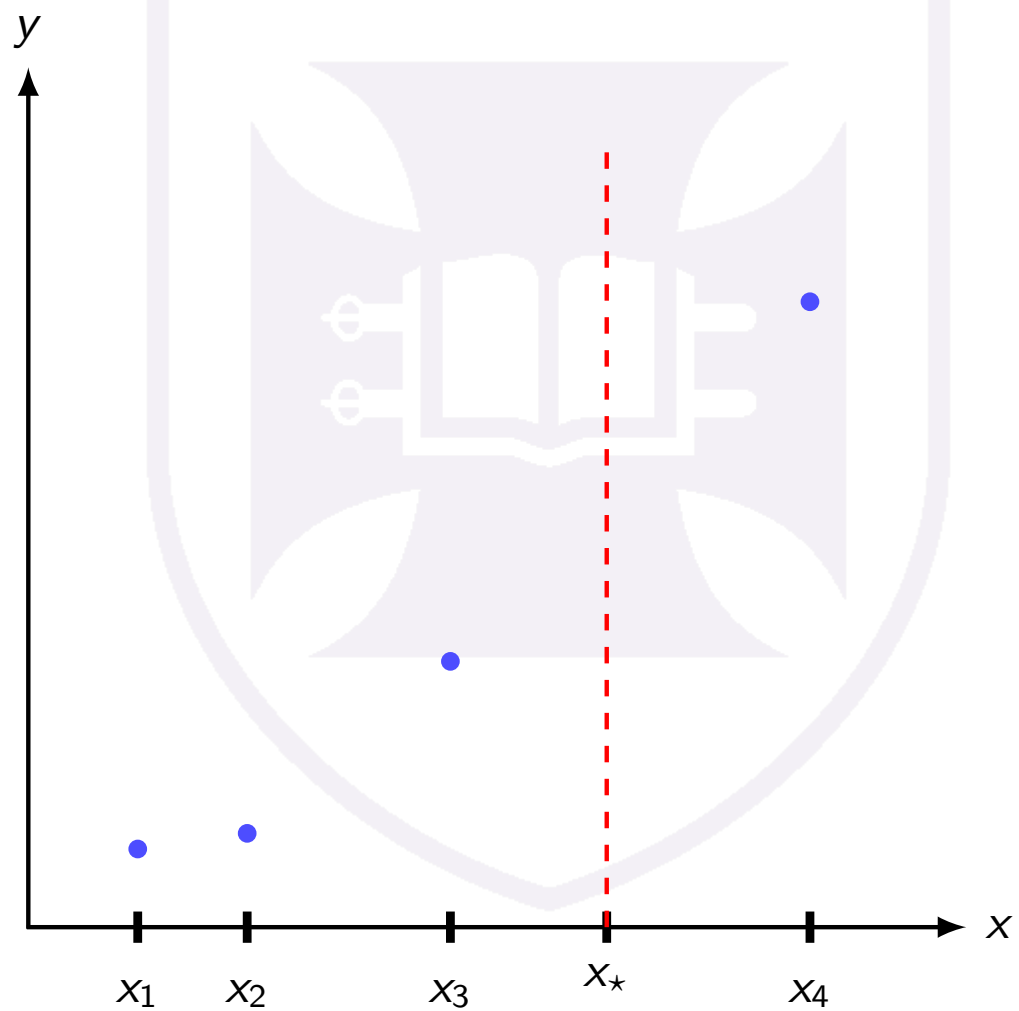
Optimizing Gaussian Processes

Honours Research Project

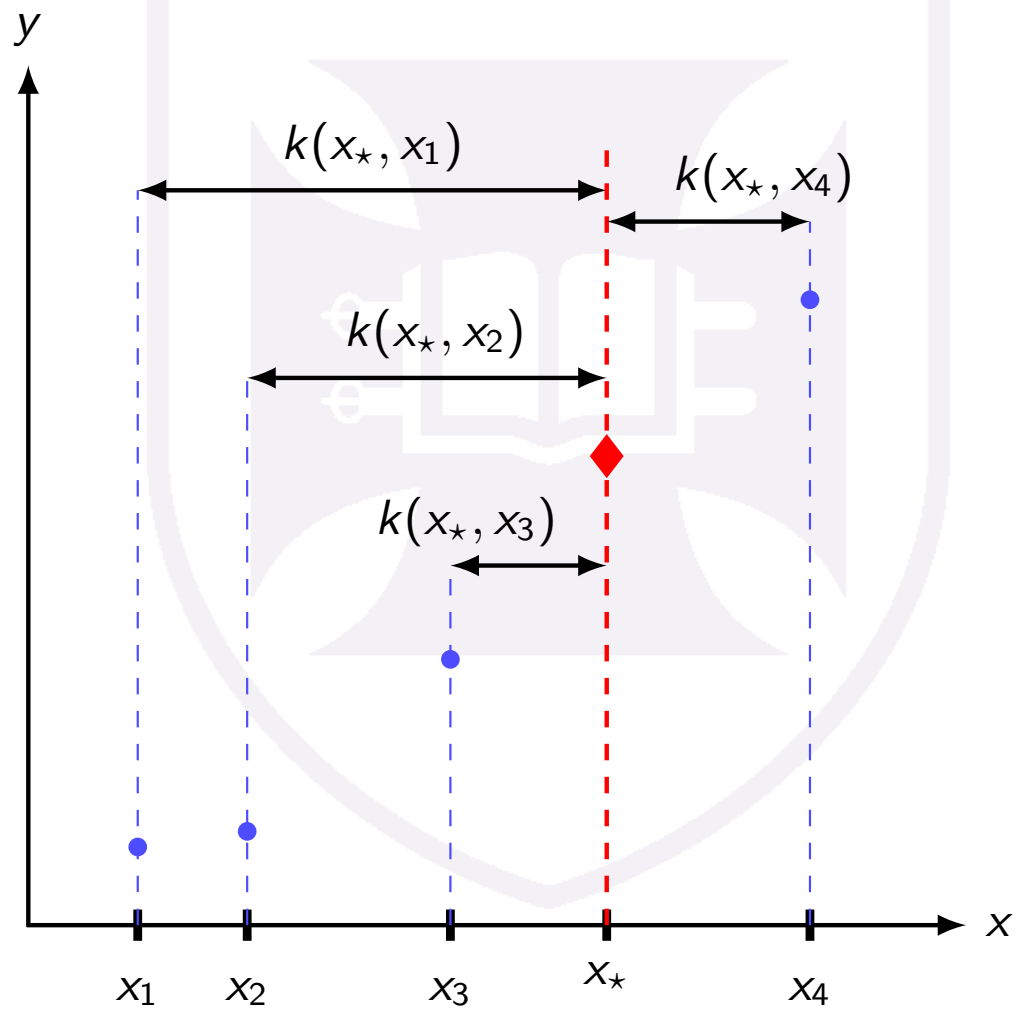
Michael Ciccotosto-Camp - 44302913

THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

Time Series Prediction

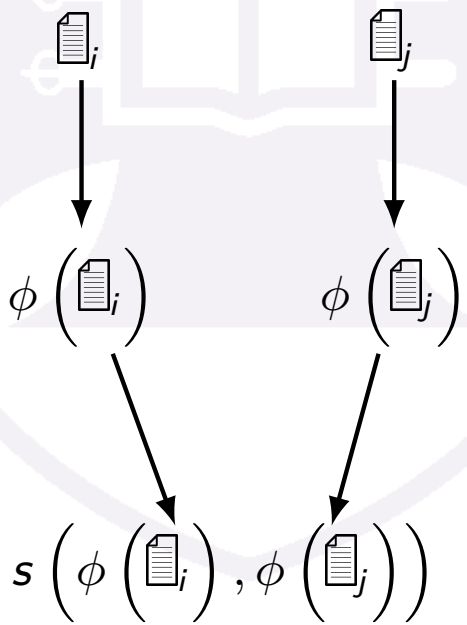


Time Series Prediction



The Kernel Trick

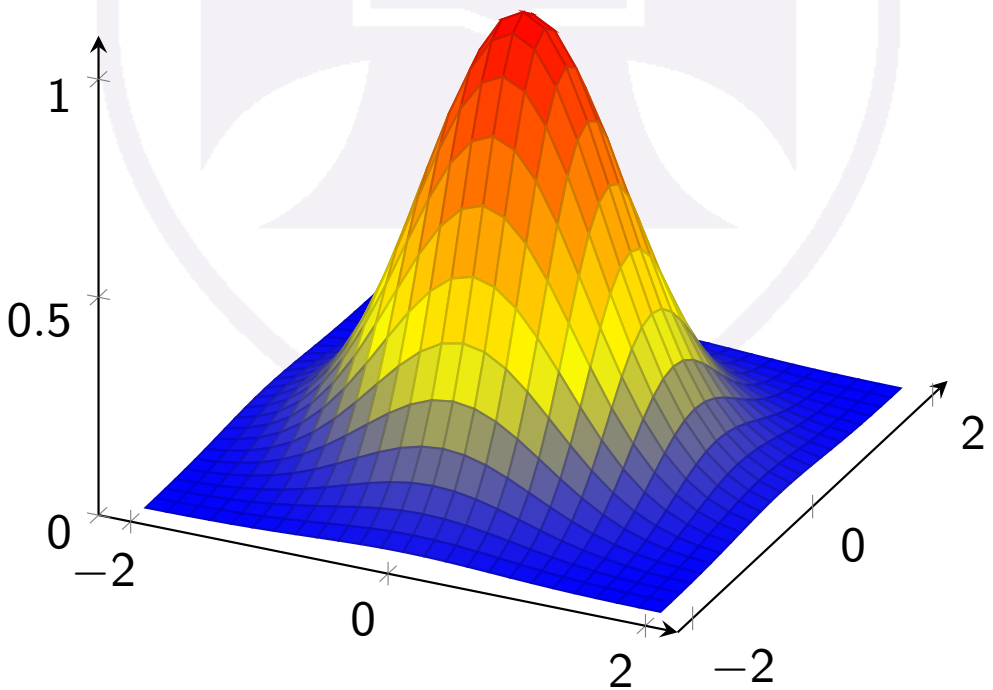
- Q: How do we get a suitable function k for computing similarity? A: Use the kernel trick!
- Suppose we have some inputs $\left[\blacksquare_1, \ldots, \blacksquare_n\right]$ (with their corressponding experimental observations $[y_1, \ldots, y_n]$), where $\blacksquare_i$ can take a number of different of form (perhaps a tree data structure or vectors of values).



- The function s provides us with some notion of similarity between inputs after they've been "transformed" into a nicer form using a *feature map* ϕ.

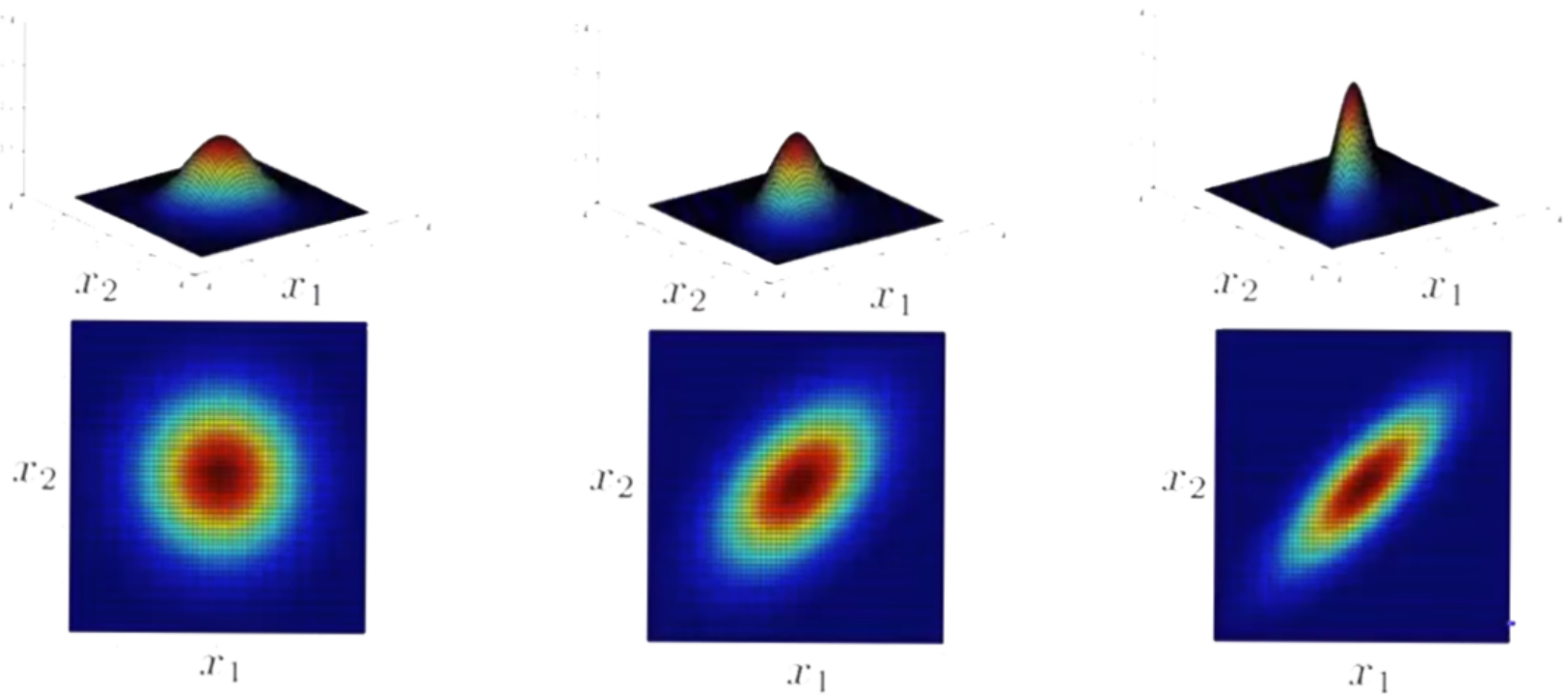
The Kernel Trick

- The kernel function k does all this computation is one step so that $k\left(\text{📄}_i, \text{📄}_j\right) = s\left(\phi\left(\text{📄}_i\right), \phi\left(\text{📄}_j\right)\right)$.
- Usually we have access to k, meaning we can avoid having to construct a feature map ϕ and similarity function s.
- A very common kernel function used is the RBF or Gaussian kernel.



Predictions

- How do we use our data to make predictions with our kernel function?
- Within the Gaussian Process paradigm we assume that our data along with the novel point at which we would like to predict form a joint Gaussian distribution.



(Machine Learning, Stanford University, https://www.coursera.org/learn/machine-learning)

Predictions

- Using the assumption that our data can be modelled as a Gaussian process, we can write out the new distribution of the observed noisy values along the points at which we wish to test the underlying function as

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}_\star \end{bmatrix} \sim \mathbb{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{x}\boldsymbol{x}} + \sigma_n^2 \mathbb{I}_{n\times n} & \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}}^\intercal \\ \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}} & \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}_\star} \end{bmatrix}\right).$$

(using the notation $\left(\boldsymbol{K}_{\boldsymbol{w}\boldsymbol{w}'}\right)_{i,j} \triangleq k\left(\boldsymbol{w}_i, \boldsymbol{w}'_j\right)$)

Predictions

- Using the assumption that our data can be modelled as a Gaussian process, we can write out the new distribution of the observed noisy values along the points at which we wish to test the underlying function as

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}_\star \end{bmatrix} \sim \mathbb{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}_{\boldsymbol{x}\boldsymbol{x}} + \sigma_n^2 \mathbb{I}_{n\times n} & \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}}^\intercal \\ \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}} & \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}_\star} \end{bmatrix}\right).$$

  (using the notation $\left(\boldsymbol{K}_{\boldsymbol{w}\boldsymbol{w}'}\right)_{i,j} \triangleq k\left(\boldsymbol{w}_i, \boldsymbol{w}'_j\right)$)
- The mean and covariance can then be computed as

$$\overline{\boldsymbol{y}_\star} = \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}}\left[\boldsymbol{K}_{\boldsymbol{x}\boldsymbol{x}} + \sigma_n^2 \mathbb{I}_{n\times n}\right]^{-1} \boldsymbol{y}$$

$$\operatorname{cov}(\boldsymbol{y}_\star) = \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}_\star} - \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}}\left[\boldsymbol{K}_{\boldsymbol{x}\boldsymbol{x}} + \sigma_n^2 \mathbb{I}_{n\times n}\right]^{-1} \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}}^\intercal.$$

Unoptimized GPR

Algorithm 1: Unoptimized GPR

input : Observations $\boldsymbol{X}, \boldsymbol{y}$ and a test input $\boldsymbol{x}_\star$.
output: A prediction $\overline{f_\star}$ with its corresponding variance $\mathbb{V}[f_\star]$.

1 $\boldsymbol{L} = \text{cholesky}\left(\boldsymbol{K}_{\boldsymbol{XX}} + \sigma_n^2 \mathbb{I}_{n \times n}\right)$
2 $\boldsymbol{\alpha} = \text{lin-solve}\left(\boldsymbol{L}^\mathsf{T}, \text{lin-solve}\left(\boldsymbol{L}, \boldsymbol{y}\right)\right)$
3 $\overline{y_\star} = \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{X}} \boldsymbol{\alpha}$
4 $\boldsymbol{v} = \text{lin-solve}\left(\boldsymbol{L}, \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{X}}\right)$
5 $\mathbb{V}[f_\star] = \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}_\star} - \boldsymbol{v}^\mathsf{T} \boldsymbol{v}$
6 **return** $\overline{f_\star}, \mathbb{V}[f_\star]$

Problems with Unoptimized GPR

Algorithm 2: Unoptimized GPR

input : Observations $\boldsymbol{X}, \boldsymbol{y}$ and a prediction inputs $\boldsymbol{x}_\star$.
output: A prediction $\overline{f_\star}$ with its corresponding variance $\mathbb{V}[f_\star]$.

1 $\boldsymbol{L} = \text{cholesky}\left(\boldsymbol{K}_{\boldsymbol{XX}} + \sigma_n^2 \mathbb{I}_{n \times n}\right)$
2 $\boldsymbol{\alpha} = \text{lin-solve}\left(\boldsymbol{L}^\intercal, \text{lin-solve}\left(\boldsymbol{L}, \boldsymbol{y}\right)\right)$
3 $\overline{f_\star} = \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{X}} \boldsymbol{\alpha}$
4 $\boldsymbol{v} = \text{lin-solve}\left(\boldsymbol{L}, \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{X}}\right)$
5 $\mathbb{V}[f_\star] = \boldsymbol{K}_{\boldsymbol{x}_\star \boldsymbol{x}_\star} - \boldsymbol{v}^\intercal \boldsymbol{v}$
6 **return** $\overline{f_\star}, \mathbb{V}[f_\star]$

- Lines 1,2 and 4 can be incredibly slow as computing $\boldsymbol{K}_{\boldsymbol{XX}}$ doing a Cholesky decomposition and performing linear solves scale poorly as the number of inputs, n, grows.

Nystrom Approximation

- The Nystrom method we seek a matrix $\boldsymbol{Q} \in \mathbb{R}^{n \times k}$ that satisfies $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A}\|_F \leq \varepsilon$, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive semi definite matrix, to form the rank$-k$ approximation

Nystrom Approximation

- The Nystrom method we seek a matrix $\boldsymbol{Q} \in \mathbb{R}^{n \times k}$ that satisfies $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A}\|_F \leq \varepsilon$, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive semi definite matrix, to form the rank$-k$ approximation

$$\boldsymbol{A} \simeq \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A}$$

Nystrom Approximation

- The Nystrom method we seek a matrix $\boldsymbol{Q} \in \mathbb{R}^{n \times k}$ that satisfies $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A}\|_F \leq \varepsilon$, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive semi definite matrix, to form the rank$-k$ approximation

$$\begin{aligned} \boldsymbol{A} &\simeq \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A} \\ &\simeq \boldsymbol{Q} \left(\boldsymbol{Q}^* \boldsymbol{A} \boldsymbol{Q}\right) \boldsymbol{Q}^* \end{aligned}$$

Nystrom Approximation

- The Nystrom method we seek a matrix $\boldsymbol{Q} \in \mathbb{R}^{n \times k}$ that satisfies $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A}\|_F \leq \varepsilon$, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive semi definite matrix, to form the rank$-k$ approximation

$$\begin{aligned} \boldsymbol{A} &\simeq \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A} \\ &\simeq \boldsymbol{Q} \left(\boldsymbol{Q}^* \boldsymbol{A} \boldsymbol{Q}\right) \boldsymbol{Q}^* \\ &= \boldsymbol{Q} \left(\boldsymbol{Q}^* \boldsymbol{A} \boldsymbol{Q}\right) \left(\boldsymbol{Q}^* \boldsymbol{A} \boldsymbol{Q}\right)^\dagger \left(\boldsymbol{Q}^* \boldsymbol{A} \boldsymbol{Q}\right) \boldsymbol{Q}^* \end{aligned}$$

Nystrom Approximation

- The Nystrom method we seek a matrix $\boldsymbol{Q} \in \mathbb{R}^{n \times k}$ that satisfies $\|\boldsymbol{A} - \boldsymbol{Q}\boldsymbol{Q}^* \boldsymbol{A}\|_F \leq \varepsilon$, where $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is positive semi definite matrix, to form the rank$-k$ approximation

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

Random Fourier Feature Approximation

- The RFF technique hinges on Bochners theorem which characterises positive definite functions (namely kernels) and states that any positive definite functions can be represented as

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) \mu_k(d\boldsymbol{\omega})$$

where μ_k is a positive finite measure on the frequencies of $\boldsymbol{\omega}$.

Random Fourier Feature Approximation

- The RFF technique hinges on Bochners theorem which characterises positive definite functions (namely kernels) and states that any positive definite functions can be represented as

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) \mu_k(d\boldsymbol{\omega})$$

  where μ_k is a positive finite measure on the frequencies of $\boldsymbol{\omega}$.
- This integral can then be approximated via the following Monte Carlo estimate

$$k(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) p(\boldsymbol{\omega})\, d\boldsymbol{\omega}$$

Random Fourier Feature Approximation

- The RFF technique hinges on Bochners theorem which characterises positive definite functions (namely kernels) and states that any positive definite functions can be represented as

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) \mu_k(d\boldsymbol{\omega})$$

  where μ_k is a positive finite measure on the frequencies of $\boldsymbol{\omega}$.
- This integral can then be approximated via the following Monte Carlo estimate

$$\begin{aligned} k(\boldsymbol{x} - \boldsymbol{y}) &= \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) p(\boldsymbol{\omega})\, d\boldsymbol{\omega} \\ &= \mathbb{E}_{\boldsymbol{\omega} \sim p(\cdot)} \left(\exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right)\right) \end{aligned}$$

Random Fourier Feature Approximation

- The RFF technique hinges on Bochners theorem which characterises positive definite functions (namely kernels) and states that any positive definite functions can be represented as

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) \mu_k(d\boldsymbol{\omega})$$

  where μ_k is a positive finite measure on the frequencies of $\boldsymbol{\omega}$.
- This integral can then be approximated via the following Monte Carlo estimate

$$\begin{aligned} k(\boldsymbol{x} - \boldsymbol{y}) &= \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) p(\boldsymbol{\omega})\, d\boldsymbol{\omega} \\ &= \mathbb{E}_{\boldsymbol{\omega} \sim p(\cdot)} \left(\exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right)\right) \\ &\simeq \frac{1}{D} \sum_{j=1}^{D} \exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{x} - \boldsymbol{y}\rangle\right) \end{aligned}$$

Random Fourier Feature Approximation

- The RFF technique hinges on Bochners theorem which characterises positive definite functions (namely kernels) and states that any positive definite functions can be represented as

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) \mu_k(d\boldsymbol{\omega})$$

  where μ_k is a positive finite measure on the frequencies of $\boldsymbol{\omega}$.
- This integral can then be approximated via the following Monte Carlo estimate

$$\begin{aligned} k(\boldsymbol{x} - \boldsymbol{y}) &= \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) p(\boldsymbol{\omega})\, d\boldsymbol{\omega} \\ &= \mathbb{E}_{\boldsymbol{\omega} \sim p(\cdot)} \left(\exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right)\right) \\ &\simeq \frac{1}{D} \sum_{j=1}^{D} \exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{x} - \boldsymbol{y}\rangle\right) \\ &= \sum_{j=1}^{D} \left(\frac{1}{\sqrt{D}} \exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{x}\rangle\right)\right) \overline{\left(\frac{1}{\sqrt{D}} \exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{y}\rangle\right)\right)} \end{aligned}$$

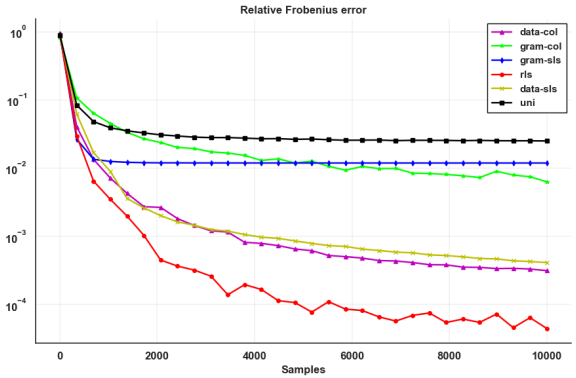
Random Fourier Feature Approximation

- The RFF technique hinges on Bochners theorem which characterises positive definite functions (namely kernels) and states that any positive definite functions can be represented as

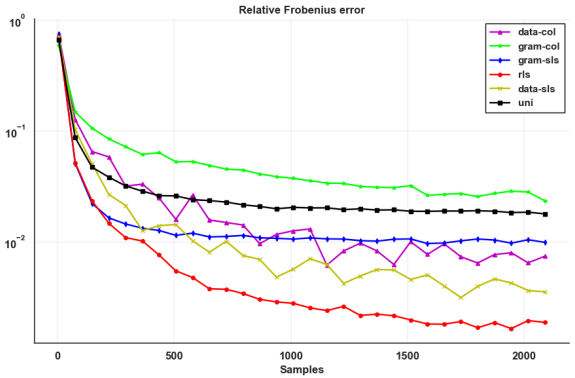
$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y}) = \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) \mu_k(d\boldsymbol{\omega})$$

  where μ_k is a positive finite measure on the frequencies of $\boldsymbol{\omega}$.
- This integral can then be approximated via the following Monte Carlo estimate

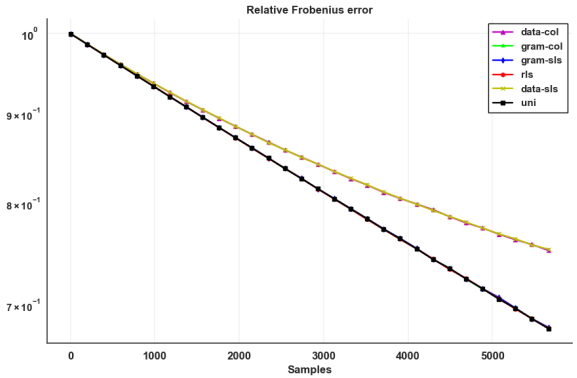
$$\begin{aligned} k(\boldsymbol{x} - \boldsymbol{y}) &= \int_{\mathbb{C}^d} \exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right) p(\boldsymbol{\omega})\, d\boldsymbol{\omega} \\ &= \mathbb{E}_{\boldsymbol{\omega} \sim p(\cdot)} \left(\exp\left(i\langle \boldsymbol{\omega}, \boldsymbol{x} - \boldsymbol{y}\rangle\right)\right) \\ &\simeq \frac{1}{D} \sum_{j=1}^{D} \exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{x} - \boldsymbol{y}\rangle\right) \\ &= \sum_{j=1}^{D} \left(\frac{1}{\sqrt{D}} \exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{x}\rangle\right)\right) \overline{\left(\frac{1}{\sqrt{D}} \exp\left(i\langle \boldsymbol{\omega}_j, \boldsymbol{y}\rangle\right)\right)} \\ &= \langle \varphi(\boldsymbol{x}), \varphi(\boldsymbol{y})\rangle_{\mathbb{C}^D} \end{aligned}$$



(a) 3D-Spatial network

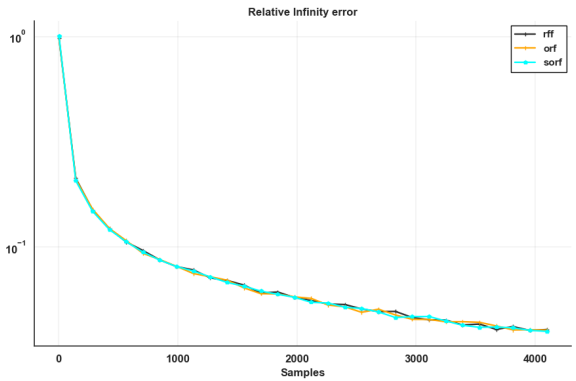


(b) Abalone

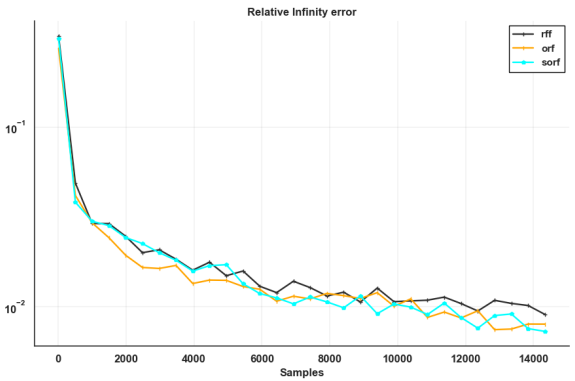


(c) Temperature

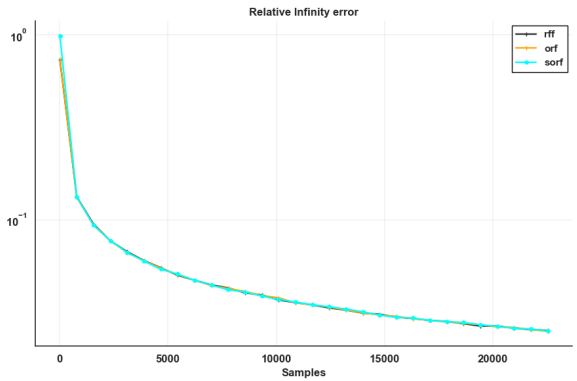
Figure: Comparison of Nystrom methods for various datasets.



(a) 3D-Spatial network

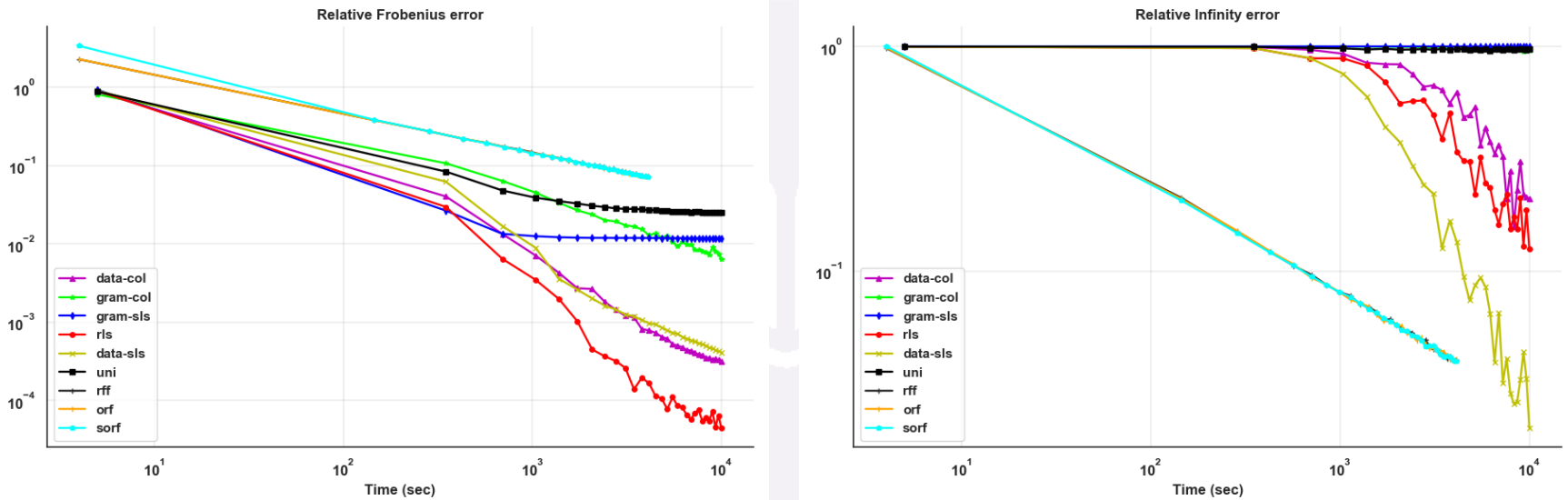


(b) Abalone



(c) Wine

Figure: Comparison of RFF methods for various datasets.



(a) Frobenius error

(b) Infinity error

Figure: Comparison between Nystrom and RFF approximations for the 3D-Spatial network data.

Moving Forward

- We saw that the Nystrom technique is better at producing approximations of the Gram matrix, $\boldsymbol{K_{XX}}$, with smaller *relative Frobenius errors* while the RFF technique is better at producing approximations with smaller *relative infinity errors*. Which is better for GP prediction?
- Recall, the other bottle neck in the GPR algorithm was the Cholesky decomposition. We can employ faster linear system solvers, namely the *Conjugate Gradient* (CG) and *Minimum Residual* (MINRES) method.