Gaussian processes for time series

- Anna Scaife's (University of Manchester) blog:

https://allofyourbases.com/2017/08/21/gaussian-processes-in-python/
https://allofyourbases.com/2017/09/04/gaussian-process-modelling-in-python/
https://allofyourbases.com/2017/09/17/predicting-the-future/
https://allofyourbases.com/2017/09/23/spinning-stars/
https://allofyourbases.com/2017/10/02/spinning-stars-ii-celerite/

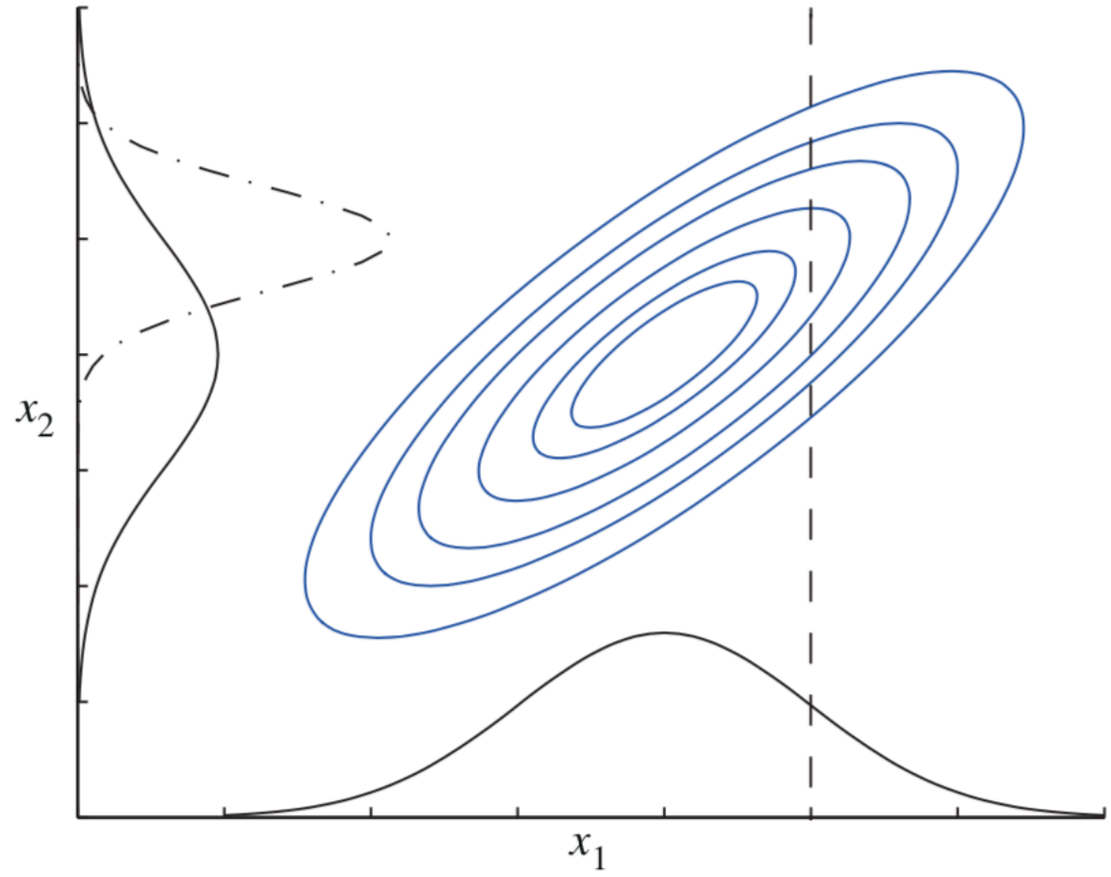
- Roberts S, Osborne M, Ebden M, Reece S, Gibson N, Aigrain S. 2013 Gaussian processes for time-series modelling. Phil Trans R Soc A 371: 20110550. http://dx.doi.org/10.1098/rsta.2011.0550

- http://www.gaussianprocess.org/

Co-variate Gaussian noise

covariance matrix

$$\mathbf{K}(\mathbf{x},\mathbf{x}) = \begin{pmatrix} k(x_1,x_1) & k(x_1,x_2) & \cdots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \cdots & k(x_2,x_n) \\ \vdots & \vdots & \vdots & \vdots \\ k(x_n,x_1) & k(x_n,x_2) & \cdots & k(x_n,x_n) \end{pmatrix}.$$



Co-variate Gaussian noise

$$\mathbf{K}(\mathbf{x},\mathbf{x}) = \begin{pmatrix} k(x_1,x_1) & 0 & ... & 0 \\ 0 & k(x_2,x_2) & ... & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & ... & k(x_n,x_n) \end{pmatrix}$$

diagonal co-variance matrix shows uncorrelated (independent) Gaussian noise

each value in this covariance matrix represents the variance at some point.

The actual y-value at each x-position is drawn from a probability distribution:

$$p(y_i) = N(\mu_i, K_{ij})$$

If the co-variance matrix is diagonal this becomes:

$$y_i = \mu_i + N(0,\sigma_i^2)$$

But, if the co-variance matrix is non-diagonal, the y-values at one position affects those at another.

The co-variance kernel

- This is the “formula” for making the co-variance matrix
- It controls how the data points affect each other
- Squared-exponential co-variance kernel:

$$k(x_n, x_m) = h^2 \exp\left(\frac{-(x_n - x_m)^2}{\lambda^2}\right)$$

- h (normalisation) and λ (width) are hyper parameters
- There are many other options for co-variance kernels, even periodic ones.

As part of Tut 4:

1. Use a squared-exponential kernel to make a co-variance matrix (300 x 300)

2. Plot the covariance matrix for $\lambda = 0.1$, 1 and 10, with a constant normalisation.

3. Plot the *y* values (just as a function of index) drawn from this distribution.

4. Comment on the difference between the distributions for different values of λ.

5. Extend this to a model, by splitting your data into test and training sets…

Modelling with Gaussian processes

- Let's try to predict the value of a function at any point based on its value at a few known points

- Assumption: *behaviour of function depends only on the covariance between measurement at different separations*

- Invert the probability statement to get y_i at some x_i: $p(y_i) = N(\mu_i, K_{ij})$

- You need to know or calculate covariance matrix and some training data.

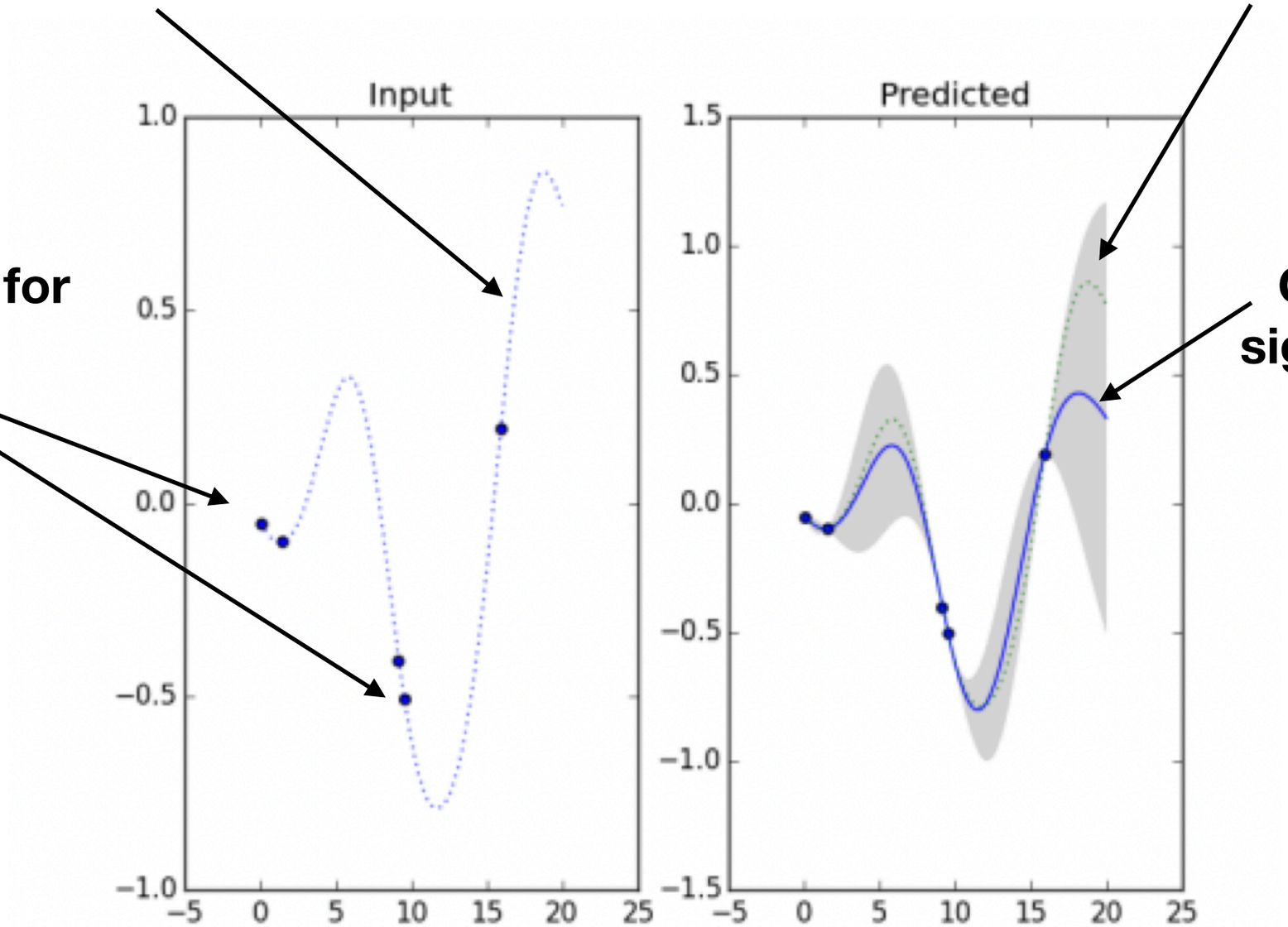
$$\mathbf{m}_* = \mathbf{K}(\mathbf{x}_*, \mathbf{x})^{\mathbf{T}}\mathbf{K}(\mathbf{x}, \mathbf{x})^{-1}\mathbf{y}$$
$$\mathbf{C}_* = \mathbf{K}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{K}(\mathbf{x}_*, \mathbf{x})^{\mathbf{T}}\mathbf{K}(\mathbf{x}, \mathbf{x})^{-1}\mathbf{K}(\mathbf{x}_*, \mathbf{x})$$

This is for measurements with zero mean

$$\mathbf{m}_* = \mathbf{k}_*^{\mathbf{T}}\mathbf{K}^{-1}\mathbf{y}$$
$$\mathbf{C}_* = \mathbf{k}(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^{\mathbf{T}}\mathbf{K}^{-1}\mathbf{k}_*$$

http://www.gaussianprocess.org/gpml/chapters/RW2.pdf

Modelling with Gaussian processes



Markov chain Monte Carlo

- Useful method for integrating over a large number of parameters
- Monte Carlo - statistically independent samples
- MCMC samples from continuous random variable, with prob density proportional to a known function
- Generate an ensemble of chains starting from arbitrarily chosen points. Move about randomly according to an algorithm that searches for where high contributions to the integral are.
- Metropolis-Hastings algorithm is a well-known implementation of the random walker.