THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Optimizing performance in Gaussian Processes

Michael Ciccotosto-Camp

Supervisor: Fred (Farbod) Roosta
Co-Supervisors: Andries Potgieter
Yan Zhao

Bachelor of Mathematics (Honours)
June 2022

The University of Queensland
School of Mathematics and Physics

Contents

Acknowledgements

I would like to deeply thank my supervisor Dr. Masoud Kamgarpour for his advice and all of his time spent with me. I consider myself lucky and am glad to have been his student for my honours year. I would also like to thank my co-supervisor Dr. Anna Puskás for the same reasons. A special thanks to Dr. Valentin Buciumas for his time spent teaching me while he was at The University of Queensland.

Symbols and Notation

Matrices are capitalized bold face letters while vectors are lowercase bold face letters.

Syntax	*Meaning*
$\triangleq$	An equality which acts as a statement
$\lvert\boldsymbol{A}\rvert$	The determinate of a matrix.
$\langle\cdot,\cdot\rangle_{\mathcal{H}}$	The inner product with respect to the Hilbert space $\mathcal{H}$, sometimes abbreviated as $\langle\cdot,\cdot\rangle$ if the Hilbert space is clear from context.
$\lVert\cdot\rVert_{\mathcal{V}}$	The norm of a vector with respect to the vector space $\mathcal{V}$, sometimes abbreviated as $\lVert\cdot\rVert$ if the vector space is clear from context.
$\boldsymbol{x}^\intercal, \boldsymbol{X}^\intercal$	The transpose operator.
$\boldsymbol{x}^*, \boldsymbol{X}^*$	The hermitian operator.
$\boldsymbol{a}.*\boldsymbol{b}$ or $\boldsymbol{A}.*\boldsymbol{B}$	Element-wise vector (matrix) multiplication, similar to Matlab.
$\propto$	Proportional to.
∇ or $\nabla_{\boldsymbol{f}}$	The partial derivative (with respect to $\boldsymbol{f}$).
∇	The Hessian.
$\sim$	Distributed according to, example $x \sim \mathcal{N}(0,1)$
$\mathbf{0}$ or $\mathbf{0}_n$ or $\mathbf{0}_{n\times m}$	The zero vector (matrix) of appropriate length (size) or the zero vector of length n or the zero matrix with dimensions $n \times m$.
$\mathbf{1}$ or $\mathbf{1}_n$ or $\mathbf{1}_{n\times m}$	The one vector (matrix) of appropriate length (size) or the one vector of length n or the one matrix with dimensions $n \times m$.
$\mathbb{1}_{n\times m}$	The matrix with ones along the diagonal and zeros on off diagonal elements.

$\boldsymbol{A}_{(\cdot,\cdot)}$	Index slicing to extract a submatrix from the elements of $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, similar to indexing slicing from the python and Matlab programming languages. Each parameter can receive a single value or a 'slice' consisting of a start and an end value separated by a semicolon. The first and second parameter describe what row and columns should be selected, respectively. A single value means that only values from the single specified row/column should be selected. A slice tells us that all rows/columns between the provided range should be selected. Additionally if now start and end values are specified in the slice then all rows/columns should be selected. For example, the slice $\boldsymbol{A}_{(1:3,j:j')}$ is the submatrix $\mathbb{R}^{3 \times (j'-j+1)}$ matrix containing the first three rows of $\boldsymbol{A}$ and columns j to j'. As another example, $\boldsymbol{A}_{(:,j)}$ is the j^{th} column of $\boldsymbol{A}$.
$\boldsymbol{A}^{\dagger}$	Denotes the unique psuedo inverse or Moore-Penore inverse of $\boldsymbol{A}$.
$\mathbb{C}$	The complex numbers.
C	The classes in a classification problem.
$\text{cholesky}\left(\boldsymbol{A}\right)$	A function to compute the Cholesky decomposition of the matrix $\boldsymbol{A}$, where $\boldsymbol{L}\boldsymbol{L}^{\intercal} = \boldsymbol{A}$.
$\text{cov}\left(\boldsymbol{f}\right)$	Gaussian process posterior covariance.
d	The number of features in the data set.
D	The dimension of the feature space of the feature mapping constructed in the Random Fourier Feature method.
$\mathcal{D}$	The dataset, $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$.
$\text{diag}\left(\boldsymbol{w}\right)$	Vector argument, a diagonal matrix containing the elements of vector $\boldsymbol{w}$.
$\text{diag}\left(\boldsymbol{W}\right)$	Matrix argument, a vector containing the diagonal elements of the matrix $\boldsymbol{W}$.
$\mathbb{E}$ or $\mathbb{E}_{q(x)}\left[z(x)\right]$	Expectation, or expectation of $z(x)$ where $x \sim q(x)$.
$\mathcal{GP}$	Gaussian process $f \sim \mathcal{GP}\left(m(\boldsymbol{x}), k\left(\boldsymbol{x}, \boldsymbol{x}'\right)\right)$, the function f is distributed as a Guassian process with mean function $m\left(\boldsymbol{x}\right)$ and covariance function $k\left(\boldsymbol{x}, \boldsymbol{x}'\right)$.

$k(\cdot,\cdot)$	A covariance or kernel matrix.
$\boldsymbol{K}_{\boldsymbol{W}\boldsymbol{W}'}$	For two data sets $\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_n]^\intercal \in \mathbb{R}^{n \times d}$ and $\boldsymbol{W}' = [\boldsymbol{w}'_1, \boldsymbol{w}'_2, \ldots, \boldsymbol{w}'_m]^\intercal \in \mathbb{R}^{n' \times d}$ the matrix $\boldsymbol{K}_{\boldsymbol{W}\boldsymbol{W}'} \in \mathbb{R}^{n \times n'}$ has elements $(\boldsymbol{K}_{\boldsymbol{W}\boldsymbol{W}'})_{i,j} = k\left(\boldsymbol{w}_i, \boldsymbol{w}'_j\right)$.
$\text{lin-solve}\,(\boldsymbol{A}, \boldsymbol{B})$	A function used to solve $\boldsymbol{X} = \boldsymbol{A}^{-1}\boldsymbol{B}$ in the linear system $\boldsymbol{A}\boldsymbol{X} = \boldsymbol{B}$.
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ or $\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	(the variable $\boldsymbol{x}$ has a) Multivariate Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$.
n and n_*	The number of training (and tests) cases.
N	The dimenion of the feature space.
$\mathbb{N}$	The natural numbers, $\mathbb{N} = \{1, 2, 3, \ldots\}$.
$\mathcal{O}(\cdot)$	Big-O notation. If a function $f \in \mathcal{O}(g)$ then the absolute value of $f(x)$ is at most a positive multiple of $g(x)$ for all sufficiently large values of x.
$y \mid x$ and $p(x \mid y)$	A conditonal random variable y given x and its probability density.
$\boldsymbol{Q}, \boldsymbol{V}$	Typically used to denote a matrix with orthonormal structure.
$\mathbb{R}$	The real numbers.
$\text{tr}\,(\boldsymbol{A})$	The trace of a matrix.
$\mathbb{V}$ or $\mathbb{V}_{q(x)}[z(x)]$	Variance, the variance of $z(x)$ when $x \sim q(x)$.
$\mathcal{X}$	Input space.
$\boldsymbol{X}$	The $n \times d$ matrix of training inputs.
$\boldsymbol{X}_*$	The $n_* \times d$ matrix of test inputs.
$\boldsymbol{x}_i$	The i^{th} training input.
$\mathbb{Z}$	The integers, $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

1. The Nystrom Method

In chapter **??** we saw that GP regression and classification relied on a Gram matrix (see definition **??**) to produce predictions. Unfortunately, from a computational perspective, constructing the Gram matrix for a data set $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ brings about a nasty bottle neck owed by the $\mathcal{O}\left(n^2\right)$ kernel evaluations. Even before the rise of ML, there has been a lot of research devoted to creating numerical methods that quickly construct a low rank approximation of large matrices, $\boldsymbol{A}$, which ordinarily are a computational burdened to build exactly. These methods are centered around the idea of capturing the columns space of the matrix that best describes the the action of $\boldsymbol{A}$ as an operator. For lack of a better explanation, Mahoney gives a fantastic summary as to why the column space is of paramount importance in these approximation techniques

> *"To understand why sampling columns (or rows) from a matrix is of interest,recall that matrices are "about" their columns and rows that is, linear combinations are taken with respect to them; one all but understands a given matrix if one understands its column space, row space, and null spaces; and understanding the subspace structure of a matrix sheds a great deal of light on the linear transformation that the matrix represents."*
> [MWM11, page 13]

Moreover, this class of algorithms lend very nice forms when $\boldsymbol{A}$ possess positive definite structure, which is exactly the case for our Gram matrix.

1.1. The Nystrom Method.

Attempting to compute an entire kernel matrix can prove to be quite a computational headache, prompting us to seek estimative alternatives. The approximation techniques studied in this chapter have been spurred on by the John-Lindenstrauss lemma stated in lemma 1.

Lemma 1 (John-Lindenstrauss)**.** *Given $0 < \varepsilon < 0$, any set of n points, X, in a high dimensional Euclidean space can be embedded into a $\ell-$dimensional Euclidean space where $\ell = \mathcal{O}(\ln(n))$ via some linear map $\boldsymbol{\Omega} \in \mathbb{R}^{n\times\ell}$ which satisfies*

$$(1-\varepsilon)\left\|\boldsymbol{u}-\boldsymbol{v}\right\|^2 \leq \left\|\boldsymbol{\Omega u}-\boldsymbol{\Omega v}\right\|^2 \leq \varepsilon\left\|\boldsymbol{u}-\boldsymbol{v}\right\|^2$$

for any $\boldsymbol{u}, \boldsymbol{v} \in X$ [MWM11, page 15].

The John-Lindenstrauss lemma tells us that $\boldsymbol{QQ}^*\boldsymbol{A}$ will serve as a good approximation to some matrix $\boldsymbol{A}$ where $\boldsymbol{QQ}^*$, in some sense, projects onto some rank-k subspace of $\boldsymbol{A}$'s column space. This is because if $\boldsymbol{QQ}^*$ closesly matches the behavior of $\boldsymbol{\Omega}$ from the lemma then the pair-wise distances between points before and after applying $\boldsymbol{QQ}^*$ should remain fairly similar. To state this a little more explicitly, for a matrix $\boldsymbol{A}$ and a positive error tolerance ε we seek a matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times k_\varepsilon}$ with orthonormal columns such that

$$\left\|\boldsymbol{A}-\boldsymbol{QQ}^*\boldsymbol{A}\right\|_F \leq \varepsilon$$

which can be expressed a more short hand notation as

$$\boldsymbol{A} \simeq \boldsymbol{QQ}^*\boldsymbol{A}. \tag{1}$$

This is commonly called the *fixed precision approximation problem*. Although, to simplify algorithmic development, a value of k is specified in advanced (instead of ε, thus removing k's dependence on ε) which

is instead given the name *fixed rank problem*. Within the fixed rank problem framework, when $\boldsymbol{A}$ is hermitian, the matrix $\boldsymbol{QQ}^*$ acts as a good projection for both the columns and row space of $\boldsymbol{A}$ so that we have both $\boldsymbol{A} \simeq \boldsymbol{QQ}^*\boldsymbol{A}$ and $\boldsymbol{A} \simeq \boldsymbol{AQQ}^*$ so that

$$\boldsymbol{A} \simeq \boldsymbol{QQ}^* (\boldsymbol{A}) \simeq \boldsymbol{QQ}^*\boldsymbol{AQQ}^*. \tag{2}$$

Furthermore, if $\boldsymbol{A}$ is positive semi-definite we can improve the quality of our approximation of our approximation at almost no additional cost [Hal11, page 32]. Using the approximation from 2

$$\begin{aligned} \boldsymbol{A} &\simeq \boldsymbol{Q} (\boldsymbol{Q}^*\boldsymbol{AQ}) \boldsymbol{Q}^* \\ &= \boldsymbol{Q} (\boldsymbol{Q}^*\boldsymbol{AQ}) (\boldsymbol{Q}^*\boldsymbol{AQ})^\dagger (\boldsymbol{Q}^*\boldsymbol{AQ}) \boldsymbol{Q}^* \\ &\simeq (\boldsymbol{AQ}) (\boldsymbol{Q}^*\boldsymbol{AQ})^\dagger (\boldsymbol{Q}^*\boldsymbol{A}) . \end{aligned} \tag{3}$$

This is known as the Nystrom method. Since any Gram matrix is positive semi-definite, we can always applied the Nystrom method to find an approximation to it. A general Nystrom framework is presented in Algorithm 1.

Algorithm 1: General Nystrom Framework

input : A positive semi-definite matrix $\boldsymbol{A}$, a matrix $\boldsymbol{Q}$ that satisfies 1.
output: A rank k approximation $\overline{\boldsymbol{A}} \simeq \boldsymbol{A}$.

$\boldsymbol{C} = \boldsymbol{AQ}$
$\boldsymbol{W} = \boldsymbol{Q}^*\boldsymbol{C}$
return $\boldsymbol{CW}^\dagger\boldsymbol{C}^*$

However, Algorithm 1 assumes that $\boldsymbol{Q}$ has already been computed. Naturally, the next question is then how do we do about efficiently constructing a suitable matrix $\boldsymbol{Q}$ that satisfies equation 1? We can do this through a very popular column sampling technique ubiquitous in numerical linear algebra literature. This technique has been driven by Theorem

Theorem 2. *Every $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ matrix contains a $k-$column submatrix $\boldsymbol{C}$ for which*

$$\left\| \boldsymbol{A} - \boldsymbol{CC}^\dagger\boldsymbol{A} \right\|_F \leq \sqrt{1 + k(n-k)} \cdot \|\boldsymbol{A} - \boldsymbol{A}_k\|$$

where $\boldsymbol{A}_k$ is the best rank$-k$ approximation of $\boldsymbol{A}$ [Hal11, page 11].

Before we delve anymore into this column sampling Nystrom technique, we will first need to cover the random matrix multiplication algorithm which serves as a backbone for this technique. Let $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ be a target matrix we would like to approximate and suppose that $\boldsymbol{A}$ can be represented as the sum of 'simpler' (for example, sparse or low-rank) matrices, $\boldsymbol{A}_i$, so that

$$\boldsymbol{A} = \sum_{i=1}^{I} \boldsymbol{A}_i. \tag{4}$$

The basic idea is to consider a Monte-Carlo approximation of equation 4 that randomly selects $\boldsymbol{A}_i$ according to the distribution $\{p_i\}_{i=1}^I$ to give an estimate

$$\boldsymbol{A} \simeq \frac{1}{c} \sum_{t=1}^{c} p_{t_i}^{-1} \boldsymbol{A}_{t_i} \tag{5}$$

where c is the number of samples and each summand is rescaled by a factor of $p_{t_i}^{-1}$ to ensure our estimate is unbiased [PGMaJT21, pages 24-27]. The random matrix multiplication algorithm works by attempting to find a Monte-Carlo estimate for $\boldsymbol{AB}$, where $\boldsymbol{A} \in \mathbb{R}^{n\times I}$ and $\boldsymbol{A} \in \mathbb{R}^{I\times m}$. Recall that any matrix multiplication can be written in its outter product form

$$\boldsymbol{AB} = \sum_{i=1}^{I} \boldsymbol{A}_{(:,i)}\boldsymbol{B}_{(i,:)}$$

[FR20, Dri06]. A straight forward way to approximate this using the Monte-Carlo estimate is to simply set each $\boldsymbol{A}_i$ in 4 to the corresponding rank-1 outter-product summand $\boldsymbol{A}_{(:,i)}\boldsymbol{B}_{(i,:)}$. This justifies the random matrix multiplication algorithm seen in Algorithm 3 [PDaMWM17, page 16].

Algorithm 2: Random Matrix Multiplication

input : $\boldsymbol{A} \in \mathbb{R}^{n\times I}$ and $\boldsymbol{A} \in \mathbb{R}^{I\times m}$, the number of samples $1 \leq c \leq n$ and a probability distribution over I, $\{p_i\}_{i=1}^{I}$.

output: Matricies $\boldsymbol{C} \in \mathbb{R}^{n\times c}$ and $\boldsymbol{R} \in \mathbb{R}^{c\times m}$.

for $t = 1, \ldots, c$ **do**

  Pick $i_t \in \{1, \ldots, n\}$ with $\mathbb{P}[i_t = k] = p_k$, independently and with replacement.

  $\boldsymbol{C}_{(:,t)} = \frac{1}{\sqrt{cp_{i_t}}}\boldsymbol{A}_{(:,i_t)}$

  $\boldsymbol{R}_{(:,t)} = \frac{1}{\sqrt{cp_{i_t}}}\boldsymbol{B}_{(i_t,:)}$

end

return $\boldsymbol{CR} = \sum_{t=1}^{c} \frac{1}{cp_{i_t}}\boldsymbol{A}_{(:,i_t)}\boldsymbol{B}_{(i_t,:)}$

This algorithm makes this idea a little more precise, taking in the two matrices to multiply together as well as a probability distribution over I to provide an estimate for $\boldsymbol{AB}$ of the form

$$\boldsymbol{AB} \simeq \sum_{t=1}^{c} \frac{1}{cp_{i_t}}\boldsymbol{A}_{(:,i_t)}\boldsymbol{B}_{(i_t,:)}.$$

Equivalently, the above can be restated as the product of two matrices $\boldsymbol{CR}$ formed by Algorithm 3, where $\boldsymbol{C}$ consists of c randomly selected rescaled columns of $\boldsymbol{A}$ and $\boldsymbol{R}$ is c randomly selected rescaled rows of $\boldsymbol{B}$. Notice that

$$\boldsymbol{CR} = \sum_{t=1}^{c} \boldsymbol{C}_{(:,i_t)}\boldsymbol{R}_{(i_t,:)} = \sum_{t=1}^{c} \left(\frac{1}{\sqrt{cp_{i_t}}}\boldsymbol{A}_{(:,i_t)}\right)\left(\frac{1}{\sqrt{cp_{i_t}}}\boldsymbol{B}_{(i_t,:)}\right) = \frac{1}{c}\sum_{t=1}^{c} \frac{1}{p_{i_t}}\boldsymbol{A}_{(:,i_t)}\boldsymbol{B}_{(i_t,:)}.$$

To make development easier, let us define a sampling and rescaling matrix, usually referred to as a sketching matrix, $\boldsymbol{S} \in \mathbb{R}^{n\times c}$ to be the the matrix with elements $\boldsymbol{S}_{i_t,t} = 1\sqrt{cp_{i_t}}$ if the i_t column of $\boldsymbol{A}$ is chosen during the t^{th} trial and all other entries of $\boldsymbol{S}$ are set to 0. Then we have

$$\boldsymbol{C} = \boldsymbol{AS} \quad \text{and} \quad \boldsymbol{R} = \boldsymbol{S}^\intercal\boldsymbol{B}$$

so that

$$\boldsymbol{CR} = \boldsymbol{ASS}^\intercal\boldsymbol{B} \simeq \boldsymbol{AB}. \tag{6}$$

Notice that $\boldsymbol{S}$ is generally a very sparse matrix and therefore is generally to constructed explicitly where the matrix products $\boldsymbol{AS}$ and $\boldsymbol{S}^\intercal\boldsymbol{B}$ are done through row and column rescaling of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ respectively [PDaMWM17, page 17]. Lemma 3 provides some bounds on $\boldsymbol{CR}$ as an estimate for $\boldsymbol{AB}$.

Lemma 3. *Let $\boldsymbol{C}$ and $\boldsymbol{R}$ be constructed as described in Algorithm 3, then*

$$\mathbb{E}\left[(\boldsymbol{CR})_{ij}\right] = (\boldsymbol{AB})_{ij}.$$

That is, $\boldsymbol{CR}$ is an unbiased estimate of $\boldsymbol{AB}$. Furthermore

$$\mathbb{V}\left[(\boldsymbol{CR})_{ij}\right] \leq \frac{1}{c}\sum_{k=1}^{n}\frac{\boldsymbol{A}_{ik}^2\boldsymbol{B}_{kj}^2}{p_k}.$$

Proof. For some fixed pair i, j for each $t = 1, \ldots, c$ define $\boldsymbol{X}_t = \left(\frac{\boldsymbol{A}_{(:,i_t)}\boldsymbol{B}_{(i_t,:)}}{cp_{i_t}}\right)_{ij} = \frac{\boldsymbol{A}_{(i,i_t)}\boldsymbol{B}_{(i_t,j)}}{cp_{i_t}}$. Thus, for any t,

$$\mathbb{E}\left[\boldsymbol{X}_t\right] = \sum_{k=1}^{n} p_k\frac{\boldsymbol{A}_{ik}\boldsymbol{B}_{kj}}{cp_k} = \frac{1}{c}\sum_{k=1}^{n}\boldsymbol{A}_{ik}\boldsymbol{B}_{kj} = \frac{1}{c}(\boldsymbol{AB})_{ij}.$$

Since we have $(\boldsymbol{CR})_{ij} = \sum_{t=1}^{c}\boldsymbol{X}_t$, it follows that

$$\mathbb{E}\left[(\boldsymbol{CR})_{ij}\right] = \mathbb{E}\left[\sum_{t=1}^{c}\boldsymbol{X}_t\right] = \sum_{t=1}^{c}\left[\mathbb{E}\boldsymbol{X}_t\right] = (\boldsymbol{AB})_{ij}.$$

Hence, $\boldsymbol{CR}$ is an unbiased estimator of $\boldsymbol{AB}$, regardless of the choice of the sampling probabilities. Using the fact that $(\boldsymbol{CR})_{ij}$ is the sum of c independent random variables, we get

$$\mathbb{V}\left[(\boldsymbol{CR})_{ij}\right] = \mathbb{V}\left[\sum_{t=1}^{c}\boldsymbol{X}_t\right] = \sum_{t=1}^{c}\mathbb{V}\left[\boldsymbol{X}_t\right].$$

Using the fact $\mathbb{V}\left[\boldsymbol{X}_t\right] \leq \mathbb{E}\left[\boldsymbol{X}_t^2\right] = \sum_{k=1}^{n}\frac{\boldsymbol{A}_{ik}^2\boldsymbol{B}_{kj}^2}{c^2p_k}$, we get

$$\mathbb{V}\left[(\boldsymbol{CR})_{ij}\right] = \sum_{t=1}^{c}\mathbb{V}\left[\boldsymbol{X}_t\right] \leq c\sum_{k=1}^{n}\frac{\boldsymbol{A}_{ik}^2\boldsymbol{B}_{kj}^2}{c^2p_k} = \frac{1}{c}\frac{\boldsymbol{A}_{ik}^2\boldsymbol{B}_{kj}^2}{p_k}.$$

□

So how does this help us with the Nystrom method? Consider using the random matrix multiplication algorithm to approximate the matrix multiplication of a Gram matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ and $\mathbb{1}^{n\times n}$. Equation 6 gives

$$\boldsymbol{KSS}^\intercal\mathbb{1}^{n\times n} = \boldsymbol{KSS}^\intercal \simeq \boldsymbol{K}.$$

We see now that the sketching matrix produced by Algorithm 3 provides a sketching matrix $\boldsymbol{S}$ that satisfies the properties of $\boldsymbol{Q}$ from equation 1 meaning that $\boldsymbol{S}$ can be used in place of $\boldsymbol{Q}$ within the Nystrom estimate from equation 3. These ideas are used together in Algorithm TODO that uses the column sampling technique from Algorithm 3 together with the general Nystrom framework

(Algorithm 1) to provide a new column sampling Nystrom method [PDaMWM05, AGaMWM13].

Algorithm 3: Nystrom Method via Column Sampling

input : Data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n]^\intercal \in \mathbb{R}^{n \times d}$, the number of samples $1 \leq c \leq n$ and a probability distribution over n, $\{p_i\}_{i=1}^n$.

output: An approximation of the Gram matrix corresponding to $\boldsymbol{X}$, that is $\overline{\boldsymbol{K}} \simeq \boldsymbol{K}$ where $\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Initialize $\boldsymbol{C}$ as an empty $n \times c$ matrix.

Pick c columns with the probability of choosing the k^{th} column $(1 \leq k \leq n)$ as $\mathbb{P}[k = i] = p_i$, independently and with replacement and let I a list of indices of the sampled columns.

for $i \in I$ **do**

  Pick $i \in \{1, \ldots, n\}$ with $\mathbb{P}[i = k] = p_k$, independently and with replacement.

  $\boldsymbol{K}_{(:,i)} = [k(\boldsymbol{x}_1, \boldsymbol{x}_i), \ldots, k(\boldsymbol{x}_n, \boldsymbol{x}_i)]^\intercal$

  $\boldsymbol{C}_{(:,i)} = \boldsymbol{K}_{(:,i)} / \sqrt{cp_i}$

end

$\boldsymbol{W} = \boldsymbol{K}_{(I,I)} \in \mathbb{R}^{c \times c}$

Rescale each entry of $\boldsymbol{W}$, $\boldsymbol{W}_{ij}$, by $1/c\sqrt{p_i p_j}$.

Compute $\boldsymbol{W}^\dagger$

return $\boldsymbol{C}\boldsymbol{W}^\dagger\boldsymbol{C}^*$

As we can tell from the algorithms inputs, this requires some sort of probability to select the columns. As seen in lemma 3 any probability distribution we use will provide an unbiased estimate, although some probability distributions can be used to lower the variance faster than others. Naively, we could just employ uniform sampling where each column in selected with equal probability. However, this is seldom a good idea since uniform sampling tend to over sample landmarks from one large cluster while under sampling or possibly entirely missing small but important clusters. As a result, the approximation for $\boldsymbol{K}$ will decline [CMaCM17, page 3]. This is demonstarted in graphic form in Figure TODO. To combat this issue, alternative probabilites density can be constructed to take into account a measure of importance in landmark selection. Indeed there has been a plethora of research that has shown the importance of using data-dependent non-uniform probability distributions to obtain proveably good error bounds on Nystrom approximations [PDaMWM05, AGaMWM13, CMaCM17, PDaMMaMWMaDPW11, MBCaCMaCM15, Kum09]. A few of the more common distributions will be discussed in the coming sections.

References

[Ras06] Carl Edward and Williams Rasmussen Christopher K. I, *Gaussian processes for machine learning / Carl Edward Rasmussen, Christopher K.I. Williams.*, Adaptive computation and machine learning, MIT Press, Cambridge, Mass., 2006 (eng).

[HHF73] H. Howard Frisinger, *Aristotle's legacy in meteorology*, Bulletin of the American Meteorological Society **54** (1973), no. 3, 198–204.

[Yul27] G. Udny Yule, *On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers*, Philosophical transactions of the Royal Society of London. Series A, Containing papers of a mathematical or physical character **226** (1927), no. 636-646, 267–298 (eng).

[Box08] George E. P. and Jenkins Box Gwilym M and Reinsel, *Time series analysis : forecasting and control / George E.P. Box, Gwilym M. Jenkins, Gregory C. Reinsel.*, 4th ed., Wiley series in probability and statistics, John Wiley, Hoboken, N.J., 2008 (eng).

[VdW19] Mark Van der Wilk, *Sparse Gaussian process approximations and applications*, University of Cambridge, 2019.

[Cao18] Yanshuai Cao, *Scaling Gaussian Processes*, University of Toronto (Canada), 2018.

[SD22] Matías and Estévez Salinero-Delgado José and Pipia, *Monitoring Cropland Phenology on Google Earth Engine Using Gaussian Process Regression*, Remote Sensing **14** (2022), no. 1, DOI 10.3390/rs14010146.

[Pot13] Andries and Lawson Potgieter Kenton and Huete, *Determining crop acreage estimates for specific winter crops using shape attributes from sequential MODIS imagery*, International Journal of Applied Earth Observation and Geoinformation **23** (2013), DOI 10.1016/j.jag.2012.09.009.

[Mur12] Kevin P. Murphy, *Machine learning : a probabilistic perspective / Kevin P. Murphy.*, Adaptive computation and machine learning, MIT Press, Cambridge, MA, 2012 (eng).

[Ber96] Z.G. Sheftel Berezansky G.F, *Functional analysis. Volume 1 / Y.M. Berezansky, Z.G. Sheftel, G.F. Us ; translated from the Russian by Peter V. Malyshev.*, 1st

ed. 1996., Operator Theory: Advances and Applications, 85, Basel ; Boston ; Berlin : BirkhaIuser Verlag, Basel ; Boston ; Berlin, 1996 (eng).

[Tre97] Lloyd N. (Lloyd Nicholas) and Bau Trefethen David, *Numerical linear algebra / Lloyd N. Trefethen, David Bau.*, SIAM Society for Industrial and Applied Mathematics, Philadelphia, 1997 (eng).

[Dem97] James W Demmel, *Applied numerical linear algebra / James W. Demmel.*, Society for Industrial and Applied Mathematics, Philadelphia, Pa., 1997 (eng).

[Ste08] Ingo and Christmann Steinwart Andreas, *Support Vector Machines*, 1st ed. 2008., Information Science and Statistics, Springer New York, New York, NY, 2008 (eng).

[Ber03] Alain and Thomas-Agnan Berlinet Christine, *Reproducing Kernel Hilbert Spaces in Probability and Statistics*, Springer, SpringerLink (Online service), Boston, MA, 2003 (eng).

[Ste99] Michael L Stein, *Interpolation of Spatial Data Some Theory for Kriging / by Michael L. Stein.*, 1st ed. 1999., Springer Series in Statistics, Springer New York : Imprint: Springer, New York, NY, 1999 (eng).

[Bos92] Bernhard and Guyon Boser Isabelle and Vapnik, *A training algorithm for optimal margin classifiers*, Proceedings of the fifth annual workshop on computational learning theory, 1992, pp. 144–152 (eng).

[Cor95] Corinna Cortes, *Support-Vector Networks*, Machine learning **20** (1995), no. 3, 273 (eng).

[Kro14] Dirk P and C.C. Chan Kroese Joshua, *Statistical Modeling and Computation by Dirk P. Kroese, Joshua C.C. Chan.*, 1st ed. 2014., Springer New York : Imprint: Springer, New York, NY, 2014 (eng).

[Fle00] R Fletcher, *Practical Methods of Optimization*, John Wiley and Sons, Incorporated, New York, 2000 (eng).

[Bis06] Christopher M Bishop, *Pattern recognition and machine learning / Christopher M. Bishop.*, Information science and statistics, Springer, New York, 2006 (eng).

[Spi90] David J and Lauritzen Spiegelhalter Steffen L, *Sequential updating of conditional probabilities on directed graphical structures*, Networks **20** (1990), no. 5, 579–605.

[MWM11] Michael W. Mahoney, *Randomized algorithms for matrices and data*, CoRR **abs/1104.5557** (2011).

[Hal11] Nathan and Martinsson Halko Per-Gunnar and Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM review **53** (2011), no. 2, 217–288.

[PGMaJT21] Per-Gunnar Martinsson and Joel Tropp, *Randomized Numerical Linear Algebra: Foundations and Algorithms*, arXiv, 2021.

[FR20] Fred Roosta, *University of Queensland MATH3204, Lecture notes in Numerical Linear Algebra and Optimisation*, University of Queensland, 2020.

[Dri06] Petros and Kannan Drineas Ravi and Mahoney, *Fast Monte Carlo Algorithms for Matrices I: Approximating Matrix Multiplication*, SIAM Journal on Computing **36** (2006), no. 1, 132-157, DOI 10.1137/S0097539704442684, available at https://doi.org/10.1137/S0097539704442684.

[PDaMWM17] Petros Drineas and Michael W. Mahoney, *Lectures on Randomized Numerical Linear Algebra*, arXiv, 2017.

[PDaMWM05] Petros Drineas and Michael W. Mahoney, *On the Nystrom Method for Approximating a Gram Matrix for Improved Kernel-Based Learning*, Journal of Machine Learning Research **6** (2005), no. 72, 2153-2175.

[AGaMWM13] Alex Gittens and Michael W. Mahoney, *Revisiting the Nystrom Method for Improved Large-Scale Machine Learning*, CoRR **abs/1303.1849** (2013), available at 1303.1849.

[CMaCM17] Cameron Musco and Christopher Musco, *Recursive Sampling for the Nystrom Method*, arXiv, 2017.

[PDe11] Petros Drineas etal., *Fast approximation of matrix coherence and statistical leverage*, CoRR **abs/1109.3843** (2011).

[MBCaCMaCM15] Michael B. Cohen and Cameron Musco and Christopher Musco, *Ridge Leverage Scores for Low-Rank Approximation*, CoRR **abs/1511.07263** (2015).

[Kum09] Sanjiv and Mohri Kumar Mehryar and Talwalkar, *Sampling techniques for the nystrom method*, Artificial intelligence and statistics, 2009, pp. 304–311.

[Pre92] William H. (William Henry) Press, *Numerical recipes in C : the art of scientific computing / William H. Press ...* [*et al.*], 2nd ed., Cambridge University Press, Cambridge, 1992 (eng).

[Wan] Guorong and Wei Wang Yimin and Qiao, *Generalized Inverses: Theory and Computations*, Developments in Mathematics, vol. 53, Springer Singapore, Singapore (eng).

[Gre97] Anne Greenbaum, *Iterative methods for solving linear systems Anne Greenbaum.*, Frontiers in applied mathematics ; 17, Society for Industrial and Applied Mathematics SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104, Philadelphia, Pa., 1997 (eng).

[Cho07] Sou-Cheng (Terrya) Choi, *Iterative methods for singular linear equations and least -squares problems*, ProQuest Dissertations Publishing, 2007 (eng).

[CHO11] Sou-Cheng T and PAIGE CHOI Christopher C and SAUNDERS, *MINRES-QLP: A KRYLOV SUBSPACE METHOD FOR INDEFINITE OR SINGULAR SYMMETRIC SYSTEMS*, SIAM journal on scientific computing **33** (2011), no. 3-4, 1810–1836 (eng).

[Rah08] Ali and Recht Rahimi Benjamin, *Random Features for Large-Scale Kernel Machines*, Advances in Neural Information Processing Systems, 2008.

[Pot21] Andres and Wu Potapczynski Luhuan and Biderman, *Bias-Free Scalable Gaussian Processes via Randomized Truncations* (2021) (eng).

[Hah33] Hans Hahn, *S. Bochner, Vorlesungen über Fouriersche Integrale: Mathematik und ihre Anwendungen, Bd. 12.) Akad. Verlagsges., Leipzig 1932, VIII. u. 229S. Preis brosch. RM 14,40, geb. RM16*, Monatshefte für Mathematik **40** (1933), no. 1, A27–A27 (ger).

[Liu21] Fanghui and Huang Liu Xiaolin and Chen, *Random Features for Kernel Approximation: A Survey on Algorithms, Theory, and Beyond*, IEEE transactions on pattern analysis and machine intelligence **PP** (2021) (eng).

[HAe16] Haim Avron etal, *Quasi-Monte Carlo Feature Maps for Shift-Invariant Kernels*, Journal of Machine Learning Research **17** (2016), no. 120, 1-38.

[DJSaJS15] Danica J. Sutherland and Jeff Schneider, *On the Error of Random Fourier Features*, 2015.

[Yu16] Felix X and Suresh Yu Ananda Theertha and Choromanski, *Orthogonal Random Features* (2016) (eng).

[Bro91] Peter J and Davis Brockwell Richard A, *Time Series: Theory and Methods*, Second Edition., Springer Series in Statistics, Springer New York, SpringerLink (Online service), New York, NY, 1991 (eng).

[Cho17] Krzysztof and Rowland Choromanski Mark and Weller, *The Unreasonable Effectiveness of Structured Random Orthogonal Embeddings* (2017) (eng).

[FaA76] Fino and Algazi, *Unified Matrix Treatment of the Fast Walsh-Hadamard Transform*, IEEE transactions on computers **C-25** (1976), no. 11, 1142–1146 (eng).

[And15] Alexandr and Indyk Andoni Piotr and Laarhoven, *Practical and Optimal LSH for Angular Distance* (2015) (eng).

[Cho20] Krzysztof and Likhosherstov Choromanski Valerii and Dohan, *Rethinking Attention with Performers* (2020) (eng).

[Boj16] Mariusz and Choromanska Bojarski Anna and Choromanski, *Structured adaptive and random spinners for fast machine learning computations* (2016) (eng).