational time from the cubic scaling cost with the number of nonzeros.

In inhomogeneous geometries where certain points benefit from more nonzeros than others, a principled way of distributing nonzeros might be to minimize KL divergence end-to-end like was done for sparsity selection. The *local* greedy algorithms select the sparsity entry that minimizes prediction error at *particular* columns of interest. In *global* greedy selection, we pick from *any* column the candidate that minimizes the overall KL divergence (2.12). We maintain a priority queue containing all

candidates from every column, keyed by the candidate's effect on the KL divergence. The data structure must support popping the largest element off the queue as well as updating the value for an element in the queue. Both operations have time complexity $\mathcal{O}(\log n)$ for n elements if implemented as an array-backed binary heap, for example.

The greedy selection algorithms already compute the effect of an entry on the KL divergence, up to monotonically increasing transformations (which preserve the ranking of candidates). But in the global context, if different columns use different transformations, then the ranking of candidates between columns is skewed. We describe the necessary modifications to compute exactly the difference in KL divergence.

5.4.1. Single column selection. Selecting an entry k for a single target only affects its conditional variance, so exactly one term in the KL divergence (2.12) changes:

$$\operatorname*{argmin}_k \left[\log(\Theta_{\mathrm{Pr},\mathrm{Pr}|I,k}) - \log(\Theta_{\mathrm{Pr},\mathrm{Pr}|I})\right] = \operatorname*{argmin}_k \left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I,k}\right) \Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I} \tag{5.5}$$

Using the original objective (3.2) to compute the change in variance from selecting k,

$$\operatorname*{argmin}_k \left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I} - \frac{\Theta^2_{\mathrm{Pr},k|I}}{\Theta_{k,k|I}}\right) \Theta^{-1}_{\mathrm{Pr},\mathrm{Pr}|I} = \operatorname*{argmax}_k \ \mathbb{C}\mathrm{orr}[y_{\mathrm{Pr}}, y_k \mid I]^2 \tag{5.6}$$

where the new objective (5.6) is easily computed as the original objective (3.2) divided by the target's conditional variance, the percentage the decrease in variance takes up.

5.4.2. Aggregated selection. The multiple-target algorithm already computes the exact difference in log determinant after selecting a candidate. The partial selection algorithm computes the log determinant itself, not the difference, so the log determinant before the selection needs to be subtracted, easily computed as the sum of the squared "diagonal" entries of L corresponding to target points from (C.2).

One heuristical improvement is to measure "bang for the buck", i.e. to consider that the number of nonzeros added from selecting a candidate varies depending on the number of targets conditioned. The KL divergence adds a candidate's effect on each target together, so candidates conditioning more targets tend to decrease the KL divergence more, even if they are less efficient per target. In practice, it is often better to divide the change in KL divergence by the number of targets conditioned.

For the multiple-target algorithm, this modification makes no difference within a group since every candidate conditions every target point, but can make a difference in the global setting if groups have different sizes. This modification can make a difference both within a column and globally for the partial algorithm since different candidates within the same group can condition a different number of targets.

6. Numerical experiments. All experiments ran on the Partnership for an Advanced Computing Environment (PACE) Phoenix cluster at the Georgia Institute of Technology, with 8 cores of a Intel Xeon Gold 6226 CPU @ 2.70GHz and 22 GB of RAM per core. The code is written in Python using standard scientific libraries `numpy` [8], `scipy` [39], `scikit-learn` [23], `matplotlib` [10] as well as Cython [1] which provides direct transpilation of Python code into C. Cython also allows Python code to access native C interfaces to the `BLAS` and Intel `oneMKL` libraries. Code for all numerical experiments can be found at https://github.com/stephen-huan/conditional-knn.

6.1. *k*-nearest neighbors selection. We compare k-nearest neighbors (k-NN) to our selection algorithm from subsection 3.1, which we call conditional k-nearest neighbors (Ck-NN), in the following toy example performing image classification. From the MNIST database of handwritten digits [18], we randomly select $N = 10^3$

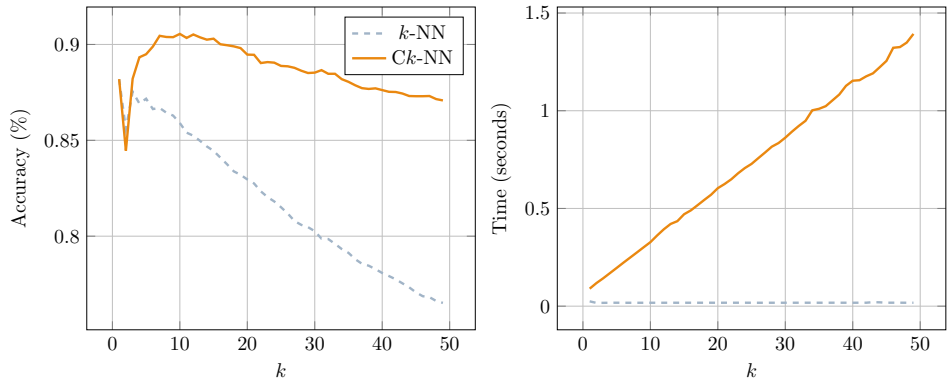

FIG. 5. *We compare selection by Euclidean distance (k-NN) to conditional selection (Ck-NN) in image classification. From the MNIST database $N = 10^3$ training images and $m = 10^2$ testing images are randomly chosen; each test image is classified by the mode label of k selected images, and this process is repeated 10^2 times for each value of k. (Left:) Accuracy with k. (Right:) the time to select k points using Ck-NN seems to scale linearly with k although it is quadratic asymptotically; possibly resulting from applying highly optimized BLAS operations to relatively small matrices.*

training images and $m = 10^2$ testing images. We classify a test image by selecting k training images and taking the mode label of those images. For k-NN, we use the Euclidean distance and for Ck-NN, we use a Matérn kernel with smoothness $\nu = 3/2$ and length scale $\ell = 2^{10}$. Accuracy is the percentage of test images classified correctly.

The accuracy of both methods decreases linearly with increasing k as shown in Figure 5. We hypothesize that nearby images are more likely to have the same label, so selecting more points increases the influence of further, differently labeled images. Ck-NN remains more accurate than k-NN for every $k > 2$, suggesting it consistently selects more informative images. We emphasize that conditioning alone causes the difference in accuracy: unconditional covariance decays monotonically with distance.

6.2. Recovery of sparse Cholesky factors. Motivated by the similarity of the selection algorithm to orthogonal matching pursuit [35], we attempt to recover an *a priori* sparse Cholesky factor L from its precision matrix $Q = LL^\top$. To generate the nonzero entries of L, for each column we pick s lower triangular indices uniformly at random and sample their values i.i.d. from $\mathcal{N}(0, 1)$. We fill L's diagonal with a "large" positive value 10 to ensure Q is well-conditioned. The selection algorithm is given s and either Q or the covariance matrix Q^{-1} depending on which results in higher accuracy in reconstructing L. For a recovered sparsity pattern X and ground truth Y we report $|X \cap Y|/|X \cup Y|$, the intersection over union (IOU). Using the KL divergence (2.9) of the recovered Cholesky factor seems mostly equivalent to IOU.

As shown in Figure 6, Ck-NN maintains a near-perfect recovery accuracy much higher than the unconditional baselines. In the setting of noisy measurements, noise sampled i.i.d from $\mathcal{N}(0, \sigma^2)$ is symmetrically added to each entry of Q ($Q_{i,j}$ receives the same noise as $Q_{j,i}$). Accuracy degrades with increasing noise for all methods, but Ck-NN is the most sensitive to noise as is shown in Figure 7. At high enough levels of noise Q can lose positive-definiteness, causing Ck-NN to break down entirely.

An important setting where Q (and possibly L) are known to be *a priori* sparse is when solving Laplacian systems. In exploratory numerical experiments we found that the factors generated by our method were unreasonably dense compared the sparsity of the Laplacian matrix, a manifestation of *fill-in*. We comment that our method is

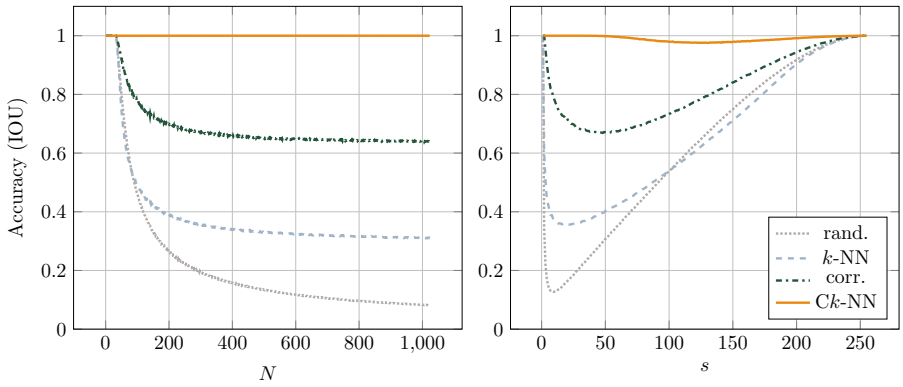

FIG. 6. *We attempt to recover a sparse Cholesky factor L of the precision matrix $Q = LL^\top$. "Ck-NN" minimizes the conditional variance of the target (diagonal) entry, "corr." maximizes correlation with the target, "k-NN" maximizes covariance with the target, and "rand." samples entries uniformly at random. All methods achieve their best accuracy given Q except Ck-NN which is given the covariance matrix Q^{-1}. (Left:) Varying the size of L, fixed density $s = 2^5$. (Right:) Varying s, fixed size $N = 2^8$. Accuracy starts to improve from the factor nearing fully dense.*

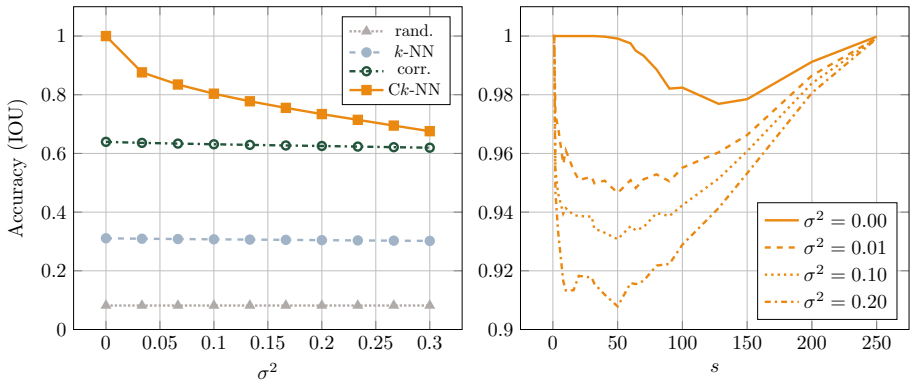

FIG. 7. *We add noise sampled i.i.d. from $\mathcal{N}(0, \sigma^2)$ to the measurements Q entrywise. The left panel shows accuracy with varying noise for $N = 2^{10}$ columns and $s = 2^5$ nonzeros per column. The right panel shows accuracy for the Ck-NN method at various noise levels for $N = 2^8$ and varying s.*

designed for *dense* kernel matrices resulting from a set of points and a "well-behaved" kernel function. For solving sparse Laplacian systems, a method designed to exploit the sparsity and graph structure of Laplacian matrices like [17] is more appropriate.

6.3. Cholesky factorization. We empirically verify that conditional sparsity selection produces more accurate sparse Cholesky factors than selection by Euclidean distance at the same density. We take N points on a slightly perturbed regular grid in $[0, 1]^2$ and use a Matérn kernel with smoothness $\nu = 5/2$ and length scale $\ell = 1$.

As a baseline for comparison, we use the single-column and aggregated variants of the KL-minimization framework of [27], which orders points by the reverse-maximin ordering described in subsection 2.2 and forms the sparsity pattern by selecting all points within a radius of $\rho\ell_i$ to the ith point, where $\rho \geq 1$ is a tuning parameter for density and ℓ_i is the length scale from the reverse-maximin ordering. We also try selecting points by k-NN, where k is chosen to match the density of the baseline.

For our method, we run the baseline with a larger $\rho' = \rho_s \cdot \rho$ to get an initial

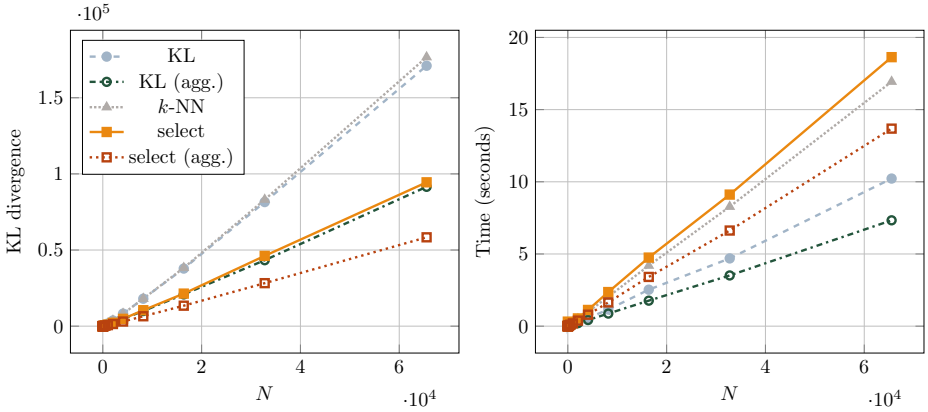

FIG. 8. *Accuracy (left) and computational time (right) of Cholesky factorization methods with varying number of points N and fixed density $\rho = 2$. "KL" is the baseline from [27], "k-NN" is selection by k-nearest neighbors, "select" is conditional selection, and "(agg.)" denotes aggregation.*

candidate set where ρ_s is a tuning parameter for the number of candidates considered. If not stated otherwise, $\rho_s = 2$ is used in the following experiments. We then use the single-column and partial variants of the conditional selection algorithm described in section 3 to subsample the actual sparsity entries from the candidate set; in these experiments, each column receives k nonzeros where k is chosen to match the original density ρ of the baseline factor. In this setting we found that using the global selection procedure described in subsection 5.4 to determine the number of nonzeros for each column led to little improvement in accuracy at a significant performance penalty.

We aggregate columns by the procedure of [27]: pick the first (w.r.t $\prec$) index i that has not been aggregated and create the group $\{j : j \in s_i, \ell_j \leq \lambda \ell_i\}$ where $\lambda \geq 1$ is a tuning parameter for group size; repeat until all indices have been aggregated. We generate the groups using the sparsity pattern from the baseline factor and use the same groups for aggregated conditional selection for a fair comparison. In all experiments we use $\lambda = 1.5$ as recommended by [27].

As Figure 8 shows, the KL divergence and computational time increase linearly with the number of points for all methods. Conditional methods are more accurate than their unconditional counterparts and the denser aggregated variants are both more accurate and faster than their non-aggregated counterparts.

The conditional selection methods achieve significantly better KL divergence compared to their unconditional counterparts for the same number of sparsity entries as Figure 9 shows. However, they do not achieve better accuracy per unit time cost from the increased cost of selection. Aggregation results in better accuracy per unit time cost but worse accuracy per nonzero entry, which may impact their computational efficiency in downstream tasks which depend on factor density like preconditioning.

6.4. Gaussian process regression. For Gaussian process regression we use the "predictions points first" method of [27] which computes a sparse Cholesky factor of the joint covariance matrix between the training and prediction points, where prediction points are ordered before training points. The desired posterior mean and covariance can then be computed efficiently from sparse submatrices of the factor. See subsection 4.2.1, Appendix D.1, or Algorithm D.1 of [27] for additional details.

We use the SARCOS dataset [38] generated from an inverse dynamics problem on a 7 degrees-of-freedom robotic arm. Given 7 joint positions, velocities and acceler-

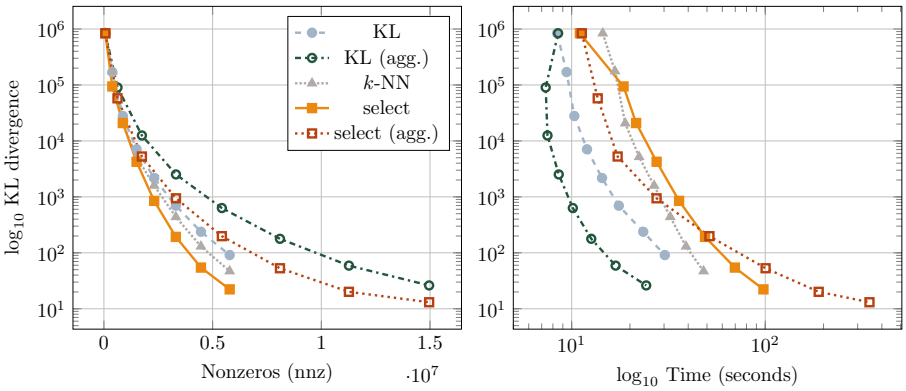

FIG. 9. *The left panel shows the KL divergence with varying density ρ and the right panel shows the accuracy to computational time trade-off over varying ρ. The number of points is $N = 2^{16}$.*

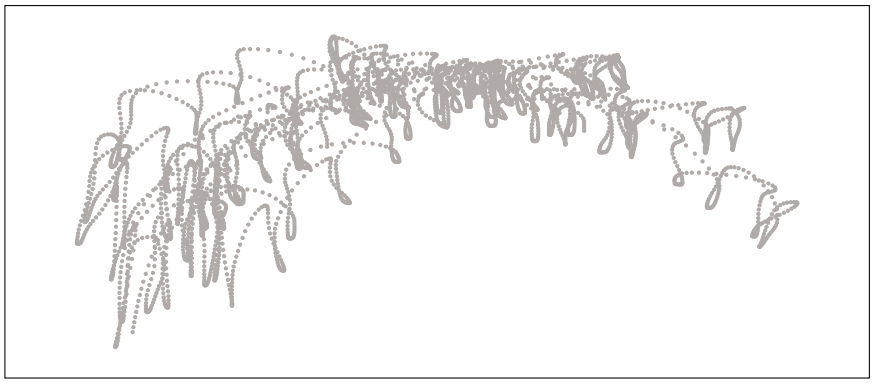

FIG. 10. *The first 5,000 points of the SARCOS training data, visualizing the first two features.*

ations for 21 features in total, the goal is to infer the torque of just the first joint. The dataset is available online at https://gaussianprocess.org/gpml/data/ and consists of 44,484 training points and 4,449 testing points which we then preprocess as follows.

We only use the first 3 features of the data and remove the 73 duplicate points this projection causes. Since the provided testing points overlap significantly with the training points, we ignore the original testing points and instead randomly partition the 44,411 remaining training points into 90% training points ($N = 39,969$) and 10% prediction points ($m = 4,442$). Since the robot arm moves smoothly, the geometry of the dataset consists of relatively continuous overlapping paths as shown in Figure 10. We then use a Matérn kernel with smoothness $\nu = 3/2$ and length scale $\ell = 1$ and draw 10^3 realizations from the resulting Gaussian process for the target variable, ignoring the original objective to ensure that the target variable is exactly Gaussian.

We consider three accuracy metrics for Gaussian process regression: the log determinant of the posterior prediction covariance matrix, the empirical coverage of the 90% posterior prediction intervals averaged over all realizations, and the average root mean square error (RMSE) of the posterior means. The log determinant is equivalent to the KL divergence by the discussion in subsection 5.1, so the results are similar to Cholesky factorization (subsection 6.3). We find that coverage is extremely accurate for all methods (within 0.1% for $\rho > 2$). Finally, the RMSE is shown in Figure 11.

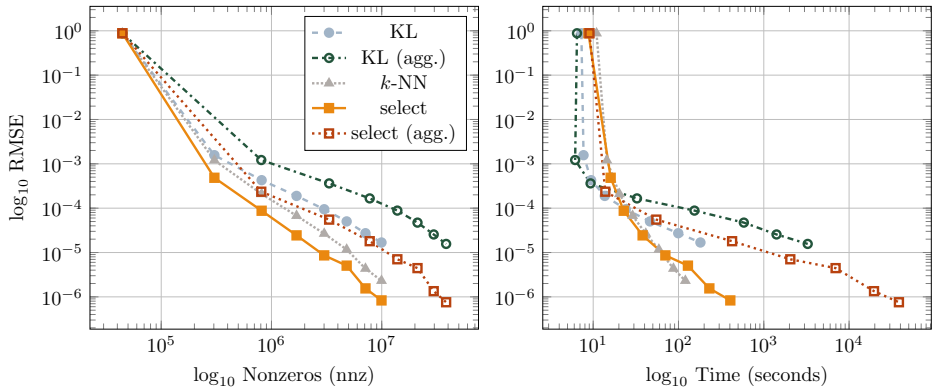

FIG. 11. *We perform Gaussian process regression by sparse Cholesky factorization of the joint covariance matrix. Like Cholesky factorization in subsection 6.3, we use a candidate set size scaling factor of $\rho_s = 2$ and an aggregation parameter of $\lambda = 1.5$. The left panel shows the difference in RMSE from exact Gaussian process regression using the same training points with varying density ρ. The right panel shows the accuracy to computational time trade-off over varying ρ.*

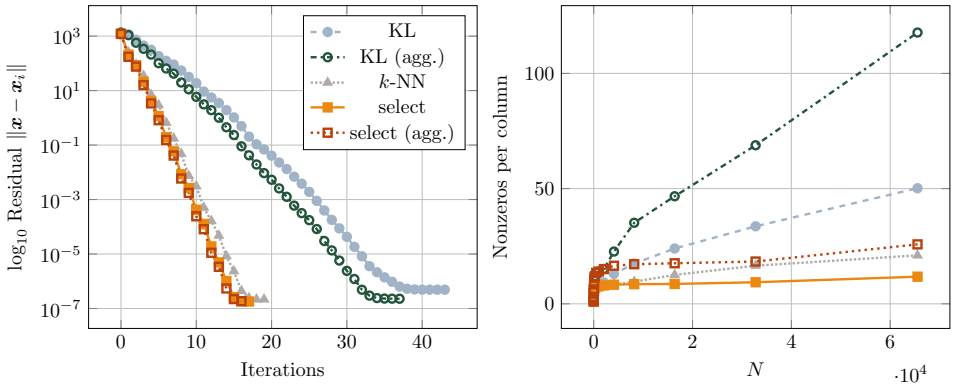

FIG. 12. *We use the conjugate gradient preconditioned by sparse Cholesky factors to solve the symmetric positive-definite system $\Theta \boldsymbol{x} = \boldsymbol{y}$. The left panel shows iteration progress for $N = 2^{16}$ points and a factor density of $\rho = 4$. The right panel shows the minimum number of nonzeros per column for conjugate gradient to converge within 50 iterations with an increasing number of points.*

Despite an increased cost to select points, the conditional methods have better accuracy per unit computational cost than their unconditional counterparts as a result of their superior accuracy at the same sparsity. However, the simpler method of k-NN also achieves comparable accuracy per unit time cost. As noted in [6], the simple method of k-NN remains hard to beat without specialized tricks.

6.5. Preconditioning for conjugate gradient. Motivated by the equivalence of functionals like the Kaporin condition number to the KL divergence, we investigate solving symmetric positive-definite systems $\Theta \boldsymbol{x} = \boldsymbol{y}$ using the conjugate gradient and a sparse Cholesky factor L as a preconditioner. We note that from (2.10) the KL divergence strongly penalizes zero eigenvalues of the preconditioned matrix $\Theta L L^\top$, improving its condition number. In order to generate the covariance matrix Θ we sample up to $N = 2^{16}$ points uniformly at random from the unit cube $[0, 1]^3$ and use a Matérn kernel with smoothness $\nu = 1/2$ and length scale $\ell = 1$. In exploratory numerical experiments, we found using higher smoothnesses like $\nu = 5/2$ led to extremely

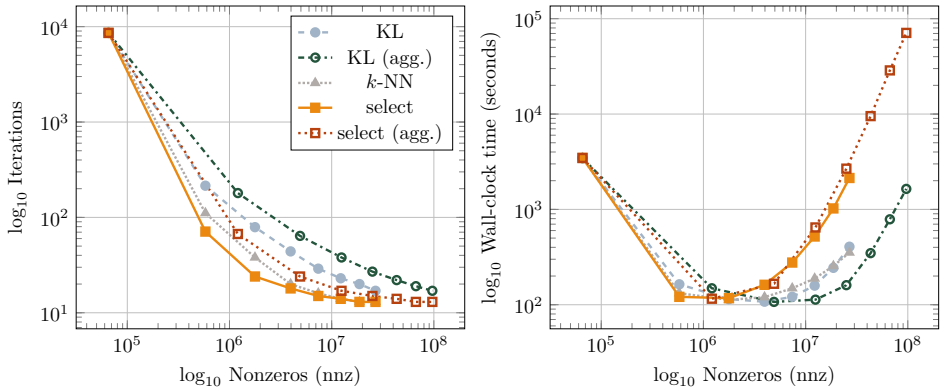

FIG. 13. *The left panel shows how the number of iterations for the conjugate gradient to converge decreases with increasing preconditioner density ρ for $N = 2^{16}$ points. The right panel shows the total wall-clock time (both to compute the preconditioner and to converge) with varying density.*

poor condition and numerical instability (over thousands of iterations to converge). Increasing length scale also worsens the condition but to a less extreme extent. Rather than generate a right hand side $\boldsymbol{y}$ directly, we first sample a solution $\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \mathrm{Id}_N)$ and then compute $\boldsymbol{y} = \Theta\boldsymbol{x}$ so that $\boldsymbol{y}$ is realistically smooth. When computing the preconditioner L by sparse Cholesky factorization, we use a candidate set size scaling factor of $\rho_s = 2$ and an aggregation parameter of $\lambda = 1.5$. We then run the conjugate gradient with L as a preconditioner until reaching a relative tolerance of 10^{-12}.

As shown in Figure 12, the conditional methods converge in half the iterations of their unconditional counterparts and aggregation barely reduces the number of iterations. The minimum number of nonzeros per column to converge within a constant number of iterations (here, 50) seems to grow logarithmically with the number of points for all methods; although the conditional methods appear near constant.

We observe a characteristic "U" shape for the total wall-clock time in Figure 13 from the trade-off between spending computational time in forming the preconditioner or in conjugate gradient iterations. Across all preconditioner densities, the aggregated variants are slower than their non-aggregated counterparts due to barely reducing the number of iterations while producing significantly denser factors with slower matrix-vector products. The time-optimal density for conditional methods is sparser than their unconditional counterparts due to using less iterations at the same sparsity and it being more expensive to select nonzeros. It is hard to directly compare the total computational time of the methods because simply increasing the number of iterations, e.g. by demanding better tolerance or by using matrices with worse condition, will prefer more accurate methods even if they are slower to form the preconditioner.

6.6. Comparison to other methods.

Local approximate Gaussian process (laGP). Our conditional variance objective for point selection (3.2) was first described in [4] for optimal experimental design, and has subsequently been named the active learning Cohn (ALC) technique. [5] and the follow-up works [6, 34] apply ALC to directed Gaussian process inference, yielding an algorithm equivalent to ours described in subsection 3.1 for a single point of interest. We note that their definition of conditional variance in Equation (5) is analogous to our (2.2), equating their conditional variance objective in Equation (8) to our (3.2) (the connection is explicitly stated in SM§4's Equation (SM.22)). They

also update the precision by the blocked equations in SM§1 like our Appendix B.3.

For inference at multiple points of interest, [5] suggests direct parallelization of the single-target algorithm over each target. They mention that the `laGP` function in their `R` package jointly considers multiple target points, but without providing details on this procedure. [34] proposes a "joint" or "path" ALC by taking the average reduction in posterior variance over target points. Instead, we generalize ALC to multiple prediction points through their posterior log determinant (5.4) and provide an explicit algorithm described in subsection 5.2. In addition, integration of the algorithm into Cholesky factorization (section 4) provides a global approximation of the Gaussian process (subsection 6.4) beyond directed local approximation.

Orthogonal matching pursuit (OMP). Our selection algorithm can be viewed as the covariance equivalent of the sparse signal recovery algorithm orthogonal matching pursuit (OMP) [35, 36]. OMP measures the approximation of a target signal by its residual after orthogonal projection onto the subspace of chosen training signals, which is efficiently calculated by maintaining a QR factorization. In contrast, our selection algorithm uses a kernel function to evaluate inner products, so orthogonalization becomes conditioning, the residual norm becomes variance, and the QR factorization becomes a Cholesky factorization. A major difference from OMP is that the computational time of conditioning dominates that of evaluating the kernel function since the feature space is relatively low-dimensional (often 2 or 3 in spatial statistics).

Sparse Cholesky factorization. Our sparse Cholesky factorization algorithm proposed in section 4 relies heavily on the KL-minimization framework of [27] and is similar to [11]. We comment that using k-nearest neighbors to select the sparsity set instead of our conditional selection algorithms essentially recovers [27] and that using the correlation objective (3.2) without conditioning on selected points recovers [11].

7. Conclusions. In this work, we develop a algorithm for directed Gaussian process regression which greedily selects training points that maximize mutual information with a target point, conditional on all points previously selected to avoid redundancy. We show that using conditional selection to pick the sparsity pattern of sparse approximate Cholesky factors of precision matrices significantly improves accuracy and performance in downstream tasks compared to selection by nearest neighbors. Single-target conditional selection is computationally efficient and can be extended to the settings of multiple-target and partial conditioning corresponding to aggregated (or supernodal) Cholesky factorization. Finally, global selection gives a principled way of distributing nonzeros over columns of the Cholesky factor. We support these claims by extensive numerical experimentation in a variety of problems.

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] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, *Cython: The Best of Both Worlds*, Computing in Science & Engineering, 13 (2011), pp. 31–39, https://doi.org/10.1109/MCSE.2010.118.

[2] K. Chalupka, C. K. I. Williams, and I. Murray, *A Framework for Evaluating Approximation Methods for Gaussian Process Regression*, Nov. 2012, https://doi.org/10.48550/arXiv.1205.6326, https://arxiv.org/abs/1205.6326.

[3] 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.

[4] D. A. Cohn, *Neural Network Exploration Using Optimal Experiment Design*, Neural Networks, 9 (1996), pp. 1071–1083, https://doi.org/10.1016/0893-6080(95)00137-9.
[5] R. B. Gramacy and D. W. Apley, *Local Gaussian process approximation for large computer experiments*, Oct. 2014, https://arxiv.org/abs/1303.0383.
[6] R. B. Gramacy and B. Haaland, *Speeding up neighborhood search in local Gaussian process prediction*, Jan. 2015, https://arxiv.org/abs/1409.0074.
[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] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, *Array programming with NumPy*, Nature, 585 (2020), pp. 357–362, https://doi.org/10.1038/s41586-020-2649-2.
[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. D. Hunter, *Matplotlib: A 2D Graphics Environment*, Computing in Science & Engineering, 9 (2007), pp. 90–95, https://doi.org/10.1109/MCSE.2007.55.
[11] M. Kang and M. Katzfuss, *Correlation-based sparse inverse Cholesky factorization for fast Gaussian-process inference*, Dec. 2021, https://arxiv.org/abs/2112.14591.
[12] 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.
[13] 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.
[14] M. Katzfuss and F. Schäfer, *Scalable Bayesian transport maps for high-dimensional non-Gaussian spatial fields*, Feb. 2022, https://arxiv.org/abs/2108.04211.
[15] 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.
[16] 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.
[17] 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.
[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE, 86 (1998), pp. 2278–2324, https://doi.org/10.1109/5.726791.
[19] H. Liu, Y.-S. Ong, X. Shen, and J. Cai, *When Gaussian Process Meets Big Data: A Review of Scalable GPs*, IEEE Transactions on Neural Networks and Learning Systems, 31 (2020), pp. 4405–4423, https://doi.org/10.1109/TNNLS.2019.2957109.
[20] 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.
[21] M. Mutný and A. Krause, *Experimental Design for Linear Functionals in Reproducing Kernel Hilbert Spaces*, May 2022, https://arxiv.org/abs/2205.13627.
[22] H. Owhadi and C. Scovel, *Operator-Adapted Wavelets, Fast Solvers, and Numerical Homogenization: From a Game Theoretic Approach to Numerical Approximation and Algorithm Design*, Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, Cambridge, 2019, https://doi.org/10.1017/9781108594967.
[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay, *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12 (2011), pp. 2825–2830.
[24] J. Quiñonero-Candela and C. E. Rasmussen, *A Unifying View of Sparse Approximate Gaussian Process Regression*, Journal of Machine Learning Research, 6 (2005), pp. 1939–1959.
[25] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, Adaptive Computation and Machine Learning, MIT Press, Cambridge, Mass, 2006.

[26] 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.
[27] 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.
[28] 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.
[29] 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.
[30] A. Smola and P. Bartlett, *Sparse Greedy Gaussian Process Regression*, in Advances in Neural Information Processing Systems, vol. 13, MIT Press, 2000.
[31] A. Spantini, D. Bigoni, and Y. Marzouk, *Inference via low-dimensional couplings*, July 2018, https://doi.org/10.48550/arXiv.1703.06131, https://arxiv.org/abs/1703.06131.
[32] M. L. Stein, *The screening effect in Kriging*, The Annals of Statistics, 30 (2002), pp. 298–323, https://doi.org/10.1214/aos/1015362194.
[33] 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.
[34] F. Sun, R. B. Gramacy, B. Haaland, E. Lawrence, and A. Walker, *Emulating satellite drag from large simulation experiments*, June 2019, https://doi.org/10.48550/arXiv.1712.00182, https://arxiv.org/abs/1712.00182.
[35] 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.
[36] 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.
[37] 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.
[38] S. Vijayakumar and S. Schaal, *Locally weighted projection regression: An O (n) algorithm for incremental real time learning in high dimensional space*, Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), (2000), pp. 1079–1086.
[39] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt, *SciPy 1.0: Fundamental algorithms for scientific computing in Python*, Nature Methods, 17 (2020), pp. 261–272, https://doi.org/10.1038/s41592-019-0686-2.
[40] T. Wada, Y. Matsumura, S. Maeda, and H. Shibuya, *Gaussian Process Regression with Dynamic Active Set and Its Application to Anomaly Detection*, in Proceedings of the International Conference on Data Science (ICDATA), 2013, p. 7.
[41] 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. Derivations in KL-minimization.

A.1. KL divergence of optimal factor. In subsection 2.3, we wish to compute the KL divergence between the covariance matrix Θ and the optimal sparse approximate Cholesky factor L computed from the closed-form expression (2.11).

Proof of Equation (2.12). From the closed-form expression for the KL divergence in (2.10) and defining $\Delta \coloneqq 2\mathbb{D}_{\mathrm{KL}}\big(\mathcal{N}(\mathbf{0},\Theta) \,\big\|\, \mathcal{N}(\mathbf{0},(LL^\top)^{-1})\big)$ for brevity of notation,

$$\Delta = \operatorname{trace}(LL^\top\Theta) - \operatorname{logdet}(LL^\top) - \operatorname{logdet}(\Theta) - N \tag{A.1}$$

Focusing on the term $\operatorname{trace}(LL^\top\Theta) = \operatorname{trace}(L^\top\Theta L)$ by the cyclic property of trace and using the sparsity of L by plugging in the definition (2.11) for each column $L_{s_i,i}$,

$$\operatorname{trace}(L^\top\Theta L) = \sum_{i=1}^N L_{s_i,i}^\top \Theta_{s_i,s_i} L_{s_i,i} \tag{A.2}$$

$$= \sum_{i=1}^N \left(\frac{\big(\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1\big)^\top}{\sqrt{\boldsymbol{e}_1^\top\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}}\right)\Theta_{s_i,s_i}\left(\frac{\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}}\right) \tag{A.3}$$

$$= \sum_{i=1}^N \frac{\boldsymbol{e}_1^\top\Theta_{s_i,s_i}^{-1}\Theta_{s_i,s_i}\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1}{\boldsymbol{e}_1^\top\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1} = \sum_{i=1}^N 1 = N \tag{A.4}$$

exactly the Jacobi scaling constraint $\operatorname{diag}(L^\top\Theta L) = 1$. Substituting back into (A.1),

$$\Delta = -\operatorname{logdet}(LL^\top) - \operatorname{logdet}(\Theta) \tag{A.5}$$

Computing the log determinant of a triangular matrix as the sum of the log of its diagonal entries and plugging in the definition (2.11) for the diagonal entries,

$$= -\sum_{i=1}^N \big[\log(\boldsymbol{e}_1^\top\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1)\big] - \operatorname{logdet}(\Theta) \tag{A.6}$$

$$= \sum_{i=1}^N \big[\log\big((\boldsymbol{e}_1^\top\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1)^{-1}\big)\big] - \operatorname{logdet}(\Theta) \tag{A.7}$$

Now we use that conditioning in covariance is marginalization in precision:

$$\Theta_{1,1|2} = \big(\Theta^{-1}\big)_{1,1}^{-1} \qquad \text{for } \Theta = \begin{pmatrix}\Theta_{1,1} & \Theta_{1,2}\\ \Theta_{2,1} & \Theta_{2,2}\end{pmatrix} \tag{A.8}$$

Transforming the marginalization $(\boldsymbol{e}_1^\top\Theta_{s_i,s_i}^{-1}\boldsymbol{e}_1)^{-1} = \big(\Theta_{s_i,s_i}^{-1}\big)_{1,1}^{-1} = \Theta_{i,i|s_i\setminus\{i\}}$ by (A.8),

$$\Delta = \sum_{i=1}^N \big[\log\big(\Theta_{i,i|s_i\setminus\{i\}}\big)\big] - \operatorname{logdet}(\Theta) \tag{A.9}$$

Now we use the chain rule of log determinants, using the same blocking as (A.8):

$$\operatorname{logdet}(\Theta) = \operatorname{logdet}(\Theta_{1,1}) + \operatorname{logdet}(\Theta_{2,2|1}) \tag{A.10}$$

Repeatedly expanding the log determinant by (A.10), working from back to front,

$$\Delta = \sum_{i=1}^N \log\big(\Theta_{i,i|s_i\setminus\{i\}}\big) - \sum_{i=1}^N \log\big(\Theta_{i,i|i+1:}\big) \tag{A.11}$$

$$= \sum_{i=1}^N \big[\log\big(\Theta_{i,i|s_i\setminus\{i\}}\big) - \log\big(\Theta_{i,i|i+1:}\big)\big] \tag{A.12}$$

∎

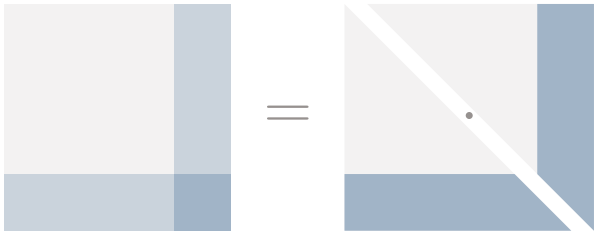

FIG. 14. *Illustration of the Cholesky factorization of a partially conditioned covariance matrix. Here grey denotes fully unconditional, blue denotes fully conditional, and the mixed color denotes interaction between the two. Surprisingly, such a matrix factors into a "pure" Cholesky factor by "gluing" the prefix of the fully unconditional factor with the suffix of the fully conditional factor.*

A.2. Aggregated KL divergence. In the aggregated Cholesky factorization setting of subsection 5.1, we have indices $\tilde{i} = \{i_1, \dots, i_m\}$ such that $i_1 \succ \cdots \succ i_m$. Let the selected entries be $\tilde{s}$; we assume every selected entry is after (w.r.t $\succ$) every index in $\tilde{i}$ so that the sparsity pattern for the kth column is $s_k = \tilde{s} \cup \{j \in \tilde{i} : j \succeq k\}$.

Proof of Equation (5.1). The KL divergence (2.12) restricted to the group $\tilde{i}$ is

$$\sum_{i \in \tilde{i}} \log(\Theta_{i|s_i \setminus \{i\}}) = \log(\Theta_{i_1|\tilde{s}}) + \log(\Theta_{i_2|\tilde{s} \cup \{i_1\}}) + \cdots + \log(\Theta_{i_m|\tilde{s} \cup \tilde{i}}) \tag{A.13}$$

where we write $\Theta_j \coloneqq \Theta_{j,j}$. Combining the first two terms by the chain rule (A.10),

$$= \operatorname{logdet}(\Theta_{\{i_1,i_2\}|\tilde{s}}) + \log(\Theta_{i_3|\tilde{s} \cup \{i_1,i_2\}}) + \cdots + \log(\Theta_{i_m|\tilde{s} \cup \tilde{i}}) \tag{A.14}$$

Proceeding by induction, we are able to reduce the entire sum to the single term

$$= \operatorname{logdet}(\Theta_{\tilde{i},\tilde{i}|\tilde{s}}) \tag{A.15}$$

∎

A.3. Partial KL divergence. In the setting of subsection 5.1.1, we have the Gaussian random variables $\boldsymbol{y} = [y_1, \dots, y_m]^\top$ corresponding to ordered column indices $i_1 \succ \cdots \succ i_m$ grouped in $\tilde{i}$. We select an index k that conditions all but the first p variables and wish to compute the covariance $\mathbb{C}\mathrm{ov}[\boldsymbol{y}_{\|k}]$ of the updated variables $\boldsymbol{y}_{\|k}$.

Proof of Equation (5.3). The original variables $\boldsymbol{y}$ have joint density multivariate Gaussian, $\boldsymbol{y} \sim \mathcal{N}(\mathbf{0}, \Theta)$ for some covariance matrix Θ so the fully conditional variables $\boldsymbol{y}_{|k}$ have posterior distribution from (2.1) and (2.2), $\boldsymbol{y}_{|k} \sim \mathcal{N}(\boldsymbol{\mu}, \Theta_{:,:|k})$ for some posterior mean $\boldsymbol{\mu}$. The covariance of unconditioned y_i and y_j is $\Theta_{i,j}$ by definition; similarly, the covariance of conditioned $y_{i|k}$ and $y_{j|k}$ is $\Theta_{i,j|k}$. We must compute the covariance between unconditioned y_i and conditioned $y_{j|k}$. Let $L = \operatorname{chol}(\Theta)$ and $L' = \operatorname{chol}(\Theta_{:,:|k})$ so that $\boldsymbol{y} = L\boldsymbol{z}$ for $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0}, \mathrm{Id})$ and $\boldsymbol{y}_{|k} = L'\boldsymbol{z} + \boldsymbol{\mu}$. By the definition of covariance,

$$\mathbb{C}\mathrm{ov}[y_i, y_{j|k}] = \mathbb{E}[(y_i - \mathbb{E}[y_i])(y_{j|k} - \mathbb{E}[y_{j|k}])] = \mathbb{E}[(L_i\boldsymbol{z})(L'_j\boldsymbol{z} + \mu_j - \mu_j)] \tag{A.16}$$

$$= \mathbb{E}[(L_{i,1}z_1 + \cdots + L_{i,m}z_m)(L'_{j,1}z_1 + \cdots + L'_{j,m}z_m)] \tag{A.17}$$

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

$$= L_{i,1}L'_{j,1}\,\mathbb{E}[z_1^2] + \cdots + L_{i,m}L'_{j,m}\,\mathbb{E}[z_m^2] \tag{A.18}$$

For any i, $\mathbb{E}[z_i^2] = \mathbb{V}\mathrm{ar}[z_i] + \mathbb{E}[z_i]^2 = 1 + 0 = 1$.

$$= L_{i,1}L'_{j,1} + \cdots + L_{i,m}L'_{j,m} = L_i^\top L'_j \tag{A.19}$$

Thus, the new covariance matrix factors into two Cholesky factors "glued" together:

$$\mathbb{C}\mathrm{ov}[\boldsymbol{y}_{\|k}] = \begin{pmatrix} L_{:p}L_{:p}^\top & L_{:p}L'^\top_{p+1:} \\ L'_{p+1:}L_{:p}^\top & L'_{p+1:}L'^\top_{p+1:} \end{pmatrix} = \begin{pmatrix} L_{:p} \\ L'_{p+1:} \end{pmatrix} \begin{pmatrix} L_{:p} \\ L'_{p+1:} \end{pmatrix}^\top \tag{A.20}$$

which is illustrated in Figure 14. Armed with this representation, we equate the log determinant of $\Theta_{\tilde{\imath},\tilde{\imath}\|k} \coloneqq \mathbb{C}\mathrm{ov}[\boldsymbol{y}_{\|k}]$ to the KL divergence in (2.12). Recalling that the determinant of a triangular matrix is the product of its diagonal entries,

$$\frac{1}{2}\operatorname{logdet}(\Theta_{\tilde{\imath},\tilde{\imath}\|k}) = \underbrace{\log(L_{1,1}) + \cdots + \log(L_{p,p})}_{\text{the same}} + \underbrace{\log(L'_{p+1,p+1}) + \cdots + \log(L'_{m,m})}_{\text{conditioned}}$$

Comparing to the KL divergence (2.12) and recalling that k is added to s_i if $i > p$,

$$\sum_{i=1}^{m} \log\left(\Theta_{i,i|s_i\setminus\{i\}}\right) = \underbrace{\log\left(\Theta_{1,1|s_1\setminus\{1\}}\right) + \cdots + \log\left(\Theta_{p,p|s_p\setminus\{p\}}\right)}_{\text{the same}} + \underbrace{\log\left(\Theta_{p+1,p+1|s_{p+1}\setminus\{p+1\}}\right) + \cdots + \log\left(\Theta_{m,m|s_m\setminus\{m\}}\right)}_{\text{conditioned}} \tag{A.21}$$

Since $L_{i,i}$ (and $L'_{i,i}$) is the square root of the posterior variance of the ith variable from the statistical perspective in (B.6), we have $2\log(L_{i,i}) = \log(\Theta_{i,i|s_i\setminus\{i\}})$ and so

$$\operatorname{logdet}(\Theta_{\tilde{\imath},\tilde{\imath}\|k}) = \sum_{i=1}^{m} \log\left(\Theta_{i,i|s_i\setminus\{i\}}\right) \tag{A.22}$$

∎

A.4. Aggregated computation of sparsity entries. We wish to compute the entries of the sparse Cholesky factor $L_{s_i,i}$ according to (2.11) in the aggregated sparsity setting of subsection 5.1. Recall that we have aggregated the column indices $i_1 \succ \cdots \succ i_m$ into $\tilde{\imath} = \{i_1, \ldots, i_m\}$ and $s_{\tilde{\imath}}$ denotes the aggregated sparsity pattern. The sparsity pattern for the ith column of the group is all those entries that satisfy lower triangularity, $s_i \coloneqq \{j \in s_{\tilde{\imath}} : j \succeq i\} \subseteq s_{\tilde{\imath}}$. Because each column's sparsity pattern is a subset of the overall sparsity pattern, it is possible to compute an outer approximation and specialize to each column efficiently. Assuming the sparsity entries $s_{\tilde{\imath}}$ are sorted according to $\prec$, let $Q \coloneqq \Theta^{-1}_{s_{\tilde{\imath}},s_{\tilde{\imath}}}$ be the precision of the aggregated sparsity pattern and let k be the ith column's index in $s_{\tilde{\imath}}$. Because $s_{\tilde{\imath}}$ is sorted according to $\prec$, the sparsity pattern for the ith column s_i is exactly the entries k and after.

$$L_{s_i,i} = \frac{\Theta^{-1}_{s_i,s_i}\boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top \Theta^{-1}_{s_i,s_i}\boldsymbol{e}_1}} = \frac{\left(Q^{-1}\right)^{-1}_{k:,k:}\boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top \left(Q^{-1}\right)_{k:,k:}\boldsymbol{e}_1}} = \frac{Q_{k:,k:|:k-1}\boldsymbol{e}_1}{\sqrt{\boldsymbol{e}_1^\top Q_{k:,k:|:k-1}\boldsymbol{e}_1}} \tag{A.23}$$

where the first equality follows from $Q^{-1} = \Theta_{s_{\tilde{\imath}},s_{\tilde{\imath}}}$ and s_i is the kth index of $s_{\tilde{\imath}}$ and after. We turn marginalization in precision into conditioning in covariance by (A.8).

So we want the kth column of Q, conditional on all columns before it. From (B.6), this can be directly read off the kth column of the Cholesky factor $L = \operatorname{chol}(Q)$ to compute (2.11) for each $i \in \tilde{\imath}$. However, computing $Q = \Theta^{-1}_{s_{\tilde{\imath}},s_{\tilde{\imath}}}$ by inverting $\Theta_{s_{\tilde{\imath}},s_{\tilde{\imath}}}$ and then additionally computing its Cholesky factor $L = \operatorname{chol}(Q)$ is a bit wasteful.

Instead of computing a *lower* triangular factor for the precision, we can compute an *upper* triangular factor for the covariance whose inverse transpose will be a lower triangular factor for the precision. Let $U = P^{\updownarrow}\operatorname{chol}(P^{\updownarrow}\Theta_{s_{\tilde{\imath}},s_{\tilde{\imath}}}P^{\updownarrow})P^{\updownarrow}$ where $P^{\updownarrow}$ is the order-reversing permutation; U is upper triangular and satisfies $UU^\top = \Theta_{s_{\tilde{\imath}},s_{\tilde{\imath}}}$ so $U^{-\top}U^{-1} = \Theta^{-1}_{s_{\tilde{\imath}},s_{\tilde{\imath}}} = Q$ and we see that $L = U^{-\top}$ is a lower triangular factor satisfying $LL^\top = Q$. Thus we can compute U as a Cholesky factor of the covariance and dynamically form the kth column of L by solving the triangular system $L_{:,k} = U^{-\top}\boldsymbol{e}_k$. This method is described in Algorithm 3.2, Figure 4, and Appendix A.2 of [27].

Appendix B. Computation in sparse Gaussian process selection.

B.1. Mutual information objective. The *mutual information* or *information gain* between two collections of random variables $\boldsymbol{y}_{\mathrm{Pr}}$ and $\boldsymbol{y}_{\mathrm{Tr}}$ is defined as

$$\mathbb{I}[\boldsymbol{y}_{\mathrm{Pr}}; \boldsymbol{y}_{\mathrm{Tr}}] \coloneqq \mathbb{H}[\boldsymbol{y}_{\mathrm{Pr}}] - \mathbb{H}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \tag{B.1}$$

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 monotonically increases with the determinant of its covariance matrix, minimizing the conditional entropy is equivalent to minimizing the determinant of the posterior covariance matrix. The determinant of a single prediction point is its variance so we can minimize the *conditional variance* of the target point.

Supposing $\boldsymbol{y}_{\mathrm{Pr}}$ is estimated by the posterior mean (2.1), the estimator is unbiased because $\mathbb{E}[\mathbb{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]] = \mathbb{E}[\boldsymbol{y}_{\mathrm{Pr}}]$ and so the expected mean squared error of the estimator $\mathrm{MSE} \coloneqq \mathbb{E}[(\boldsymbol{y}_{\mathrm{Pr}} - \mathbb{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}])^2]$ is simply the conditional variance because

$$\mathrm{MSE} = \mathbb{E}[\mathbb{E}[(\boldsymbol{y}_{\mathrm{Pr}} - \mathbb{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}])^2 \mid \boldsymbol{y}_{\mathrm{Tr}}]] = \mathbb{E}[\mathbb{V}\mathrm{ar}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]] = \mathbb{V}\mathrm{ar}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \tag{B.2}$$

from the quirk that the posterior covariance (2.2) does not depend on observing $\boldsymbol{y}_{\mathrm{Tr}}$.

So maximizing the mutual information is equivalent to minimizing the posterior variance (or log determinant) which is in turn equivalent to minimizing the expected mean squared error of the prediction. Yet another perspective arises from comparing the definition of mutual information (B.1) to the EV-VE identity (B.4),

$$\mathbb{H}[\boldsymbol{y}_{\mathrm{Pr}}] = \mathbb{H}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] + \mathbb{I}[\boldsymbol{y}_{\mathrm{Pr}}; \boldsymbol{y}_{\mathrm{Tr}}] \tag{B.3}$$

$$\mathbb{V}\mathrm{ar}[\boldsymbol{y}_{\mathrm{Pr}}] = \mathbb{E}[\mathbb{V}\mathrm{ar}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]] + \mathbb{V}\mathrm{ar}[\mathbb{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]] \tag{B.4}$$

On the left hand side, variance is monotone with entropy. On the right hand side, the conditional variance is monotone with conditional entropy. Because the two terms on the right hand side add to a constant, minimizing the conditional variance is equivalent to maximizing the variance of conditional expectation, which, by process of elimination, corresponds to the mutual information. We can intuitively interpret the variance of the conditional expectation as measuring the extent to which the estimator for $\boldsymbol{y}_{\mathrm{Pr}}$, the conditional expectation $\mathbb{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]$, varies with observed values of $\boldsymbol{y}_{\mathrm{Tr}}$.

B.2. Cholesky factorization as iterative conditioning. Factoring the joint covariance matrix Θ blocked as (A.8) by two steps of block Gaussisan elimination,

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

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

$$\mathrm{chol}(\Theta) = \begin{pmatrix} \mathrm{Id} & 0 \\ \Theta_{2,1}\Theta_{1,1}^{-1} & \mathrm{Id} \end{pmatrix} \begin{pmatrix} \mathrm{chol}(\Theta_{1,1}) & 0 \\ 0 & \mathrm{chol}(\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}) \end{pmatrix} \tag{B.5}$$

$$= \begin{pmatrix} \mathrm{chol}(\Theta_{1,1}) & 0 \\ \Theta_{2,1}\,\mathrm{chol}(\Theta_{1,1})^{-\top} & \mathrm{chol}(\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}) \end{pmatrix} \tag{B.6}$$

where the conditional expectation (2.1) corresponds to $\Theta_{2,1}\Theta_{1,1}^{-1}$ and the conditional covariance (2.2) corresponds to $\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}$. Blocking Θ such that 1 is the current column and 2 is all the indices past 1, we observe the column of the Cholesky factor is a conditional covariance $\Theta_{2,1}$ divided by $\mathrm{chol}(\Theta_{1,1})^{\top}$, the square root of the variance (the Cholesky factor of a scalar is its square root). We conclude that the kth column of the Cholesky factor equals $\boldsymbol{u}$ in (3.1) since iteratively conditioning on the columns $i_1, i_2, \ldots, i_{k-1}$ is equivalent to conditioning on I by the quotient rule (2.4).

B.3. Updating precision after insertion. Given the precision matrix $\Theta_{1,1}^{-1}$ we wish to compute the inverse of the covariance matrix $\Theta_{1,1}$ with added rows and columns, i.e. to compute Θ^{-1} for $\Theta = \begin{pmatrix}\Theta_{1,1} & \Theta_{1,2}\\ \Theta_{2,1} & \Theta_{2,2}\end{pmatrix}$. Using the same block $LDL^\top$ factorization as (B.6) with the Schur complement $\Theta_{2,2} - \Theta_{2,1}\Theta_{1,1}^{-1}\Theta_{1,2}$ denoted $\Theta_{2,2|1}$,

$$\begin{pmatrix}\Theta_{1,1} & \Theta_{1,2}\\ \Theta_{2,1} & \Theta_{2,2}\end{pmatrix} = \begin{pmatrix}\mathrm{Id} & 0\\ \Theta_{2,1}\Theta_{1,1}^{-1} & \mathrm{Id}\end{pmatrix}\begin{pmatrix}\Theta_{1,1} & 0\\ 0 & \Theta_{2,2|1}\end{pmatrix}\begin{pmatrix}\mathrm{Id} & \Theta_{1,1}^{-1}\Theta_{1,2}\\ 0 & \mathrm{Id}\end{pmatrix} \tag{B.7}$$

Inverting both sides of the equation and multiplying,

$$\Theta^{-1} = \begin{pmatrix}\mathrm{Id} & -\Theta_{1,1}^{-1}\Theta_{1,2}\\ 0 & \mathrm{Id}\end{pmatrix}\begin{pmatrix}\Theta_{1,1}^{-1} & 0\\ 0 & \Theta_{2,2|1}^{-1}\end{pmatrix}\begin{pmatrix}\mathrm{Id} & 0\\ -\Theta_{2,1}\Theta_{1,1}^{-1} & \mathrm{Id}\end{pmatrix} \tag{B.8}$$

$$= \begin{pmatrix}\Theta_{1,1}^{-1} + (\Theta_{1,1}^{-1}\Theta_{1,2})\Theta_{2,2|1}^{-1}(\Theta_{2,1}\Theta_{1,1}^{-1}) & -(\Theta_{1,1}^{-1}\Theta_{1,2})\Theta_{2,2|1}^{-1}\\ -\Theta_{2,2|1}^{-1}(\Theta_{2,1}\Theta_{1,1}^{-1}) & \Theta_{2,2|1}^{-1}\end{pmatrix} \tag{B.9}$$

In the context of adding a new entry k 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 C.1. If we let $\boldsymbol{v} \coloneqq \Theta_{I,I}^{-1}\Theta_{I,k}$, then

$$= \begin{pmatrix}\Theta_{I,I}^{-1} + \Theta_{k,k|I}^{-1}\boldsymbol{v}\boldsymbol{v}^T & -\Theta_{k,k|I}^{-1}\boldsymbol{v}\\ -\Theta_{k,k|I}^{-1}\boldsymbol{v}^\top & \Theta_{k,k|I}^{-1}\end{pmatrix} \tag{B.10}$$

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

B.4. Updating precision after conditioning. Given the precision matrix of the prediction points conditional on the selected entries, $\Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1}$, we want to take into account selecting an index k, or to compute $\Theta_{\mathrm{Pr},\mathrm{Pr}|I,k}^{-1}$, which is a rank-one update to the covariance matrix (but not necessarily the precision matrix) from (3.1). We can directly apply the Sherman–Morrison–Woodbury formula which states that:

$$\Theta_{1,1|2}^{-1} = \Theta_{1,1}^{-1} + (\Theta_{1,1}^{-1}\Theta_{1,2})\,\Theta_{2,2|1}^{-1}\,(\Theta_{2,1}\Theta_{1,1}^{-1}) \tag{B.11}$$

Expanding the Schur complement from the posterior covariance (2.2),

$$(\Theta_{1,1} - \Theta_{1,2}\Theta_{2,2}^{-1}\Theta_{2,1})^{-1} = \Theta_{1,1}^{-1} + (\Theta_{1,1}^{-1}\Theta_{1,2})\,\Theta_{2,2|1}^{-1}\,(\Theta_{2,1}\Theta_{1,1}^{-1}) \tag{B.12}$$

For brevity of notation, define the vectors $\boldsymbol{u} \coloneqq \Theta_{1,2}$ and $\boldsymbol{v} \coloneqq \Theta_{1,1}^{-1}\Theta_{1,2} = \Theta_{1,1}^{-1}\boldsymbol{u}$.

$$(\Theta_{1,1} - \Theta_{2,2}^{-1}\boldsymbol{u}\boldsymbol{u}^\top)^{-1} = \Theta_{1,1}^{-1} + \Theta_{2,2|1}^{-1}\boldsymbol{v}\boldsymbol{v}^\top \tag{B.13}$$

So we see that a rank-one update to $\Theta_{1,1}$ then inverting is a rank-one update to $\Theta_{1,1}^{-1}$. In our context, $\Theta_{1,1} = \Theta_{\mathrm{Pr},\mathrm{Pr}|I}, \boldsymbol{u} = \Theta_{\mathrm{Pr},k|I}, \Theta_{2,2} = \Theta_{k,k|I}$ so $\Theta_{2,2|1}^{-1} = \Theta_{k,k|\mathrm{Pr},I}^{-1}$ (from the quotient rule (2.4)). By definition, $\boldsymbol{v} = \Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1}\boldsymbol{u}$. Thus, the update in context is

$$\left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I} - \frac{\Theta_{\mathrm{Pr},k|I}\Theta_{\mathrm{Pr},k|I}^\top}{\Theta_{k,k|I}}\right)^{-1} = \Theta_{1,1}^{-1} + \Theta_{k,k|\mathrm{Pr},I}^{-1}\boldsymbol{v}\boldsymbol{v}^\top \tag{B.14}$$

which is the update in line 18 of Algorithm C.4. Since the update is a rank-one update, it can be computed in $\mathcal{O}(|\mathrm{Pr}|^2) = \mathcal{O}(m^2)$.

B.5. Updating log determinant after conditioning. In the context of subsection 5.2, given the log determinant of the covariance matrix of the prediction points conditional on the selected entries, $\operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}|I})$, we wish to compute the log determinant after we add an index k to I, that is, to compute $\operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}|I,k})$.

Proof of Equation (5.4). From the posterior (3.1), selecting a new point is a rank-one downdate on the current covariance matrix:

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

Applying the matrix determinant lemma,

$$= \operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}|I}) + \log\left(1 - \frac{\Theta_{\mathrm{Pr},k|I}^{\top}\Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1}\Theta_{\mathrm{Pr},k|I}}{\Theta_{k,k|I}}\right) \tag{B.15}$$

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

$$= \operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}|I}) + \log\left(\frac{\Theta_{k,k|I} - \Theta_{k,\mathrm{Pr}|I}\Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1}\Theta_{\mathrm{Pr},k|I}}{\Theta_{k,k|I}}\right) \tag{B.16}$$

By the quotient rule (2.4), we combine the conditioning.

$$= \operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}|I}) + \log\left(\frac{\Theta_{k,k|I,\mathrm{Pr}}}{\Theta_{k,k|I}}\right) \tag{B.17}$$

∎

Appendix C. Algorithms. Reference Python and Cython code for all algorithms can be found at https://github.com/stephen-huan/conditional-knn.

C.1. Single-target selection. To compute the objective (3.2) for each candidate index j, we start with the unconditional variance $\Theta_{j,j}$ and covariance $\Theta_{\mathrm{Pr},j}$, updating these quantities when an index k is selected. If we can compute $\boldsymbol{u}$ for k, then we can update j's conditional variance by subtracting u_j^2 and update its conditional covariance by subtracting $u_j u_{\mathrm{Pr}}$. We have two efficient strategies to compute $\boldsymbol{u}$.

Precision approach. (Algorithm C.1). The precision of the selected entries $\Theta_{I,I}^{-1}$ can be updated whenever a new index is added to I in time complexity $\mathcal{O}(s^2)$ (see Appendix B.3). With $\Theta_{I,I}^{-1}$ in hand $\boldsymbol{u}$ is computed directly according to the posterior covariance (2.2). On each of the s rounds of selection it takes $\mathcal{O}(s^2)$ to update the precision and $\mathcal{O}(Ns)$ to compute $\boldsymbol{u}$, for an overall time complexity of $\mathcal{O}(Ns^2)$.

Cholesky approach. (Algorithm C.2). The second strategy exploits the iterative conditioning in Cholesky factorization: we maintain only the first $|I|$ columns corresponding to selected points from the Cholesky factor of the joint covariance matrix between the training and target points; the kth column of this factor is precisely $\boldsymbol{u}$ (see Appendix B.2 for details). Adding a new column to the end of this factor can be efficiently computed with left-looking in time complexity $\mathcal{O}(Ns)$ (see Algorithm C.6), resulting in the same $\mathcal{O}(Ns^2)$ time complexity as the explicit precision approach.

C.2. Multiple-target selection. Context is provided in subsection 5.2.

Precision approach. (Algorithm C.4). The precision approach initializes the precision $\Theta_{\mathrm{Pr},\mathrm{Pr}}^{-1}$ for m prediction points in time $\mathcal{O}(m^3)$ and computes the initial conditional variances $\Theta_{k,k|\mathrm{Pr}}$ for N candidates by direct application of the posterior (2.2) in time $\mathcal{O}(Nm^2)$. For each of the s rounds of selecting candidates, it costs $\mathcal{O}(s^2)$ and $\mathcal{O}(m^2)$ 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 are given in Appendix B.4. Given the precisions, $\boldsymbol{u}$ (3.1) and $\boldsymbol{u}_{\mathrm{Pr}} \coloneqq \frac{\Theta_{:,k|I,\mathrm{Pr}}}{\sqrt{\Theta_{k,k|I,\mathrm{Pr}}}}$ are computed as usual according to (2.2) in time $\mathcal{O}(Ns)$ and $\mathcal{O}(Nm)$. Finally, for each candidate j the conditional variance $\Theta_{j,j|I}$ is updated by subtracting u_j^2, the conditional covariance $\Theta_{\mathrm{Pr},k|I}$ is updated for each prediction point index c by subtracting $u_j u_c$, and the conditional variance $\Theta_{j,j|I,\mathrm{Pr}}$ is updated by subtracting $(u_{\mathrm{Pr}})_j^2$. The total time complexity is $\mathcal{O}(Ns^2 + Nm^2 + m^3)$. The space complexity is $\mathcal{O}(s^2+m^2)$ to store both precision matrices as well as $\mathcal{O}(Nm)$ memory to store the conditional covariances of each candidate with each target.

Cholesky approach. (Algorithm C.5). The Cholesky approach storing two partial Cholesky factors is considerably simpler than the explicit precision approach from the symmetry of the factors and the lack of explicitly computing conditional covariances. We first add each prediction point to one Cholesky factor by left-looking (Algorithm C.6) exactly as in the single-target algorithm, for a time complexity of $\mathcal{O}((N+m)m^2)$. Whenever a candidate is selected, both Cholesky factors are updated in time $\mathcal{O}((N+m)(m+s))$ dominated by updating the initialized Cholesky factor. The desired covariances $\boldsymbol{u}$ and $\boldsymbol{u}_{\mathrm{Pr}}$ are directly read off the columns of the Cholesky factors and both conditional variances $\Theta_{j,j|I}$ and $\Theta_{j,j|I,\mathrm{Pr}}$ are computed like the precision approach. For s points selected, the total time complexity is $\mathcal{O}(Ns^2+Nm^2+m^3)$. The space complexity is $\mathcal{O}(Ns+Nm+m^2)$ to store two $(N+m)\times(m+s)$ sized partial Cholesky factors for the joint covariance matrix of training and prediction points.

C.3. Partial selection. We now compute the log determinant of the target covariance matrix if a selected index k only conditions *suffixes* of the target points, skipping the first p targets, say. Following the suggestive form of the target covariance matrix (5.2), we maintain a Cholesky factor L taking from the factor of the joint covariance matrix Θ only the $|I| + m$ columns corresponding to selected points and target points, sorted w.r.t $\succ$. Inserting the index k into its proper column in L,

$$L \leftarrow \begin{pmatrix} L_{:,:p} & \boldsymbol{u} & L'_{:,p+1:} \end{pmatrix} \qquad \boldsymbol{u} = \frac{\Theta_{:,k|:p}}{\sqrt{\Theta_{k,k|:p}}} \tag{C.1}$$

where L' is the Cholesky factor of the covariance matrix conditional on k and the updated form of L after insertion is derived from the statistical perspective in (B.6).

To account for this insertion, the first p columns $L_{:,:p}$ are unchanged, the newly inserted column $\boldsymbol{u}$ of the form (3.1) can be computed as usual with left-looking (see Algorithm C.6), and the remaining columns $L'_{:,p+1:}$ are from the Cholesky factor of Θ after a rank-one downdate by $\boldsymbol{u}$, allowing the updated factor $L'_{:,p+1:}$ to be efficiently computed from the original factor $L_{:,p+1:}$. Specifically we adapt Lemma 1 of [16] to make no assumption on the row ordering of L, in order to implement the downdate in-place (see Algorithm C.10). For a Cholesky factor of size $R \times C$, the time complexity of the downdate is $\mathcal{O}(RC)$ compared to $\mathcal{O}(RC^2)$ if recomputed from scratch.

For N candidates, m targets, and a budget of s points to select, the Cholesky factor L has size $(N + m) \times (s + m)$. When adding a new index k that ignores the first p targets, the first p columns of L are correspondingly ignored, the new column $\boldsymbol{u}$ is computed with left-looking in time $\mathcal{O}((N + m)p)$, and the columns past p are updated with downdating in time $\mathcal{O}((N + m)(s + m - p))$ for a total time complexity of $\mathcal{O}(s(N+m)(s+m))$. With the factor L in hand, we discuss actually computing the log determinant objective (5.3) after adding a candidate index j. We directly compute each individual target point's variance, conditional on j as well as previously selected points, and simply add these variances together to get the overall objective.

We proceed by induction on the columns of L, assuming we know the candidate's variance conditional on all points prior; starting at the first column we know the candidate's unconditional variance. We proceed to the next column by reading the requisite statistical quantities from L (C.1) and conditioning (2.4). For the ith column,

$$\Theta_{i,i|:i-1} = L_{i,i}^2 \tag{C.2}$$

$$\Theta_{j,i|:i-1} = L_{j,i} \cdot L_{i,i} \tag{C.3}$$

$$\Theta_{i,i|:i-1,j} = \Theta_{i,i|:i-1} - \Theta_{j,i|:i-1}^2 / \Theta_{j,j|:i-1} \tag{C.4}$$

$$\Theta_{j,j|:i-1,i} = \Theta_{j,j|:i-1} - \Theta_{j,i|:i-1}^2 / \Theta_{i,i|:i-1} = \Theta_{j,j|:i} \tag{C.5}$$

where (C.5) satisfies the conditions of the inductive hypothesis for the next column and (C.4) is the desired conditional variance when the ith point is a target.

For each of the N candidates, it requires $\mathcal{O}(1)$ work per column for $s + m$ columns. Over s selections, the total time complexity is $\mathcal{O}(sN(s+m))$ which is dominated by the time to update the Cholesky factor, meaning the partial selection algorithm matches the asymptotic time complexity of the multiple-target algorithm. However, the downdate of the Cholesky factor is implemented with repeated BLAS level-one `daxpy` operations while the bulk of left-looking takes place in a BLAS level-two `dgemv` operation. Higher level operations often have better constant-factor performance for the same asymptotic time complexity. In addition, the objective for the partial selection algorithm is more complicated to compute than its multiple-target counterpart. Psuedocode for the algorithm is provided in Algorithm C.7 to Algorithm C.10.

Algorithm C.1 Point selection by explicit precision

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

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

Algorithm C.2 Point selection by Cholesky factorization

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

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

FIG. 15. *Algorithms for single-target selection.*

Algorithm C.3 Directed sparse Gaussian process regression by selection

Input: $X_{\mathrm{Tr}}, \boldsymbol{y}_{\mathrm{Tr}}, X_{\mathrm{Pr}}, K(\cdot,\cdot), s$
Output: $\mathbb{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}], \mathbb{C}\mathrm{ov}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]$

Compute I using Algorithm C.1 or Algorithm C.2
$\Theta_{\mathrm{Tr,Tr}} \leftarrow K(X_{\mathrm{Tr}}[I], X_{\mathrm{Tr}}[I])$
$\Theta_{\mathrm{Pr,Pr}} \leftarrow K(X_{\mathrm{Pr}}, X_{\mathrm{Pr}})$
$\Theta_{\mathrm{Tr,Pr}} \leftarrow K(X_{\mathrm{Tr}}[I], X_{\mathrm{Pr}})$
$\mathbb{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \leftarrow \Theta_{\mathrm{Pr,Tr}} \Theta_{\mathrm{Tr,Tr}}^{-1} \boldsymbol{y}_{\mathrm{Tr}}[I]$
$\mathbb{C}\mathrm{ov}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \leftarrow \Theta_{\mathrm{Pr,Pr}} - \Theta_{\mathrm{Tr,Pr}}^\top \Theta_{\mathrm{Tr,Tr}}^{-1} \Theta_{\mathrm{Tr,Pr}}$
return $\mathbb{E}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}], \mathbb{C}\mathrm{ov}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}]$

Algorithm C.4 Multiple-target selection by explicit precision

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

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

Algorithm C.5 Multiple-target selection by Cholesky factorization

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

$N \leftarrow |X_{\mathrm{Tr}}|$
$m \leftarrow |X_{\mathrm{Pr}}|$
$X \leftarrow \begin{pmatrix} X_{\mathrm{Tr}} \\ X_{\mathrm{Pr}} \end{pmatrix}$
$I \leftarrow \emptyset$
$-I \leftarrow \{1, \ldots, N\}$
$L \leftarrow \mathbf{0}^{(N+m)\times s}$
$L_{\mathrm{Pr}} \leftarrow \mathbf{0}^{(N+m)\times(s+m)}$
$\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I}) \leftarrow \mathrm{diag}(K(X_{\mathrm{Tr}}, X_{\mathrm{Tr}}))$
$\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}}) \leftarrow \mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I})$
for $i \in \{1, \ldots, m\}$ **do**
  Update L_{Pr} and $\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}})$ with $k = N + i$ by Algorithm C.6.
end for
while $|-I| > 0$ and $|I| < s$ **do**
  $k \leftarrow \min_{j\in -I} \frac{\Theta_{j,j|I,\mathrm{Pr}}}{\Theta_{j,j|I}}$
  $I \leftarrow I \cup \{k\}$
  $-I \leftarrow -I \setminus \{k\}$
  $i \leftarrow |I|$
  Update L and $\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I})$ by Algorithm C.6.
  Update L_{Pr} and $\mathrm{diag}(\Theta_{\mathrm{Tr},\mathrm{Tr}|I,\mathrm{Pr}})$ with $i = |I| + m$ by Algorithm C.6.
end while
return I

Algorithm C.6 Update Cholesky factor

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

$N \leftarrow |\mathrm{diag}(\Theta)|$
$L_{:,i} \leftarrow K(X, X_k) - L_{:,:i-1}L_{k,:i-1}^\top$
$L_{:,i} \leftarrow \frac{L_{:,i}}{\sqrt{L_{k,i}}}$
for $j \in \{1, \ldots, N\}$ **do**
  $\Theta_{j,j} \leftarrow \Theta_{j,j} - L_{j,i}^2$
end for

FIG. 16. *Algorithms for multiple-target selection.*

Algorithm C.7 Partial point selection

Input: $X_{\mathrm{Tr}}, X_{\mathrm{Pr}}, \prec, K(\cdot,\cdot), s$
Output: I

$N \leftarrow |X_{\mathrm{Tr}}|$
$m \leftarrow |X_{\mathrm{Pr}}|$
$X \leftarrow \begin{pmatrix} X_{\mathrm{Tr}} \\ X_{\mathrm{Pr}} \end{pmatrix}$
$I \leftarrow \emptyset$
$-I \leftarrow \{1, \ldots, N\}$
$\mathcal{I} \leftarrow \emptyset$
$L \leftarrow \mathbf{0}^{(N+m)\times(s+m)}$
for $i \in \{1, \ldots, m\}$ **do**
  Update L and $\mathcal{I}$ with $k = N + i$ by Algorithm C.10.
end for
while $|-I| > 0$ and $|I| < s$ **do**
  $i \leftarrow |I| + m$
  best $\leftarrow \infty$
  for $j \in -I$ **do**
    Find j's score by Algorithm C.9
    if score $<$ best **then**
      best $\leftarrow$ score
      $k \leftarrow j$
    end if
  end for
  $I \leftarrow I \cup \{k\}$
  $-I \leftarrow -I \setminus \{k\}$
  Update L and $\mathcal{I}$ by Algorithm C.10.
end while
return I

Algorithm C.8 Find j's index in $\mathcal{I}$

Input: $\mathcal{I}, \prec, i, j$
Output: p

for $p \in \{1, \ldots, i\}$ **do**
  if $j \succeq \mathcal{I}_p$ **then**
    return p
  end if
end for
return $i + 1$

Algorithm C.9 Compute j's objective

Input: $\mathcal{I}, \prec, i, j, L, X, N, K(\cdot,\cdot)$
Output: score

Compute p by Algorithm C.8
score $\leftarrow 0$
$\Theta_{j,j} = K(X_j, X_j)$
for $c \in \{1, \ldots, i\}$ **do**
  $\mathcal{C} \leftarrow \mathcal{I}_c$
  $\Theta_{\mathcal{C},\mathcal{C}|:c-1} \leftarrow L^2_{\mathcal{C},c}$
  $\Theta_{j,\mathcal{C}|:c-1} \leftarrow L_{j,c} \cdot \Theta_{\mathcal{C},\mathcal{C}|:c-1}$
  if $\mathcal{C} \geq N$ **then**
    $\sigma^2 \leftarrow \Theta_{\mathcal{C},\mathcal{C}|:c-1}$
    if $c \geq p$ **then**
      $\sigma^2 \leftarrow \Theta_{\mathcal{C},\mathcal{C}|:c-1} - \frac{\Theta^2_{j,\mathcal{C}|:c-1}}{\Theta_{j,j|:c-1}}$
    end if
    score $\leftarrow$ score $+ \log(\sigma^2)$
  end if
  $\Theta_{j,j|:c} \leftarrow \Theta_{j,j|:c-1} - \frac{\Theta^2_{j,\mathcal{C}|:c-1}}{\Theta_{\mathcal{C},\mathcal{C}|:c-1}}$
end for
return score

Algorithm C.10 Insert k into L

Input: $\mathcal{I}, \prec, i, k, L, X, K(\cdot,\cdot)$
Output: $L, \mathcal{I}$

Find p by Algorithm C.8 for $j = k$
$\mathcal{I} \leftarrow \{\mathcal{I}_{:p-1}, j, \mathcal{I}_{p:}\}$
$\boldsymbol{u} \leftarrow K(X, X_k) - L_{:,:i-1} L^\top_{k,:i-1}$
$\boldsymbol{u} \leftarrow \boldsymbol{u} / \sqrt{L_{k,i}}$
$\boldsymbol{v} \leftarrow \boldsymbol{u}$
for $c \in \{p, \ldots, i\}$ **do**
  $\mathcal{C} \leftarrow \mathcal{I}_c$
  $\gamma \leftarrow \sqrt{L^2_{\mathcal{C},c} - \boldsymbol{v}^2_{\mathcal{C}}}$
  $\alpha \leftarrow L_{\mathcal{C},c} / \gamma$
  $\beta \leftarrow \boldsymbol{v}_{\mathcal{C}} / \gamma$
  $L_{:,c} \leftarrow \alpha L_{:,c} - \beta \boldsymbol{v}$
  $\boldsymbol{v} \leftarrow \frac{1}{\alpha}\boldsymbol{v} - \frac{\beta}{\alpha} L_{:,c}$
end for
$L \leftarrow \begin{pmatrix} L_{:,:p-1} & \boldsymbol{u} & L_{:,p:} \end{pmatrix}$
return $L, \mathcal{I}$

FIG. 17. *Algorithms for partial selection.*