Appendix

A Bayesian Information Criterion (BIC)

The BIC is a model selection criterion that is the marginal likelihood with a model complexity penalty:

$$BIC = \log p(y|\hat{\theta}) - \frac{1}{2} p \log(N)$$

for observations y, number of observations N, maximum likelihood estimate (MLE) of model hyperparameters $\hat{\theta}$, number of hyperparameters p. It is derived as an approximation to the log model evidence $\log p(y)$.

B Compositional Kernel Search Algorithm

Algorithm 2: Compositional Kernel Search Algorithm

```
Input: data $x_1, \dots, x_n \in \mathbb{R}^D, y_1, \dots, y_n \in \mathbb{R}$,base
       kernel set $\mathcal{B}$
Output: $k$, the resulting kernel
For each base kernel on each dimension, fit GP to
  data (i.e. optimise hyperparams by ML-II) and set $k$
  to be kernel with highest BIC.
for depth=1:T (either fix T or repeat until BIC no
  longer increases) do
    Fit GP to following kernels and set $k$ to be the one
      with highest BIC:
    (1) All kernels of form $k + B$ where $B$ is any base
      kernel on any dimension
    (2) All kernels of form $k \times B$ where $B$ is any base
      kernel on any dimension
    (3) All kernels where a base kernel in $k$ is replaced
      by another base kernel
```

C Base Kernels

$$\text{LIN}(x, x') = \sigma^2 (x - l)(x' - l)$$

$$\text{SE}(x, x') = \sigma^2 \exp\Big(- \frac{(x - x')^2}{2l^2} \Big)$$

$$\text{PER}(x, x') = \sigma^2 \exp\Big(- \frac{2 \sin^2(\pi(x - x')/p)}{l^2} \Big)$$

D Matrix Identities

Lemma 1 (Woodbury's Matrix Inversion Lemma)**.** $(A + UBV)^{-1} = A^{-1} - A^{-1}U(B^{-1} + VA^{-1}U)^{-1}VA^{-1}$

So setting $A = \sigma^2 I$ (Nyström) or $\sigma^2 I + diag(K - \hat{K})$ (FIC) or $\sigma^2 I + blockdiag(K - \hat{K})$ (PIC), $U = \Phi^T = V$, $B = I$, we get:

$$(A + \Phi^\top \Phi)^{-1} = A^{-1} - A^{-1}\Phi^\top (I + \Phi A^{-1} \Phi^\top)^{-1} \Phi A^{-1}$$

Lemma 2 (Sylvester's Determinant Theorem)**.** $\det(I + AB) = \det(I + BA) \;\; \forall A \in \mathbb{R}^{m \times n} \;\; \forall B \in \mathbb{R}^{n \times m}$

Hence:

$$\begin{aligned} \det(\sigma^2 I + \Phi^\top \Phi) &= (\sigma^2)^n \det(I + \sigma^{-2} \Phi^\top \Phi) \\ &= (\sigma^2)^n \det(I + \sigma^{-2} \Phi \Phi^\top) \\ &= (\sigma^2)^{n-m} \det(\sigma^2 I + \Phi \Phi^\top) \end{aligned}$$

E Proof of Proposition 1

Proof. If PCG converges, the upper bound for NIP is exact. We showed in Section 4.1 that the convergence happened in only a few iterations. Moreover Cortes et al [7] shows that the lower bound for NIP can be rather loose in general.

So it suffices to prove that the upper bound for NLD is tighter than the lower bound for NLD. Let $(\lambda_i)_{i=1}^N, (\hat{\lambda}_i)_{i=1}^N$ be the ordered eigenvalues of $K + \sigma^2 I, \hat{K} + \sigma^2 I$ respectively. Since $K - \hat{K}$ is positive semi-definite (e.g. [3]), we have $\lambda_i \geq \hat{\lambda}_i \geq 2\sigma^2 \; \forall i$ (using the assumption in the proposition). Now the slack in the upper bound is:

$$\begin{aligned} &-\frac{1}{2} \log\det(\hat{K} + \sigma^2 I) - (-\frac{1}{2} \log\det(K + \sigma^2 I)) \\ &\quad = \frac{1}{2} \sum_{i=1}^N (\log \lambda_i - \log \hat{\lambda}_i) \end{aligned}$$

Hence the slack in the lower bound is:

$$\begin{aligned} &-\frac{1}{2} \log\det(K + \sigma^2 I) \\ &\quad - \Big[-\frac{1}{2} \log\det(\hat{K} + \sigma^2 I) - \frac{1}{2\sigma^2} \operatorname{Tr}(K - \hat{K}) \Big] \\ &\quad = -\frac{1}{2} \sum_{i=1}^N (\log \lambda_i - \log \hat{\lambda}_i) + \frac{1}{2\sigma^2} \sum_{i=1}^N (\lambda_i - \hat{\lambda}_i) \end{aligned}$$

Now by concavity and monotonicity of log, and since $\hat{\lambda} \geq 2\sigma^2$, we have:

$$\frac{\log \lambda_i - \log \hat{\lambda}_i}{\lambda_i - \hat{\lambda}_i} \leq \frac{1}{2\sigma^2}$$

$$\Rightarrow \sum_{i=1}^N (\log \lambda_i - \log \hat{\lambda}_i) \leq \frac{1}{2\sigma^2} \sum_{i=1}^N (\lambda_i - \hat{\lambda}_i)$$

$$\begin{aligned} &\Rightarrow \frac{1}{2} \sum_{i=1}^N (\log \lambda_i - \log \hat{\lambda}_i) \\ &\quad \leq \frac{1}{2\sigma^2} \sum_{i=1}^N (\lambda_i - \hat{\lambda}_i) - \frac{1}{2} \sum_{i=1}^N (\log \lambda_i - \log \hat{\lambda}_i) \end{aligned}$$

□

F Convergence of hyperparameters from optimising lower bound to optimal hyperparameters

Note from Figure 7 that the hyperparameters found by optimising the lower bound converges to the hyperparameters found by the exact GP when optimising the exact marginal likelihood, giving empirical evidence for the second claim in Section 3.3.

G Parallelising SKC

Note that SKC can be parallelised across the random hyperparameter initialisations, and also across the kernels at each depth for computing the BIC intervals. In fact, SKC is even more parallelisable with the kernel buffer: say at a certain depth, we have two kernels remaining to be optimised and evaluated before we can move onto the next depth. If the buffer size is 5, say, then we can in fact move on to the next depth and grow the kernel search tree on the top 3 kernels of the buffer, without having to wait for the 2 kernel evaluations to be complete. This saves a lot of computation time wasted by idle cores waiting for all kernel evaluations to finish before moving on to the next depth of the kernel search tree.

H Optimisation

Since we wish to use the learned kernels for interpretation, it is important to have the hyperparameters lie in a sensible region after the optimisation. In other words, we wish to regularise the hyperparameters during optimisation. For example, we want the SE kernel to learn a globally smooth function with local variation. When naïvely optimising the lower bound, sometimes the length scale and the signal variance becomes very small, so the SE kernel explains all the variation in the signal and ends up connecting the dots. We wish to avoid this type of behaviour. This can be achieved by giving priors to hyperparameters and optimising the energy (log prior added to the log marginal likelihood) instead, as well as using sensible initialisations. Looking at Figure 8, we see that using a strong prior with a sensible random initialisation (see Appendix I for details) gives a sensible smoothly varying function, whereas for all the three other cases, we have the length scale and signal variance shrinking to small values, causing the GP to overfit to the data. Note that the weak prior is the default prior used in the GPstuff software [39].

Careful initialisation of hyperparameters and inducing points is also very important, and can have strong influence the resulting optima. It is sensible to have the optimised hyperparameters of the parent kernel in the search tree be inherited and used to initialise the hyperparameters of the child. The new hyperparameters of the child must be initialised with random restarts, where the variance is small enough to ensure that they lie in a sensible region, but large enough to explore a good portion of this region. As for the inducing points, we want to spread them out to capture both local and global structure. Trying both K-means and a random subset of training data, we conclude that they give similar results and resort to a random subset. Moreover we also have the option of learning the inducing points. However, this will be considerably more costly and show little improvement over fixing them, as we show in Section 4. Hence we do not learn the inducing points, but fix them to a given randomly chosen set.

Hence for SKC, we use *maximum a posteriori* (MAP) estimates instead of MLE for the hyperparameters to calculate the BIC, since the priors have a noticeable effect on the optimisation. This is justified [23] and has been used for example in [11], where they argue that using the MLE to estimate the BIC for Gaussian mixture models can fail due to singularities and degeneracies.

I Hyperparameter initialisation and priors

$Z \sim \mathcal{N}(0,1), TN(\sigma^2, I)$ is a Gaussian with mean 0 and variance σ^2 truncated at the interval I then renormalised.

Signal noise
$\sigma^2 = 0.1 \times \exp(Z/2)$
$p(\log \sigma^2) = \mathcal{N}(0, 0.2)$

LIN
$\sigma^2 = \exp(V)$ where $V \sim TN(1, [-\infty, 0]), l = \exp(\frac{Z}{2})$
$p(\log \sigma^2) = logunif$
$p(\log l) = logunif$

SE
$l = \exp(Z/2), \sigma^2 = 0.1 \times \exp(Z/2)$
$p(\log l) = \mathcal{N}(0, 0.01), p(\log \sigma^2) = logunif$

PER
$p_{min} = \log(10 \times \frac{\max(x) - \min(x)}{N})$ (shortest possible period is 10 time steps)
$p_{max} = \log(\frac{\max(x) - \min(x)}{5})$ (longest possible period is a fifth of the range of data set)
$l = \exp(Z/2), p = \exp(p_{min} + W)$ or $\exp(p_{max} + U)$, $\sigma^2 = 0.1 \times \exp(Z/2)$ w.p. $\frac{1}{2}$
where $W \sim \mathcal{TN}(-0.5, [0, \infty])$,

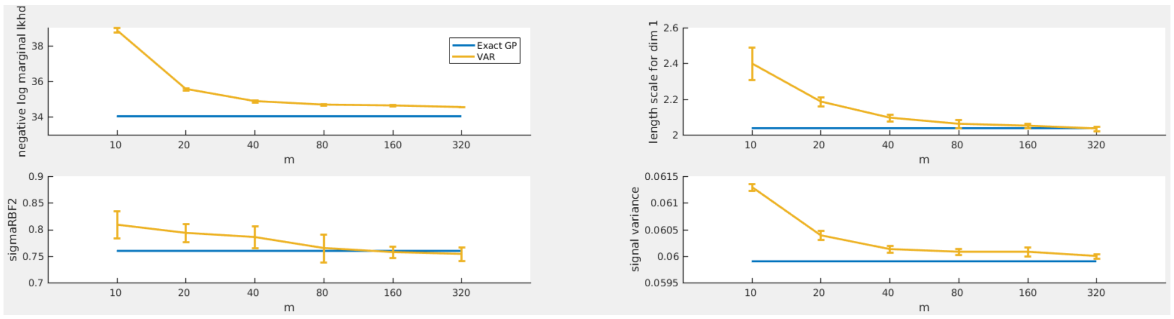


Figure 7: Log marginal likelihood and hyperparameter values after optimising the lower bound with ARD kernel on a subset of the Power plant data for different values of m. This is compared against the exact GP values when optimising the true log marginal likelihood. Error bars show mean $\pm$ 1 standard deviation over 10 random iterations.

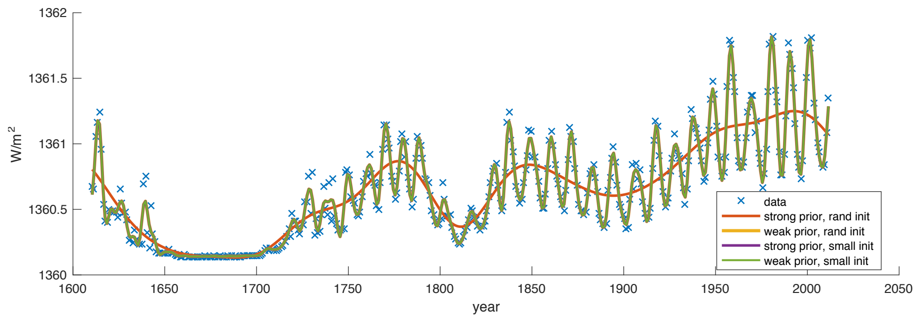


Figure 8: GP predictions on solar data set with SE kernel for different priors and initialisations.

$U \sim \mathcal{TN}(-0.5, [-\infty, 0]))$
$p(\log l) = t(\mu = 0, \sigma^2 = 1, \nu = 4)$,
$p(\log p) = \mathcal{LN}(p_{min} - 0.5, 0.25)$ or $\mathcal{LN}(p_{max} - 2, 0.5)$ w.p. $\frac{1}{2}$
$p(\log \sigma^2) = logunif$ where $\mathcal{LN}(\mu, \sigma^2)$ is log Gaussian, $t(\mu, \sigma^2, \nu)$ is the student's t-distribution.

J Computation times

Look at Table 2. For all three data sets, the GP optimisation time is much greater than the sum of the Var GP optimisation time and the upper bound (NLD + NIP) evaluation time for $m \leq 80$. Hence the savings in computation time for SKC is significant even for these small data sets.

Note that we show the lower bound optimisation time against the upper bound evaluation time instead of the evaluation times for both, since this is what happens in SKC - the lower bound has to be optimised for each kernel, whereas the upper bound only has to be evaluated once.

K Mauna and Solar plots and hyperparameter values found by SKC

Solar The solar data has 26 cycles over 285 years, which gives a periodicity of around 10.9615 years. Using SKC with $m = 40$, we find the kernel: SE $\times$ PER $\times$ SE $\equiv$ SE $\times$ PER. The value of the period hyperparameter in PER is 10.9569 years, hence SKC finds the periodicity to 3 s.f. with only 40 inducing points. The SE term converts the global periodicity to local periodicity, with the extent of the locality governed by the length scale parameter in SE, equal to 45. This is fairly large, but smaller than the range of the domain (1610-2011), indicating that the periodicity spans over a long time but isn't quite global. This is most likely due to the static solar irradiance between the years 1650-1700, adding a bit of noise to the periodicities.

Mauna The annual periodicity in the data and the linear trend with positive slope is clear. Linear regression gives us a slope of 1.5149. SKC with $m = 40$ gives the kernel: SE + PER + LIN. The period hyperparameter in PER takes value 1, hence SKC successfully finds the right periodicity. The offset l and magnitude σ^2 parameters of LIN allow us to calculate the slope

Table 2: Mean and standard deviation of computation times (in seconds) for full GP optimisation, Var GP optimisation(with and without learning inducing points), NLD and NIP (PCG using PIC preconditioner) upper bounds over 10 random iterations.

		Solar	Mauna	Concrete
GP		29.1950 ± 5.1430	164.8828 ± 58.7865	403.8233 ± 127.0364
Var GP	m=10	7.0259 ± 4.3928	6.0117 ± 3.8267	5.4358 ± 0.7298
	m=20	8.3121 ± 5.4763	11.9245 ± 6.8790	10.2410 ± 2.5109
	m=40	10.2263 ± 4.1025	17.1479 ± 10.7898	19.6678 ± 4.3924
	m=80	9.6752 ± 6.5343	28.9876 ± 13.0031	47.2225 ± 13.1955
	m=160	25.6330 ± 8.7934	91.0406 ± 39.8409	158.9199 ± 18.1276
	m=320	76.3447 ± 20.3337	202.2369 ± 96.0749	541.4835 ± 99.6145
NLD	m=10	0.0019 ± 0.0001	0.0033 ± 0.0002	0.0113 ± 0.0004
	m=20	0.0026 ± 0.0001	0.0046 ± 0.0002	0.0166 ± 0.0007
	m=40	0.0043 ± 0.0001	0.0079 ± 0.0003	0.0286 ± 0.0005
	m=80	0.0084 ± 0.0002	0.0154 ± 0.0004	0.0554 ± 0.0012
	m=160	0.0188 ± 0.0006	0.0338 ± 0.0007	0.1188 ± 0.0030
	m=320	0.0464 ± 0.0032	0.0789 ± 0.0036	0.2550 ± 0.0074
NIP	m=10	0.0474 ± 0.0092	0.1020 ± 0.0296	0.2342 ± 0.0206
	m=20	0.0422 ± 0.0130	0.1274 ± 0.0674	0.1746 ± 0.0450
	m=40	0.0284 ± 0.0075	0.0846 ± 0.0430	0.2345 ± 0.0483
	m=80	0.0199 ± 0.0081	0.0553 ± 0.0250	0.2176 ± 0.0376
	m=160	0.0206 ± 0.0053	0.0432 ± 0.0109	0.2136 ± 0.0422
	m=320	0.0250 ± 0.0019	0.0676 ± 0.0668	0.2295 ± 0.0433
Var GP,	m=10	23.4 ± 14.6	42.0 ± 33.0	110.0 ± 302.5
learn IP	m=20	38.5 ± 17.5	62.0 ± 66.0	70.0 ± 97.0
	m=40	124.7 ± 99.0	320.0 ± 236.0	307.0 ± 341.4
	m=80	268.6 ± 196.6	1935.0 ± 1103.0	666.0 ± 41.0
	m=160	1483.6 ± 773.8	10480.0 ± 5991.0	4786.0 ± 406.9
	m=320	2923.8 ± 1573.5	39789.0 ± 23870.0	25906.0 ± 820.9

by the formula $\sigma^2(x-l)^\top[\sigma^2(x-l)(x-l)^\top + \sigma_n^2 I]^{-1}y$ where σ_n^2 is the noise variance in the learned likelihood. This formula is obtained from the posterior mean of the GP, which is linear in the inputs for the linear kernel. This value amounts to 1.5150, hence the slope found by SKC is accurate to 3 s.f.

L Optimising the upper bound

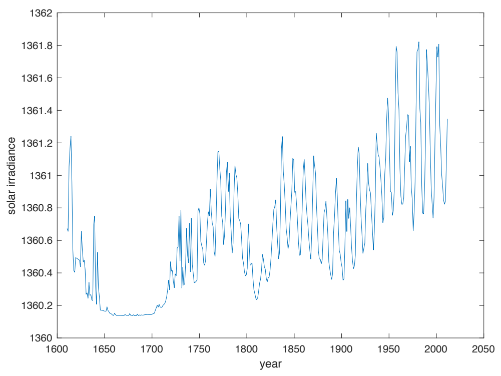
If the upper bound is tighter and more robust with respect to choice of inducing points, why don't we optimise the upper bound to find hyperparameters? If this were to be possible then we can maximise this to get an upper bound of the exact marginal likelihood with optimal hyperparameters. In fact this holds for any analytic upper bound whose value and gradients can be evaluated cheaply. Hence for any m, we can find an interval that contains the true optimised marginal likelihood. So if this interval is dominated by an interval of another kernel, we can discard the kernel and there is no need to evaluate the bounds for bigger values of m. Now we wish to use values of m such that we can choose the right kernel (or kernels) at each depth of the search tree with minimal computation. This gives rise to an exploitation-exploration trade-off, whereby we want to balance between raising m for tight intervals that allow us to discard unsuitable kernels whose intervals fall strictly below that of other kernels, and quickly moving on to the next depth in the search tree to search for finer structure in the data. The search algorithm is highly parallelisable, and thus we may raise m simultaneously for all candidate kernels. At deeper levels of the search tree, there may be too many candidates for simultaneous computation, in which case we may select the ones with the highest upper bound to get tighter intervals. Such attempts are listed below.

Note the two inequalities for the NLD and NIP terms:

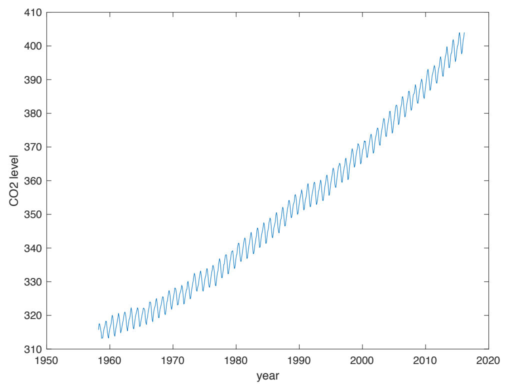
$$-\frac{1}{2}\log\det(\hat{K}+\sigma^2 I) - \frac{1}{2\sigma^2}\operatorname{Tr}(K-\hat{K}) \leq -\frac{1}{2}\log\det(K+\sigma^2 I) \leq -\frac{1}{2}\log\det(\hat{K}+\sigma^2 I) \tag{5}$$

$$-\frac{1}{2}y^\top(\hat{K}+\sigma^2 I)^{-1}y \leq -\frac{1}{2}y^\top(K+\sigma^2 I)^{-1}y \leq -\frac{1}{2}y^\top(K+(\sigma^2+\operatorname{Tr}(K-\hat{K}))I)^{-1}y \tag{6}$$

Where the first two inequalities come from [3], the



(a) Solar



(b) Mauna

Figure 9: Plots of small time series data: Solar and Mauna

third inequality is a direct consequence of $K - \hat{K}$ being postive semi-definite, and the last inequality is from Michalis Titsias' lecture slides [3].

Also from 4, we have that

$$-\frac{1}{2}\log\det(\hat{K}+\sigma^2 I)+\frac{1}{2}\alpha^\top(K+\sigma^2 I)\alpha-\alpha^\top y$$

is an upper bound $\forall\alpha\in\mathbb{R}^N$. Thus one idea of obtaining a cheap upper bound to the optimised marginal likelihood was to solve the following maximin optimisation problem:

$$\max_\theta \min_{\alpha\in\mathbb{R}^N} -\frac{1}{2}\log\det(\hat{K}+\sigma^2 I)+\frac{1}{2}\alpha^\top(K+\sigma^2 I)\alpha-\alpha^\top y$$

One way to solve this cheaply would be by coordinate descent, where one maximises with respect to θ fixing α, then minimises with respect to α fixing θ. However σ tends to blow up in practice. This is because the expression is $O(-\log\sigma^2+\sigma^2)$ for fixed α, hence maximising with respect to σ pushes it towards infinity.

An alternative is to sum the two upper bounds above to get the upper bound

$$-\frac{1}{2}\log\det(\hat{K}+\sigma^2 I)-\frac{1}{2}y^\top(K+(\sigma^2+\mathrm{Tr}(K-\hat{K}))I)^{-1}y$$

However we found that maximising this bound gives quite a loose upper bound unless $m = O(N)$. Hence this upper bound is not very useful.

[3]http://www.aueb.gr/users/mtitsias/papers/titsiasNipsVar14.pdf

M Random Fourier Features

Random Fourier Features (RFF) (a.k.a. Random Kitchen Sinks) was introduced by [26] as a low rank approximation to the kernel matrix. It uses the following theorem

Theorem 3 (Bochner's Theorem [29]). *A stationary kernel $k(d)$ is positive definite if and only if $k(d)$ is the Fourier transform of a non-negative measure.*

to give an unbiased low-rank approximation to the Gram matrix $K = \mathbb{E}[\Phi^\top\Phi]$ with $\Phi\in\mathbb{R}^{m\times N}$. A bigger m lowers the variance of the estimate. Using this approximation, one can compute determinants and inverses in $O(Nm^2)$ time. In the context of kernel composition in 2, RFFs have the nice property that samples from the spectral density of the sum or product of kernels can easily be obtained as sums or mixtures of samples of the individual kernels (see Appendix N). We use this later to give a memory-efficient upper bound on the exact log marginal likelihood in Appendix P.

N Random Features for Sums and Products of Kernels

For RFF the kernel can be approximated by the inner product of random features given by samples from its spectral density, in a Monte Carlo approximation, as follows:

$$\begin{aligned}
k(x-y) &= \int_{\mathbb{R}^D} e^{iv^\top(x-y)}d\mathbb{P}(v)\\
&\propto \int_{\mathbb{R}^D} p(v)e^{iv^\top(x-y)}dv\\
&= \mathbb{E}_{p(v)}[e^{iv^\top x}(e^{iv^\top y})^*]\\
&= \mathbb{E}_{p(v)}[Re(e^{iv^\top x}(e^{iv^\top y})^*)]\\
&\approx \frac{1}{m}\sum_{k=1}^m Re(e^{iv_k{}^\top x}(e^{iv_k{}^\top y})^*)\\
&= \mathbb{E}_{b,v}[\phi(x)^\top\phi(y)]
\end{aligned}$$

where $\phi(x) = \sqrt{\frac{2}{m}}(cos(v_1{}^\top x+b_1),\ldots,cos(v_m{}^\top x+b_m))$ with spectral frequencies v_k iid samples from $p(v)$ and b_k iid samples from $U[0,2\pi]$.

Let k_1, k_2 be two stationary kernels, with respective

spectral densities p_1, p_2 so that
$k_1(d) = a_1\hat{p_1}(d), k_2(d) = a_2\hat{p_2}(d)$, where $\hat{p}(d) := \int_{\mathbb{R}^D} p(v)e^{iv^\top d}dv$. We use this convention as the Fourier transform. Note $a_i = k_i(0)$.

$$\begin{aligned}(k_1 + k_2)(d) &= a_1 \int p_1(v)e^{iv^\top d}dv + a_2 \int p_2(v)e^{iv^\top d}dv \\ &= (a_1 + a_2)\hat{p_+}(d)\end{aligned}$$

where $p_+(v) = \frac{a_1}{a_1+a_2}p_1(v) + \frac{a_2}{a_1+a_2}p_2(v)$, a mixture of p_1 and p_2. So to generate RFF for $k_1 + k_2$, generate $v \sim p_+$ by generating $v \sim p_1$ w.p. $\frac{a_1}{a_1+a_2}$ and $v \sim p_2$ w.p. $\frac{a_2}{a_1+a_2}$
Now for the product, suppose

$$(k_1 \cdot k_2)(d) = a_1a_2\hat{p_1}(d)\hat{p_2}(d) = a_1a_2\hat{p_*}(d)$$

Then $p_*(d)$ is the inverse fourier transform of $\hat{p_1}\hat{p_2}$, which is the convolution $p_1 * p_2(d) := \int_{\mathbb{R}^D} p_1(z)p_2(d-z)dz$. So to generate RFF for $k_1 \cdot k_2$, generate $v \sim p_*$ by generating $v_1 \sim p_1, v_2 \sim p_2$ and setting $v = v_1 + v_2$. This is not applicable for non-stationary kernels, such as the linear kernel. We deal with this problem as follows:

Suppose ϕ_1, ϕ_2 are random features such that
$k_1(x, x') = \phi_1(x)^\top\phi_1(x'), \phi_2(x)^\top\phi_2(x'), \phi_i : \mathbb{R}^D \to \mathbb{R}^m$.
It is straightforward to verify that

$$\begin{aligned}(k_1 + k_2)(x, x') &= \phi_+(x)^\top\phi_+(x') \\ (k_1 \cdot k_2)(x, x') &= \phi_*(x)^\top\phi_*(x')\end{aligned}$$

where $\phi_+(\cdot) = (\phi_1(\cdot)^\top, \phi_2(\cdot)^\top)^\top$ and $\phi_*(\cdot) = \phi_1(\cdot) \otimes \phi_2(\cdot)$. However we do not want the number of features to grow as we add or multiply kernels, since it will grow exponentially. We want to keep it to be m features. So we subsample m entries from ϕ_+ (or ϕ_*) and scale by factor $\sqrt{2}$ ($\sqrt{m}$ for ϕ_*), which will still give us unbiased estimates of the kernel provided that each term of the inner product $\phi_+(x)^\top\phi_+(x')$ (or $\phi_*(x)^\top\phi_*(x')$) is an unbiased estimate of $(k_1+k_2)(x, x')$(or $(k_1 \cdot k_2)(x, x')$). This is how we generate random features for linear kernels combined with other stationary kernels, using the features $\phi(x) = \frac{\sigma}{\sqrt{m}}(x, \ldots, x)^\top$.

O Spectral Density for PER

From [35], we have that the spectral density of the PER kernel is:

$$\sum_{n=-\infty}^{\infty} \frac{I_n(l^{-2})}{\exp(l^{-2})}\delta\left(v - \frac{2\pi n}{p}\right)$$

where I is the modified Bessel function of the first kind.

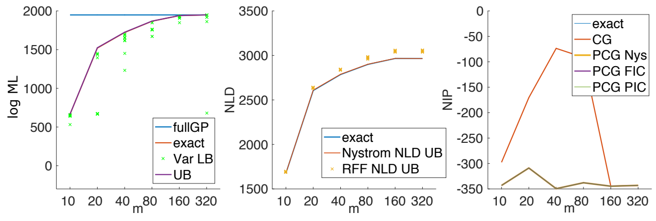
P An upper bound to NLD using Random Fourier Features

Note that the function $f(X) = -\log\det(X)$ is convex on the set of positive definite matrices [6]. Hence by Jensen's inequality we have, for $\Phi^\top\Phi$ an unbiased estimate of K:

$$\begin{aligned}-\frac{1}{2}\log\det(K + \sigma^2 I) &= f(K + \sigma^2 I) \\ &= f(\mathbb{E}[\Phi^\top\Phi + \sigma^2 I]) \\ &\leq \mathbb{E}[f(\Phi^\top\Phi + \sigma^2 I)]\end{aligned}$$

Hence $-\frac{1}{2}\log\det(\Phi^\top\Phi + \sigma^2 I)$ is a stochastic upper bound to NLD that can be calculated in $O(Nm^2)$. An example of such an unbiased estimator Φ is given by RFF.

Q Further Plots


(a) Mauna: fix inducing points

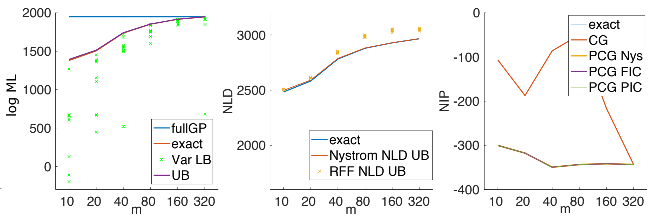
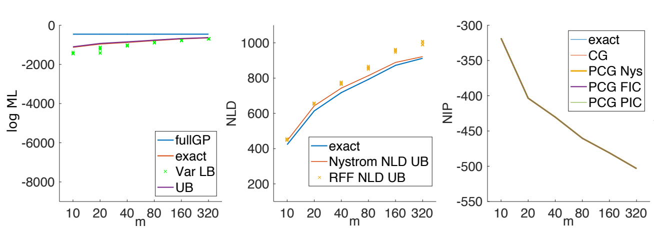

(b) Mauna: learn inducing points

Figure 10: Same as 1a and 1b but for Mauna Loa data.


(a) Concrete: fix inducing points

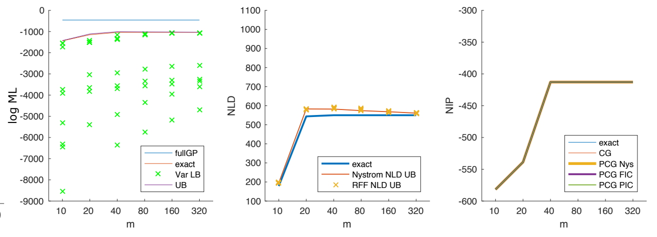

(b) Concrete: learn inducing points

Figure 11: Same as 1a and 1b but for Concrete data.

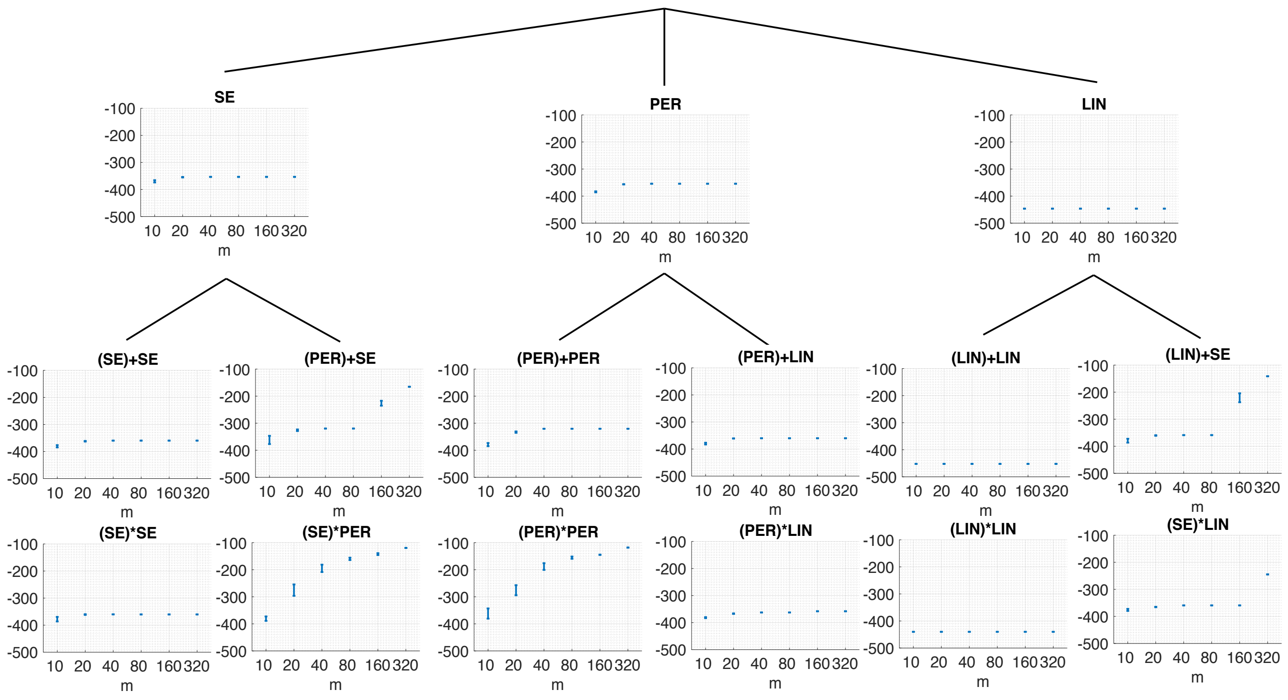


Figure 12: Kernel search tree for SKC on solar data up to depth 2. We show the upper and lower bounds for different numbers of inducing points m.

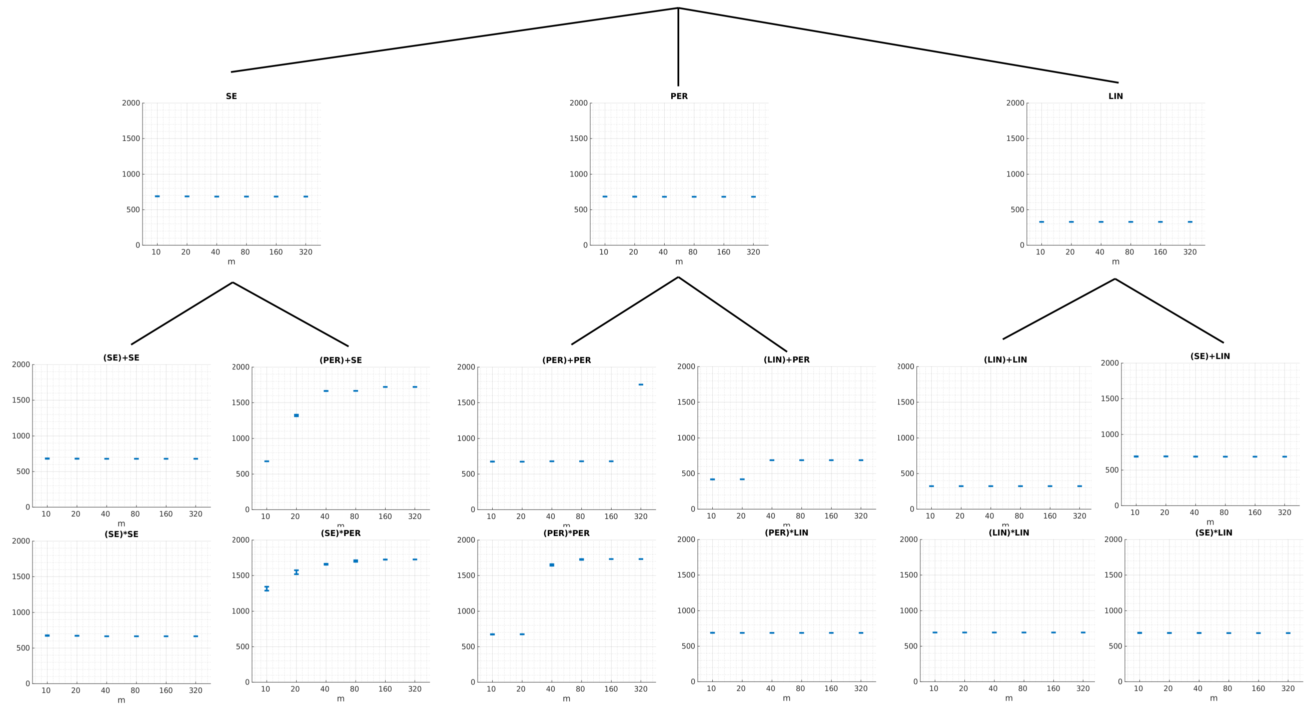


Figure 13: Same as Figure 12 but for Mauna data.

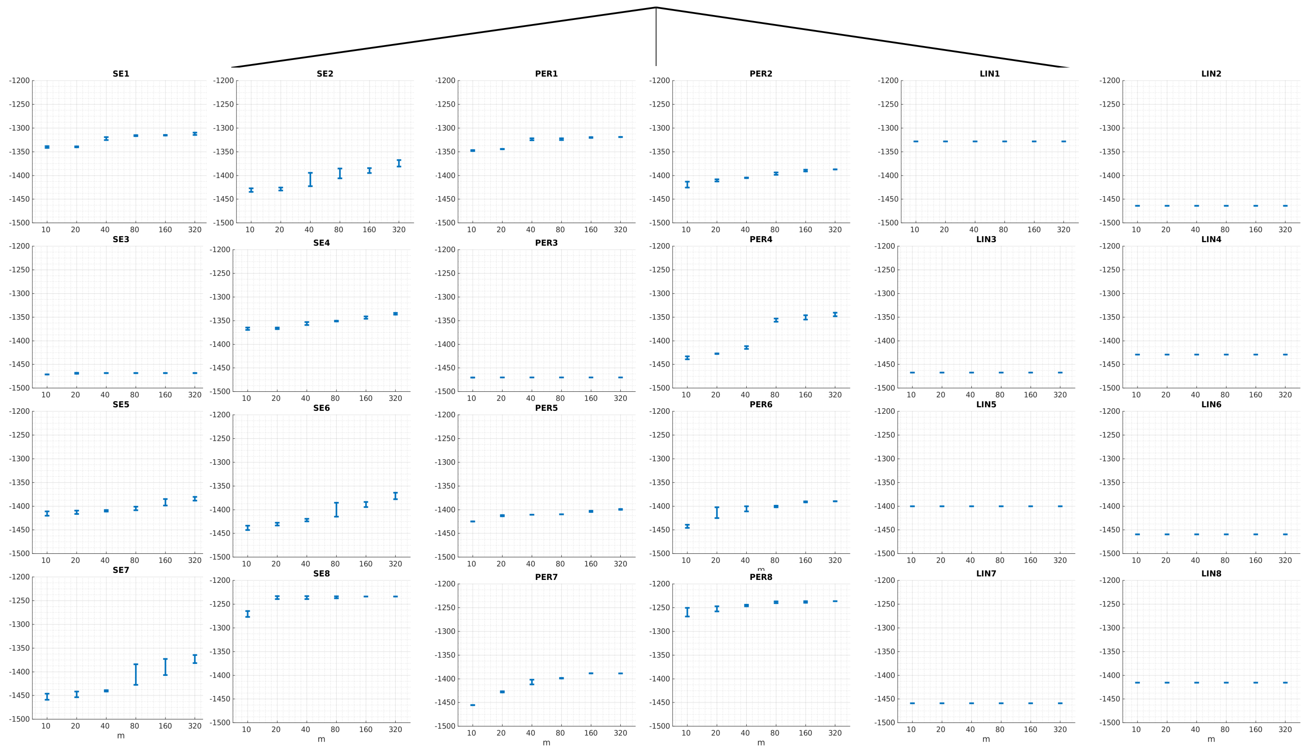


Figure 14: Same as Figure 12 but for concrete data and up to depth 1.