Deep learning estimation of the spectral density of functional time series on large domains

Piotr Kokoszka

Colorado State University

Joint work with

Neda Mohammadi

University of Texas El Paso

Soham Sarkar

Indian Statistical Institute

Introduction

Spectral density of a scalar time series:

$$f(\theta) = \frac{1}{2\pi} \sum_{h=-\infty}^{\infty} c_h e^{-\mathrm{i}h\theta}, \quad \theta \in [-\pi, \pi].$$

Spectral density of a functional time series is a function on $[-\pi, \pi]$ whose values are Hilbert-Schmidt operators:

$$F^X(\theta) = \frac{1}{2\pi} \sum_{h\in\mathbb{Z}} C_h \exp(-\mathrm{i}h\theta), \quad \theta \in [-\pi, \pi].$$

Introduction

The autocovariance operators

$$C_h = \mathbb{E}[(X_{t+h} - \mu) \otimes (X_t - \mu)] \quad (X_t \in L^2(\mathcal{Q}))$$

are estimated by the kernels

$$\hat{c}_h(u, v) = \frac{1}{N} \sum_{k=1}^{N-h} X_{h+k}(u) X_k(v), \quad h \geq 0.$$

(Assume $\mu = 0$, a similar formula for $h < 0$.)

Introduction

We consider the case of the observations $X_t(u)$, where u is a point of a grid $\mathcal{Q}$ (2D or 3D in practice).

Examples: climate data, brains scans $D = |\mathcal{Q}| \sim 10^4 - 10^5$ points.

For such data, $\hat{c}_h(u, v)$ must be evaluated at $10^8 - 10^{10}$ points.

Matrices of such dimensions are difficult to manipulate and sample autocovariance matrices are even theoretically not consistent. Some statistical calculations become infeasible even on powerful, broadly available, computers.

Objective: Derive a method of estimating the spectral density F^X without estimating the autocovariance operators.

Introduction

Key ingredients of the solution:

1. The deep learning CovNet method of Sarkar and Panaretos (2024, JRSSB)
(Networks for the estimation of the the functional variance kernel)

2. The dynamic functional principal components of Hörmann, Kidziński and Hallin (2015, JRSSB)
(Spectral principal components for functional time series)

(There is a similar solution to the two sample problem for functional data that uses CovNet ideas and and the method of Horváth, Kokoszka and Reeder (2013, JRSSB).)

Dynamic FPCs

We omit mathematical assumptions and precise statements, $\{X_t\}$ is stationary.

Hörmann *et al.* (2015) showed that

$$X_t(u) = \sum_{m \geq 1} \sum_{h \in \mathbb{Z}} Y_{m,t+h} \varphi_{m,h}(u), \tag{1}$$

where the scalar sequences $\{Y_{m,t}\}$ and $\{Y_{m',t}\}$ are uncorrelated at all lags if $m \neq m'$.

The components of decomposition (1) can be estimated and approximated, but this requires the computation of the sample autocorrelations, so we will not use these methods, merely the validity of expansion (1).

Deep networks

We consider multilayer perceptrons (MLPs)

$$\mathfrak{g}_r : \mathcal{Q} \to \mathbb{R}, \quad r = 1, 2, \dots R.$$

The count R is a hyperparameter. Each $\mathfrak{g}_r$ is defined by the iterations

$$\begin{aligned} u_{1,r} =& \sigma(W_{1,r} u + B_{1,r}), \quad u \in \mathcal{Q}, \\ u_{j+1,r} =& \sigma(W_{j+1,r} u_{j,r} + B_{j+1,r}), \quad j = 1, 2, \dots, J-1, \\ \mathfrak{g}_r(u) =& c_r \sigma(w_r^\top u_{J,r} + b_r), \quad r = 1, \dots R. \end{aligned}$$

We use fraktur to denote networks.
The hyperparameter J is the depth of the network.
The elements of the matrices $W_{j,r}$ and vectors $B_{j,r}$ are the parameters that are learned through training.

Approximation by deep networks

We denote by $\mathcal{C}^{\mathrm{nn}}$ any class of networks of the form

$$\mathfrak{g} = \sum_{r=1}^{R} \mathfrak{g}_r, \quad R \in \mathbb{N}.$$

(This representation is motivated by approximations of the covariance function by separable covariance functions.)

In light of (1), for $M, L \in \mathbb{N}$, we consider the stationary sequence of random networks (fields)

$$\widetilde{\mathfrak{X}}_t(u) = \sum_{h=-L}^{L} \sum_{m=1}^{M} \xi_{m,t+h} \mathfrak{g}_{m,h}(u), \quad \mathfrak{g}_{m,h} \in \mathcal{C}^{\mathrm{nn}} \subset L^2(\mathcal{Q}). \qquad (2)$$

Theory: The $\xi_{m,t}$ are random variables with the same properties as the DFPC scores $Y_{m,t}$.

Practice: The $\xi_{m,t}$ are additional parameters to be learned.

Approximation by deep networks

Compare:

$\{X_t\}$ has spectral density kernel

$$f^X(\theta)(u,v) = \sum_{m\geq 1} \lambda_m(\theta)\varphi^\dagger_m(\theta)(u)\overline{\varphi}^\dagger_m(\theta)(v), \quad u,v \in \mathcal{Q}, \tag{3}$$

$\{\widetilde{\mathfrak{X}}_t\}$ has spectral density kernel

$$F^{\widetilde{\mathfrak{X}}}(\theta)(u,v) = \sum_{m=1}^{M} F^{\xi}_{m,m}(\theta)\mathfrak{g}^\ddagger_m(\theta)(u)\mathfrak{g}^\ddagger_m(\theta)(v). \tag{4}$$

(The daggers are suitable Fourier transforms.)
One can show that $F^{\widetilde{\mathfrak{X}}}$ becomes close to F^X as the hyperparameters grow.

A loss function

$$\begin{aligned}\ell &:= \int_{-\pi}^{\pi} \left\| \hat{f}^{X}(\theta) - \hat{f}^{\widetilde{\mathfrak{X}}}(\theta) \right\|_{\mathcal{S}} d\theta \\ &= \int_{-\pi}^{\pi} \left\| \sum_{|h|\le q} \omega\left(\frac{h}{q}\right) \left[\widehat{C}_h^{X} - \widehat{C}_h^{\mathfrak{X}}\right] \exp(-\mathrm{i}h\theta) \right\|_{\mathcal{S}} d\theta.\end{aligned}$$

The loss ℓ depends on hundreds of parameters: the matrices $W_{j,r}$, the vectors $B_{j,r}$, the $\xi_{m,t}$, denote them by $\boldsymbol{\eta}$.

Key observation:
Due to cancelations, the matrices $\widehat{C}_h^{X}$ and $\widehat{C}_h^{\mathfrak{X}}$ do not have to be computed. The loss ell can be expressed a sum of four terms, each containing $O(D)$ sums like

$$\langle \widetilde{\mathfrak{X}}_{h+k}, X_{h'+k'} \rangle = \int_{\mathcal{Q}} \widetilde{\mathfrak{X}}_{h+k}(u) X_{h'+k'}(u) du.$$

Algorithm

1. Code the expression for $\ell = \ell(\boldsymbol{\eta})$.
(It involves only $O(D)$ operations.)

2. Use deep learning (forward-backward passes with a single batch) to make $\ell(\boldsymbol{\eta})$ as small as desirable or possible.
Denote the minimizer by $\hat{\boldsymbol{\eta}}$.

3. Approximate the the autocovariance kernels by (cf. (2))

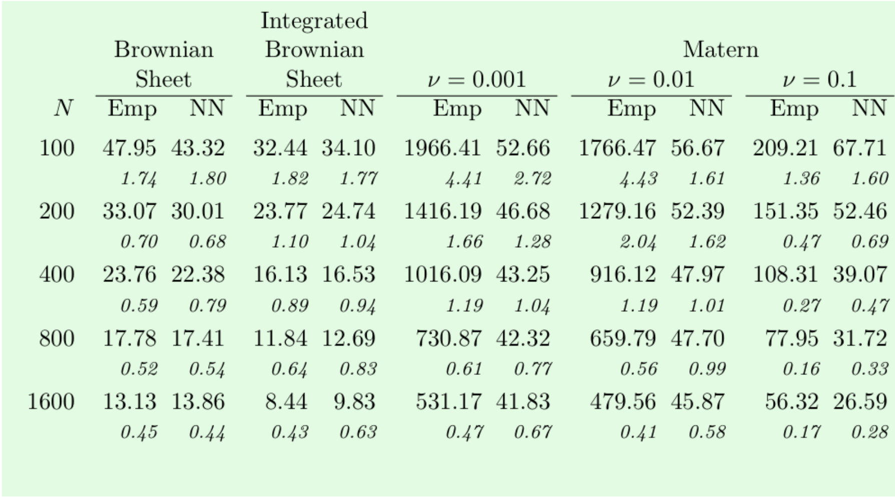
$$\hat{c}_h^{\widetilde{\mathfrak{X}}}(\hat{\boldsymbol{\eta}})(u,v)$$
$$= \frac{1}{N} \sum_{k=1}^{N} \sum_{j,j'=-L}^{L} \sum_{m,m'=1}^{M} \left(\hat{\xi}_{m,h+k+j} \hat{\xi}_{m',k+j'} \right) \mathfrak{g}(\hat{\boldsymbol{\eta}})_{m,j}(u) \mathfrak{g}(\hat{\boldsymbol{\eta}})_{m',j'}(v).$$

4. Compute the estimated cospectrum

$$\hat{p}^X(\theta)(u,v) = \frac{1}{2\pi} \sum_{|h| \le q} \omega\left(\frac{h}{q}\right) \hat{c}_h^{\widetilde{\mathfrak{X}}}(\hat{\boldsymbol{\eta}})(u,v) \cos(h\theta)$$

and an analogous quadspectrum (with sin).

Simulations: Integrated errors for $D = 50 \times 50$

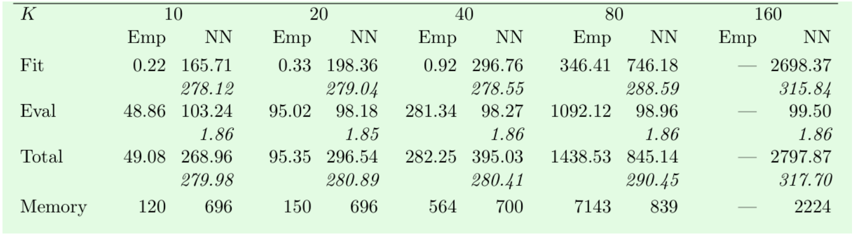


	Brownian Sheet		Integrated Brownian Sheet		Matern					
					$\nu = 0.001$		$\nu = 0.01$		$\nu = 0.1$	
N	Emp	NN	Emp	NN	Emp	NN	Emp	NN	Emp	NN
100	47.95	43.32	32.44	34.10	1966.41	52.66	1766.47	56.67	209.21	67.71
	1.74	*1.80*	*1.82*	*1.77*	*4.41*	*2.72*	*4.43*	*1.61*	*1.36*	*1.60*
200	33.07	30.01	23.77	24.74	1416.19	46.68	1279.16	52.39	151.35	52.46
	0.70	*0.68*	*1.10*	*1.04*	*1.66*	*1.28*	*2.04*	*1.62*	*0.47*	*0.69*
400	23.76	22.38	16.13	16.53	1016.09	43.25	916.12	47.97	108.31	39.07
	0.59	*0.79*	*0.89*	*0.94*	*1.19*	*1.04*	*1.19*	*1.01*	*0.27*	*0.47*
800	17.78	17.41	11.84	12.69	730.87	42.32	659.79	47.70	77.95	31.72
	0.52	*0.54*	*0.64*	*0.83*	*0.61*	*0.77*	*0.56*	*0.99*	*0.16*	*0.33*
1600	13.13	13.86	8.44	9.83	531.17	41.83	479.56	45.87	56.32	26.59
	0.45	*0.44*	*0.43*	*0.63*	*0.47*	*0.67*	*0.41*	*0.58*	*0.17*	*0.28*

Simulations: $N = 250$, $D = K \times K$

	Brownian Sheet		Integrated Brownian Sheet				Matern			
					$\nu = 0.001$		$\nu = 0.01$		$\nu = 0.1$	
K	Emp	NN	Emp	NN	Emp	NN	Emp	NN	Emp	NN
10	33.53	31.58	26.47	22.51	1413.66	172.78	1276.75	222.37	147.18	82.57
	0.97	*0.95*	*0.99*	*1.00*	*4.15*	*6.59*	*4.01*	*7.93*	*1.04*	*1.59*
20	31.19	29.06	21.60	20.53	1301.69	80.31	1176.46	84.45	138.13	57.98
	0.76	*0.84*	*0.97*	*1.06*	*2.83*	*3.20*	*2.61*	*3.02*	*0.78*	*0.90*
40	29.11	26.60	18.28	18.40	1275.89	47.07	1153.01	51.27	135.11	47.21
	0.76	*0.79*	*0.98*	*1.09*	*1.64*	*1.39*	*1.53*	*1.21*	*0.53*	*0.76*
80	30.72	28.34	20.07	20.26	1267.63	42.38	1145.32	47.85	135.58	45.96
	0.80	*0.87*	*0.96*	*1.10*	*1.12*	*1.17*	*0.99*	*0.94*	*0.54*	*0.75*
160	—	26.76	—	19.27	—	37.58	—	43.96	—	43.53
		0.79		*1.12*		*0.64*		*0.53*		*0.60*

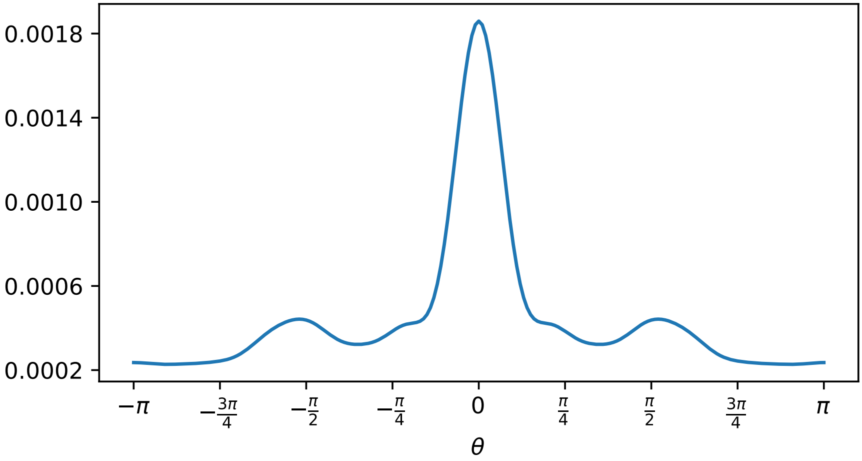
Simulations: $N = 250$, time (s) and memory (MB)



K	10		20		40		80		160	
	Emp	NN	Emp	NN	Emp	NN	Emp	NN	Emp	NN
Fit	0.22	165.71	0.33	198.36	0.92	296.76	346.41	746.18	—	2698.37
		278.12		*279.04*		*278.55*		*288.59*		*315.84*
Eval	48.86	103.24	95.02	98.18	281.34	98.27	1092.12	98.96	—	99.50
		1.86		*1.85*		*1.86*		*1.86*		*1.86*
Total	49.08	268.96	95.35	296.54	282.25	395.03	1438.53	845.14	—	2797.87
		279.98		*280.89*		*280.41*		*290.45*		*317.70*
Memory	120	696	150	696	564	700	7143	839	—	2224

Computer with 64 GiB RAM, AMD Ryzen 9 5900X (3.7 GHz) CPU, NVIDIA GeForce RTX 3090 GPU, and Ubuntu 24.04.2 LTS (64-bit) OS

Norm of the cospectrum of fMRI scans


$-\pi$
$-\frac{3\pi}{4}$
$-\frac{\pi}{2}$
$-\frac{\pi}{4}$
$\frac{\pi}{4}$
$\frac{\pi}{2}$
$\frac{3\pi}{4}$
π
θ