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.
$\boldsymbol{x}^{\mathsf{T}}, \boldsymbol{X}^{\mathsf{T}}$	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.
$\mathrm{cholesky}\left(\boldsymbol{A}\right)$	A function to compute the Cholesky decomposition of the matrix $\boldsymbol{A}$, where $\boldsymbol{L}\boldsymbol{L}^{\intercal} = \boldsymbol{A}$.
$\mathrm{cov}\left(\boldsymbol{f}\right)$	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, $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$.
$\mathrm{diag}\left(\boldsymbol{w}\right)$	Vector argument, a diagonal matrix containing the elements of vector $\boldsymbol{w}$.
$\mathrm{diag}\left(\boldsymbol{W}\right)$	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, $\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$.

Introduction

Time series prediction (and related regressional tasks) is a subject of high interest across many disciplines of science and mathematics. The history of time series can be traced back to the birth of science in ancient Greece where Aristotle devised a systematic approach to weather forecasting in 350 BC in his famous treatise *Meteorologica*. This method was later used to help predict when certain meteorological induced events, such as the flooding of the Nile river [HHF73]. Statistical modelling for time series prediction would not come until the 20th century where development of AutoRegressive Moving Average (ARMA) models which where first mentioned by Yule [Yul27] in 1927 and later popularized by Box and Jenkins in their book *Time Series Analysis* published in 1970 [Box08].

Given a data set of n observations $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$, where each input $x_i \in \mathbb{R}_{>0}$ is a time value and $y_i \in \mathbb{R}$ is a output or experimental observation that acts a function of time, the goal of time series prediction is to try and best predict a value $y_\star$ at time $x_\star$. With computing power becoming ever more advanced and affordable, many have taken to Machine Learning (ML) to develop sophisticated models to address the problem of creating accurate yet computationally inexpensive time series predictors. Broadly speaking, ML is any class of heuristic algorithm that attempts to refine and develope some model to perform a "simple" task by learning through user provided input. ML is founded on the idea that any form of task learning is done through sensory input taken from the surrounding environment. More formally speaking, ML attempts to generate a function $f : X \to Y$, for some input set X and observation or output set Y, were the outputs given by f closely aligns to actual observations. It is tacitly assumed that the phenomena we are studying follows laws which admit mathematical formulation and that experimental results can be reproduced to some degree of accuracy. Typically, experiments will never produce exact values of the underpinning law, g. Instead experimental observations, y_i, will include a small amount of random error so that $y_i = g(x_i) + \varepsilon_i$ where $\varepsilon_i \overset{\text{iid}}{\sim} \mathcal{N}\left(0, \sigma^2\right)$.

A ML model will attempt to make accurate predictions using some simplified formulation of the world. The distribution corresponding to the probability of a prediction within the context of the "state of the world" is referred to as the *likelihood*. The uncertainty within the likelihood stems from the predictive limits of the model. These limitation usually arise as a consequence of selecting a model which is either too simple or complex. The "state of the world" is sometimes internally captured by the model as a set of mutable parameters $\boldsymbol{\theta}$. The process of taking observations and using them to form predictions is called *inference* which, in some sense, is synonymous with learning [VdW19].

ML can be applied to time series prediction in a fairly straight forward manner by simply teaching a ML algorithm the time series data set, $\mathcal{D}$, to hopefully produce a function f that serves as a good approximant for event prediction.

In this thesis we shall focus on a particular class of ML algorithms called Baysian models which, unsurprisingly, makes use of Bayesian statistics to drive inference. In Baysian models a *prior* distribution is used to quantify the uncertainty of the current state of the model before any observations are made. The model can then be updated once data is observed by using the likelihood to give a *posterior* distribution which represents the reduced uncertainty after "teaching" the model new observations. Methods of teaching a model how to change its behavior using a new set of observations often involves the use of a *loss function* L. The loss function is used as an aid in deciding what action, a, should be taken in to best minimize uncertainty. The best action, roughly speaking, can be evaluated as

$$a_{\text{opt}} = \arg\min_{a} \int L\left(y_\star, a\right) p\left(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y}\right)\, dy_\star.$$

Interestingly, the best action does not rely so much on the model's internalized parameters but rather on the predictive distribution $p\left(y_\star \mid \boldsymbol{x}_\star, \boldsymbol{X}, \boldsymbol{y}\right)$ [VdW19]. This key insight has spawned a class of ML algorithms that focuses on infering the function f directly by computing $p\left(f \mid \mathcal{D}\right)$ instead of finding optimal internal parameters using $p\left(\boldsymbol{\theta} \mid \mathcal{D}\right)$ [Mur12]. Models that perform inference in this manner are called *non-parameteric* models. Within the *non-parameteric* model paradigm, the predictive distribution can be represented as

$$p\left(y_\star \mid x_\star, \boldsymbol{X}, \boldsymbol{y}\right) = \int p\left(y_\star \mid f, x_\star\right) p\left(f \mid \boldsymbol{X}, \boldsymbol{y}\right)\, df$$

and once new data is observed the posterior can be updated using Baye's rule

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \qquad p\left(\boldsymbol{f}, f_\star \mid \boldsymbol{y}\right) = \frac{p\left(\boldsymbol{y} \mid \boldsymbol{f}\right) p\left(\boldsymbol{f}, f_\star\right)}{p\left(\boldsymbol{y}\right)}$$

[Ras06]. This thesis will focus on a particular non-parameteric Bayesian ML model called Gaussian processes (GPs). The over arching idea of GPs is to assign a prior probability to every possible function mapping from X to Y. While this does not appear to be computationally tractable as this would due to the seemingly uncountable infinite number of mappings that would require checking, it turns out, these computations can infact be carried out given we are only seeking predictions at a finite number of points using a finite number of observations. GPs occupy a special place within the realm of ML since they account for uncertainty in a principled way, are relatively simple to implement and are highly modular allowing them to easily be incorporated into a larger systems. It is no surprise then that while other kernel methods (such as kernelized k^{th} nearest neighbors and ridge regression) are still overshadowed by their neural network cousins, GPs have made a quiet comeback in the ML community [Cao18].

The following example highlights a particular GP success story: a team of researchers led by Andries Potgieter at QAAFI (UQ) are currently investigating new digital approaches to accurately derive crop phenology stages (i.e. mid green, peak, flowering, grain filling and harvest) measured at field scale across large regions. Such methods can be used to better inform farmers and industry on the optimised time to plant various crops to minimize crop loss from environmental stresses such as frost and fungal disease. This involves analysing crop growth from previous seasons (i.e. 2018-2021) to forecast when certain phenological stages will take place in the current harvest. Outputs form this tool will allow producers to accurately map the temporal and spatial extend of phenology at a field and farm levels across different regions and seasons. This problem is readily converted into a time series problem. Originally, Potgieter's team surveyed a number of different parameteric models to carry out forecasting. However, the parameteric models we serverely limited in their ability to inform when key phenological stages would take place. After seeing the success of applying GPs to other remote senesing tasks [SD22] investigated the use of GPs in their own research to find that they could produce much higher resolution predictions from which they could infer a far richer phenological timeline [Pot13]. A comparision of using GPs over other parameteric models is shown in Figure 1. Potgieter's team found that the only draw back to using GPs was the lengthy run time required to create predictions and fears that collecting new

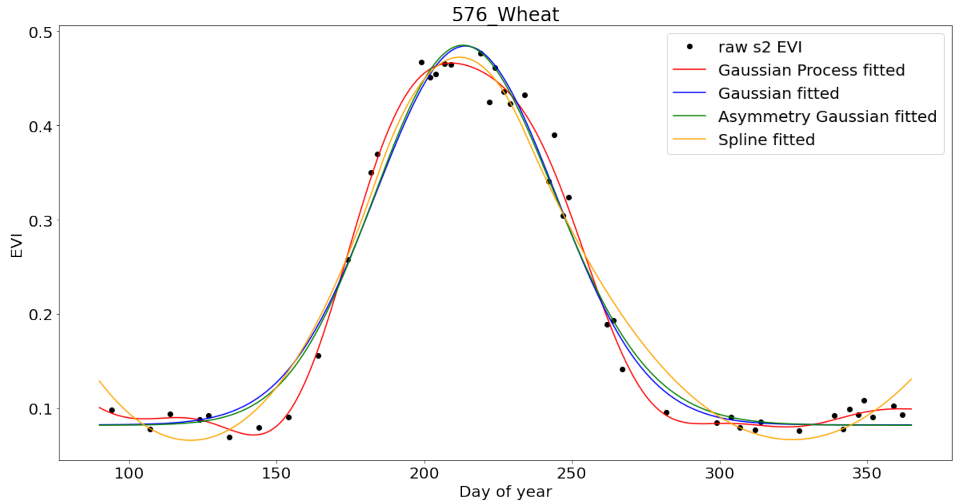


FIGURE 1. Potgieter's team found that GPs where superior in terms of predicting a phenological timeline for a number of common seasonal crops over other parametric models.

data each season will only exacerbate the issue. This is a common problem shared by anyone wanting to use GPs. Due to their unwieldy $\mathcal{O}\left(n^3\right)$ runtime, where n is the number of observations, GPs become impractical to apply on datasets with $n > 10^5$ samples. As such, the goal of this thesis is to explore various avenues one can take to replace some of the more intense calculations of GPs with computationally more efficient approximations without overly sacrificing accuracy.

Chapter **??** will give a more mathematical treatment of GPs starting from the ground through a review of some fundamental material from functional analysis also the theory behind the motivation of GPs before finally concluding with concrete algorithms for GP regression and classification. Chapters FIX and **??** will cover techniques for approximating a large matrix used with GPs that provides information on how similar each observation is to one other. Chapter **??** then gives alternative methods for solving linear systems, an essential component required for the GP algorithm to work.

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 $\ell = \mathcal{O}\left(\ln(n)\right)$ 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 $\boldsymbol{Q}\boldsymbol{Q}^*$ acts as a good projection for both the columns and row space of $\boldsymbol{A}$ so that we have both $\boldsymbol{A} \simeq \boldsymbol{Q}\boldsymbol{Q}^*\boldsymbol{A}$ and $\boldsymbol{A} \simeq \boldsymbol{A}\boldsymbol{Q}\boldsymbol{Q}^*$ so that

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

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

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

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 $\overline{\boldsymbol{A}} \simeq \boldsymbol{A}$.

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

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*

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

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 2 [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 2, 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 2, 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,

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

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 2 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 2 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 $1/c\sqrt{p_i p_j}$.

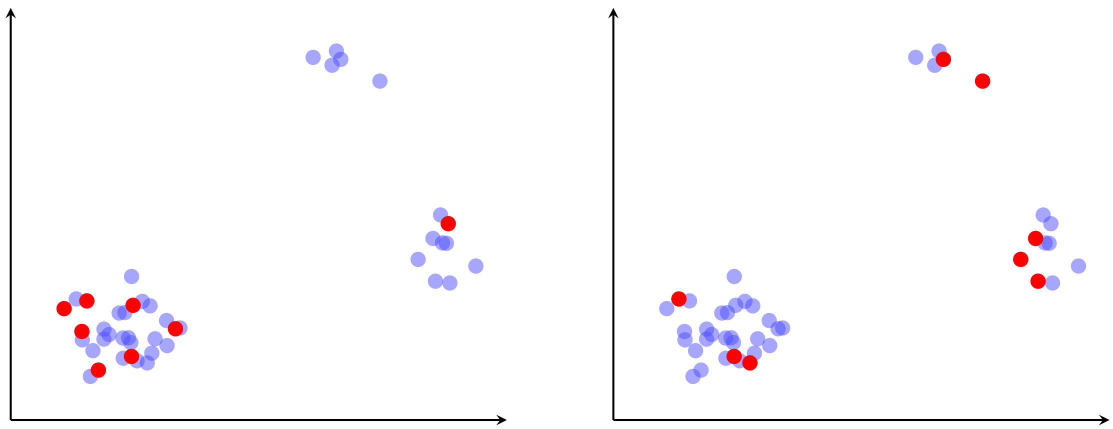
Compute $\boldsymbol{W}^\dagger$

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 2. 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, PDe11, MBCaCMaCM15, Kum09]. A few of the more common distributions will be discussed in the coming sections.

1.2. **Column Probabilities.** Recall that the Nystrom method from Algorithm 3 is largely dependent on the random matrix multiplication algorithm (Algorithm 2) to produce a suitable sketching matrix. Moreover, improvements in the sketching matrix produced by the random matrix multiplication algorithmare reflected as smaller errors in the Nystrom approximation. Now, consider using the random matrix multiplication algorithm to approximate $\boldsymbol{A}\boldsymbol{A}^\intercal$ by setting $\boldsymbol{B} = \boldsymbol{A}$. The output is an approximation of the form

$$\boldsymbol{A}\boldsymbol{A}^\intercal \simeq \boldsymbol{C}\boldsymbol{C}^\intercal = \boldsymbol{C}\boldsymbol{R}.$$



(A) (B)

FIGURE 2. Employing uniform sampling in the column sampling Nystrom estimate can lead to oversampling from denser parts of the data set. Instead data dependent probability densities are commonly used to better cover the relevant data. Example taken from [CMaCM17, page 4].

The probability distribution

$$p_i = \frac{\left\|\boldsymbol{A}_{(i,:)}\right\|_2^2}{\|\boldsymbol{A}\|_F}.$$

aims to minimize the error between $\boldsymbol{A}\boldsymbol{A}^\intercal$ and the approximation $\boldsymbol{C}\boldsymbol{C}^\intercal$. As a result, we should expect that $\boldsymbol{C}$ becomes a better estimate for $\boldsymbol{A}\boldsymbol{S}$, implying that the sketching matrix, $\boldsymbol{S}$, is using a better sampling and landmark selection criteria. Drineas and Mahoney give a precise bound on this error presented in theorem 4 [PDaMWM05, page 2158].

Theorem 4. *Given $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $1 \le c \le n$ and the probability distribution $\{p_i\}_{i=1}^n$ described in equation 1.2. Construct $\boldsymbol{C}$ using algorithm 2, then*

$$\mathbb{E}\left[\|\boldsymbol{A}\boldsymbol{A}^\intercal - \boldsymbol{C}\boldsymbol{C}^\intercal\|_F\right] \le \frac{1}{\sqrt{c}}\|\boldsymbol{A}\|_F^2$$

[PDaMWM05, page 2158].

To show theorem 4, we can actually prove something a little more general.

Lemma 5. *Given $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times p}$, $1 \le c \le n$ and the probability distribution $\{p_i\}_{i=1}^n$ as follows*

$$p_i = \frac{\left\|\boldsymbol{A}_{(k,:)}\right\|_2 \left\|\boldsymbol{B}_{(:,k)}\right\|_2}{\sum_{j=1}^n \left\|\boldsymbol{A}_{(k,:)}\right\|_2 \left\|\boldsymbol{B}_{(:,k)}\right\|}.$$

Construct $\boldsymbol{C}$ using algorithm 2, using the probability distribution described above, then

$$\mathbb{E}\left[\|\boldsymbol{A}\boldsymbol{B} - \boldsymbol{C}\boldsymbol{R}\|_F\right] \le \frac{1}{\sqrt{c}}\|\boldsymbol{A}\|_F^2\|\boldsymbol{B}\|_F^2.$$

This choice of probability distribution minimises $\mathbb{E}\left[\|\boldsymbol{A}\boldsymbol{B} - \boldsymbol{C}\boldsymbol{R}\|_F\right]$ among all possible sampling probabilites [Dri06, pages 9-12].

Proof. First note that

$$\sum_{i=1}^{m}\sum_{j=1}^{p}\mathbb{E}\left[(\boldsymbol{AB}-\boldsymbol{CR})_{ij}^2\right]=\sum_{i=1}^{m}\sum_{j=1}^{p}\mathbb{V}\left[(\boldsymbol{CR})_{ij}\right].$$

Thus from lemma 3, it follows that

$$\begin{aligned}&\mathbb{E}\left[\|\boldsymbol{AB}-\boldsymbol{CR}\|_F^2\right]\\ =&\frac{1}{c}\sum_{i=1}^{n}\frac{1}{p_k}\left(\sum_{i=1}^{m}\boldsymbol{A}_{ik}^2\right)\left(\sum_{j=1}^{p}\boldsymbol{B}_{kj}^2\right)-\frac{1}{c}\|\boldsymbol{AB}\|_F^2\\ =&\frac{1}{c}\sum_{i=1}^{n}\frac{1}{p_k}\left\|\boldsymbol{A}_{(i,:)}\right\|_2^2\cdot\left\|\boldsymbol{B}_{(:,i)}\right\|_2^2-\frac{1}{c}\|\boldsymbol{AB}\|_F^2.\end{aligned}$$

Substituting in a probability of

$$p_i=\frac{\left\|\boldsymbol{A}_{(i,:)}\right\|_2\left\|\boldsymbol{B}_{(:,i)}\right\|_2}{\sum_{j=1}^{n}\left\|\boldsymbol{A}_{(j,:)}\right\|_2\left\|\boldsymbol{B}_{(:,j)}\right\|}.$$

yields

$$\begin{aligned}\mathbb{E}\left[\|\boldsymbol{AB}-\boldsymbol{CR}\|_F^2\right]=&\frac{1}{c}\left(\sum_{i=1}^{n}\left\|\boldsymbol{A}_{(i,:)}\right\|_2\left\|\boldsymbol{B}_{(:,k)}\right\|_2\right)^2-\frac{1}{c}\|\boldsymbol{AB}\|_F^2\\ \leq&\frac{1}{c}\|\boldsymbol{A}\|_F^2\|\boldsymbol{B}\|_F^2.\end{aligned}$$

To verify that this choice of probability distribution minimises $\mathbb{E}\left[\|\boldsymbol{AB}-\boldsymbol{CR}\|_F\right]$ define the function

$$f(p_1,\ldots,p_n)=\sum_{i=1}^{n}\frac{1}{p_i}\left\|\boldsymbol{A}_{(i,:)}\right\|_2^2\cdot\left\|\boldsymbol{B}_{(:,i)}\right\|_2^2$$

which characterises the dependence of $\mathbb{E}\left[\|\boldsymbol{AB}-\boldsymbol{CR}\|_F\right]$ on the probability distribution. To minimise f subject to $\sum_{i=1}^{n}p_i=1$, we introduce the Lagrange multiplier λ and define the function

$$g(p_1,\ldots,p_n)=f(p_i,\ldots,p_n)+\lambda\left(\sum_{i=1}^{n}p_i-1\right).$$

The minimum is then

$$0=\frac{\partial g}{\partial p_i}=-\frac{1}{p_i^2}\left\|\boldsymbol{A}_{(k,:)}\right\|_2^2\cdot\left\|\boldsymbol{B}_{(:,k)}\right\|_2^2+\lambda.$$

Thus

$$p_i=\frac{\left\|\boldsymbol{A}_{(i,:)}\right\|_2\cdot\left\|\boldsymbol{B}_{(:,i)}\right\|_2}{\sqrt{\lambda}}=\frac{\left\|\boldsymbol{A}_{(i,:)}\right\|_2\cdot\left\|\boldsymbol{B}_{(:,i)}\right\|_2}{\sum_{j=1}^{n}\left\|\boldsymbol{A}_{(j,:)}\right\|_2\left\|\boldsymbol{B}_{(:,j)}\right\|_2}$$

where the second equality comes from solving for $\sqrt{\lambda}$ in $\sum_{i=1}^{n-1}p_i=1$. These probabilities are indeed minimizing since $\frac{\partial^2 g}{\partial p_i^2}>0$ for every i such that $\left\|\boldsymbol{A}_{(i,:)}\right\|_2^2\cdot\left\|\boldsymbol{B}_{(:,i)}\right\|_2^2>0$. □

1.3. **Leverage Scores.**

1.3.1. *Statistical Leverage Scores.* Our next distribution originates from the least-squares problem. Breifly, in an over constrained least-squares problem, where $\boldsymbol{A} \in \mathbb{R}^{n\times m}$, $\boldsymbol{b} \in \mathbb{R}^{n}$, for $m \ll n$ there usually is not any $\boldsymbol{x} \in \mathbb{R}^m$ for which $\boldsymbol{Ax} = \boldsymbol{b}$. Instead, alternative criteria are used to seek a $\boldsymbol{x}$ which in some way comes closest to satisfying this equality. Perhaps one of the more popular criterion is to minimize the ℓ^2-norm, that is

$$\boldsymbol{x}_{opt} = \arg\min_{x} \|\boldsymbol{Ax} - \boldsymbol{b}\|$$

[MWM11, page 19-21]. This is what the least-squares problem is. The optimal value for $\boldsymbol{x}$ can be solved as $\boldsymbol{x}_{opt} = (\boldsymbol{A}^*\boldsymbol{A})^{-1}\boldsymbol{A}^*\boldsymbol{b}$. The least-squares solution is commonly used to find the best weight vector (in this case $\boldsymbol{x}$) for a linear model, given a dataset. Fitted or predicted values are usually obtained from $\hat{\boldsymbol{b}} = \boldsymbol{Hb}$ where the projector onto the column space of $\boldsymbol{A}$

$$\boldsymbol{H} = \boldsymbol{A}\left(\boldsymbol{A}^\intercal\boldsymbol{A}\right)^{-1}\boldsymbol{A}^\intercal$$

is sometimes referred to as the *hat matrix*. The element $\boldsymbol{H}_{ij}$ has the direct interpretation as the influence or statistical leverage exerted on $\hat{\boldsymbol{b}}_i$. Thus, examining the hat matrix can reveal to us columns of $\boldsymbol{A}$ which bear a significant impact on $\hat{\boldsymbol{b}}$ [Hoa78, page 17]. Relatedly, if the element $\boldsymbol{H}_{ii}$ is particularly large this is indicative of the i^{th} column of $\boldsymbol{A}$ having great influence in determining values of $\hat{\boldsymbol{b}}$, justifying the interpretation of $\boldsymbol{H}_{ii}$ as statistical leverage scores.

The statistical leverage scores are maximised when $\boldsymbol{A}_{(:,i)}$ is linearly independent from $\boldsymbol{A}$'s other columns and decreases when it aligns with many other columns or when the value of $\left\|\boldsymbol{A}_{(:,i)}\right\|$ is small [MBCaC-MaCM15, page 5]. To compute the statistical leverage scores, if $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\intercal$ is the SVD of $\boldsymbol{A}$, then

$$\begin{aligned}
\boldsymbol{H}_{ii} &= \left(\boldsymbol{A}\left(\boldsymbol{A}^\intercal\boldsymbol{A}\right)^{-1}\boldsymbol{A}^\intercal\right)_{ii} \\
&= \left(\boldsymbol{U}\boldsymbol{\Sigma}^2\left(\boldsymbol{\Sigma}^2\right)^{-1}\boldsymbol{U}\right)_{ii} \\
&= \left\|\boldsymbol{U}_{(i,:)}\right\|_2^2.
\end{aligned}$$

Note that $\boldsymbol{H}_{ii}$ may not constitute as a probability distribution, as may the other leverage scores which we will soon discuss. This is easily enough fixed by normalisation. The idea behind using statistical leverage scores as a probability distribution in the Nystrom method is that statistical leverage scores help us priorities selecting columns that are more linearly independent from other columns so that the range of our approximate more closely aligns with the range of our original $\boldsymbol{A}$.

1.3.2. *Rank$-k$ Statistical Leverage Scores.* We can generalize this notion of statistical leverage scores to include lower rank approximations. As before let $\boldsymbol{A} = \boldsymbol{U\Sigma V}^\intercal$ be the SVD of $\boldsymbol{A}$. The SVD can be partitioned as

$$\boldsymbol{U} = [\boldsymbol{U}_1, \boldsymbol{U}_2] \qquad \boldsymbol{\Sigma} = \begin{bmatrix}\boldsymbol{\Sigma}_1 & \\ & \boldsymbol{\Sigma}_2\end{bmatrix} \qquad \boldsymbol{V} = [\boldsymbol{V}_1, \boldsymbol{V}_2].$$

Here $\boldsymbol{U}_1$ contains the first k columns of $\boldsymbol{U}$, $\boldsymbol{V}_1$ the first k rows of $\boldsymbol{V}$ and $\boldsymbol{\Sigma}_1$ is a $k \times k$ matrix containing the top k singular values across its diagonal. The matrix $\boldsymbol{A}_k = \boldsymbol{U}_1\boldsymbol{\Sigma}_1\boldsymbol{V}_1$ then forms the best rank$-k$ approximation to $\boldsymbol{A}$. The statistical leverage scores relative to the best rank$-k$ approximation are again

$\boldsymbol{H}_{ii}$, but this time $\boldsymbol{H}$ is computed only using the best rank$-k$ approximation of $\boldsymbol{A}$, that is $\boldsymbol{A}_k$. These low rank scores can be evaluated as

$$\ell_i^k \triangleq \left(\boldsymbol{A}_k \left(\boldsymbol{A}_k^{\intercal}\boldsymbol{A}_k\right)^{-1} \boldsymbol{A}_k^{\intercal}\right)_{ii} = \left\|(\boldsymbol{U}_1)_{(i,:)}\right\|_2^2.$$

What makes low-rank statistical leverage scores particularly appealing is that they can be approximated quickly with a truncated SVD [AGaMWM13, pages 3-4].

1.3.3. *Ridge Leverage Scores.* The low rank leverage scores we saw in equation 1.3.2 will not always be unique and can be sensitive to perturbations [MBCaCMaCM15, page 6]. As you could guess, the prediction results can very drastically when $\boldsymbol{A}$ is modified slightly or when we only have access to partial information on the matrix. This largely undermines the the possibility of computing good quality low rank approximations of statistical leverage scores. This motivates the next class of leverage score, ridge leverage scores. Ridge leverage scores are similar to statistical leverage scores although a ridge regression term (hence the name) is within the hat matrix for a given regularization parameter λ. The $\lambda-$ridge leverage score is defined as

$$r_i^{\lambda} \triangleq \left(\boldsymbol{A}\left(\boldsymbol{A}^{\intercal}\boldsymbol{A} + \lambda \mathbb{1}_{n\times n}\right)^{-1} \boldsymbol{A}^{\intercal}\right)_{ii}.$$

A regularization parameter of

$$\lambda = \frac{\|\boldsymbol{A} - \boldsymbol{A}_k\|_F^2}{k}$$

is typically used since this choice of λ will guarantee that the sum of the ridge leverage scores (keep in mind that the raw ridge leverage do not necessarily form a probability distribution) is bounded by $2k$, stated more formally in lemma 6.

Lemma 6. *When using a regularization parameter of $\lambda = \frac{\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2}{k}$ we have $\sum_{i=1}^n r_i^{\lambda} \leq 2k$* [MBCaCMaCM15, pages 6-7].

Proof. Writing r_i^{λ} using the SVD of $\boldsymbol{A}$ where $\lambda = \frac{\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2}{k}$ gives

$$\begin{aligned} r_i^{\lambda} &= \boldsymbol{A}_{(i,:)}\left(\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^{\intercal} + \frac{\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2}{k}\boldsymbol{U}\boldsymbol{U}^{\intercal}\right)^{-1}\boldsymbol{A}_{(i,:)}^{\intercal} \\ &= \boldsymbol{A}_{(i,:)}\left(\boldsymbol{U}\overline{\boldsymbol{\Sigma}}^2\boldsymbol{U}^{\intercal}\right)^{-1}\boldsymbol{A}_{(i,:)}^{\intercal} \\ &= \boldsymbol{A}_{(i,:)}\left(\boldsymbol{U}\overline{\boldsymbol{\Sigma}}^{-2}\boldsymbol{U}^{\intercal}\right)\boldsymbol{A}_{(i,:)}^{\intercal} \end{aligned}$$

where $\overline{\boldsymbol{\Sigma}}_{ii}^2 = \sigma_i^2(\boldsymbol{A}) + \frac{\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2}{k}$. We then have

$$\begin{aligned} \sum_{i=1}^n r_i^{\lambda} &= \operatorname{tr}\left(\boldsymbol{A}^{\intercal}\boldsymbol{U}\overline{\boldsymbol{\Sigma}}^{-2}\boldsymbol{U}^{\intercal}\boldsymbol{A}\right) \\ &= \operatorname{tr}\left(\boldsymbol{V}\boldsymbol{\Sigma}\overline{\boldsymbol{\Sigma}}^{-2}\boldsymbol{\Sigma}\boldsymbol{V}^{\intercal}\right) \\ &= \operatorname{tr}\left(\boldsymbol{\Sigma}^2\overline{\boldsymbol{\Sigma}}^{-2}\right). \end{aligned}$$

Here we have

$$\left(\boldsymbol{\Sigma}^2\overline{\boldsymbol{\Sigma}}^{-2}\right)_{ii} = \frac{\sigma_i^2(\boldsymbol{A})}{\sigma_i^2(\boldsymbol{A}) + \frac{\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2}{k}}.$$

For $i \leq k$ we simply upper bound this by 1, yielding

$$\operatorname{tr}\left(\boldsymbol{\Sigma}^2\overline{\boldsymbol{\Sigma}}^{-2}\right) = k + \sum_{i=k+1}^{n} \frac{\sigma_i^2\left(\boldsymbol{A}\right)}{\sigma_i^2\left(\boldsymbol{A}\right) + \frac{\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2}{k}} \leq k + \sum_{i=k+1}^{n} \frac{\sigma_i^2\left(\boldsymbol{A}\right)}{\frac{\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2}{k}} = k + \frac{\sum_{i=k+1}^{n} \sigma_i^2\left(\boldsymbol{A}\right)}{\frac{\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2}{k}} \leq k + k.$$

□

From now on (unless otherwise stated) the regularization parameter seen in 1.3.3 will always be used for ridge leverage scores where the notation

$$r_i^k \triangleq \left(\boldsymbol{A}\left(\boldsymbol{A}^\intercal\boldsymbol{A} + \left(\frac{\|\boldsymbol{A}-\boldsymbol{A}_k\|_F^2}{k}\right)\mathbb{1}_{n\times n}\right)^{-1}\boldsymbol{A}^\intercal\right)_{ii}$$

will be used to show that the best rank$-k$ matrix is used in the regularization parameter. Adding regularization to the hat matrix offers a smoother alternative which 'washes out' small singular directions meaning they are sampled with proportionally lower probability [MBCaCMaCM15, page 6].

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.

[Hoa78] David C and Welsch Hoaglin Roy E, *The Hat Matrix in Regression and ANOVA*, The American statistician **32** (1978), no. 1, 17–22 (eng).

[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).

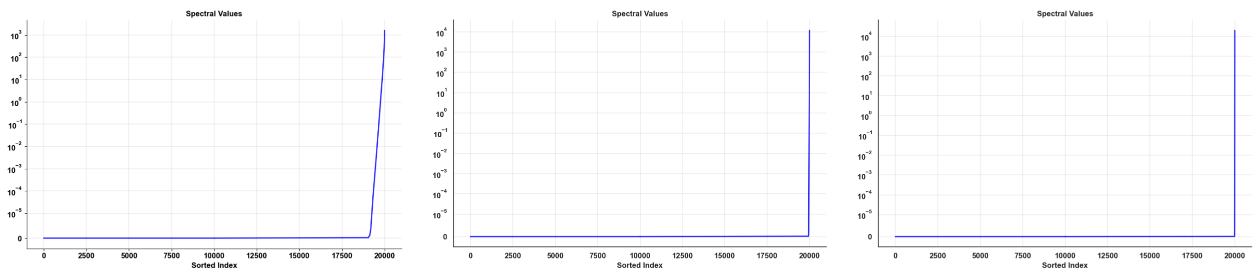
[Cho20] Krzysztof and Likhosherstov Choromanski Valerii and Dohan, *Rethinking Attention with Performers* (2020) (eng).

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

Appendix A. Additional Results

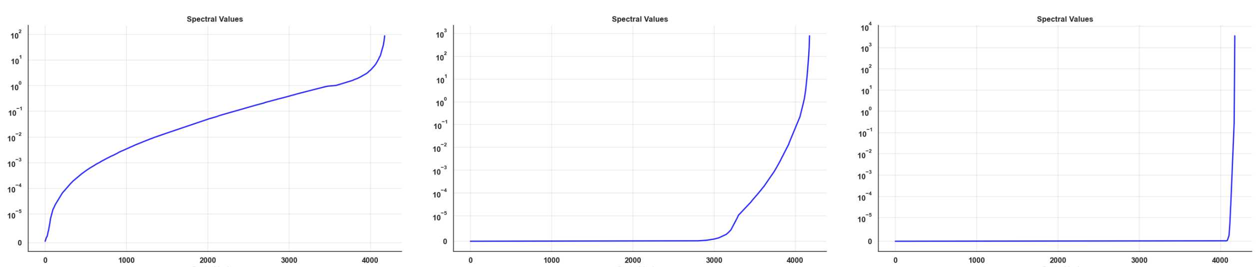
Additional results that may have not been included in the main text for conciseness.

A.1. **Gram Matrix Spectral Values.**



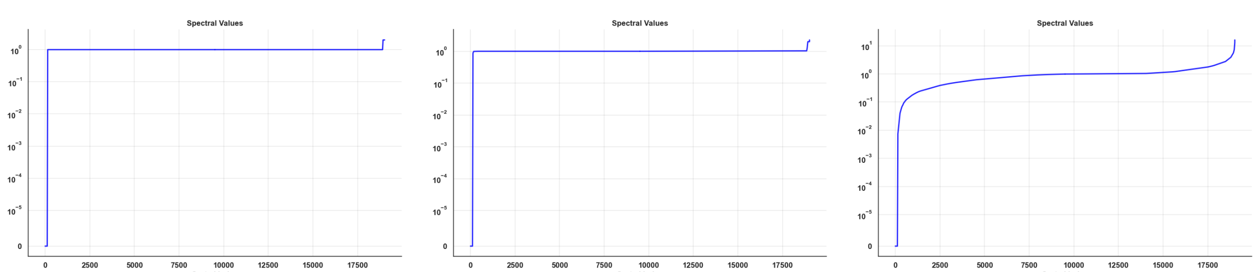
(A) $\sigma = 0.1$ (B) $\sigma = 1.0$ (C) $\sigma = 10.0$

Figure 3. Spectral values for 3D-spatial network data.



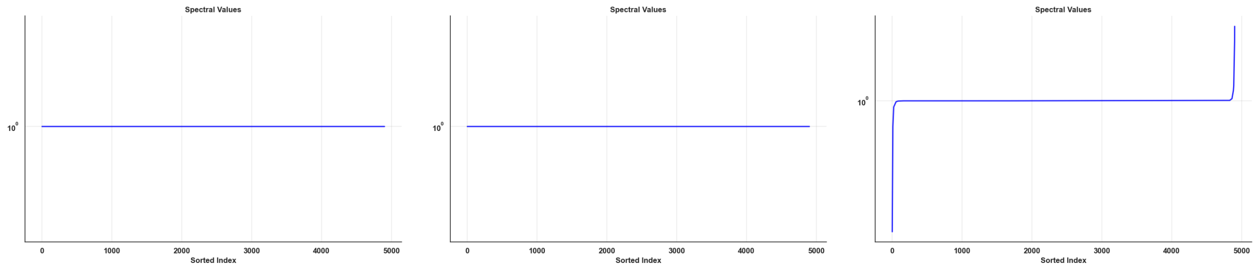
(A) $\sigma = 0.1$ (B) $\sigma = 1.0$ (C) $\sigma = 10.0$

Figure 4. Spectral values for abalone data.



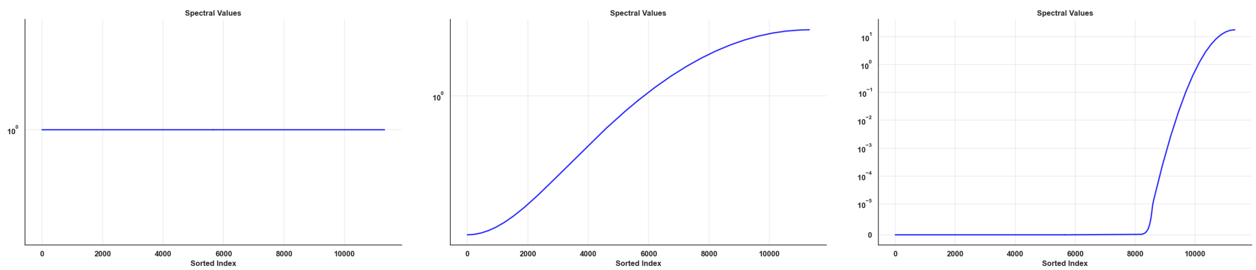
(A) $\sigma = 0.1$ (B) $\sigma = 1.0$ (C) $\sigma = 10.0$

Figure 5. Spectral values for magic04 data.



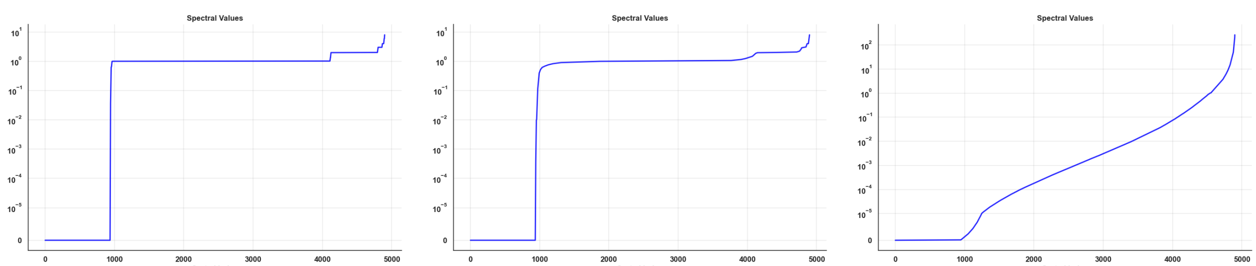
(A) $\sigma = 0.1$ (B) $\sigma = 1.0$ (C) $\sigma = 10.0$

FIGURE 6. Spectral values for stock market data.



(A) $\sigma = 0.1$ (B) $\sigma = 1.0$ (C) $\sigma = 10.0$

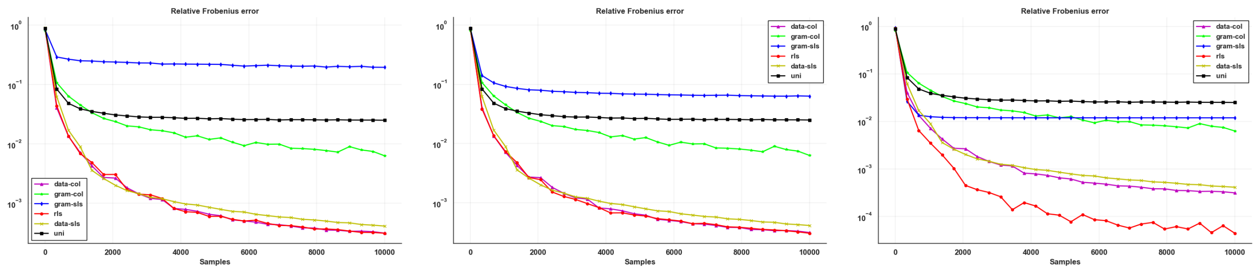
FIGURE 7. Spectral values for temperature data.



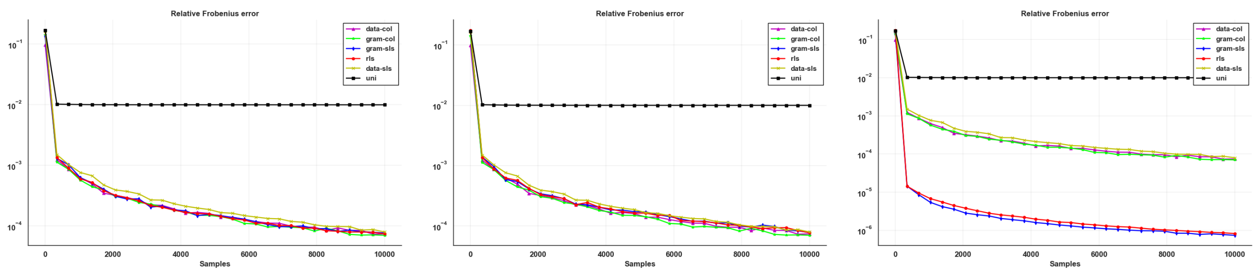
(A) $\sigma = 0.1$ (B) $\sigma = 1.0$ (C) $\sigma = 10.0$

FIGURE 8. Spectral values for wine data.

A.2. **Nystrom Errors.**

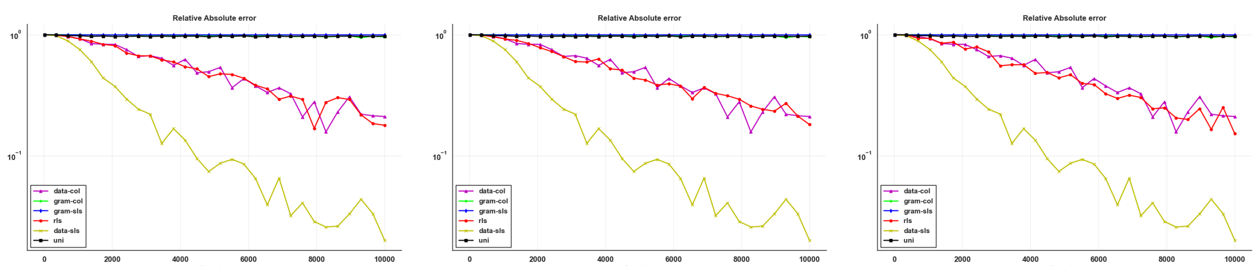


(A) $\sigma = 0.1,\ k = 10$ (B) $\sigma = 0.1,\ k = 20$ (C) $\sigma = 0.1,\ k = 80$

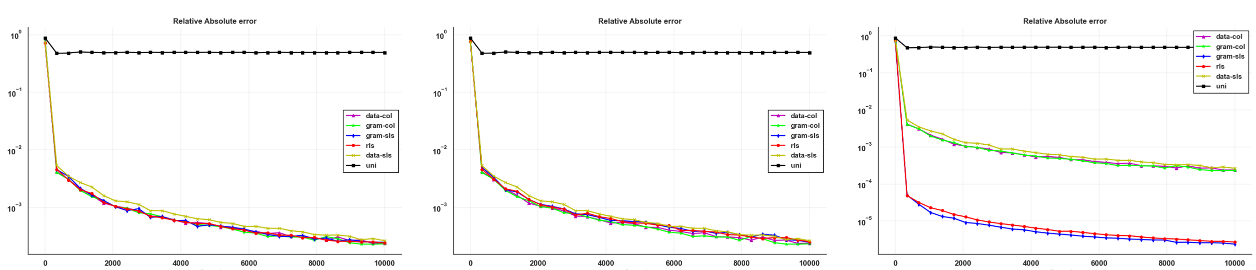


(D) $\sigma = 1.0,\ k = 10$ (E) $\sigma = 1.0,\ k = 20$ (F) $\sigma = 1.0,\ k = 40$

FIGURE 9. Nystrom Frobenius errors for the 3D-spatial network data set.

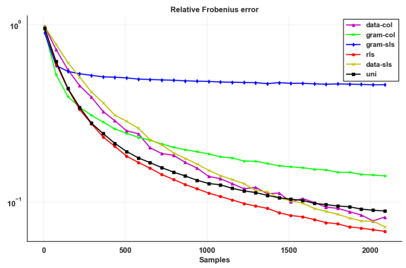


(A) $\sigma = 0.1,\ k = 10$ (B) $\sigma = 0.1,\ k = 20$ (C) $\sigma = 0.1,\ k = 40$

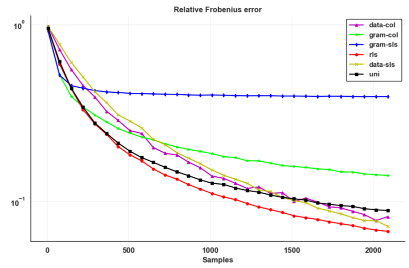


(D) $\sigma = 1.0,\ k = 10$ (E) $\sigma = 1.0,\ k = 20$ (F) $\sigma = 1.0,\ k = 40$

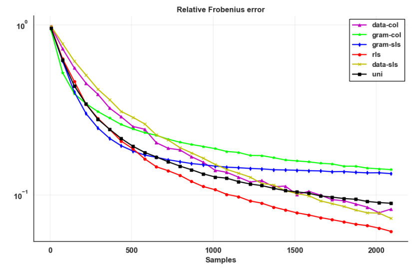
FIGURE 10. Nystrom absolute errors for the 3D-spatial network data set.



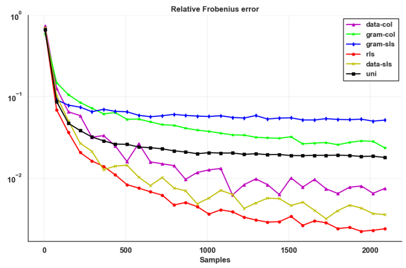
(A) $\sigma = 0.1,\ k = 10$



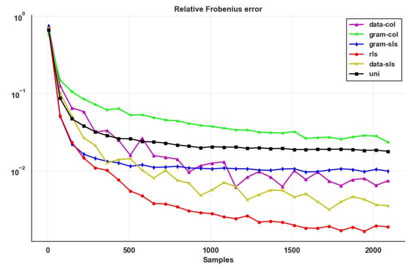
(B) $\sigma = 0.1,\ k = 20$



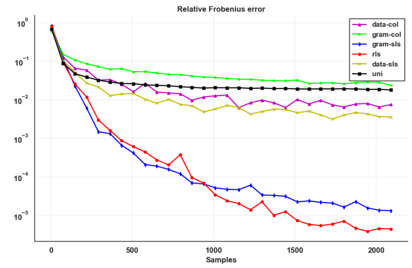
(C) $\sigma = 0.1,\ k = 80$



(D) $\sigma = 1.0,\ k = 10$

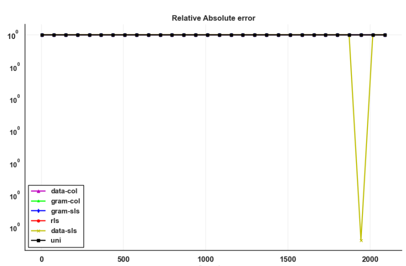


(E) $\sigma = 1.0,\ k = 20$

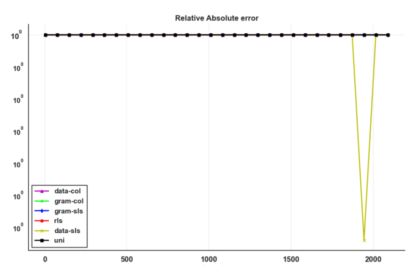


(F) $\sigma = 1.0,\ k = 80$

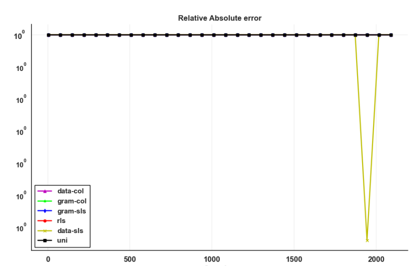
FIGURE 11. Nystrom Frobenius errors for the Abalone data set.



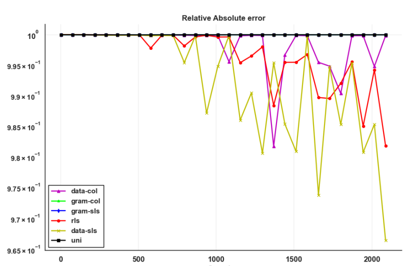
(A) $\sigma = 0.1,\ k = 10$



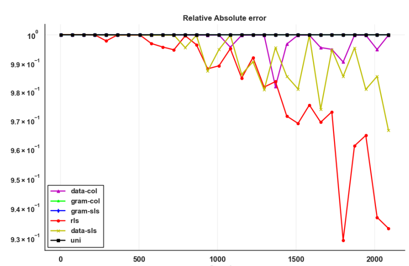
(B) $\sigma = 0.1,\ k = 20$



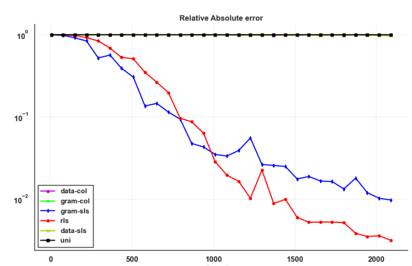
(C) $\sigma = 0.1,\ k = 80$



(D) $\sigma = 1.0,\ k = 10$

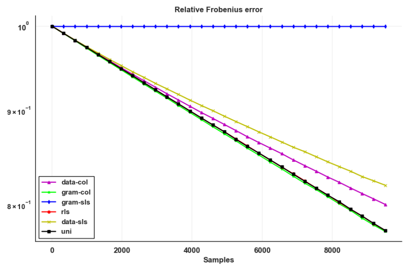


(E) $\sigma = 1.0,\ k = 20$

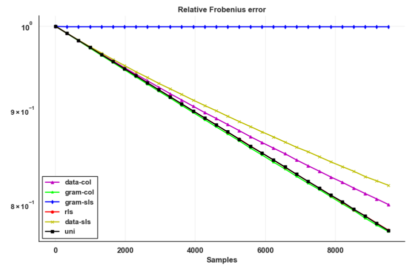


(F) $\sigma = 1.0,\ k = 80$

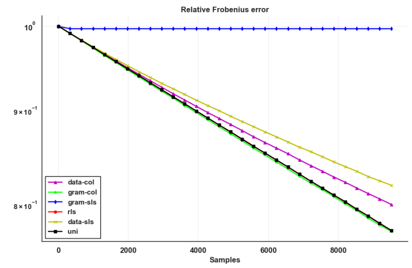
FIGURE 12. Nystrom absolute errors for the Magic data set.



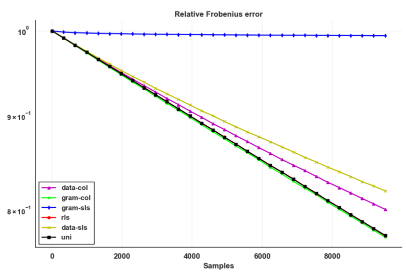
(A) $\sigma = 0.1,\ k = 10$



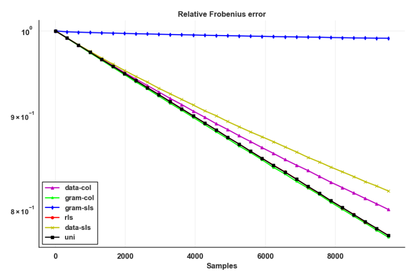
(B) $\sigma = 0.1,\ k = 20$



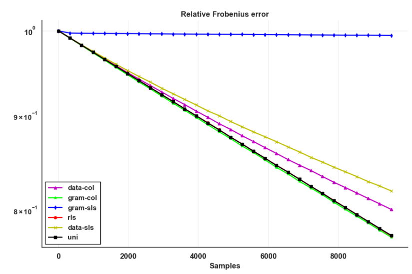
(C) $\sigma = 0.1,\ k = 80$



(D) $\sigma = 1.0,\ k = 10$

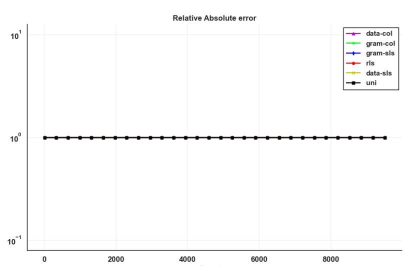


(E) $\sigma = 1.0,\ k = 20$

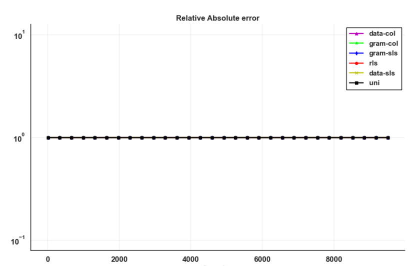


(F) $\sigma = 1.0,\ k = 80$

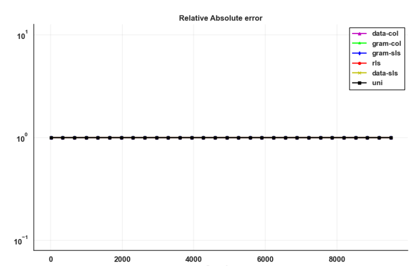
FIGURE 13. Nystrom Frobenius errors for the Magic data set.



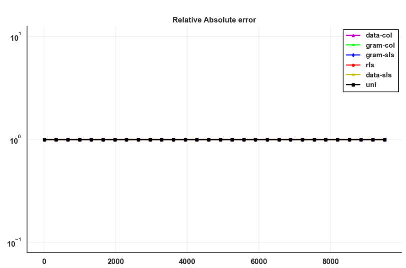
(A) $\sigma = 0.1,\ k = 10$



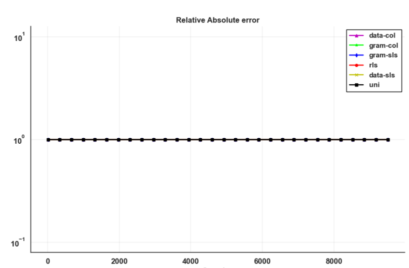
(B) $\sigma = 0.1,\ k = 20$



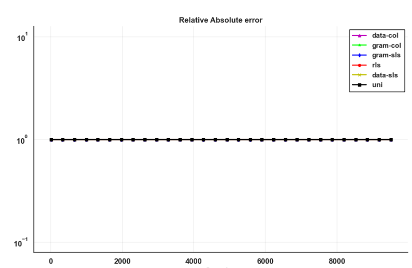
(C) $\sigma = 0.1,\ k = 80$



(D) $\sigma = 1.0,\ k = 10$

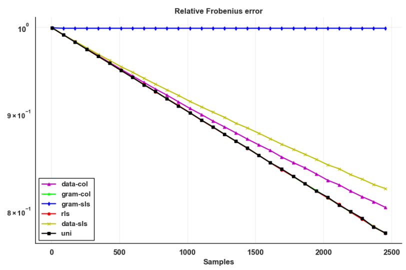


(E) $\sigma = 1.0,\ k = 20$

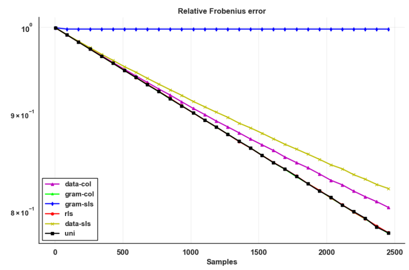


(F) $\sigma = 1.0,\ k = 80$

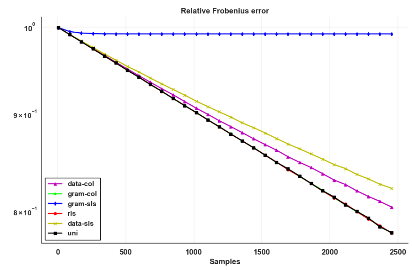
FIGURE 14. Nystrom absolute errors for the Magic data set.



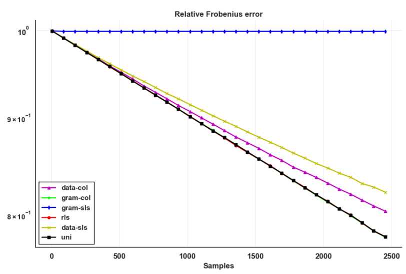
(A) $\sigma = 0.1,\ k = 10$



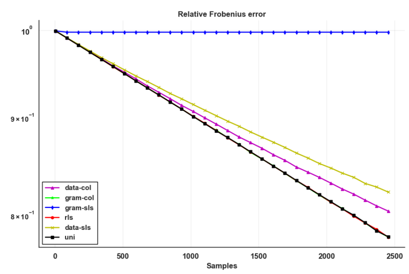
(B) $\sigma = 0.1,\ k = 20$



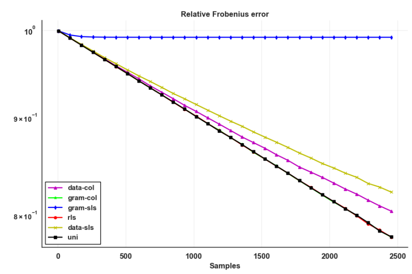
(C) $\sigma = 0.1,\ k = 80$



(D) $\sigma = 1.0,\ k = 10$

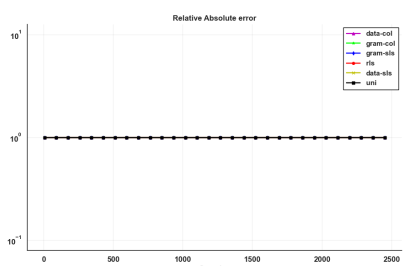


(E) $\sigma = 1.0,\ k = 20$

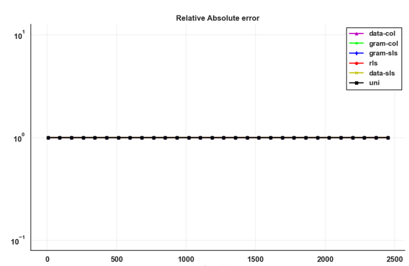


(F) $\sigma = 1.0,\ k = 80$

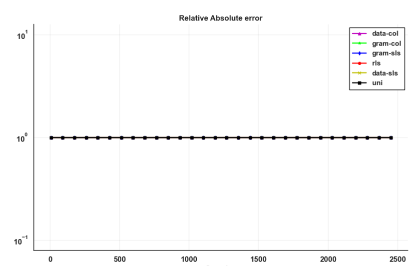
FIGURE 15. Nystrom Frobenius errors for the stock market data set.



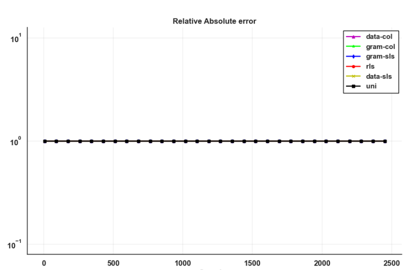
(A) $\sigma = 0.1,\ k = 10$



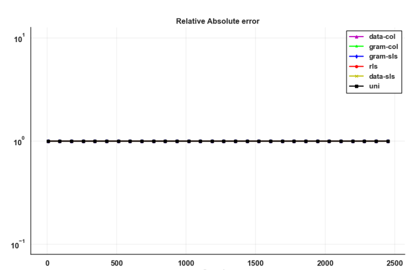
(B) $\sigma = 0.1,\ k = 20$



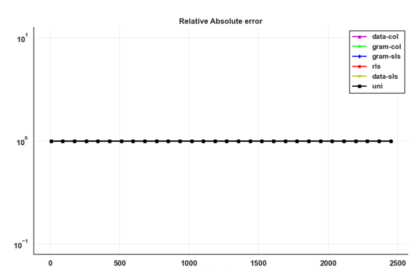
(C) $\sigma = 0.1,\ k = 80$



(D) $\sigma = 1.0,\ k = 10$

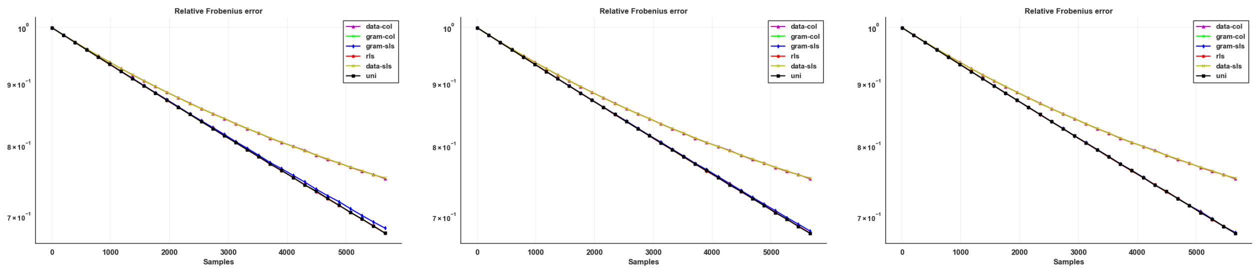


(E) $\sigma = 1.0,\ k = 20$

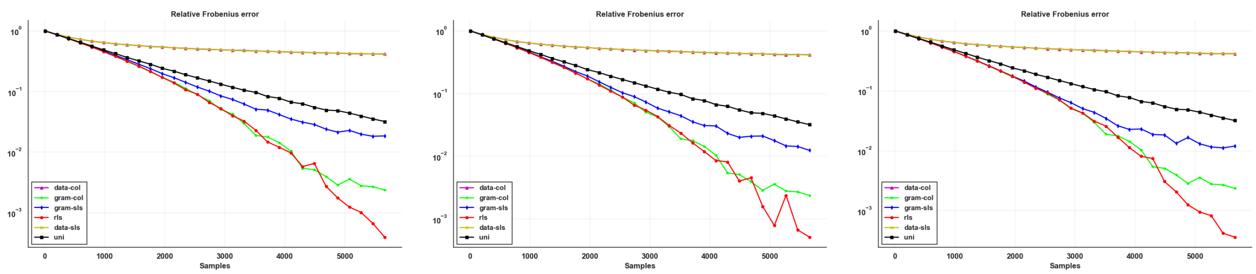


(F) $\sigma = 1.0,\ k = 80$

FIGURE 16. Nystrom absolute errors for the stock market data set.

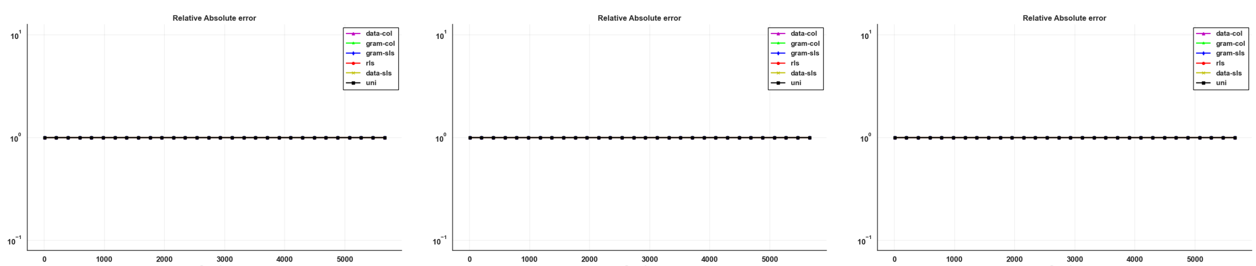


(A) $\sigma = 1.0, \ k = 10$ (B) $\sigma = 1.0, \ k = 20$ (C) $\sigma = 1.0, \ k = 80$

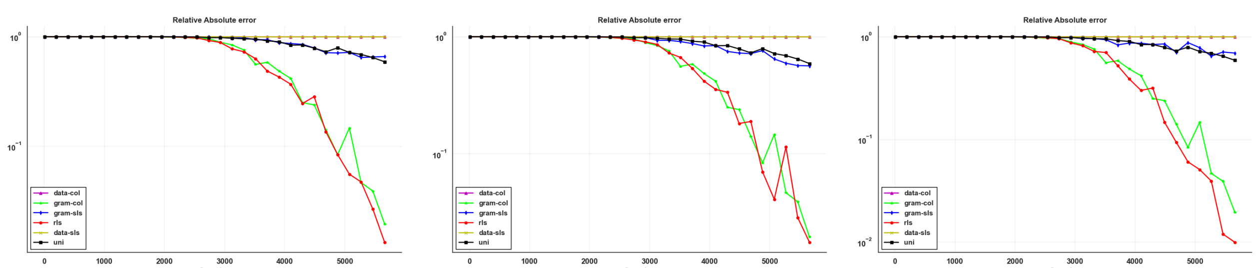


(D) $\sigma = 10.0, \ k = 10$ (E) $\sigma = 10.0, \ k = 20$ (F) $\sigma = 10.0, \ k = 80$

FIGURE 17. Nystrom Frobenius errors for the temperature data set.

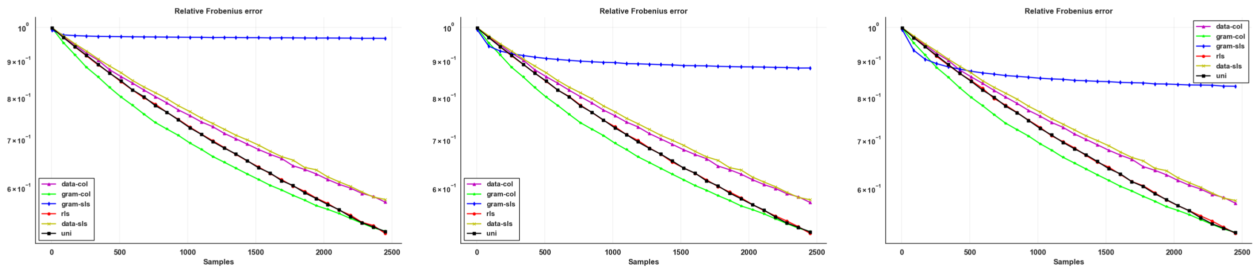


(A) $\sigma = 1.0, \ k = 10$ (B) $\sigma = 1.0, \ k = 20$ (C) $\sigma = 1.0, \ k = 80$

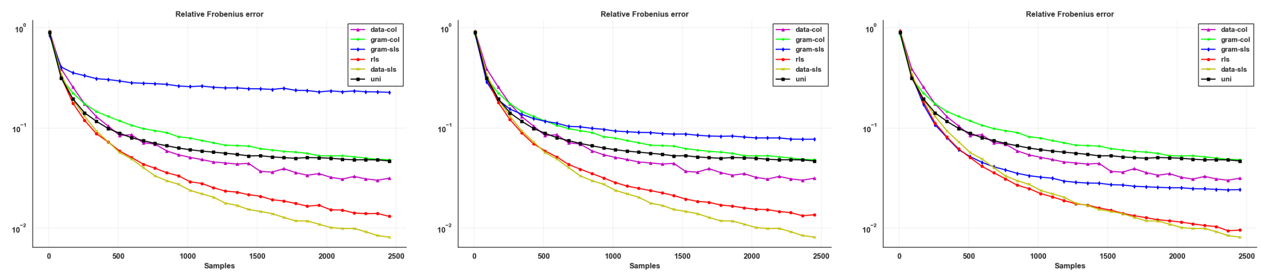


(D) $\sigma = 10.0, \ k = 10$ (E) $\sigma = 10.0, \ k = 20$ (F) $\sigma = 10.0, \ k = 80$

FIGURE 18. Nystrom absolute errors for the temperature data set.

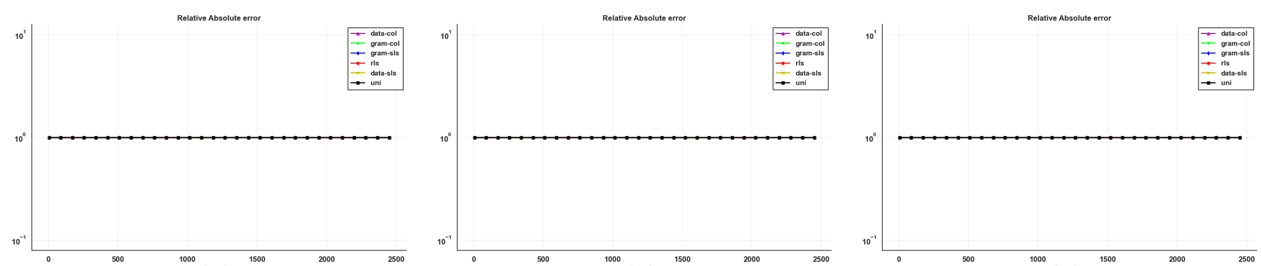


(A) $\sigma = 2.1,\ k = 5$ (B) $\sigma = 2.1,\ k = 20$ (C) $\sigma = 2.1,\ k = 40$

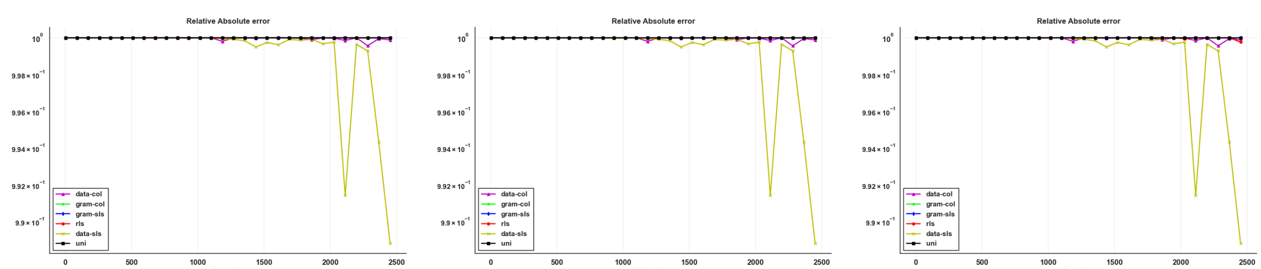


(D) $\sigma = 10.0,\ k = 5$ (E) $\sigma = 10.0,\ k = 20$ (F) $\sigma = 10.0,\ k = 80$

FIGURE 19. Nystrom Frobenius errors for the wine data set.



(A) $\sigma = 2.1,\ k = 5$ (B) $\sigma = 2.1,\ k = 20$ (C) $\sigma = 2.1,\ k = 40$



(D) $\sigma = 10.0,\ k = 5$ (E) $\sigma = 10.0,\ k = 20$ (F) $\sigma = 10.0,\ k = 80$

FIGURE 20. Nystrom absolute errors for the wine data set.