# EXPERIMENTAL DESIGN FOR FAST LINEAR ALGEBRA

STEPHEN HUAN[*] AND FLORIAN SCHÄFER[†]

**Abstract.** experimental design for linear algebra

**Key words.** to do

**AMS subject classifications.**

## 1. Introduction.

[3] test citation

## 2. Greedy selection for directed inference.

### 2.1. Conditional *k*-th nearest neighbors.

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

**Algorithm 2.1** Idealized $k$-NN regression

```
Given $(X_{\text{Tr}}, \boldsymbol{y}_{\text{Tr}}) = \{(\boldsymbol{x}_1, y_1), \dots, (\boldsymbol{x}_n, y_n)\}$ and $\boldsymbol{x}_{\text{Pr}}$
    1. Select the $k$ points in $X_{\text{Tr}}$ most informative about $\boldsymbol{x}_{\text{Pr}}$
    2. Combine the labels of the selected points to generate a prediction
```

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

**Algorithm 2.2** $k$-NN regression

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

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

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

**Algorithm 2.3** Kernel $k$-NN regression

```
Given kernel function $K(\boldsymbol{x}, \boldsymbol{y})$
    1. Select the $k$ points $\{\boldsymbol{x}_{i_1}, \dots, \boldsymbol{x}_{i_k}\} \subseteq X_{\text{Tr}}$ most similar to $\boldsymbol{x}_{\text{Pr}}$
    2. Compute $\boldsymbol{y}_{\text{Pr}}$ by an average weighted by similarity
```

[*]Georgia Institute of Technology

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

Although the kernel $k$-NN approach is more general than its normed counterpart, it still suffers from a fundamental issue. Suppose the closest point to $\boldsymbol{x}_{\mathrm{Pr}}$ has many duplicates in the training set. Then the algorithm will select the same point multiple times, even though in some sense the duplicate point has stopped giving additional information about the prediction point. In order to fix this issue, we should be selecting new points *conditional* on the points we've already selected. This preserves the idealized algorithm of selecting points based on the information they tell us about the prediction point — once we've selected a point, conditioning on it reduces the information similar points tells us, encouraging diverse point selection.

**Algorithm 2.4** Conditional kernel $k$-NN regression

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

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

mention sensor placement/spatial statistics perspective/literature

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

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

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

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

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

cite sparse Gaussian regression papers

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

$$\mathrm{I}[\boldsymbol{y}_{\mathrm{Pr}}; \boldsymbol{y}_{\mathrm{Tr}}] = \mathrm{H}[\boldsymbol{y}_{\mathrm{Pr}}] - \mathrm{H}[\boldsymbol{y}_{\mathrm{Pr}} \mid \boldsymbol{y}_{\mathrm{Tr}}] \tag{2.3}$$

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

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

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

**2.4. A greedy approach.** We now consider how to efficiently minimize the conditional variance objective using a greedy approach. At each iteration, we pick the training point which most reduces the conditional variance of the prediction point. Let $I = \{i_1, i_2, \ldots, i_j\}$ be the set of indexes of training points selected already. Let the prediction point have index $n$, the last index. For a candidate index $k$, we update the covariance matrix after conditioning on $y_k$, in addition to the indices already selected according to (2.2):

$$\begin{aligned}
\Theta_{:,:\mid I,k} &= \Theta_{:,:\mid I} - \Theta_{:,k\mid I}\Theta_{k,k\mid I}^{-1}\Theta_{k,:\mid I} \\
&= \Theta_{:,:\mid I} - \frac{\Theta_{:,k\mid I}\Theta_{:,k\mid I}^{\top}}{\Theta_{kk\mid I}} \qquad (2.4)
\end{aligned}$$

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

$$\frac{\operatorname{Cov}[\boldsymbol{y}_{\mathrm{Pr}}, \boldsymbol{y}_{\mathrm{Tr},k}]^2}{\operatorname{Var}[\boldsymbol{y}_{\mathrm{Tr},k}, \boldsymbol{y}_{\mathrm{Tr},k}]} = \frac{\Theta_{nk|I}^2}{\Theta_{kk|I}} \tag{2.5}$$

For each candidate $k$, we need to keep track of its conditional variance and conditional covariance with the prediction point after conditioning on the points already selected to compute (2.5). We then simply choose the candidate with the largest decrease in predictive variance. To keep track of the conditional variance and covariance, we can simply start with the initial values given by $\Theta_{kk}$ and $\Theta_{nk}$ and update after selecting an index $j$. We compute $\boldsymbol{u}$ for $j$ directly according to (2.2) and update $k$'s conditional variance by subtracting $u_k^2$ and update its conditional covariance by subtracting $u_k u_n$.

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

The second approach is to take advantage of the quotient rule of Schur complementation. Stated statistically, the quotient rule states that conditioning on $I$ and then conditioning on $J$ is the same as conditioning on $I \cup J$. Because computing the Cholesky factorization of $\Theta$ involves sequence of rank-one updates to $\Theta$, Cholesky factorization can be viewed as iterative conditioning, where the $i$th column of the Cholesky factor corresponds precisely to the corresponding $\boldsymbol{u}$ for $i$ since a iterative sequence of conditioning on $i_1, i_2 \ldots$ is equivalent to conditioning on $I$ by the quotient rule. The Cholesky factorization can be efficiently computed without excess dependence on $N$ with left-looking, so the conditioning only happens when we need it. For each of the $s$ rounds of selection, it costs $\mathcal{O}(Ns)$ to compute the next column of the Cholesky factorization, for a total cost of $\mathcal{O}(Ns^2)$, matching the time complexity of the explicit inverse approach.

psuedocode

**Algorithm 2.5** Point selection with explicit inverse
```
Input: $\Theta$
Output: $L$
```

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

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

We now need to be able to efficiently compute the effect of selecting an index $k$ on the log determinant. From (2.4), we know that selecting an index is a rank-one update on the prediction covariance. Using the matrix determinant lemma,

(2.6)
$$\begin{aligned}\operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I\cup\{k\}}\right) &= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I} - \frac{\Theta_{\mathrm{Pr},k|I}\Theta_{\mathrm{Pr},k|I}^{\top}}{\Theta_{kk|I}}\right)\\ &= \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I}\right) + \log\left(1 - \frac{\Theta_{\mathrm{Pr},k|I}^{\top}\Theta_{\mathrm{Pr},\mathrm{Pr}|I}^{-1}\Theta_{\mathrm{Pr},k|I}}{\Theta_{kk|I}}\right)\end{aligned}$$

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

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

By the quotient rule, we combine the conditioning:

$$(2.7)\qquad = \operatorname{logdet}\left(\Theta_{\mathrm{Pr},\mathrm{Pr}|I}\right) + \log\left(\frac{\Theta_{kk|I,\mathrm{Pr}}}{\Theta_{kk|I}}\right)$$

The greedy objective (2.7) tells us that to minimize the log determinant, we can simply select the index $k$ with the smallest ratio between the conditional variance after conditioning on the previously selected points as well as the prediction points, and the conditional variance after just conditioning on the selected points. Intuitively this tells us that we can place sensors *backwards*, where we imagine placing sensors at the *prediction points* instead of the candidates. We then measure the conditional variance at a candidate, and pick the candidate whose conditional variance decreases the most (relative from what it started out as). Intuitively, these candidates are likely to give information about the prediction points, because the prediction points give information about the candidate.

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

For the inverse algorithm, using (2.2) directly, for $m$ prediction points it costs $\mathcal{O}(m^3)$ to compute $\Theta_{\mathrm{Pr},\mathrm{Pr}}^{-1}$ and then $\mathcal{O}(Nm^2)$ to compute $\Theta_{kk|\mathrm{Pr}}$ for the $N$ candidates $k$. For each of the $s$ rounds of selecting candidates, it costs $s^2$ and $m^2$ to update the inverses $\Theta_{I,I}^{-1}$ and $\Theta_{\mathrm{Pr},\mathrm{Pr}}^{-1}$ respectively, where $\Theta_{\mathrm{Pr},\mathrm{Pr}}^{-1}$ is efficiently updated after the rank-one update in (2.6) with the Sherman–Morrison–Woodbury formula. Given the inverses, $\boldsymbol{u} = \frac{\Theta_{:,k|I}}{\sqrt{\Theta_{kk|I}}}$ and $\boldsymbol{u}_{\mathrm{Pr}} = \frac{\Theta_{:,k|I,\mathrm{Pr}}}{\sqrt{\Theta_{kk|I,\mathrm{Pr}}}}$ are computed as usual according to (2.2) in time $Ns$ and $Nm$. Finally, for each candidate $j$ the conditional variance $\Theta_{jj|I}$ is updated by subtracting $u_j^2$, the conditional covariance $\Theta_{\mathrm{Pr},k|I}$ is updated for each prediction point index $c$ each by subtracting $u_j u_c$, and the conditional variance

$\Theta_{jj|I,\mathrm{Pr}}$ is updated by subtracting ${u_{\mathrm{Pr}}}_j^2$. The total time complexity after simplification is $\mathcal{O}(Ns^2 + Nm^2 + m^3)$.

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

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

psuedocode

**Algorithm 2.6** Multiple prediction point selection with explicit inverse

```
Input: $\Theta$
Output: $L$
```

### 2.6. Near optimality by submodularity.

## 3. Greedy selection for *global* approximation by KL-minimization.

We have a covariance matrix $\Theta$ and wish to compute the Cholesky factorization of $\Theta$ into a lower triangular factor $L$ such that $\Theta = LL^\top$. justify importance/downstream applications of Cholesky factorization. This can be done in $\mathcal{O}(N^3)$ with standard algorithms, which is often prohibitive. Recall the problem of inference in Gaussian process regression as described in subsection 2.2 also took $\mathcal{O}(N^3)$ to invert the covariance matrix $\Theta$. Thus, similar to Guassian process regression, we will use *sparsity* to mitigate the computational cost. In fact, we will be able to re-use our previous algorithms Algorithms 2.5 and 2.6 on each column of the Cholesky factorization.

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

Because of sparsity, we can only get an approximate Cholesky factor $L$, $\hat{L}$ belonging to a pre-specified sparsity pattern $S$ — a set of (row, column) indices that are allowed to be nonzero. In order to measure the performance of the estimator, we treat the matrices as covariance matrices of centered Gaussian processes (mean $\mathbf{0}$). In order to compare the resulting distributions, we use the *KL divergence* according to [2], or the expected difference in log-densities:

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

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

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

Theorem 3.1. *[2]. The non-zero entries of the ith column of $L$ in (3.1) are:*

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

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

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

Because $L$ is lower triangular, its determinant is just the product of its diagonal entries:

$$= -2 \sum_{i=1}^{N} \log(L_{ii}) - \operatorname{logdet}(\Theta)$$

Plugging (3.3) for its diagonal entry,

$$= -\sum_{i=1}^{N} \log(\boldsymbol{e}_1^\top \Theta_{s_i, s_i}^{-1} \boldsymbol{e}_1) - \operatorname{logdet}(\Theta)$$

$$= \sum_{i=1}^{N} \log \left( (\boldsymbol{e}_1^\top \Theta_{s_i, s_i}^{-1} \boldsymbol{e}_1)^{-1} \right) - \operatorname{logdet}(\Theta) \tag{3.4}$$

In order to maximize (3.4), we can maximize over each column independently, since each term in the sum only depends on a single column. We want to minimize $(\boldsymbol{e}_1^\top \Theta_{s_i, s_i}^{-1} \boldsymbol{e}_1)^{-1}$, the term corresponding to the diagonal entry in the inverse of the submatrix of $\Theta$ corresponding to the entries we've taken. We can give this value statistical interpretation by taking advantage of Schur complementation:

$$\Theta_{\mathrm{Pr}, \mathrm{Pr} \mid \mathrm{Tr}} = ((\Theta^{-1})_{\mathrm{Pr}, \mathrm{Pr}})^{-1} \tag{3.5}$$

where $\Theta$ is blocked according to

$$\Theta = \begin{pmatrix} \Theta_{\mathrm{Tr}, \mathrm{Tr}} & \Theta_{\mathrm{Tr}, \mathrm{Pr}} \\ \Theta_{\mathrm{Pr}, \mathrm{Tr}} & \Theta_{\mathrm{Pr}, \mathrm{Pr}} \end{pmatrix} \tag{3.6}$$

Thus, we see that

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

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

**3.1. Aggregated sparsity pattern.** We can also use the Gaussian process regression viewpoint to efficiently aggregate multiple columns, that is, to use the same sparsity pattern for multiple columns. We denote aggregating the column indices $i_1, \dots, i_m$ into the same group as $\tilde{i} = \{i_1, i_2, \dots i_m\}$, letting $s_{\tilde{i}} = \bigcup_{i \in \tilde{i}} s_i$ be the aggregated sparsity pattern, and letting $\tilde{s} = s_{\tilde{i}} - \tilde{i}$ be the set of selected entries excluding the diagonal entries. Each $s_i = \tilde{s} \cup \{j \in \tilde{i} \mid j \geq i\}$, that is, the sparsity pattern of the $i$ column is the selected entries plus all the diagonal entries lower than it. We will enforce that all the selected entries, excluding the indices of the diagonals of the columns themselves, are below the lowest index so that indices are not selected "partially" — that is, an index could be above some indices in the aggregated columns, and therefore invalid to add to their column, but below others. That is, we restrict the candidate indices $k > \max \tilde{i}$ so that the selected index can be added to each column in $\tilde{i}$ without violating the lower triangularity of $L$. We now show that the KL-minimization objective on the aggregated indices corresponds precisely to (2.6), the objective multiple point Gaussian regression with the chain rule of log determinant through Schur complementation.

$$\operatorname{logdet}(\Theta) = \operatorname{logdet}(\Theta_{\mathrm{Pr},\mathrm{Pr}\mid\mathrm{Tr}}) + \operatorname{logdet}(\Theta_{\mathrm{Tr},\mathrm{Tr}}) \tag{3.8}$$

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

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

We see that the objective (3.9) is equivalent to the objective (2.6), that is, to minimize the log determinant of the conditional covariance matrix corresponding to a set of prediction points, conditional on the selected entries. We can therefore directly use Algorithm 2.6 on the aggregated columns, where the prediction points correspond to indices in the aggregation and where we restrict the candidates $k$ to those below each column in the aggregation, $k > \max \tilde{i}$.

pseudocode, $U$ and computing submatrices

Hence the sparse Cholesky factorization motivated by KL divergence can be viewed as the sparse Gaussian process regression over each column, where entries are selected to maximize mutual information with the entry on the diagonal of the current column.

The explicit inverse algorithm computes $\Theta_{11}^{-1}$, which can be directly used to generate the columns of $L$ according to (3.3) without extra computational cost.

### 3.2. Review of KL approximation.

## 4. Numerical experiments.

**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] A. Krause, A. Singh, and C. Guestrin, *Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies*, J. Mach. Learn. Res., 9 (2008), p. 235–284.

[2] F. Schäfer, M. Katzfuss, and H. Owhadi, *Sparse Cholesky factorization by Kullback-Leibler minimization*, arXiv preprint arXiv:2004.14455, (2020).

[3] F. Schäfer, T. J. Sullivan, and H. Owhadi, *Compression, inversion, and approximate PCA of dense kernel matrices at near-linear computational complexity*, arXiv preprint arXiv:1706.02205, (2017).

add proofs, if any, in appendix