Estimating Nonlinear Neural Response Functions using GP Priors and Kronecker Methods
– Supplementary Information –

Cristina Savin
IST Austria
Klosterneuburg, AT 3400
`csavin@ist.ac.at`

Gasper Tkačik
IST Austria
Klosterneuburg, AT 3400
`tkacik@ist.ac.at`

The derivations below rely strongly on [1–4]. We provide a summary here for reading convenience and refer the reader to the original references for further details. The code for our implementation is available online (https://crissavin@bitbucket.org/crissavin/pfgp.git).

S1 Inference and learning

The generative model of the data takes the form of a log-normal Cox process, where spike counts $\mathbf{y}_i$ are generated by an inhomogeneous Poisson process with input-dependent mean $\lambda_i = \lambda\left(\mathbf{x}^{(i)}\right)$:

$$\mathrm{P}(\mathbf{y}|\mathbf{x}) = \prod_i \mathrm{Poisson}\left(y_i; \lambda_i\right), \qquad \text{where} \quad \mathrm{Poisson}\left(y; \lambda\right) = \frac{1}{y!}\lambda^y \mathrm{e}^{-\lambda}. \tag{1}$$

where the inputs $\mathbf{x}$ are defined on a d-dimensional lattice and the spike counts are measured within a time window δt. Assumptions concerning the nature of the functional dependency linking inputs to outputs is formalized as a gaussian process (GP) prior for the log-firing rate $f = \log \lambda(\mathbf{x}) \sim \mathcal{GP}(\mu, k_\beta)$ with constant mean $\mu_i = \alpha$ and covariance function $k(\cdot, \cdot)$ specified by hyperparameters β. The choice of covariance kernel determines the flexibility of the model and the computational complexity of the implementation. We postpone the discussion on the tradeoffs associated with different covariances for the moment, and focus on inference and learning in the general case.

First, the posterior over latent variable $\mathbf{f}$ given the data $\mathcal{D} = \{y_i, \mathbf{x}_i\}_{i=1:N}$ for fixed hyperparameters θ[1] takes the form:

$$\mathrm{P}\left(\mathbf{f}|\mathcal{D}\right) = \frac{\mathrm{P}\left(\mathbf{y}|\mathbf{f}\right)\mathrm{P}\left(\mathbf{f}|\mathbf{x}\right)}{\mathrm{P}\left(\mathbf{y}|\mathbf{x}\right)} \tag{2}$$

where $\mathrm{P}\left(\mathbf{y}|\mathbf{f}\right) = \prod_i \mathrm{P}\left(y_i|f_i\right)$ and $\mathrm{P}\left(\mathbf{f}|\mathbf{x}\right) = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \mathbf{K})$, where $\boldsymbol{\mu}$ and $\mathbf{K}$ are the GP mean and covariance for inputs $\mathbf{x}$. Because of the Poisson likelihood this expression does not have a simple closed form, but can be approximated with a normal distribution using the Laplace method [1].

In more detail, the Laplace approximation entails a 2nd order Taylor expansion of $\log \mathrm{P}\left(\mathbf{f}|\mathcal{D}\right)$ around the maximum of the posterior $\hat{\mathrm{f}} = \mathrm{argmax}_{\mathbf{f}} \mathrm{P}\left(\mathbf{f}|\mathbf{y}, \mathbf{x}\right)$, resulting in an approximate posterior:

$$q\left(\mathbf{f}|\mathbf{x}, \mathbf{y}\right) = \mathcal{N}\left(\mathbf{f}; \hat{\mathbf{f}}, \left(\mathbf{K} + \mathbf{H}^{-1}\right)^{-1}\right) \tag{3}$$

where the Hessian of the log-likelihood $\mathbf{H} = -\nabla\nabla \log \mathrm{P}(\mathbf{y}|\mathbf{f}) \mid_{\hat{\mathbf{f}}}$ is diagonal with $H_{ii} = \exp(\hat{f}_i)$ since the noise is conditionally independent Poisson given $\mathbf{f}$. Our linkage function is convex and log-concave so the posterior has a unique maximum $\hat{\mathbf{f}}$ which can be computed numerically as the

---

[1] To simplify notation, we omit the θ dependency here.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

maximum of unnormalized log-posterior $\Psi\left(\mathbf{f}\right)=\log\mathrm{P}\left(\mathbf{y}|\mathbf{f}\right)+\log\mathrm{P}\left(\mathbf{f}|\mathbf{x}\right)$, via Newton's method. This involves updates of the form (implementation details follow below):

$$\mathbf{f}^{\mathrm{new}}\leftarrow\mathbf{f}^{\mathrm{old}}+\left(\nabla\nabla\Psi\right)^{-1}\nabla\Psi. \tag{4}$$

The hyperparameters $\theta=\{\alpha,\beta\}$ are learned by maximizing the marginal likelihood: [2]

$$\mathrm{P}\left(\mathbf{y}|\mathbf{x}\right)=\int\mathrm{P}(\mathbf{y}|\mathbf{f})\mathrm{P}(\mathbf{f}|\mathbf{x})d\mathbf{f}=\int\exp\left(\Psi(\mathbf{f})\right)d\mathbf{f}. \tag{5}$$

This integral can be computed analytically using the same Laplace approximation,

$$\Psi(\mathbf{f})\approx\Psi\left(\hat{\mathbf{f}}\right)-\frac{1}{2}\left(\mathbf{f}-\hat{\mathbf{f}}\right)^{\mathrm{T}}\cdot\left(\mathbf{K}+\mathbf{H}^{-1}\right)^{-1}\cdot\left(\mathbf{f}-\hat{\mathbf{f}}\right) \tag{6}$$

resulting in the log marginal likelihood:

$$\mathcal{L}(\Theta)=-\frac{1}{2}\hat{\mathbf{f}}^{\mathrm{T}}\mathbf{K}^{-1}\hat{\mathbf{f}}+\log\mathrm{P}\left(\mathbf{y}|\hat{\mathbf{f}}\right)+\log|\mathbf{I}+\mathbf{K}\mathbf{H}|. \tag{7}$$

Consistent with past approaches, we use $\theta^*=\mathrm{argmax}_{\theta}\mathrm{P}\left(\theta|\mathrm{y}\right)$ to infer the tuning function for a set of test inputs $\mathbf{x}^*$.

Predicting the tuning function at a set of test locations $\mathbf{x}^*$ requires computing the mean and covariance of latent $\mathbf{f}^*$, after marginalizing $\mathbf{f}$ under the normal approximation q:

$$\begin{aligned}\mathbb{E}_q[\mathbf{f}^*|\mathcal{D},\mathbf{x}^*] &= \alpha+K\left(\mathbf{x}^*,\mathbf{x}\right)\cdot\mathbf{K^{-1}}\cdot\hat{\mathbf{f}}=\alpha+K\left(\mathbf{x}^*,\mathbf{x}\right)\cdot\nabla\mathrm{P}\left(\mathbf{y}|\mathbf{f}\right)|_{\hat{\mathbf{f}}} & (8)\\ \mathbb{V}_q[\mathbf{f}^*|\mathcal{D},\mathbf{x}^*] &= K\left(\mathbf{x}^*,\mathbf{x}^*\right)-K\left(\mathbf{x}^*,\mathbf{x}\right)\cdot(\mathbf{K}+\mathbf{H}^{-1})^{-1}\cdot K\left(\mathbf{x},\mathbf{x}^*\right). & (9)\end{aligned}$$

Since we have used an exponential linkage function, the predicted tuning function for an individual test point $\lambda(\mathbf{x}^*)=\exp(\mathbf{f}^*)$, is log-normal with mean $\exp(\mu_{f^*}+\sigma^2_{f^*}/2)$ and variance $\left(\exp(\sigma^2_{f^*})-1\right)\exp\left(2\mu_{f^*}+\sigma^2_{f^*}\right)$, where $\mu_{f^*}=\mathbb{E}_q[f^*|\mathcal{D},\mathbf{x}^*]$ and $\sigma^2_{f^*}=\mathbb{V}_q[f^*|\mathcal{D},\mathbf{x}^*]$.

S2 Efficient implementation using Kronecker methods

As we have seen above, numerical optimization is required for determining $\mathbf{f}^*$ and θ^*. This involves solving linear systems and determinants for $N\times N$ matrices, often via Cholesky decomposition, with computational cost $\mathcal{O}\left(N^3\right)$ and memory cost $\mathcal{O}\left(N^2\right)$, respectively, in the general case. These computations can be substantially sped up if additional assumptions are made concerning the inputs and the structure of $\mathbf{K}$. Here we rely on the fact that the inputs $\mathbf{x_i}$ lie on a (non-uniform) cartesian grid with arbitrary elements $x_i\in\mathbb{R}^{D_i}$ where D_i is the size of the i-th grid dimension (for simplicity we take these to all be the same in the following). Second, we assume that the prior covariance is described by a *tensor product kernel*, meaning that it can be written as a product of (arbitrary) kernels operating on scalar input spaces:

$$k(\mathbf{x},\mathbf{y})=\prod_{i=1}^{d}k_i(x_i,y_i). \tag{10}$$

Tensor products kernel seem a natural choice when different input dimensions live in different spaces with specific functional dependencies (e.g. for space-time data we may want to assume smoothness over space and periodicity in time). While not all popular kernels fall into this category, many of them do. For instance the popular squared-exponential kernel can be written in this form. The particular form of a tensor product kernel implies that the covariance matrix has Kronecker structure, i.e. it can be written as a Kronecker product

$$\mathbf{K}=\bigotimes_i\mathbf{K}_i \tag{11}$$

This dramatically simplifies computations.

---

[2] While we use an uninformative prior for θ here for simplicity, it may be useful to extent this to a hierarchical version with shared hyperparameters across simultaneously recorded cells or alternatively within cells across sessions.

S2.1 Kronecker product properties

A detailed explanation of Kronecker algebra properties is beyond the scope of this text, but we'll review here the most relevant features (see also chapter 5 and corresponding appendix from [2]). Briefly, the Kronecker product of two matrices $\mathbf{A}$ and $\mathbf{B}$ of size $m \times n$ and $p \times q$, respectively, is a $mp \times nq$ matrix of the form:

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{11}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \tag{12}$$

Because of this particular structure, traditional matrix operations on $\mathbf{K}$ can be rewritten in terms of operations on the (much smaller) component matrices $\mathbf{K}_i$:

$$\begin{array}{rc} \text{Transpose :} & \mathbf{K}^{\mathrm{T}} = \bigotimes_i \mathbf{K}_i^{\mathrm{T}} \qquad (13) \\ \text{Inverse :} & \mathbf{K}^{-1} = \bigotimes_i \mathbf{K}_i^{-1} \qquad (14) \\ \text{Eigen} - \text{decomposition :} & \mathbf{K} = \bigotimes_i \mathbf{Q}_i \cdot \bigotimes_i \mathbf{\Lambda}_i \cdot \bigotimes_i \mathbf{Q}_i^{\mathrm{T}} \qquad (15) \\ \text{Determinants :} & \det\left(\mathbf{K}\right) = \prod_i \det\left(\mathbf{K}_i\right)^{D_i} \qquad (16) \\ \text{Trace :} & \mathrm{tr}\left(\mathbf{K}\right) = \prod_i \mathrm{tr}\left(\mathbf{K}_i\right) \qquad (17) \\ \text{Vec :} & \mathrm{vec}\left(\mathbf{AXB}\right) = \mathbf{A} \otimes \mathbf{B}\ \mathrm{vec}\left(\mathbf{X}\right) \qquad (18) \end{array}$$

where the eigen-decomposition of the individual components is $\mathbf{K}_i = \mathbf{Q}_i \Lambda_i \mathbf{Q}_i^{\mathrm{T}}$ and vec($\mathbf{X}$) is a vector obtained by concatenating column-wise the elements in matrix $\mathbf{X}$.

Fast matrix-vector products (needed for the Newton updates) can be computed using tensor algebra (based on Eq.18, see Suppl. Info. in [3] for full derivation):

$$\left(\bigotimes_i \mathbf{K}_i\right) \cdot \mathbf{u} = \mathrm{vec}\left(\left[\mathbf{K}_1, \ldots \left[\mathbf{K}_{d-1}, \left[\mathbf{K}_d, \mathbf{U}\right]\right]\right]\right) \overset{\text{def}}{=} \mathrm{kronmvprod}\left(\mathbf{K}_1, \ldots \mathbf{K}_d, \mathbf{u}\right) \tag{19}$$

where the brackets denote a matrix product, followed by the reshaping of the resulting matrix, $\left[\mathbf{K}_d, \mathbf{U}\right] = \mathrm{reshape}\left(\mathbf{K}_d\mathbf{U}, N^{\frac{1}{d}}, N^{\frac{d-1}{d}}\right)$ and $\mathbf{U} = \mathrm{reshape}\left(\mathbf{u}, N^{\frac{1}{d}}, N^{\frac{d-1}{d}}\right)$.

S2.2 Kronecker GP implementation

For numerical stability, the update equations for Newton's method are first rewritten in the form [1]:

$$\mathbf{f}^{\text{new}} \leftarrow \mathbf{Ka} \tag{20}$$

using the sequence of transformations:

$$\begin{array}{rcl} \mathbf{B} & = & \mathbf{I} + \mathbf{H}^{\frac{1}{2}}\mathbf{K}\mathbf{H}^{\frac{1}{2}} \qquad (21) \\ \mathbf{Q} & = & \mathbf{H}^{\frac{1}{2}}\mathbf{B}^{-1}\mathbf{H}^{\frac{1}{2}} \qquad (22) \\ \mathbf{b} & = & \mathbf{H}\left(\mathbf{f} - \boldsymbol{\mu}\right) + \nabla \mathrm{P}(\mathbf{y}|\mathbf{f}) \qquad (23) \\ \mathbf{a} & = & \mathbf{b} - \mathbf{QKb} \qquad (24) \end{array}$$

After the variable change we still need to invert $\mathbf{B}$, which unfortunately does not have Kronecker structure. This problem can be circumvented by using linear conjugate gradients (LCG), an iterative method that replaces the costly inversion with matrix-vector products. In this representation, the variance of the predictive distribution $\mathrm{P}\left(\mathbf{f}^*|\mathcal{D}, \mathbf{x}^*, \theta\right)$ becomes:

$$\mathbb{V}_q[\mathbf{f}^*|\mathcal{D}, \mathbf{x}^*] = \mathbf{K}_{**} - \mathbf{K}_*\mathbf{Q}\,\mathbf{K}_*^{\mathbf{T}}. \tag{25}$$

Evaluating the marginal likelihood requires computing $\log|\mathbf{I} + \mathbf{KH}|$. Using the eigenvalue decomposition of $\mathbf{K}$, obtained in $\mathcal{O}\left(dN^{\frac{3}{d}}\right)$, one can bound this complexity term (using Fiedler's bound for the log determinant of a sum of Hermitian positive semidefinite matrices, see [4] for details) by:

$$\log|\mathbf{I} + \mathbf{KH}| \leq \sum_i \log(1 + e_i h_i) \tag{26}$$

where e_i and h_i are the eigenvalues of $\mathbf{K}$ and $\mathbf{H}$, respectively.[3] This results in a lower bound for the marginal likelihood of the form:

$$\mathcal{L}(\theta) \geq -\frac{1}{2}\hat{\mathbf{f}}^{\mathrm{T}}\mathbf{K}^{-1}\hat{\mathbf{f}} + \log \mathrm{P}\left(\mathbf{y}|\hat{\mathbf{f}}\right) - \frac{1}{2}\sum_i \log(1 + e_i h_i) \tag{27}$$

where the inversion of $\mathbf{K}$ can be done efficiently using Eq.14.

Pseudocode for the final algorithm is reproduced from [4] as Algorithm 1. For simplicity, we assume here that the grid has the same cardinality across all dimensions, i.e. $D_i = N^{\frac{1}{d}}$, for all i. Given this assumption, matrix-vector products (function kronmvprod) can be evaluated in $\mathcal{O}\left(dN^{\frac{d+1}{d}}\right)$. Consequently, computing $\mathbf{f}^*$ requires $\mathcal{O}\left(mdN^{\frac{d+1}{d}}\right)$ operations, where m is the number of Newton steps needed for convergence. Memory-wise, we only need to store the matrices $\mathbf{K}_i$ with total size $\mathcal{O}\left(dN^{\frac{2}{d}}\right)$.

Algorithm 1 Kronecker GP inference and learning

```
procedure GP($\theta$, $\boldsymbol{\mu}$, $\mathbf{K}_{1:d}$, $\mathrm{P}(\mathbf{y}|\mathbf{f})$, $\mathbf{y}$)
    $\boldsymbol{\alpha} \leftarrow 0$
    repeat
        $\mathbf{f} \leftarrow \mathbf{K}\boldsymbol{\alpha}$                                   ▷ using kronmvprod
        $\mathbf{H} \leftarrow -\nabla\nabla\mathrm{P}(\mathbf{y}|\mathbf{f})$                      ▷ diagonal matrix
        $\mathbf{b} = \mathbf{H}(\mathbf{f} - \boldsymbol{\mu}) + \nabla\mathrm{P}(\mathbf{y}|\mathbf{f})$
        $\mathbf{z} \leftarrow \mathrm{CG}(\mathbf{B}, \mathbf{H})$     ▷ use conjugate gradient to solve $\mathbf{B}\mathbf{z} = \mathbf{H}^{-\frac{1}{2}}\mathbf{b}$
        $\Delta\boldsymbol{\alpha} \leftarrow \mathbf{H}^{\frac{1}{2}}\mathbf{z} - \boldsymbol{\alpha}$
        $\hat{\xi} \leftarrow \mathrm{argmin}_{\xi}\Psi(\boldsymbol{\alpha} + \xi\Delta\boldsymbol{\alpha})$                   ▷ line search
        $\boldsymbol{\alpha} \leftarrow \boldsymbol{\alpha} + \hat{\xi}\Delta\boldsymbol{\alpha}$                          ▷ actual update step
    until convergence
    $\mathbf{e} \leftarrow \mathrm{eig}(\mathbf{K})$                              ▷ exploit Kronecker structure
    $Z =\leftarrow \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}(\mathbf{f} - \boldsymbol{\mu}) + \frac{1}{2}\sum_i \log(1 + e_i h_i) - \log \mathrm{P}(\mathbf{y}|\mathbf{f})$   ▷ marginal likelihood bound
    return $\mathbf{f}$,$\boldsymbol{\alpha}$, $Z$
```

An important limitation of the Kronecker approach is the fact that N grows exponentially with d, which restricts the practical applicability of the algorithm to up to 7 or 8 dimensions. Nonetheless, many nonlinear tuning functions we seek to estimate satisfy this constraint. This problem can be somewhat alleviated by limiting the input set to the biologically-relevant subregion of the hypercube. For instance, in the spatial domain we restrict the test positions to be within the borders of the box (either circular or cross-shaped). A clever discretization of the individual dimensions can also help.

S3 Comparison of different tuning function estimators

Traditional approaches for estimating nonlinear tuning functions such as place fields rely on histograms, usually followed by smoothing. Despite their appealing simplicity, such estimators require very large amounts of data [5]. Furthermore, the degree of smoothing is usually decided arbitrarily, with potentially misleading results, for instance generating place-field like structure out of noise in the limit of very little data or very low firing rates.

Alternative methods increase estimation efficiency by a probabilistic treatment, that also provides a principled way for setting smoothing parameters. These benefits come at the cost of a significant increase in computational time. Several algorithms of this class have been proposed in the literature, which differ mainly in their treatment of the tradeoff between model flexibility and computational tractability. At one extreme, [5] makes strong assumptions about the covariance structure (sparse band-diagonal inverse covariance, with dependences restricted to nearest-neighbour interactions) but with the fastest implementation (linear in $\mathbf{N}$). At the other extreme, [6] enforces no constraints on $\mathbf{K}$, with cubic computational cost, but relies on an active learning framework to make sure that the very limited data it uses is most informative. Our approach sits between these two in that it achieves

---

[3] $\mathbf{H}$ is diagonal, so the only eigen-decomposition that matters computationally is that of $\mathbf{K}$.

almost linear computational cost ($N = 10^5$ learning done in minutes) but with much more flexibility in terms of the functional structure captured by the prior, especially when combined with expressive kernels such as SM. To provide some intuition for the flexibility of different priors, we show a set of tuning functions drawn from a log-normal Cox process with different covariances (Fig. S1). The Kronecker kernel is as expressive as the standard SE kernel (they are formally equivalent), with different hyperparameters capturing different spatial scales and degrees of smoothness. In contrast, Rad-like covariances have inherently small length scales (due to the nearest-neighbour correlations). Lastly, the Kronecker-SM kernel can represent a combination of structures at different spatial scales.

References

[1] Rasmussen, C.E. & Williams, C.K.I. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)* (The MIT Press, 2005).

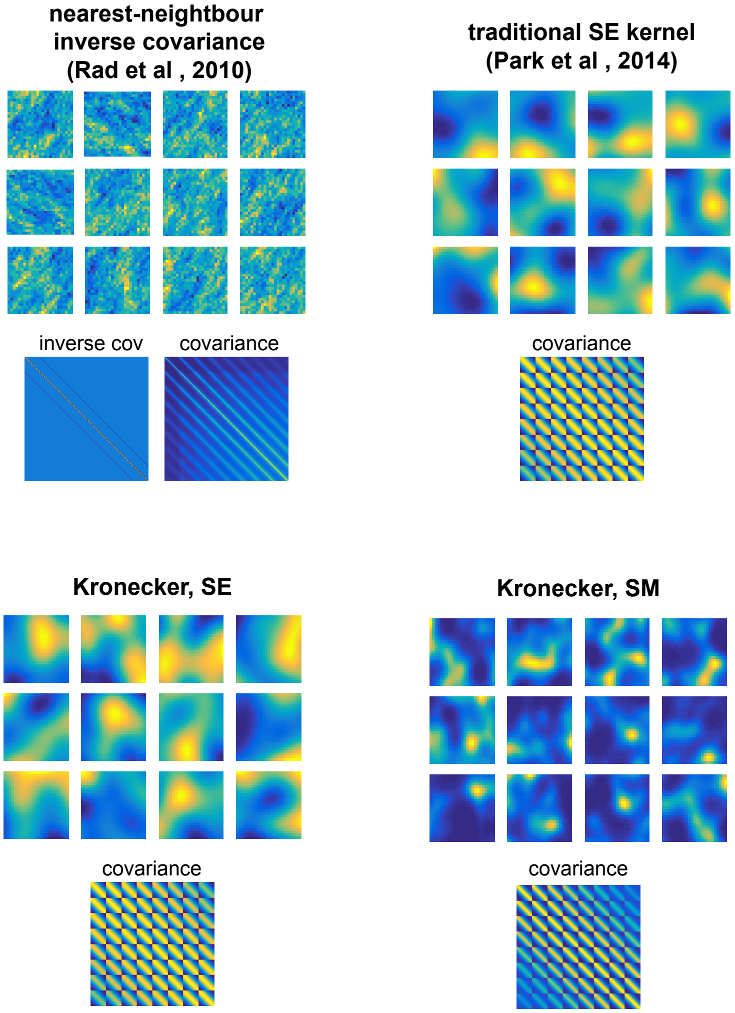
[2] Saatçi, Y. *Scalable inference for structured Gaussian process models*. PhD thesis, Cambridge University, UK, (2012).

[3] Wilson, A., Gilboa, E., Nehorai, A. & Cunningham, J. Fast kernel learning for multidimensional pattern extrapolation. in *Advances in Neural Information Processing Systems* (MIT Press, 2014).

[4] Flaxman, A., Wilson, A., Neill, D., Nickisch, H. & Smola, A. Fast Kronecker inference in Gaussian Processes with non-Gaussian likelihoods. in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)* , 607–616, (2015).

[5] Rad, K.R. & Paninski, L. Efficient, adaptive estimation of two-dimensional firing rate surfaces via Gaussian process methods. *Network* **21**, 142–168 (2010).

[6] Park, M., Weller, J.P., Horwitz, G.D. & Pillow, J.W. Bayesian active learning of neural firing rate maps with transformed gaussian process priors. *Neural computation* **26**, 1519–1541 (2014).



nearest-neightbour inverse covariance (Rad et al , 2010)

traditional SE kernel (Park et al , 2014)

Kronecker, SE

Kronecker, SM

Figure S1: **Flexibility of different kernels.** Top row: Sample tuning functions $\boldsymbol{\lambda} = \exp(\mathbf{f})$ on a 24×24 grid, with $\mathbf{f}$ drawn from a GP prior with zero mean different covariances. From left to right: band-diagonal inverse covariance (parameters as in Fig.2 from [5]), traditional SE kernel with $\rho_1 = \rho_2 = 0.3$, and $\sigma_1 = \sigma_2 = 0.5$, Kronecker SE with same parameters; Kronecker spectral mixture kernel with 5 components; weights, spectral means and variances drawn independently from uniform distribution, with weights further normalised to L1 norm 1. Bottom row: corresponding covariance structure (first 200×200 elements of K.)