Appendix A. Compositional Kernel Search Algorithm

Algorithm 2: Compositional Kernel Search Algorithm

Input: data $x_1, \ldots, x_n \in \mathbb{R}^D, y_1, \ldots, y_n \in \mathbb{R}$,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 smallest BIC.
for *depth=1:T (either fix T or repeat until BIC no longer decreases)* **do**
  Fit GP to following kernels and set k to be the one with lowest 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
end

Appendix B. Base Kernels

$$\begin{aligned}
\mathrm{LIN}(x, x') &= \sigma^2 x x' \\
\mathrm{SE}(x, x') &= \sigma^2 \exp\Big(-\frac{(x - x')^2}{2l^2} \Big) \\
\mathrm{PER}(x, x') &= \sigma^2 \exp\Big(-\frac{2\sin^2(\pi(x - x')/p)}{l^2} \Big)
\end{aligned}$$

Appendix C. Random Fourier Features for Sums and Products of Kernels

Theorem 1 (Bochner's Theorem (Rudin, 1964)) *A stationary kernel k(d) is positive definite if and only if k(d) is the Fourier transform of a non-negative measure.*

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:

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

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 $a_i = k_i(0)$.

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

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

$$(k_1 \cdot k_2)(d) = a_1 a_2 \hat{p_1}(d)\hat{p_2}(d) = a_1 a_2 \hat{p_*}(d)$$

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 show how this is dealt with in Appendix D.

Appendix D. Random Features for Sums and Products of Kernels

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

$$(k_1 + k_2)(x, x') = \phi_+(x)^\top \phi_+(x') \text{ where } \phi_+(\cdot) = (\phi_1(\cdot)^\top, \phi_2(\cdot)^\top)^\top$$
$$(k_1 \cdot k_2)(x, x') = \phi_*(x)^\top \phi_*(x') \text{ where } \phi_*(\cdot) = \phi_1(\cdot) \otimes \phi_2(\cdot)$$

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 $\phi(x) = \frac{\sigma}{\sqrt{m}}(x, \ldots, x)^\top$.

Appendix E. Spectral Density for PER

From Solin and Särkkä (2014), 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.

Appendix F. Matrix Identities

Lemma 2 (Woodbury's Matrix Inversion Lemma)
$(A + UBV)^{-1} = A^{-1} - A^{-1}U(B^{-1} + VA^{-1}U)^{-1}VA^{-1}$

Lemma 3 (Sylvester's Determinant Theorem)
$\det(I + AB) = \det(I + BA) \forall A \in \mathbb{R}^{m \times n} \forall B \in \mathbb{R}^{n \times m}$

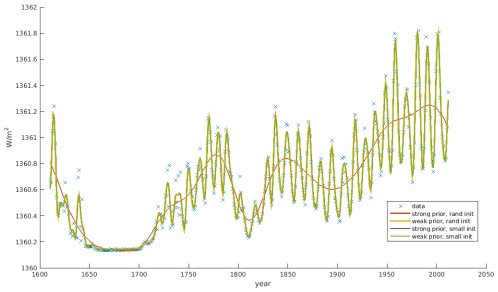


Figure 3: GP predictions on solar data set with SE kernel for different priors and initialisations.

Appendix G. 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 3, we see that using a strong prior with a sensible random initialisation (see Appendix H 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.

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 set.

Appendix H. 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 $V \sim TN(1, [-\infty, 0])$
$p(\log \sigma^2) = logunif$

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)
$l = \exp(Z/2), p = \exp(p_{min} + W), \sigma^2 = 0.1 \times \exp(Z/2)$ where $W \sim \mathcal{TN}(-0.5, [0, \infty))$
$p(\log l) = t(\mu = 0, \sigma^2 = 1, \nu = 4), p(\log p) = \mathcal{LN}(p_m in - 0.5, 1), p(\log \sigma^2) = logunif$ where $\mathcal{LN}(\mu, \sigma^2)$ is log Gaussian, $t(\mu, \sigma^2, \nu)$ is the student's t-distribution.

Appendix I. Computation Times for CKS

Table 1: Mean and standard deviation of computation times (in seconds) for full GP optimisation, Var GP optimisation, 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

Appendix J. Analytic upper bound to marginal likelihood

If we can find an analytic upper bound to the marginal likelihood, whose value and gradients can be evaluated in linear time, then we can maximise this to get an upper bound of exact likelihood with optimal hyperparameters. 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 to find an analytic upper bound are shown below.

First note that

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

is an upper bound to the marginal likelihood, up to a constant. Moreover from Equation (5), 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:

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

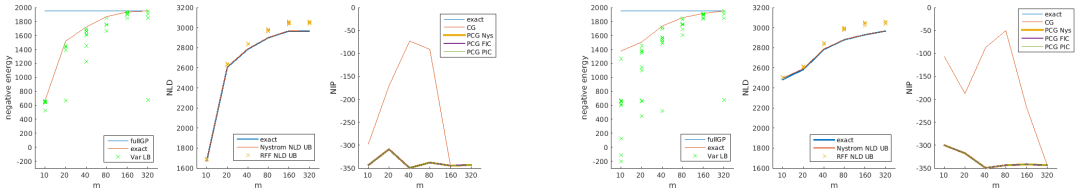
One way to solve this cheaply would be by coordinate descent, where one maximises with respect to θ fixing α, then minimises with respect to α fixing θ. However σ tends to blow up in practice. This is because the expression is $O(-\log\sigma^2+\sigma^2)$ for fixed α, hence maximising with respect to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

An alternative is to use the approximate upper bound

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

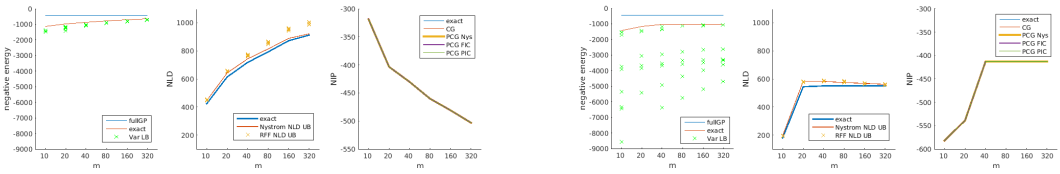
However we see that maximising this bound gives quite a loose upper bound unless $m=O(N)$. Hence this upper bound is not very useful.

Appendix K. Further Plots



(a) Mauna: fix inducing points (b) Mauna: learn inducing points

Figure 4: Same as Figures 1(a) and 1(b) but for Mauna Loa data.



(a) Concrete: fix inducing points (b) Concrete: learn inducing points

Figure 5: Same as Figures 1(a) and 1(b) but for Concrete data.

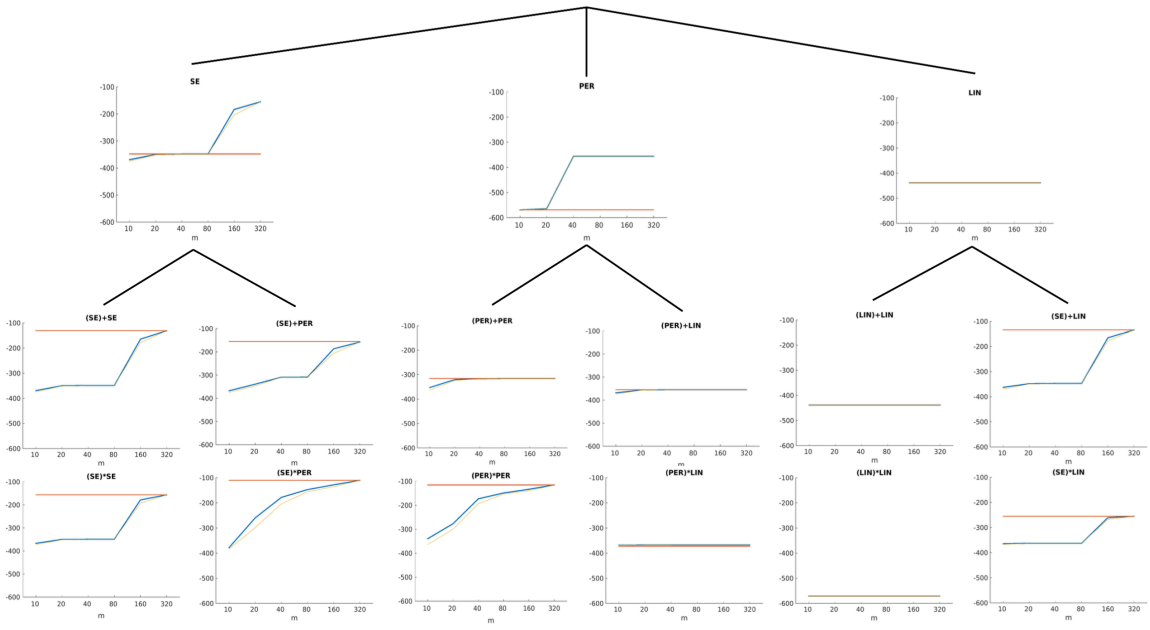


Figure 6: Kernel Tree search on solar data up to depth 2. The red, blue and yellow lines correspond to the best GP negative energy out of 10 random initialisations, the best lower bound out of 10 random initialisations, and the upper bound corresponding to the best lower bound, obtained by m iterations of PCG with PIC preconditioner.

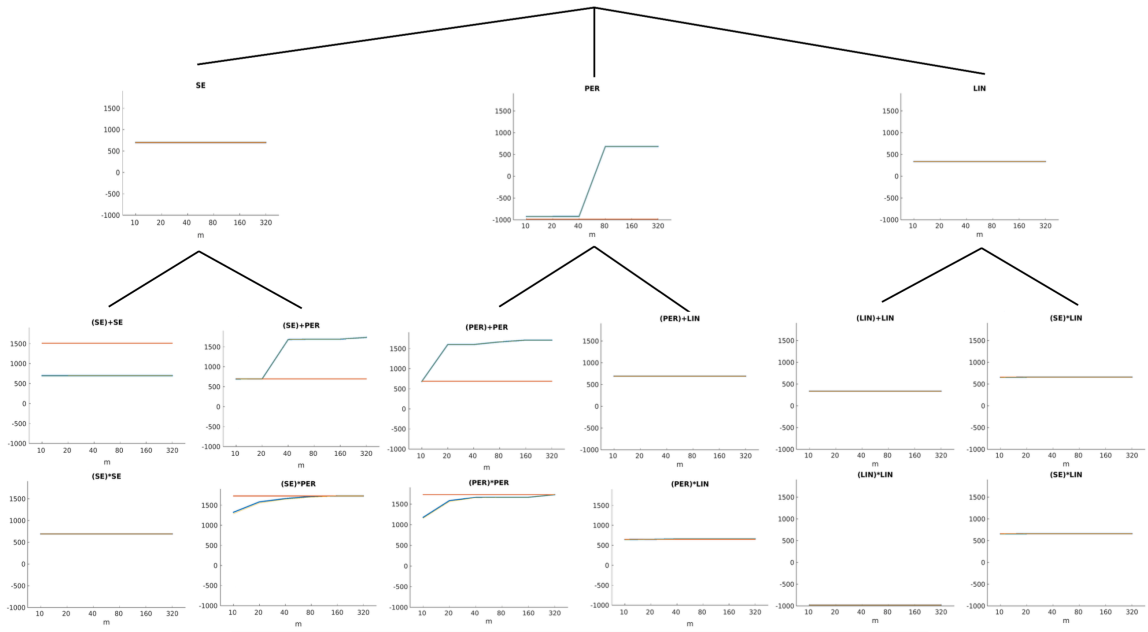


Figure 7: Same as Figure 6 but for Mauna data.

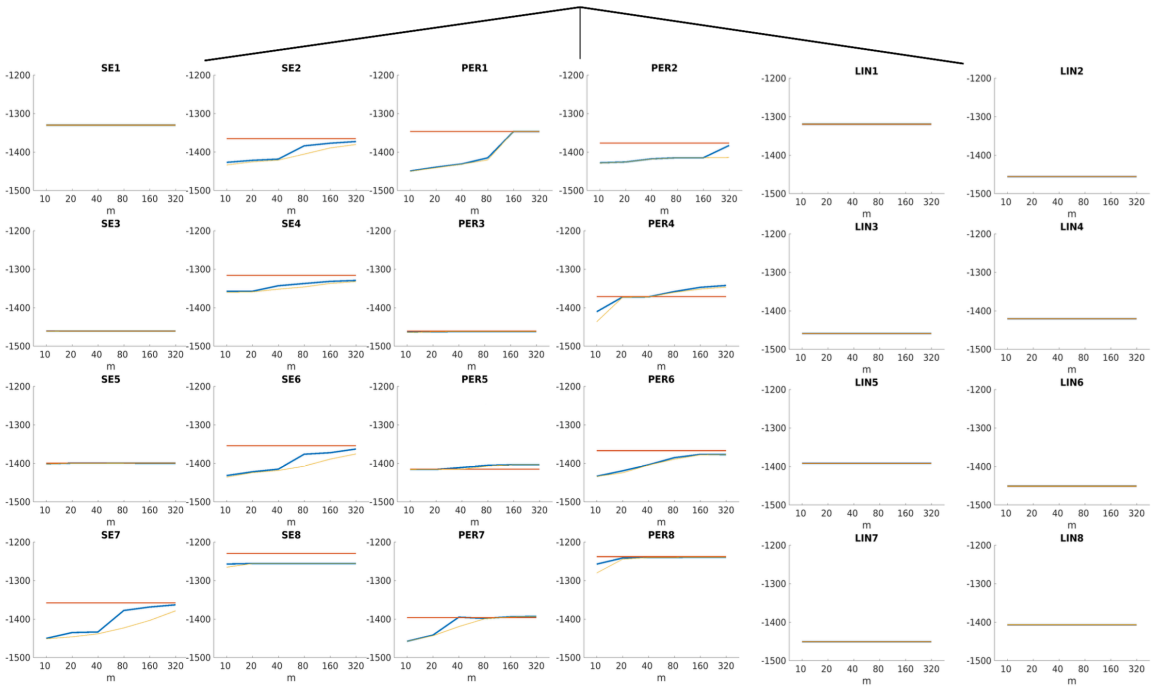


Figure 8: Same as Figure 6 but for concrete data and up to depth 1.