

Fitting Covariance and Multioutput Gaussian Processes

Neil D. Lawrence

GPWS
14th January 2014



Outline

Parametric Models are a Bottleneck

Constructing Covariance

GP Limitations

Kalman Filter

Outline

Parametric Models are a Bottleneck

Constructing Covariance

GP Limitations

Kalman Filter

Nonparametric Gaussian Processes

- ▶ This work takes us from parametric to non-parametric.
- ▶ The limit implies infinite dimensional \mathbf{w} .
- ▶ Gaussian processes are generally non-parametric: combine data with covariance function to get model.
- ▶ This representation *cannot* be summarized by a parameter vector of a fixed size.

The Parametric Bottleneck

- ▶ Parametric models have a representation that does not respond to increasing training set size.
- ▶ Bayesian posterior distributions over parameters contain the information about the training data.
 - ▶ Use Bayes' rule from training data, $p(\mathbf{w}|\mathbf{y}, \mathbf{X})$,
 - ▶ Make predictions on test data

$$p(y_*|\mathbf{X}_*, \mathbf{y}, \mathbf{X}) = \int p(y_*|\mathbf{w}, \mathbf{X}_*) p(\mathbf{w}|\mathbf{y}, \mathbf{X}) d\mathbf{w}.$$

- ▶ \mathbf{w} becomes a bottleneck for information about the training set to pass to the test set.
- ▶ Solution: increase M so that the bottleneck is so large that it no longer presents a problem.
- ▶ How big is big enough for M ? Non-parametrics says $M \rightarrow \infty$.

The Parametric Bottleneck

- ▶ Now no longer possible to manipulate the model through the standard parametric form.

The Parametric Bottleneck

- ▶ Now no longer possible to manipulate the model through the standard parametric form.
- ▶ However, it *is* possible to express *parametric* as GPs:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi: (\mathbf{x}_i)^\top \phi: (\mathbf{x}_j).$$

The Parametric Bottleneck

- ▶ Now no longer possible to manipulate the model through the standard parametric form.
- ▶ However, it *is* possible to express *parametric* as GPs:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi: (\mathbf{x}_i)^\top \phi: (\mathbf{x}_j).$$

- ▶ These are known as degenerate covariance matrices.

The Parametric Bottleneck

- ▶ Now no longer possible to manipulate the model through the standard parametric form.
- ▶ However, it is possible to express *parametric* as GPs:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi: (\mathbf{x}_i)^\top \phi: (\mathbf{x}_j).$$

- ▶ These are known as degenerate covariance matrices.
- ▶ Their rank is at most M , non-parametric models have full rank covariance matrices.

The Parametric Bottleneck

- ▶ Now no longer possible to manipulate the model through the standard parametric form.
- ▶ However, it is possible to express *parametric* as GPs:

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j).$$

- ▶ These are known as degenerate covariance matrices.
- ▶ Their rank is at most M , non-parametric models have full rank covariance matrices.
- ▶ Most well known is the “linear kernel”, $k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j$.

Making Predictions

- ▶ For non-parametrics prediction at new points \mathbf{f}_* is made by conditioning on \mathbf{f} in the joint distribution.

Making Predictions

- ▶ For non-parametrics prediction at new points \mathbf{f}_* is made by conditioning on \mathbf{f} in the joint distribution.
- ▶ In GPs this involves combining the training data with the covariance function and the mean function.

Making Predictions

- ▶ For non-parametrics prediction at new points \mathbf{f}_* is made by conditioning on \mathbf{f} in the joint distribution.
- ▶ In GPs this involves combining the training data with the covariance function and the mean function.
- ▶ Parametric is a special case when conditional prediction can be summarized in a *fixed* number of parameters.

Making Predictions

- ▶ For non-parametrics prediction at new points \mathbf{f}_* is made by conditioning on \mathbf{f} in the joint distribution.
- ▶ In GPs this involves combining the training data with the covariance function and the mean function.
- ▶ Parametric is a special case when conditional prediction can be summarized in a *fixed* number of parameters.
- ▶ Complexity of parametric model remains fixed regardless of the size of our training data set.

Making Predictions

- ▶ For non-parametrics prediction at new points \mathbf{f}_* is made by conditioning on \mathbf{f} in the joint distribution.
- ▶ In GPs this involves combining the training data with the covariance function and the mean function.
- ▶ Parametric is a special case when conditional prediction can be summarized in a *fixed* number of parameters.
- ▶ Complexity of parametric model remains fixed regardless of the size of our training data set.
- ▶ For a non-parametric model the required number of parameters grows with the size of the training data.

Covariance Functions and Mercer Kernels

- ▶ Mercer Kernels and Covariance Functions are similar.

Covariance Functions and Mercer Kernels

- ▶ Mercer Kernels and Covariance Functions are similar.
- ▶ the kernel perspective does not make a probabilistic interpretation of the covariance function.

Covariance Functions and Mercer Kernels

- ▶ Mercer Kernels and Covariance Functions are similar.
- ▶ the kernel perspective does not make a probabilistic interpretation of the covariance function.
- ▶ Algorithms can be simpler, but probabilistic interpretation is crucial for kernel parameter optimization.

Outline

Parametric Models are a Bottleneck

Constructing Covariance

GP Limitations

Kalman Filter

Constructing Covariance Functions

- ▶ Sum of two covariances is also a covariance function.

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}')$$

Constructing Covariance Functions

- ▶ Product of two covariances is also a covariance function.

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}')$$

Multiply by Deterministic Function

- ▶ If $f(\mathbf{x})$ is a Gaussian process.
- ▶ $g(\mathbf{x})$ is a deterministic function.
- ▶ $h(\mathbf{x}) = f(\mathbf{x})g(\mathbf{x})$
- ▶ Then

$$k_h(\mathbf{x}, \mathbf{x}') = g(\mathbf{x})k_f(\mathbf{x}, \mathbf{x}')g(\mathbf{x}')$$

where k_h is covariance for $h(\cdot)$ and k_f is covariance for $f(\cdot)$.

Covariance Functions

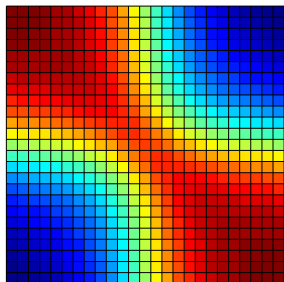
MLP Covariance Function

$$k(\mathbf{x}, \mathbf{x}') = \alpha \sin\left(\frac{w\mathbf{x}^\top \mathbf{x}' + b}{\sqrt{w\mathbf{x}^\top \mathbf{x} + b + 1} \sqrt{w\mathbf{x}'^\top \mathbf{x}' + b + 1}}\right)$$

- ▶ Based on infinite neural network model.

$$w = 40$$

$$b = 4$$



Covariance Functions

MLP Covariance Function

$$k(\mathbf{x}, \mathbf{x}') = \alpha \sin\left(\frac{w\mathbf{x}^\top \mathbf{x}' + b}{\sqrt{w\mathbf{x}^\top \mathbf{x} + b + 1} \sqrt{w\mathbf{x}'^\top \mathbf{x}' + b + 1}}\right)$$

- ▶ Based on infinite neural network model.

$$w = 40$$

$$b = 4$$

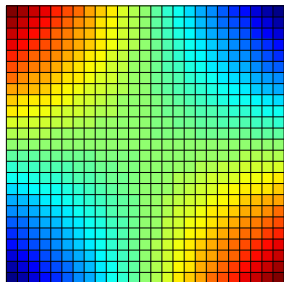
Covariance Functions

Linear Covariance Function

$$k(\mathbf{x}, \mathbf{x}') = \alpha \mathbf{x}^\top \mathbf{x}'$$

- ▶ Bayesian linear regression.

$$\alpha = 1$$



Covariance Functions

Linear Covariance Function

$$k(\mathbf{x}, \mathbf{x}') = \alpha \mathbf{x}^\top \mathbf{x}'$$

- ▶ Bayesian linear regression.

$$\alpha = 1$$

Gaussian Process Interpolation

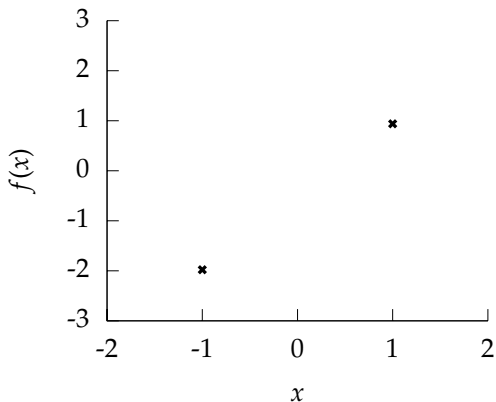


Figure : Real example: BACCO (see *e.g.* (Oakley and O'Hagan, 2002)). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

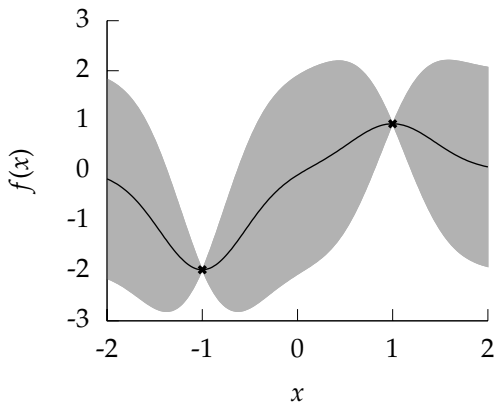


Figure : Real example: BACCO (see *e.g.* (Oakley and O'Hagan, 2002)). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

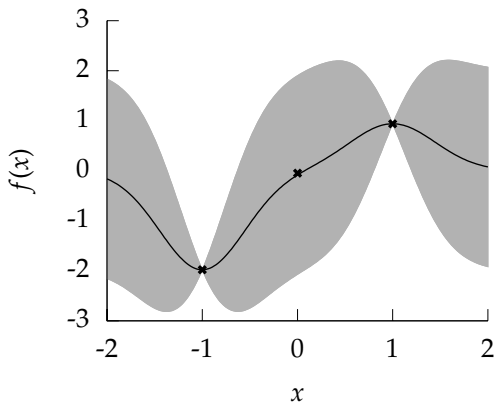


Figure : Real example: BACCO (see *e.g.* (Oakley and O'Hagan, 2002)). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

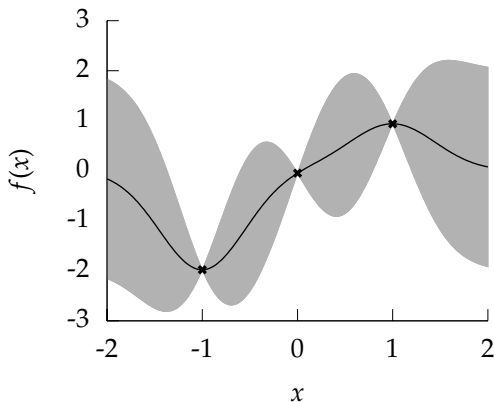


Figure : Real example: BACCO (see *e.g.* (Oakley and O'Hagan, 2002)). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

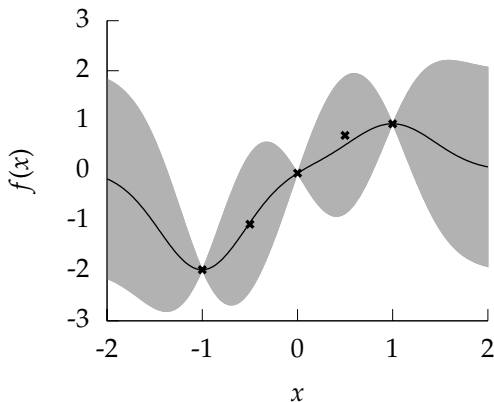


Figure : Real example: BACCO (see *e.g.* (Oakley and O'Hagan, 2002)). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

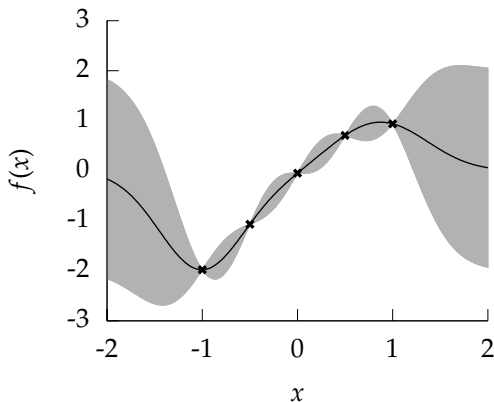


Figure : Real example: BACCO (see *e.g.* (Oakley and O'Hagan, 2002)). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

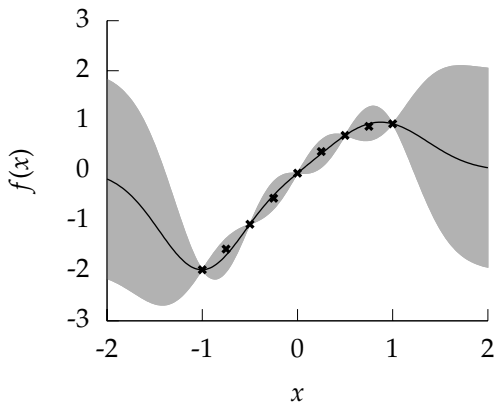


Figure : Real example: BACCO (see *e.g.* (Oakley and O'Hagan, 2002)). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

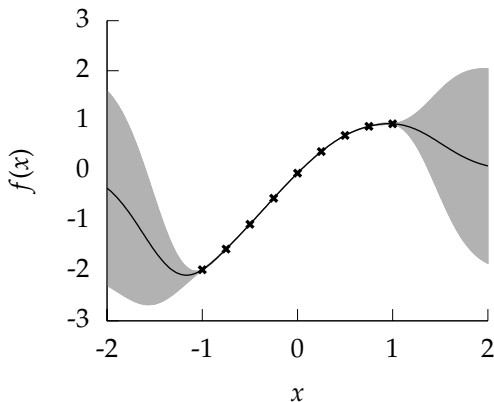


Figure : Real example: BACCO (see *e.g.* (Oakley and O'Hagan, 2002)). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Noise

- ▶ Gaussian noise model,

$$p(y_i|f_i) = \mathcal{N}(y_i|f_i, \sigma^2)$$

where σ^2 is the variance of the noise.

- ▶ Equivalent to a covariance function of the form

$$k(\mathbf{x}_i, \mathbf{x}_j) = \delta_{i,j}\sigma^2$$

where $\delta_{i,j}$ is the Kronecker delta function.

- ▶ Additive nature of Gaussians means we can simply add this term to existing covariance matrices.

Gaussian Process Regression

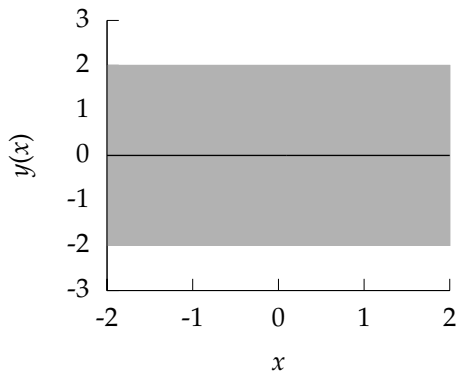


Figure : Examples include WiFi localization, C14 calibration curve.

Gaussian Process Regression

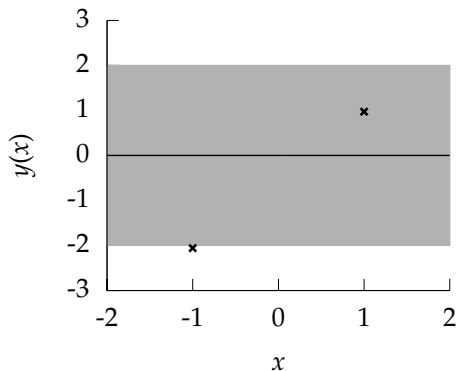


Figure : Examples include WiFi localization, C14 calibration curve.

Gaussian Process Regression

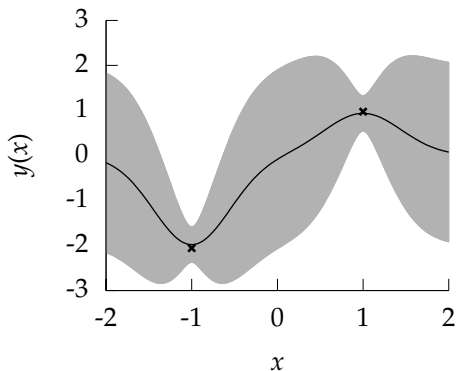


Figure : Examples include WiFi localization, C14 calibration curve.

Gaussian Process Regression

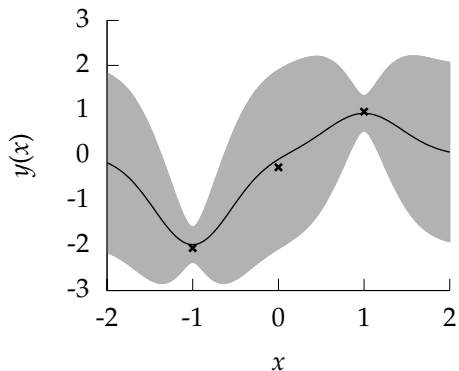


Figure : Examples include WiFi localization, C14 calibration curve.

Gaussian Process Regression

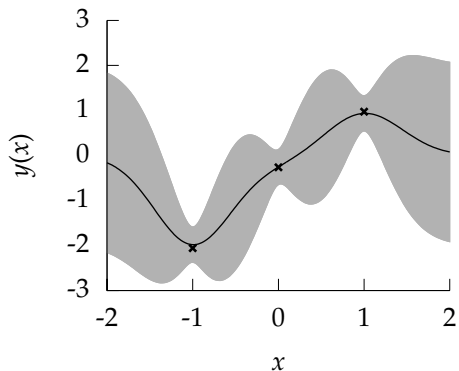


Figure : Examples include WiFi localization, C14 calibration curve.

Gaussian Process Regression

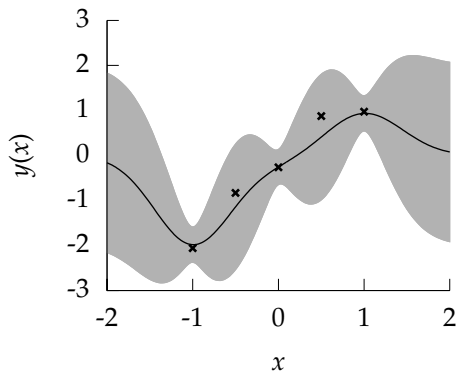


Figure : Examples include WiFi localization, C14 calibration curve.

Gaussian Process Regression

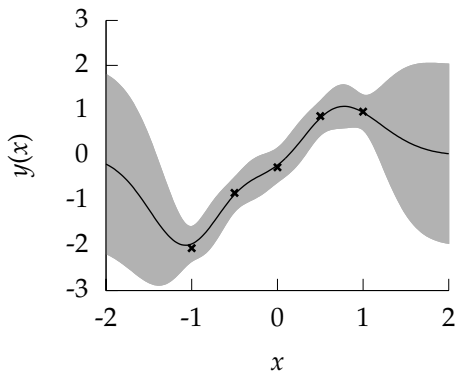


Figure : Examples include WiFi localization, C14 calibration curve.

Gaussian Process Regression

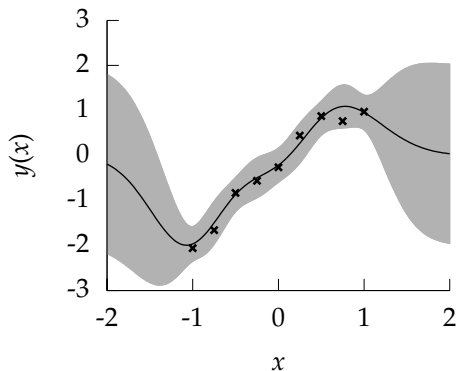


Figure : Examples include WiFi localization, C14 calibration curve.

Gaussian Process Regression

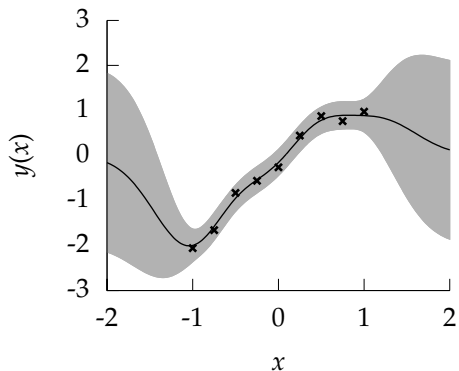


Figure : Examples include WiFi localization, C14 calibration curve.

General Noise Models

Graph of a GP

- ▶ Relates input variables, \mathbf{X} , to vector, \mathbf{y} , through \mathbf{f} given kernel parameters θ .
- ▶ Plate notation indicates independence of $y_i|f_i$.
- ▶ In general $p(y_i|f_i)$ is non-Gaussian.
- ▶ We approximate with Gaussian
 $p(y_i|f_i) \approx \mathcal{N}(m_i|f_i, \beta_i^{-1})$.

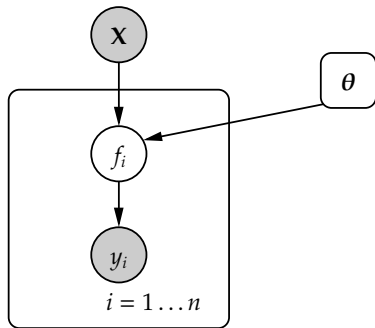


Figure : The Gaussian process depicted graphically.

Gaussian Noise

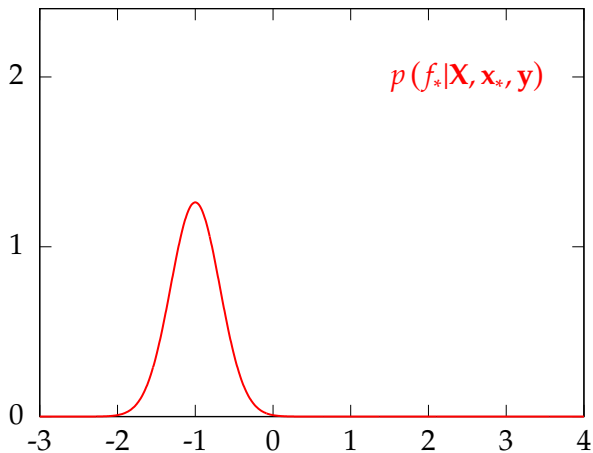


Figure : Inclusion of a data point with Gaussian noise.

Gaussian Noise

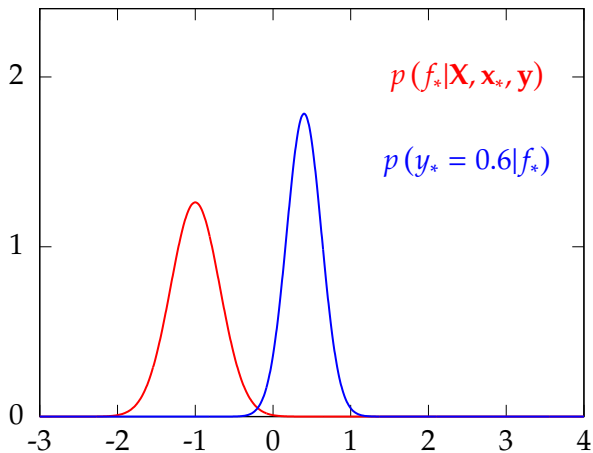


Figure : Inclusion of a data point with Gaussian noise.

Gaussian Noise

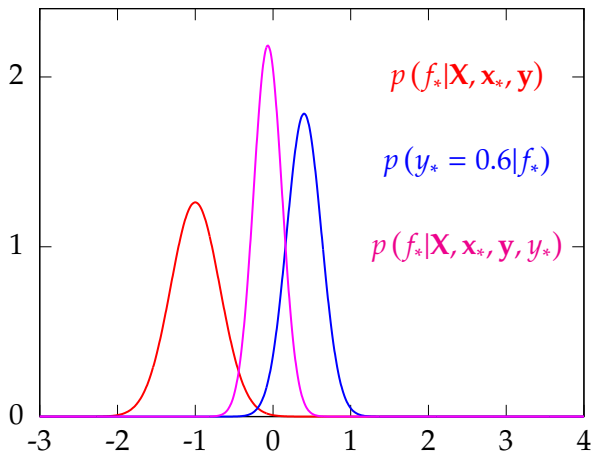


Figure : Inclusion of a data point with Gaussian noise.

Expectation Propagation

Local Moment Matching

- ▶ Easiest to consider a single previously unseen data point, y_* , \mathbf{x}_* .
- ▶ Before seeing data point, prediction of f_* is a GP, $q(f_*|\mathbf{y}, \mathbf{X})$.
- ▶ Update prediction using Bayes' Rule,

$$p(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*) = \frac{p(y_*|f_*)p(f_*|\mathbf{y}, \mathbf{X}, \mathbf{x}_*)}{p(\mathbf{y}, y_*|\mathbf{X}, \mathbf{x}_*)}.$$

This posterior is not a Gaussian process if $p(y_*|f_*)$ is non-Gaussian.

Classification Noise Model

Probit Noise Model

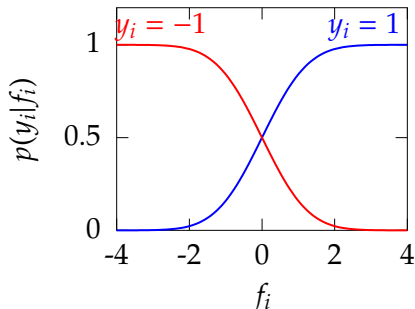


Figure : The probit model (classification). The plot shows $p(y_i|f_i)$ for different values of y_i . For $y_i = 1$ we have

$$p(y_i|f_i) = \Phi(f_i) = \int_{-\infty}^{f_i} \mathcal{N}(z|0, 1) dz.$$

Expectation Propagation II

Match Moments

- ▶ Idea behind EP — approximate with a Gaussian process at this stage by matching moments.
- ▶ This is equivalent to minimizing the following KL divergence where $q(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*)$ is constrained to be a GP.

$$q(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*) = \operatorname{argmin}_{q(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*)} \operatorname{KL}(p(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*) \| q(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*))$$

- ▶ This is equivalent to setting

$$\langle f_* \rangle_{q(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*)} = \langle f_* \rangle_{p(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*)}$$

$$\langle f_*^2 \rangle_{q(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*)} = \langle f_*^2 \rangle_{p(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*)}$$

Expectation Propagation III

Equivalent Gaussian

- ▶ This is achieved by replacing $p(y_*|f_*)$ with a Gaussian distribution

$$p(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*) = \frac{p(y_*|f_*)p(f_*|\mathbf{y}, \mathbf{X}, \mathbf{x}_*)}{p(\mathbf{y}, y_*|\mathbf{X}, \mathbf{x}_*)}$$

becomes

$$q(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*) = \frac{\mathcal{N}(m_*|f_*, \beta_m^{-1})p(f_*|\mathbf{y}, \mathbf{X}, \mathbf{x}_*)}{p(\mathbf{y}, y_*|\mathbf{X}, \mathbf{x}_*)}.$$

Classification

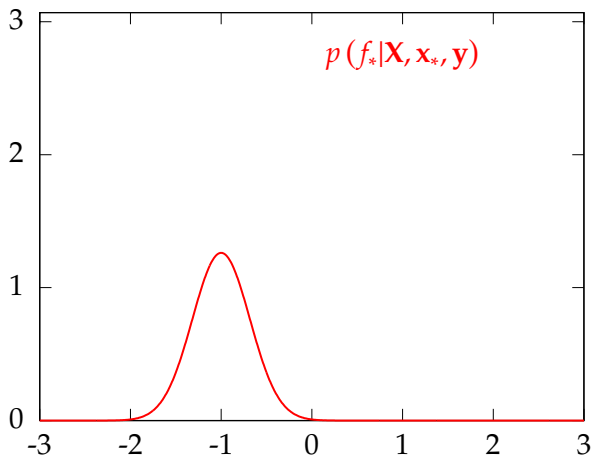


Figure : An EP style update with a classification noise model.

Classification

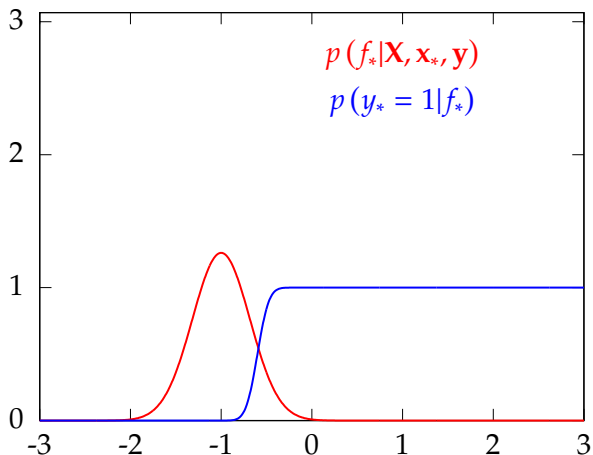


Figure : An EP style update with a classification noise model.

Classification

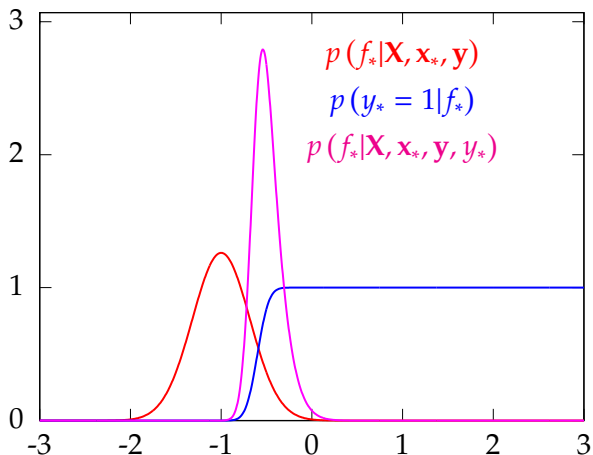


Figure : An EP style update with a classification noise model.

Classification

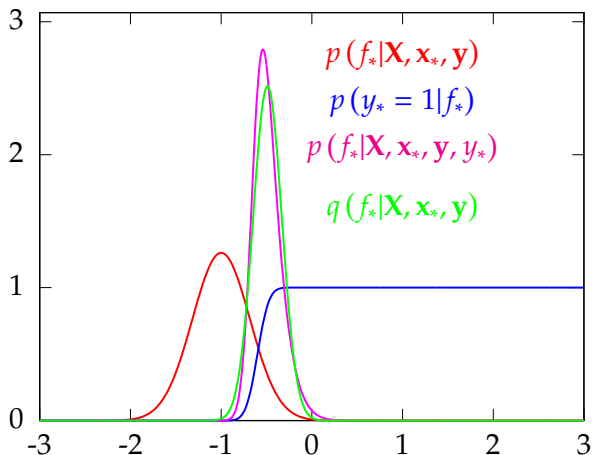


Figure : An EP style update with a classification noise model.

Ordinal Noise Model

Ordered Categories

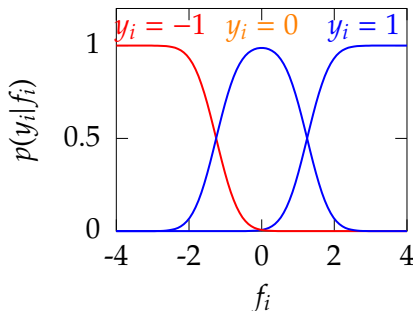


Figure : The ordered categorical noise model (ordinal regression). The plot shows $p(y_i|f_i)$ for different values of y_i . Here we have assumed three categories.

Ordinal Regression

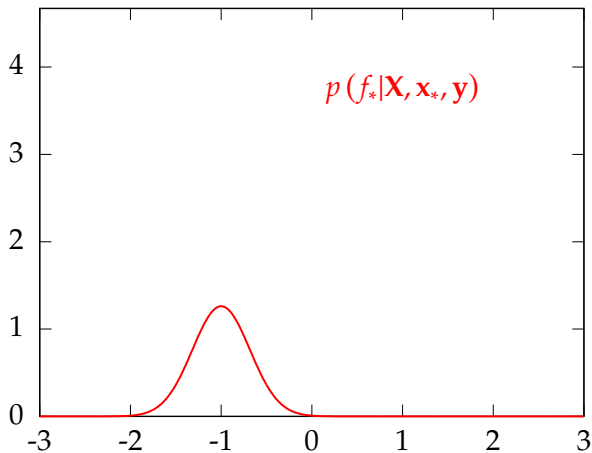


Figure : An EP style update with an ordered category noise model.

Ordinal Regression

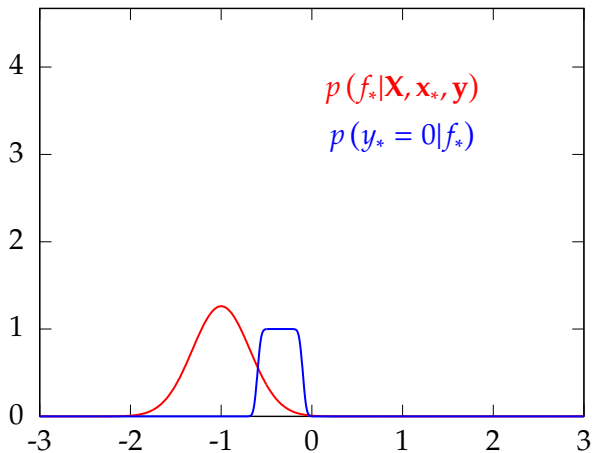


Figure : An EP style update with an ordered category noise model.

Ordinal Regression

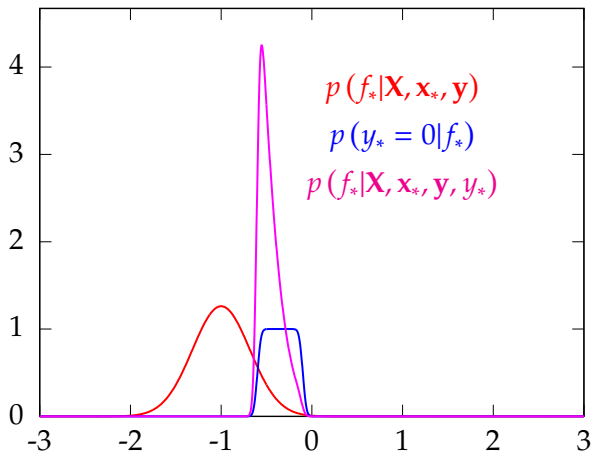


Figure : An EP style update with an ordered category noise model.

Ordinal Regression

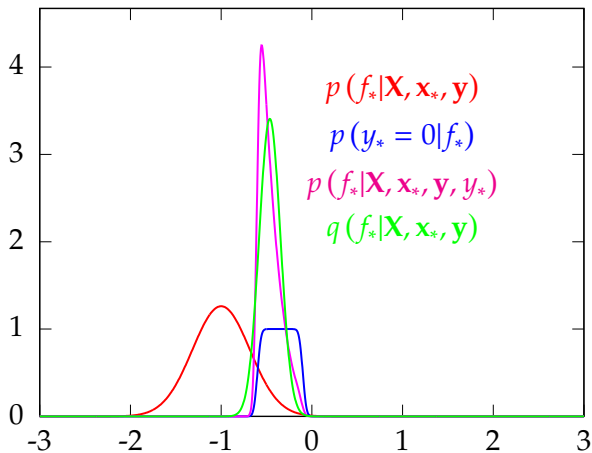


Figure : An EP style update with an ordered category noise model.

Laplace Approximation

- ▶ Equivalent Gaussian is found by making a local 2nd order Taylor approximation at the mode.
- ▶ Laplace was the first to suggest this¹, so it's known as the *Laplace approximation*.

Learning Covariance Parameters

Can we determine covariance parameters from the data?

$$\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{K}|} \exp\left(-\frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}\right)$$

The parameters are *inside* the covariance function (matrix).

$$k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j; \theta)$$

Learning Covariance Parameters

Can we determine covariance parameters from the data?

$$\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}) = \frac{1}{(2\pi)^{\frac{n}{2}} |\mathbf{K}|} \exp\left(-\frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}\right)$$

The parameters are *inside* the covariance function (matrix).

$$k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j; \theta)$$

Learning Covariance Parameters

Can we determine covariance parameters from the data?

$$\log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}) = -\frac{1}{2} \log |\mathbf{K}| - \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2} - \frac{n}{2} \log 2\pi$$

The parameters are *inside* the covariance function (matrix).

$$k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j; \theta)$$

Learning Covariance Parameters

Can we determine covariance parameters from the data?

$$E(\boldsymbol{\theta}) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

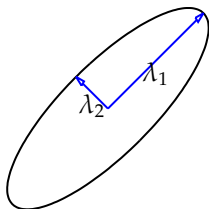
The parameters are *inside* the covariance function (matrix).

$$k_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j; \boldsymbol{\theta})$$

Eigendecomposition of Covariance

A useful decomposition for understanding the objective function.

$$\mathbf{K} = \mathbf{R}\mathbf{\Lambda}^2\mathbf{R}^\top$$



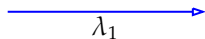
Diagonal of $\mathbf{\Lambda}$ represents distance along axes.

\mathbf{R} gives a rotation of these axes.

where $\mathbf{\Lambda}$ is a *diagonal* matrix and $\mathbf{R}^\top\mathbf{R} = \mathbf{I}$.

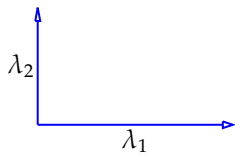
Capacity control: $\log |\mathbf{K}|$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



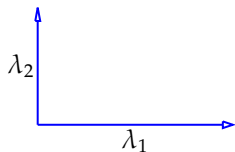
Capacity control: $\log |\mathbf{K}|$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



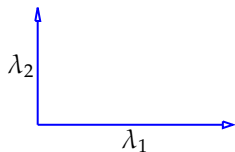
Capacity control: $\log |\mathbf{K}|$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



Capacity control: $\log |\mathbf{K}|$

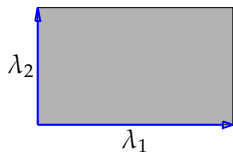
$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



$$|\mathbf{\Lambda}| = \lambda_1 \lambda_2$$

Capacity control: $\log |\mathbf{K}|$

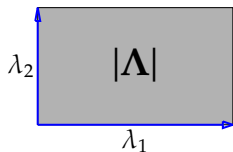
$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



$$|\mathbf{\Lambda}| = \lambda_1 \lambda_2$$

Capacity control: $\log |\mathbf{K}|$

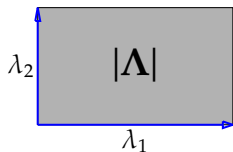
$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



$$|\mathbf{\Lambda}| = \lambda_1 \lambda_2$$

Capacity control: $\log |\mathbf{K}|$

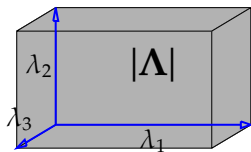
$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$



$$|\mathbf{\Lambda}| = \lambda_1 \lambda_2$$

Capacity control: $\log |\mathbf{K}|$

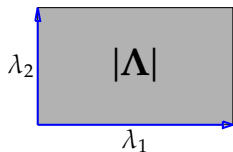
$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$



$$|\mathbf{\Lambda}| = \lambda_1 \lambda_2 \lambda_3$$

Capacity control: $\log |\mathbf{K}|$

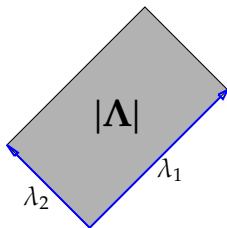
$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$



$$|\mathbf{\Lambda}| = \lambda_1 \lambda_2$$

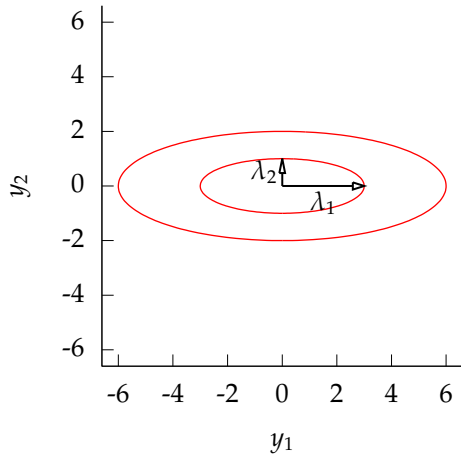
Capacity control: $\log |\mathbf{K}|$

$$\mathbf{R}\mathbf{\Lambda} = \begin{bmatrix} w_{1,1} & w_{1,2} \\ w_{2,1} & w_{2,2} \end{bmatrix}$$

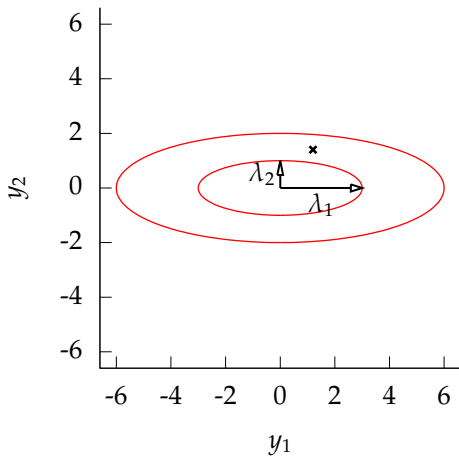


$$|\mathbf{R}\mathbf{\Lambda}| = \lambda_1\lambda_2$$

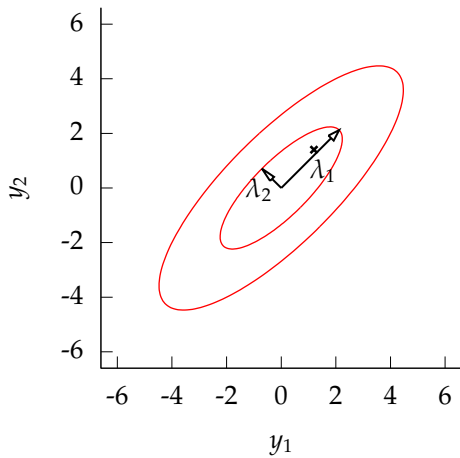
Data Fit: $\frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$



Data Fit: $\frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$

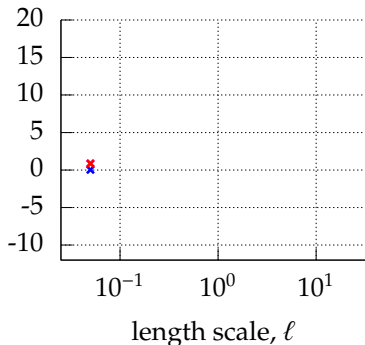
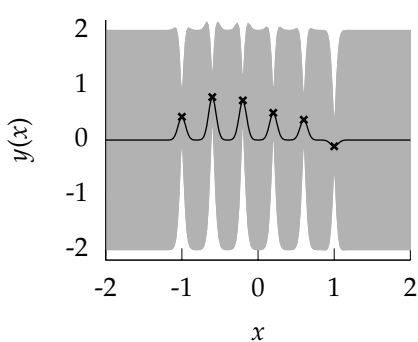


Data Fit: $\frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$



Learning Covariance Parameters

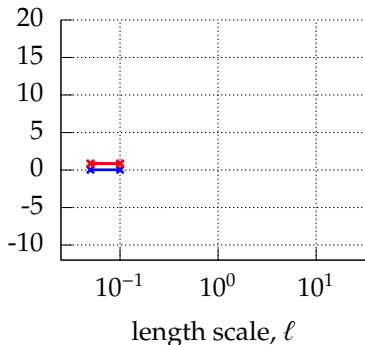
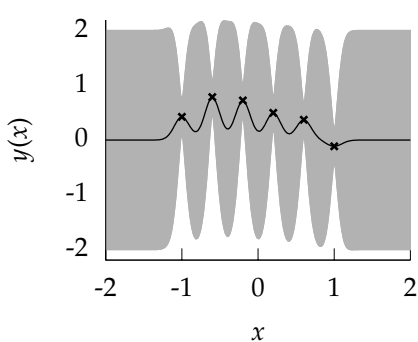
Can we determine length scales and noise levels from the data?



$$E(\theta) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

Learning Covariance Parameters

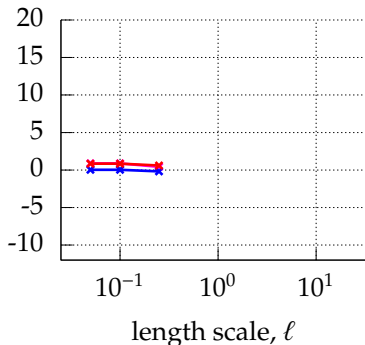
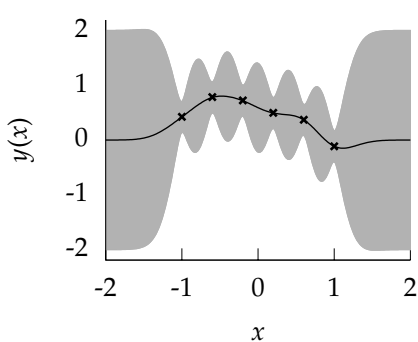
Can we determine length scales and noise levels from the data?



$$E(\theta) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

Learning Covariance Parameters

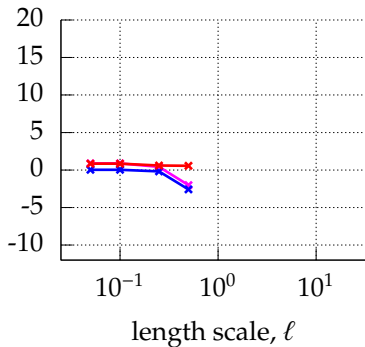
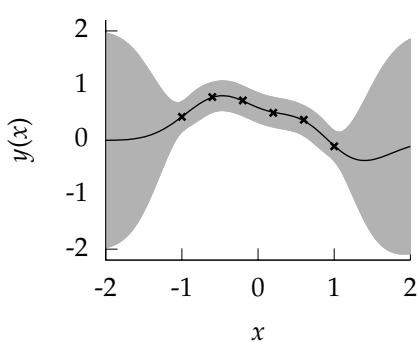
Can we determine length scales and noise levels from the data?



$$E(\theta) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

Learning Covariance Parameters

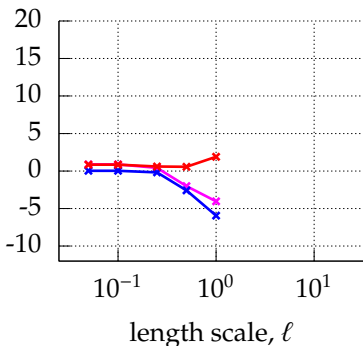
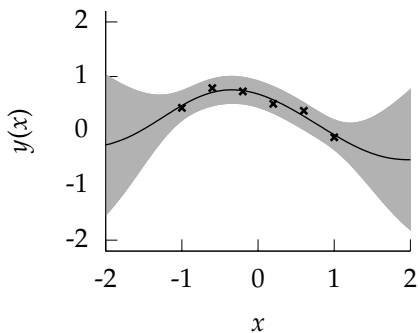
Can we determine length scales and noise levels from the data?



$$E(\theta) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

Learning Covariance Parameters

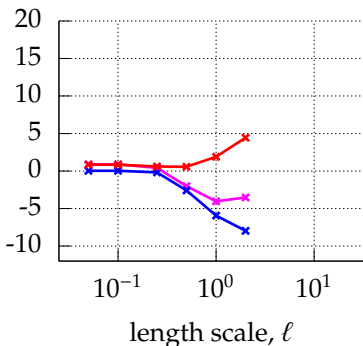
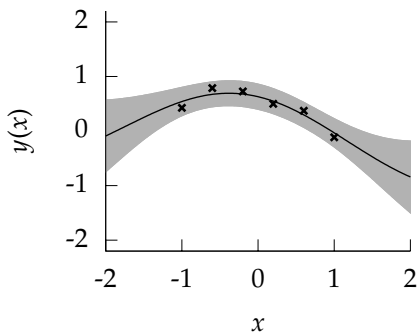
Can we determine length scales and noise levels from the data?



$$E(\theta) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

Learning Covariance Parameters

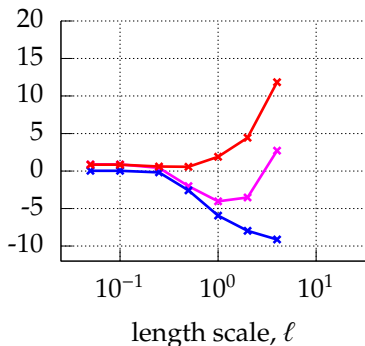
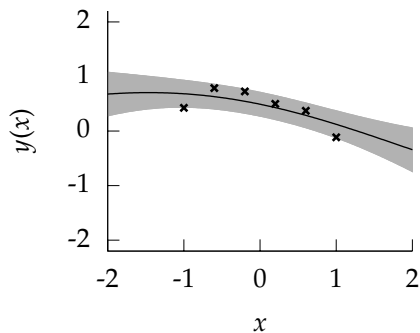
Can we determine length scales and noise levels from the data?



$$E(\theta) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

Learning Covariance Parameters

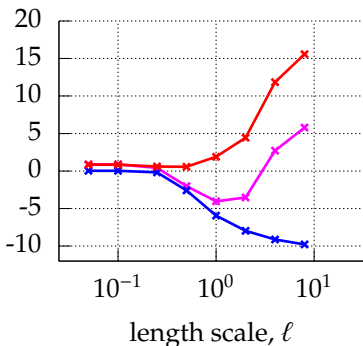
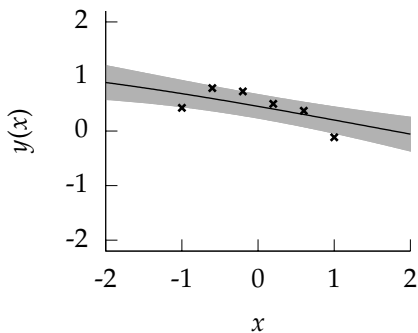
Can we determine length scales and noise levels from the data?



$$E(\theta) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

Learning Covariance Parameters

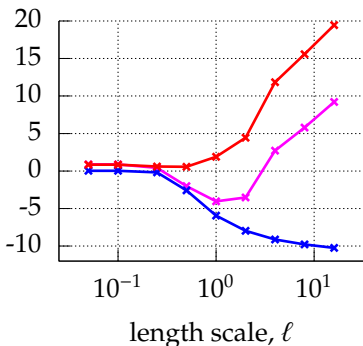
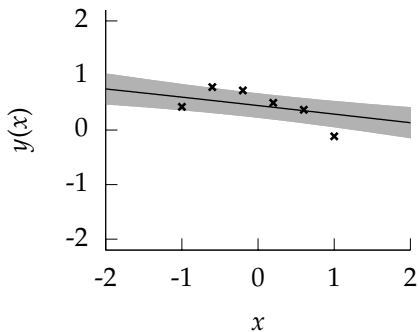
Can we determine length scales and noise levels from the data?



$$E(\theta) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

Learning Covariance Parameters

Can we determine length scales and noise levels from the data?



$$E(\theta) = \frac{1}{2} \log |\mathbf{K}| + \frac{\mathbf{y}^\top \mathbf{K}^{-1} \mathbf{y}}{2}$$

Gene Expression Example

- ▶ Given given expression levels in the form of a time series from Della Gatta et al. (2008).
- ▶ Want to detect if a gene is expressed or not, fit a GP to each gene (Kalaitzis and Lawrence, 2011).

RESEARCH ARTICLE

Open Access

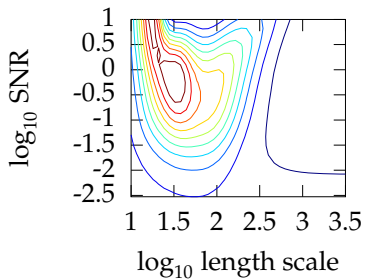
A Simple Approach to Ranking Differentially Expressed Gene Expression Time Courses through Gaussian Process Regression

Alfredo A Kalaitzis^{*} and Neil D Lawrence^{*}

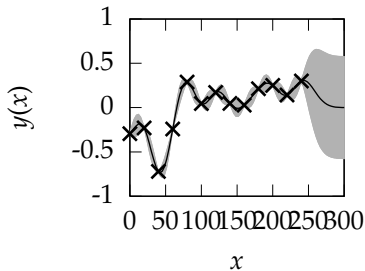
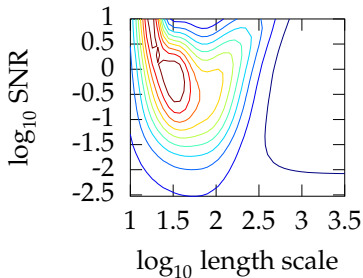
Abstract

Background: The analysis of gene expression from time series underpins many biological studies. Two basic forms of analysis recur for data of this type: removing inactive (quiet) genes from the study and determining which genes are differentially expressed. Often these analysis stages are applied disregarding the fact that the data is drawn from a time series. In this paper we propose a simple model for accounting for the underlying temporal nature of the data based on a Gaussian process.

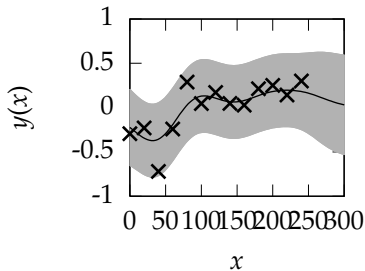
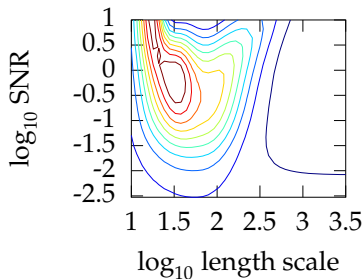
Results: We review Gaussian process (GP) regression for estimating the continuous trajectories underlying in gene expression time-series. We present a simple approach which can be used to filter quiet genes, or for the case of time series in the form of expression ratios, quantify differential expression. We assess via ROC curves the rankings produced by our regression framework and compare them to a recently proposed hierarchical Bayesian model for the analysis of gene expression time-series (BATS). We compare on both simulated and experimental data showing that the proposed approach considerably outperforms the current state of the art.



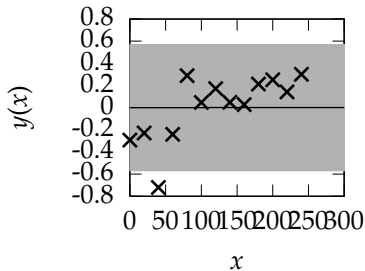
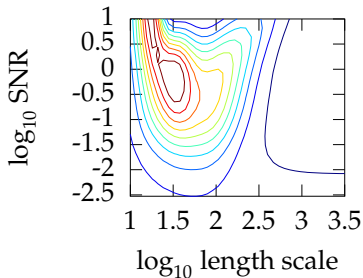
Contour plot of Gaussian process likelihood.



Optima: length scale of 1.2221 and \log_{10} SNR of 1.9654
 log likelihood is -0.22317.



Optima: length scale of 1.5162 and \log_{10} SNR of 0.21306
likelihood is -0.23604.



Optima: length scale of 2.9886 and \log_{10} SNR of -4.506
 log likelihood is -2.1056.

Outline

Parametric Models are a Bottleneck

Constructing Covariance

GP Limitations

Kalman Filter

Limitations of Gaussian Processes

- ▶ Inference is $O(n^3)$ due to matrix inverse (in practice use Cholesky).
- ▶ Gaussian processes don't deal well with discontinuities (financial crises, phosphorylation, collisions, edges in images).
- ▶ Widely used exponentiated quadratic covariance (RBF) can be too smooth in practice (but there are many alternatives!!).

Outline

Parametric Models are a Bottleneck

Constructing Covariance

GP Limitations

Kalman Filter

Simple Markov Chain

- ▶ Assume 1-d latent state, a vector over time, $\mathbf{x} = [x_1 \dots x_T]$.
- ▶ Markov property,

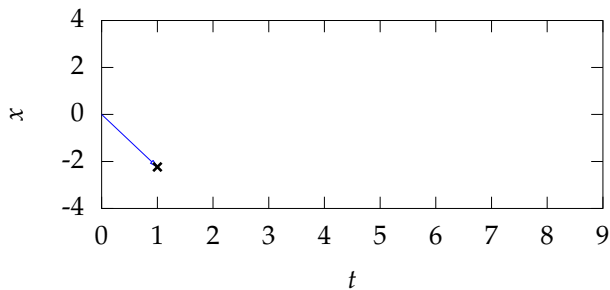
$$\begin{aligned}x_i &= x_{i-1} + \epsilon_i, \\ \epsilon_i &\sim \mathcal{N}(0, \alpha) \\ \implies x_i &\sim \mathcal{N}(x_{i-1}, \alpha)\end{aligned}$$

- ▶ Initial state,

$$x_0 \sim \mathcal{N}(0, \alpha_0)$$

- ▶ If $x_0 \sim \mathcal{N}(0, \alpha)$ we have a Markov chain for the latent states.
- ▶ Markov chain it is specified by an initial distribution (Gaussian) and a transition distribution (Gaussian).

Gauss Markov Chain

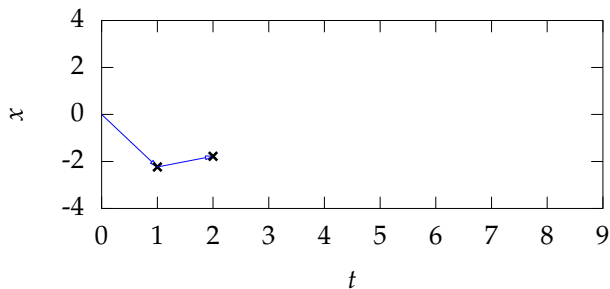


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

$$x_0 = 0.000, \quad \epsilon_1 = -2.24$$

$$x_1 = 0.000 - 2.24 = -2.24$$

Gauss Markov Chain

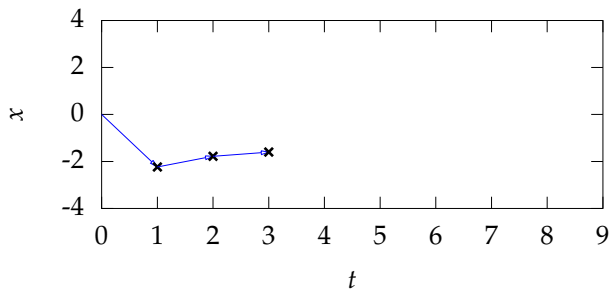


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

$$x_1 = -2.24, \quad \epsilon_2 = 0.457$$

$$x_2 = -2.24 + 0.457 = -1.78$$

Gauss Markov Chain

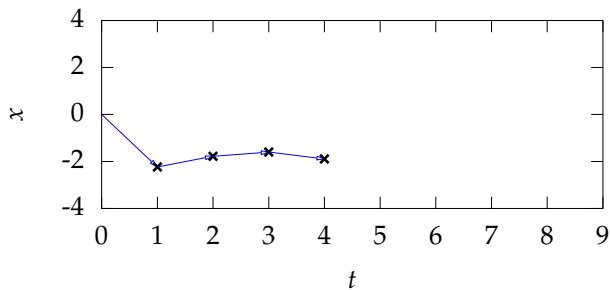


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

$$x_2 = -1.78, \quad \epsilon_3 = 0.178$$

$$x_3 = -1.78 + 0.178 = -1.6$$

Gauss Markov Chain

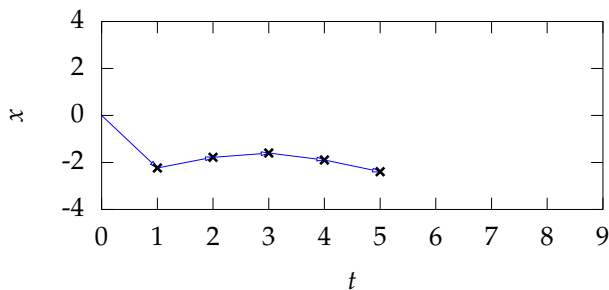


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

$$x_3 = -1.6, \quad \epsilon_4 = -0.292$$

$$x_4 = -1.6 - 0.292 = -1.89$$

Gauss Markov Chain

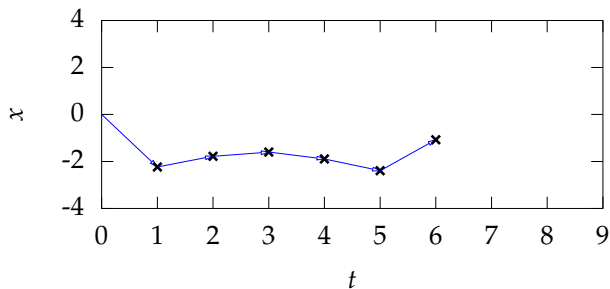


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

$$x_4 = -1.89, \quad \epsilon_5 = -0.501$$

$$x_5 = -1.89 - 0.501 = -2.39$$

Gauss Markov Chain

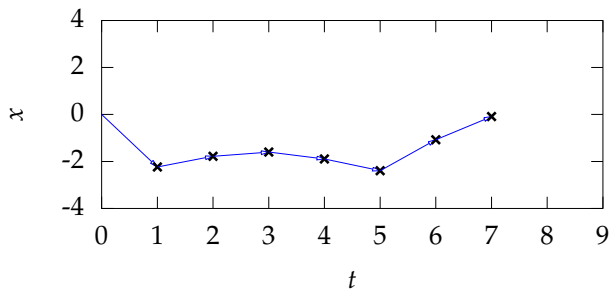


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

$$x_5 = -2.39, \quad \epsilon_6 = 1.32$$

$$x_6 = -2.39 + 1.32 = -1.08$$

Gauss Markov Chain

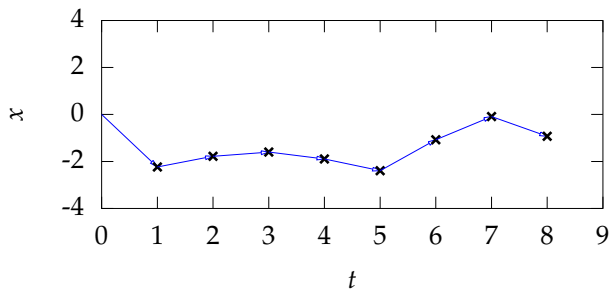


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

$$x_6 = -1.08, \quad \epsilon_7 = 0.989$$

$$x_7 = -1.08 + 0.989 = -0.0881$$

Gauss Markov Chain

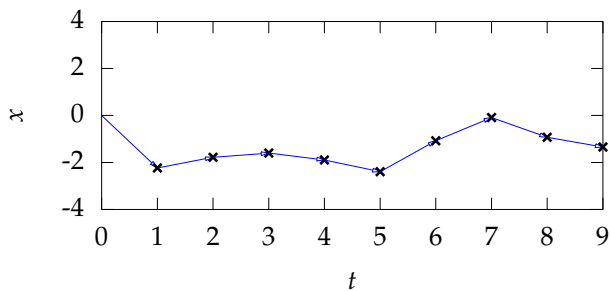


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

$$x_7 = -0.0881, \quad \epsilon_8 = -0.842$$

$$x_8 = -0.0881 - 0.842 = -0.93$$

Gauss Markov Chain



$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0, 1)$$

$$x_8 = -0.93, \quad \epsilon_9 = -0.41$$

$$x_9 = -0.93 - 0.410 = -1.34$$

Multivariate Gaussian Properties: Reminder

If

$$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{C})$$

and

$$\mathbf{x} = \mathbf{W}\mathbf{z} + \mathbf{b}$$

then

$$\mathbf{x} \sim \mathcal{N}(\mathbf{W}\boldsymbol{\mu} + \mathbf{b}, \mathbf{W}\mathbf{C}\mathbf{W}^\top)$$

Multivariate Gaussian Properties: Reminder

Simplified: If

$$\mathbf{z} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

and

$$\mathbf{x} = \mathbf{W}\mathbf{z}$$

then

$$\mathbf{x} \sim \mathcal{N}(0, \sigma^2 \mathbf{W}\mathbf{W}^\top)$$

Matrix Representation of Latent Variables

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \end{bmatrix}$$

$$x_1 = \epsilon_1$$

Matrix Representation of Latent Variables

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \end{bmatrix}$$

$$x_2 = \epsilon_1 + \epsilon_2$$

Matrix Representation of Latent Variables

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \end{bmatrix}$$

$$x_3 = \epsilon_1 + \epsilon_2 + \epsilon_3$$

Matrix Representation of Latent Variables

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \end{bmatrix}$$

$$x_4 = \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4$$

Matrix Representation of Latent Variables

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \end{bmatrix}$$

$$x_5 = \epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 + \epsilon_5$$

Matrix Representation of Latent Variables

$$\mathbf{x} = \mathbf{L}_1 \times \boldsymbol{\epsilon}$$

Multivariate Process

- ▶ Since \mathbf{x} is linearly related to ϵ we know \mathbf{x} is a Gaussian process.
- ▶ Trick: we only need to compute the mean and covariance of \mathbf{x} to determine that Gaussian.

$$\mathbf{x} = \mathbf{L}_1 \boldsymbol{\epsilon}$$

$$\langle \mathbf{x} \rangle = \langle \mathbf{L}_1 \boldsymbol{\epsilon} \rangle$$

$$\langle \mathbf{x} \rangle = \mathbf{L}_1 \langle \boldsymbol{\epsilon} \rangle$$

$$\langle \mathbf{x} \rangle = \mathbf{L}_1 \langle \boldsymbol{\epsilon} \rangle$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \alpha \mathbf{I})$$

Latent Process Mean

$$\langle \mathbf{x} \rangle = \mathbf{L}_1 \mathbf{0}$$

Latent Process Mean

$$\langle \mathbf{x} \rangle = \mathbf{0}$$

Latent Process Covariance

$$\mathbf{xx}^\top = \mathbf{L}_1 \boldsymbol{\epsilon} \boldsymbol{\epsilon}^\top \mathbf{L}_1^\top$$

$$\mathbf{x}^\top = \boldsymbol{\epsilon}^\top \mathbf{L}^\top$$

Latent Process Covariance

$$\langle \mathbf{x}\mathbf{x}^\top \rangle = \langle \mathbf{L}_1 \boldsymbol{\epsilon} \boldsymbol{\epsilon}^\top \mathbf{L}_1^\top \rangle$$

Latent Process Covariance

$$\langle \mathbf{x}\mathbf{x}^T \rangle = \mathbf{L}_1 \langle \boldsymbol{\epsilon}\boldsymbol{\epsilon}^T \rangle \mathbf{L}_1^T$$

Latent Process Covariance

$$\langle \mathbf{x}\mathbf{x}^\top \rangle = \mathbf{L}_1 \langle \boldsymbol{\epsilon}\boldsymbol{\epsilon}^\top \rangle \mathbf{L}_1^\top$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \alpha \mathbf{I})$$

Latent Process Covariance

$$\langle \mathbf{x}\mathbf{x}^T \rangle = \alpha \mathbf{L}_1 \mathbf{L}_1^T$$

$$\mathbf{x} = \mathbf{L}_1 \boldsymbol{\epsilon}$$

$$\mathbf{x} = \mathbf{L}_1 \boldsymbol{\epsilon}$$
$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \alpha \mathbf{I})$$

$$\mathbf{x} = \mathbf{L}_1 \boldsymbol{\epsilon}$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \alpha \mathbf{I})$$



$$\mathbf{x} = \mathbf{L}_1 \boldsymbol{\epsilon}$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \alpha \mathbf{I})$$

\implies

$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \alpha \mathbf{L}_1 \mathbf{L}_1^\top)$$

Covariance for Latent Process II

- ▶ Make the variance dependent on time interval.
- ▶ Assume variance grows *linearly* with time.
- ▶ Justification: sum of two Gaussian distributed random variables is distributed as Gaussian with sum of variances.
- ▶ If variable's movement is additive over time (as described) variance scales linearly with time.

Covariance for Latent Process II

- ▶ Given

$$\epsilon \sim \mathcal{N}(\mathbf{0}, \alpha \mathbf{I}) \implies \epsilon \sim \mathcal{N}(\mathbf{0}, \alpha \mathbf{L}_1 \mathbf{L}_1^\top).$$

Then

$$\epsilon \sim \mathcal{N}(\mathbf{0}, \Delta t \alpha \mathbf{I}) \implies \epsilon \sim \mathcal{N}(\mathbf{0}, \Delta t \alpha \mathbf{L}_1 \mathbf{L}_1^\top).$$

where Δt is the time interval between observations.

Covariance for Latent Process II

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \alpha \Delta t \mathbf{I}), \quad \mathbf{x} \sim \mathcal{N}(0, \alpha \Delta t \mathbf{L}_1 \mathbf{L}_1^\top)$$

Covariance for Latent Process II

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \alpha\Delta t\mathbf{I}), \quad \mathbf{x} \sim \mathcal{N}(0, \alpha\Delta t\mathbf{L}_1\mathbf{L}_1^\top)$$

$$\mathbf{K} = \alpha\Delta t\mathbf{L}_1\mathbf{L}_1^\top$$

Covariance for Latent Process II

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \alpha \Delta t \mathbf{I}), \quad \mathbf{x} \sim \mathcal{N}(0, \alpha \Delta t \mathbf{L}_1 \mathbf{L}_1^\top)$$

$$\mathbf{K} = \alpha \Delta t \mathbf{L}_1 \mathbf{L}_1^\top$$

$$k_{i,j} = \alpha \Delta t \mathbf{l}_{:,i}^\top \mathbf{l}_{:,j}$$

where $\mathbf{l}_{:,k}$ is a vector from the k th row of \mathbf{L}_1 : the first k elements are one, the next $T - k$ are zero.

Covariance for Latent Process II

$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \alpha \Delta t \mathbf{I}), \quad \mathbf{x} \sim \mathcal{N}(0, \alpha \Delta t \mathbf{L}_1 \mathbf{L}_1^\top)$$

$$\mathbf{K} = \alpha \Delta t \mathbf{L}_1 \mathbf{L}_1^\top$$

$$k_{i,j} = \alpha \Delta t \mathbf{l}_{:,i}^\top \mathbf{l}_{:,j}$$

where $\mathbf{l}_{:,k}$ is a vector from the k th row of \mathbf{L}_1 : the first k elements are one, the next $T - k$ are zero.

$$k_{i,j} = \alpha \Delta t \min(i, j)$$

define $\Delta t_i = t_i$ so

$$k_{i,j} = \alpha \min(t_i, t_j) = k(t_i, t_j)$$

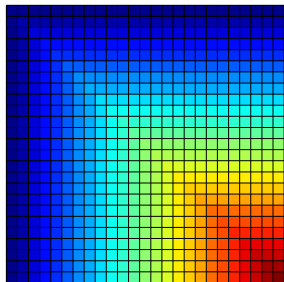
Covariance Functions

Where did this covariance matrix come from?

Markov Process

$$k(t, t') = \alpha \min(t, t')$$

- ▶ Covariance matrix is built using the *inputs* to the function t .



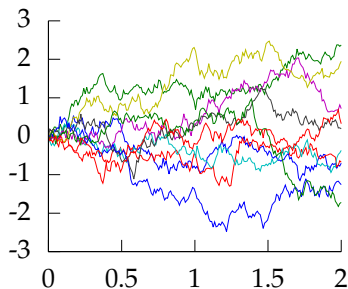
Covariance Functions

Where did this covariance matrix come from?

Markov Process

$$k(t, t') = \alpha \min(t, t')$$

- ▶ Covariance matrix is built using the *inputs* to the function t .



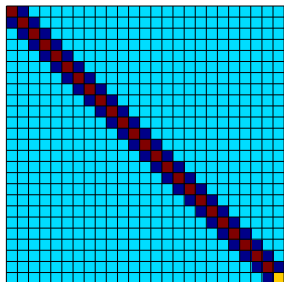
Covariance Functions

Where did this covariance matrix come from?

Markov Process

Visualization of inverse covariance (precision).

- ▶ Precision matrix is sparse: only neighbours in matrix are non-zero.
- ▶ This reflects *conditional* independencies in data.
- ▶ In this case *Markov* structure.



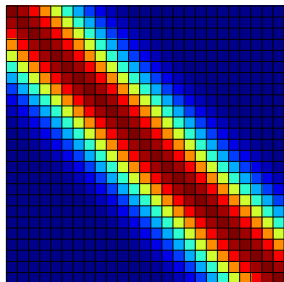
Covariance Functions

Where did this covariance matrix come from?

Exponentiated Quadratic Kernel Function (RBF, Squared Exponential, Gaussian)

$$k(\mathbf{x}, \mathbf{x}') = \alpha \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\ell^2}\right)$$

- ▶ Covariance matrix is built using the *inputs* to the function \mathbf{x} .
- ▶ For the example above it was based on Euclidean distance.
- ▶ The covariance function is also known as a kernel.



Covariance Functions

Where did this covariance matrix come from?

Exponentiated Quadratic Kernel Function (RBF, Squared Exponential, Gaussian)

$$k(\mathbf{x}, \mathbf{x}') = \alpha \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|_2^2}{2\ell^2}\right)$$

- ▶ Covariance matrix is built using the *inputs* to the function \mathbf{x} .
- ▶ For the example above it was based on Euclidean distance.
- ▶ The covariance function is also known as a kernel.

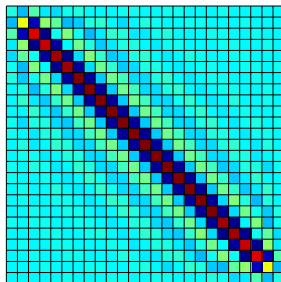
Covariance Functions

Where did this covariance matrix come from?

Exponentiated Quadratic

Visualization of inverse covariance (precision).

- ▶ Precision matrix is not sparse.
- ▶ Each point is dependent on all the others.
- ▶ In this case non-Markovian.



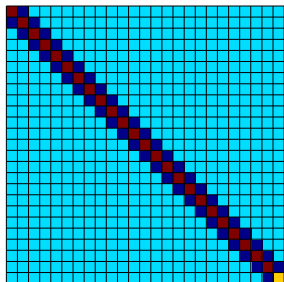
Covariance Functions

Where did this covariance matrix come from?

Markov Process

Visualization of inverse covariance (precision).

- ▶ Precision matrix is sparse: only neighbours in matrix are non-zero.
- ▶ This reflects *conditional* independencies in data.
- ▶ In this case *Markov* structure.



Simple Kalman Filter I

- ▶ We have state vector $\mathbf{X} = [\mathbf{x}_1 \dots \mathbf{x}_q] \in \mathbb{R}^{T \times q}$ and if each state evolves independently we have

$$p(\mathbf{X}) = \prod_{i=1}^q p(\mathbf{x}_{:,i})$$
$$p(\mathbf{x}_{:,i}) = \mathcal{N}(\mathbf{x}_{:,i} | \mathbf{0}, \mathbf{K}).$$

- ▶ We want to obtain outputs through:

$$\mathbf{y}_{i,:} = \mathbf{W}\mathbf{x}_{i,:}$$

Stacking and Kronecker Products I

- ▶ Represent with a 'stacked' system:

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I} \otimes \mathbf{K})$$

where the stacking is placing each column of \mathbf{X} one on top of another as

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{:,1} \\ \mathbf{x}_{:,2} \\ \vdots \\ \mathbf{x}_{:,q} \end{bmatrix}$$

Kronecker Product

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes \mathbf{K} = \begin{bmatrix} a\mathbf{K} & b\mathbf{K} \\ c\mathbf{K} & d\mathbf{K} \end{bmatrix}$$

Kronecker Product

$$\begin{bmatrix} \text{dark gray} & \text{medium gray} \\ \text{medium gray} & \text{white} \end{bmatrix} \otimes \begin{bmatrix} \text{red} & \text{green} \\ \text{green} & \text{blue} \end{bmatrix} = \begin{bmatrix} \text{dark red} & \text{dark green} & \text{red} & \text{green} \\ \text{dark green} & \text{dark blue} & \text{green} & \text{blue} \\ \text{red} & \text{green} & \text{red} & \text{green} \\ \text{green} & \text{blue} & \text{green} & \text{blue} \end{bmatrix}$$

The diagram illustrates the Kronecker product of two 2x2 matrices. The first matrix has four quadrants: dark gray (top-left), medium gray (top-right), medium gray (bottom-left), and white (bottom-right). The second matrix has four quadrants: red (top-left), green (top-right), green (bottom-left), and blue (bottom-right). The resulting 4x4 matrix is a 2x2 grid of 2x2 blocks. Each block is a scaled and color-shifted version of the second matrix. The top-left block is dark red, top-right is dark green, bottom-left is red, and bottom-right is green. The bottom-left and bottom-right blocks are identical to the second matrix.

Stacking and Kronecker Products I

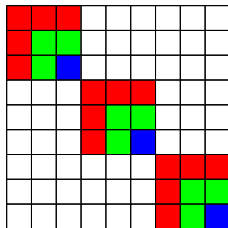
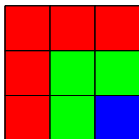
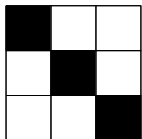
- ▶ Represent with a 'stacked' system:

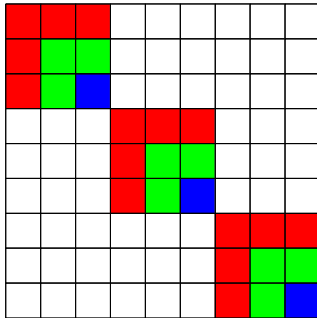
$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I} \otimes \mathbf{K})$$

where the stacking is placing each column of \mathbf{X} one on top of another as

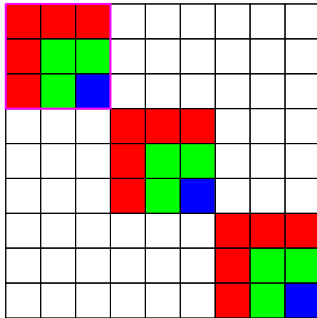
$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{:,1} \\ \mathbf{x}_{:,2} \\ \vdots \\ \mathbf{x}_{:,q} \end{bmatrix}$$

Column Stacking

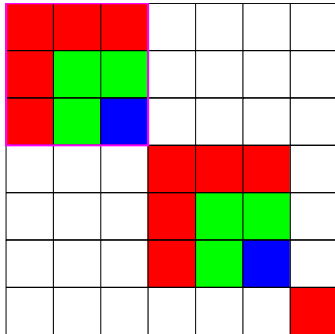




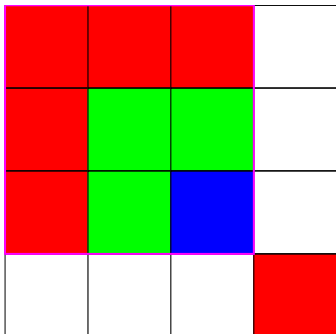
For this stacking the marginal distribution over *time* is given by the block diagonals.



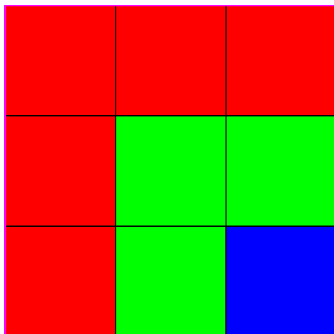
For this stacking the marginal distribution over *time* is given by the block diagonals.



For this stacking the marginal distribution over *time* is given by the block diagonals.



For this stacking the marginal distribution over *time* is given by the block diagonals.



For this stacking the marginal distribution over *time* is given by the block diagonals.

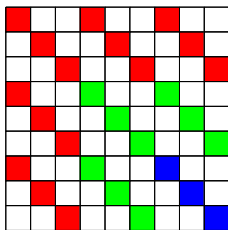
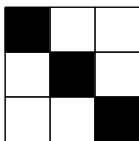
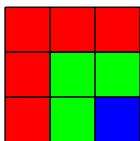
Two Ways of Stacking

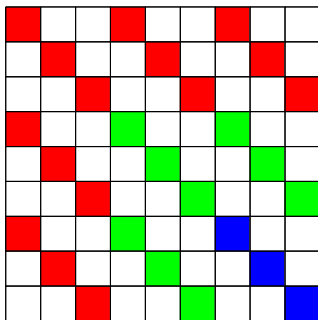
Can also stack each row of \mathbf{X} to form column vector:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{1,:} \\ \mathbf{x}_{2,:} \\ \vdots \\ \mathbf{x}_{T,:} \end{bmatrix}$$

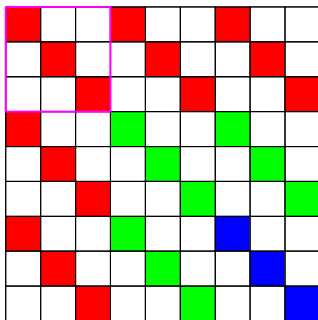
$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{K} \otimes \mathbf{I})$$

Row Stacking

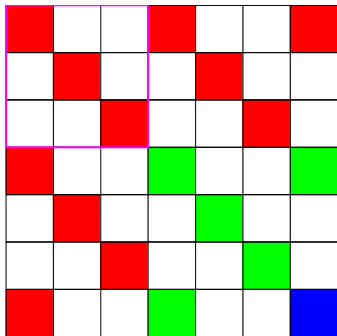




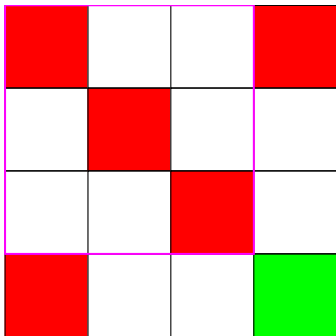
For this stacking the marginal distribution over the latent *dimensions* is given by the block diagonals.



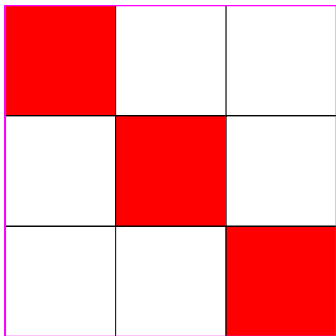
For this stacking the marginal distribution over the latent *dimensions* is given by the block diagonals.



For this stacking the marginal distribution over the latent *dimensions* is given by the block diagonals.



For this stacking the marginal distribution over the latent *dimensions* is given by the block diagonals.



For this stacking the marginal distribution over the latent *dimensions* is given by the block diagonals.

Observed Process

The observations are related to the latent points by a linear mapping matrix,

$$\mathbf{y}_{i,:} = \mathbf{W}\mathbf{x}_{i,:} + \boldsymbol{\epsilon}_{i,:}$$
$$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \sigma^2 \mathbf{I})$$

Mapping from Latent Process to Observed

$$\begin{bmatrix} \mathbf{W} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{W} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{W} \end{bmatrix} \times \begin{bmatrix} \mathbf{x}_{1,:} \\ \mathbf{x}_{2,:} \\ \mathbf{x}_{3,:} \end{bmatrix} = \begin{bmatrix} \mathbf{W}\mathbf{x}_{1,:} \\ \mathbf{W}\mathbf{x}_{2,:} \\ \mathbf{W}\mathbf{x}_{3,:} \end{bmatrix}$$

Output Covariance

This leads to a covariance of the form

$$(\mathbf{I} \otimes \mathbf{W})(\mathbf{K} \otimes \mathbf{I})(\mathbf{I} \otimes \mathbf{W}^\top) + \mathbf{I}\sigma^2$$

Using $(\mathbf{A} \otimes \mathbf{B})(\mathbf{C} \otimes \mathbf{D}) = \mathbf{AC} \otimes \mathbf{BD}$ This leads to

$$\mathbf{K} \otimes \mathbf{W}\mathbf{W}^\top + \mathbf{I}\sigma^2$$

or

$$\mathbf{y} \sim \mathcal{N}(0, \mathbf{W}\mathbf{W}^\top \otimes \mathbf{K} + \mathbf{I}\sigma^2)$$

Kernels for Vector Valued Outputs: A Review

Foundations and Trends[®] in
Machine Learning
Vol. 4, No. 3 (2011) 195–266
© 2012 M. A. Álvarez, L. Rosasco and N. D. Lawrence
DOI: 10.1561/22000000036



Kernels for Vector-Valued Functions: A Review

By Mauricio A. Álvarez,
Lorenzo Rosasco and Neil D. Lawrence

Kronecker Structure GPs

- ▶ This Kronecker structure leads to several published models.

$$(\mathbf{K}(\mathbf{x}, \mathbf{x}'))_{d,d'} = k(\mathbf{x}, \mathbf{x}')k_T(d, d'),$$

where k has \mathbf{x} and k_T has n as inputs.

- ▶ Can think of multiple output covariance functions as covariances with augmented input.
- ▶ Alongside \mathbf{x} we also input the d associated with the *output* of interest.

Separable Covariance Functions

- ▶ Taking $\mathbf{B} = \mathbf{W}\mathbf{W}^\top$ we have a matrix expression across outputs.

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathbf{B},$$

where \mathbf{B} is a $p \times p$ symmetric and positive semi-definite matrix.

- ▶ \mathbf{B} is called the *coregionalization* matrix.
- ▶ We call this class of covariance functions *separable* due to their product structure.

Sum of Separable Covariance Functions

- ▶ In the same spirit a more general class of kernels is given by

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^q k_j(\mathbf{x}, \mathbf{x}') \mathbf{B}_j.$$

- ▶ This can also be written as

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \sum_{j=1}^q \mathbf{B}_j \otimes k_j(\mathbf{X}, \mathbf{X}),$$

- ▶ This is like several Kalman filter-type models added together, but each one with a different set of latent functions.
- ▶ We call this class of kernels sum of separable kernels (SoS kernels).

Geostatistics

- ▶ Use of GPs in Geostatistics is called kriging.
- ▶ These multi-output GPs pioneered in geostatistics: prediction over vector-valued output data is known as *cokriging*.
- ▶ The model in geostatistics is known as the *linear model of coregionalization* (LMC, Journel and Huijbregts (1978); Goovaerts (1997)).
- ▶ Most machine learning multitask models can be placed in the context of the LMC model.

Weighted sum of Latent Functions

- ▶ In the linear model of coregionalization (LMC) outputs are expressed as linear combinations of independent random functions.
- ▶ In the LMC, each component f_d is expressed as a linear sum

$$f_d(\mathbf{x}) = \sum_{j=1}^q w_{d,j} u_j(\mathbf{x}).$$

where the latent functions are independent and have covariance functions $k_j(\mathbf{x}, \mathbf{x}')$.

- ▶ The processes $\{f_j(\mathbf{x})\}_{j=1}^q$ are independent for $q \neq j'$.

Kalman Filter Special Case

- ▶ The Kalman filter is an example of the LMC where $u_i(\mathbf{x}) \rightarrow x_i(t)$.
- ▶ I.e. we've moved from time input to a more general input space.
- ▶ In matrix notation:

1. Kalman filter

$$\mathbf{F} = \mathbf{W}\mathbf{X}$$

2. LMC

$$\mathbf{F} = \mathbf{W}\mathbf{U}$$

where the rows of these matrices \mathbf{F} , \mathbf{X} , \mathbf{U} each contain q samples from their corresponding functions at a different time (Kalman filter) or spatial location (LMC).

Intrinsic Coregionalization Model

- ▶ If one covariance used for latent functions (like in Kalman filter).
- ▶ This is called the intrinsic coregionalization model (ICM, Goovaerts (1997)).
- ▶ The kernel matrix corresponding to a dataset \mathbf{X} takes the form

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

Autokrigeability

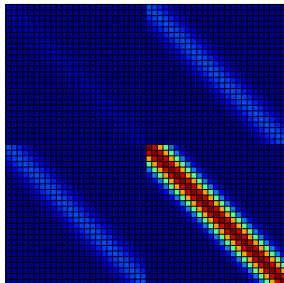
- ▶ If outputs are noise-free, maximum likelihood is equivalent to independent fits of \mathbf{B} and $k(\mathbf{x}, \mathbf{x}')$ (Helterbrand and Cressie, 1994).
- ▶ In geostatistics this is known as autokrigeability (Wackernagel, 2003).
- ▶ In multitask learning its the cancellation of intertask transfer (Bonilla et al., 2008).

Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}\mathbf{w}^\top \otimes k(\mathbf{X}, \mathbf{X}).$$

$$\mathbf{w} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

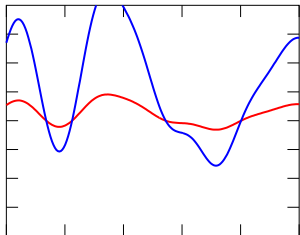
$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$



Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}\mathbf{w}^\top \otimes k(\mathbf{X}, \mathbf{X}).$$

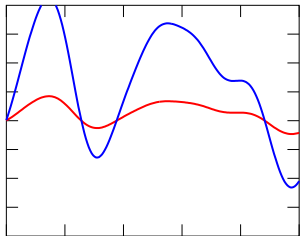
$$\mathbf{w} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$



Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}\mathbf{w}^\top \otimes k(\mathbf{X}, \mathbf{X}).$$

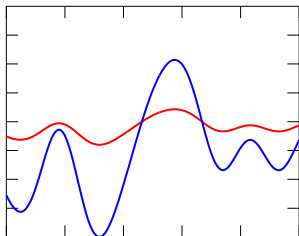
$$\mathbf{w} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$



Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}\mathbf{w}^\top \otimes k(\mathbf{X}, \mathbf{X}).$$

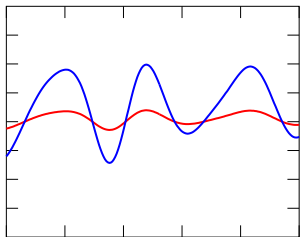
$$\mathbf{w} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$



Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}\mathbf{w}^\top \otimes k(\mathbf{X}, \mathbf{X}).$$

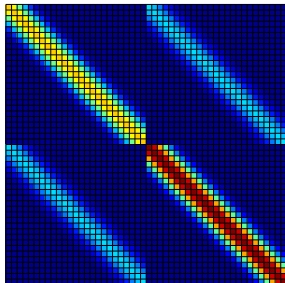
$$\mathbf{w} = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$
$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$



Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

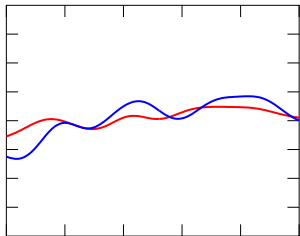
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

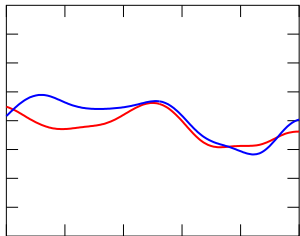
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

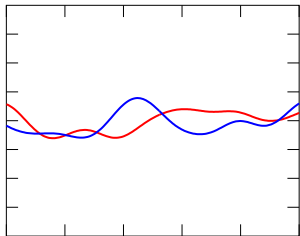
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

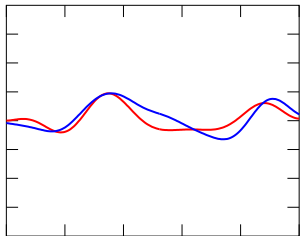
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



Intrinsic Coregionalization Model

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B} \otimes k(\mathbf{X}, \mathbf{X}).$$

$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



LMC Samples

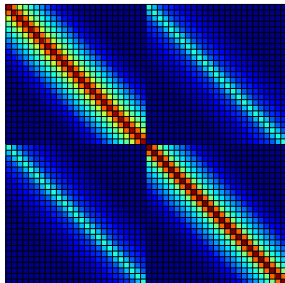
$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X}, \mathbf{X})$$

$$\mathbf{B}_1 = \begin{bmatrix} 1.4 & 0.5 \\ 0.5 & 1.2 \end{bmatrix}$$

$$\ell_1 = 1$$

$$\mathbf{B}_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.3 \end{bmatrix}$$

$$\ell_2 = 0.2$$



LMC Samples

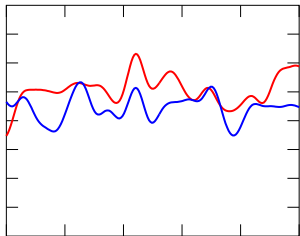
$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X}, \mathbf{X})$$

$$\mathbf{B}_1 = \begin{bmatrix} 1.4 & 0.5 \\ 0.5 & 1.2 \end{bmatrix}$$

$$\ell_1 = 1$$

$$\mathbf{B}_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.3 \end{bmatrix}$$

$$\ell_2 = 0.2$$



LMC Samples

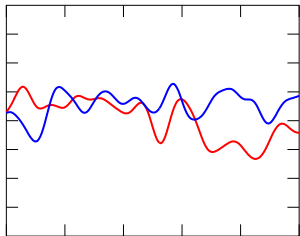
$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X}, \mathbf{X})$$

$$\mathbf{B}_1 = \begin{bmatrix} 1.4 & 0.5 \\ 0.5 & 1.2 \end{bmatrix}$$

$$\ell_1 = 1$$

$$\mathbf{B}_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.3 \end{bmatrix}$$

$$\ell_2 = 0.2$$



LMC Samples

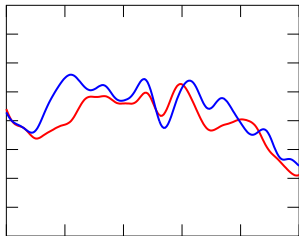
$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X}, \mathbf{X})$$

$$\mathbf{B}_1 = \begin{bmatrix} 1.4 & 0.5 \\ 0.5 & 1.2 \end{bmatrix}$$

$$\ell_1 = 1$$

$$\mathbf{B}_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.3 \end{bmatrix}$$

$$\ell_2 = 0.2$$



LMC Samples

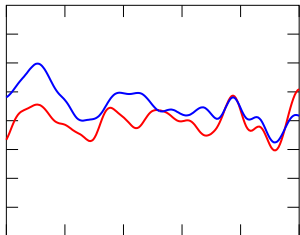
$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X}, \mathbf{X})$$

$$\mathbf{B}_1 = \begin{bmatrix} 1.4 & 0.5 \\ 0.5 & 1.2 \end{bmatrix}$$

$$\ell_1 = 1$$

$$\mathbf{B}_2 = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.3 \end{bmatrix}$$

$$\ell_2 = 0.2$$



LMC in Machine Learning and Statistics

- ▶ Used in machine learning for GPs for multivariate regression and in statistics for computer emulation of expensive multivariate computer codes.
- ▶ Imposes the correlation of the outputs explicitly through the set of coregionalization matrices.
- ▶ Setting $\mathbf{B} = \mathbf{I}_p$ assumes outputs are conditionally independent given the parameters θ . (Minka and Picard, 1997; Lawrence and Platt, 2004; Yu et al., 2005).
- ▶ More recent approaches for multiple output modeling are different versions of the linear model of coregionalization.

Semiparametric Latent Factor Model

- ▶ Coregionalization matrices are rank 1 Teh et al. (2005).
rewrite equation (??) as

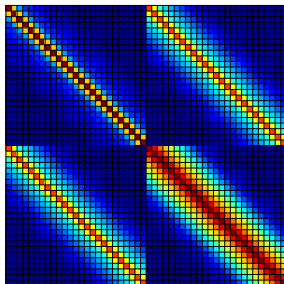
$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \sum_{j=1}^q \mathbf{w}_{:,j} \mathbf{w}_{:,j}^{\top} \otimes k_j(\mathbf{X}, \mathbf{X}).$$

- ▶ Like the Kalman filter, but each latent function has a *different* covariance.
- ▶ Authors suggest using an exponentiated quadratic characteristic length-scale for each input dimension.

Semiparametric Latent Factor Model Samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}_{:,1} \mathbf{w}_{:,1}^\top \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{w}_{:,2} \mathbf{w}_{:,2}^\top \otimes k_2(\mathbf{X}, \mathbf{X})$$

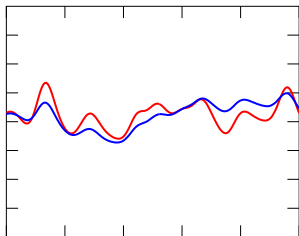
$$\mathbf{w}_1 = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$
$$\mathbf{w}_2 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$



Semiparametric Latent Factor Model Samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}_{:,1} \mathbf{w}_{:,1}^\top \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{w}_{:,2} \mathbf{w}_{:,2}^\top \otimes k_2(\mathbf{X}, \mathbf{X})$$

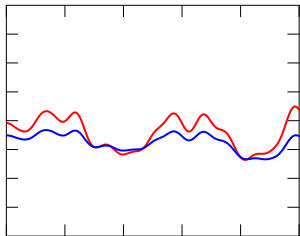
$$\mathbf{w}_1 = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$
$$\mathbf{w}_2 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$



Semiparametric Latent Factor Model Samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}_{:,1} \mathbf{w}_{:,1}^\top \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{w}_{:,2} \mathbf{w}_{:,2}^\top \otimes k_2(\mathbf{X}, \mathbf{X})$$

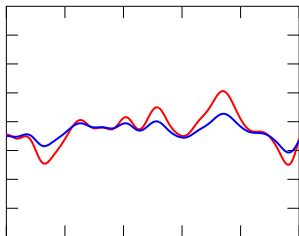
$$\mathbf{w}_1 = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$
$$\mathbf{w}_2 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$



Semiparametric Latent Factor Model Samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}_{:,1} \mathbf{w}_{:,1}^\top \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{w}_{:,2} \mathbf{w}_{:,2}^\top \otimes k_2(\mathbf{X}, \mathbf{X})$$

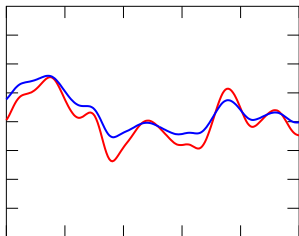
$$\mathbf{w}_1 = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$
$$\mathbf{w}_2 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$



Semiparametric Latent Factor Model Samples

$$\mathbf{K}(\mathbf{X}, \mathbf{X}) = \mathbf{w}_{:,1} \mathbf{w}_{:,1}^\top \otimes k_1(\mathbf{X}, \mathbf{X}) + \mathbf{w}_{:,2} \mathbf{w}_{:,2}^\top \otimes k_2(\mathbf{X}, \mathbf{X})$$

$$\mathbf{w}_1 = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$
$$\mathbf{w}_2 = \begin{bmatrix} 1 \\ 0.5 \end{bmatrix}$$



Gaussian processes for Multi-task, Multi-output and Multi-class

- ▶ Bonilla et al. (2008) suggest ICM for multitask learning.
- ▶ Use a PPCA form for \mathbf{B} : similar to our Kalman filter example.
- ▶ Refer to the autokrigeability effect as the cancellation of inter-task transfer.
- ▶ Also discuss the similarities between the multi-task GP and the ICM, and its relationship to the SLFM and the LMC.

Multitask Classification

- ▶ Mostly restricted to the case where the outputs are conditionally independent given the hyperparameters ϕ (Minka and Picard, 1997; Williams and Barber, 1998; Lawrence and Platt, 2004; Seeger and Jordan, 2004; Yu et al., 2005; Rasmussen and Williams, 2006).
- ▶ Intrinsic coregionalization model has been used in the multiclass scenario. Skolidis and Sanguinetti (2011) use the intrinsic coregionalization model for classification, by introducing a probit noise model as the likelihood.
- ▶ Posterior distribution is no longer analytically tractable: approximate inference is required.

Computer Emulation

- ▶ A statistical model used as a surrogate for a computationally expensive computer model.
- ▶ Higdon et al. (2008) use the linear model of coregionalization to model images representing the evolution of the implosion of steel cylinders.
- ▶ In Conti and O'Hagan (2009) use the ICM to model a vegetation model: called the Sheffield Dynamic Global Vegetation Model (Woodward et al., 1998).

References I

- E. V. Bonilla, K. M. Chai, and C. K. I. Williams. Multi-task Gaussian process prediction. In J. C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, Cambridge, MA, 2008. MIT Press.
- S. Conti and A. O'Hagan. Bayesian emulation of complex multi-output and dynamic computer models. *Journal of Statistical Planning and Inference*, 140(3):640–651, 2009. [DOI].
- G. Della Gatta, M. Bansal, A. Ambesi-Impiombato, D. Antonini, C. Missero, and D. di Bernardo. Direct targets of the *trp63* transcription factor revealed by a combination of gene expression profiling and reverse engineering. *Genome Research*, 18(6):939–948, Jun 2008. [URL]. [DOI].
- P. Goovaerts. *Geostatistics For Natural Resources Evaluation*. Oxford University Press, 1997. [Google Books].
- J. D. Helderbrand and N. A. C. Cressie. Universal cokriging under intrinsic coregionalization. *Mathematical Geology*, 26(2):205–226, 1994.
- D. M. Higdon, J. Gattiker, B. Williams, and M. Rightley. Computer model calibration using high dimensional output. *Journal of the American Statistical Association*, 103(482):570–583, 2008.
- A. G. Journel and C. J. Huijbregts. *Mining Geostatistics*. Academic Press, London, 1978. [Google Books].
- A. A. Kalaitzis and N. D. Lawrence. A simple approach to ranking differentially expressed gene expression time courses through Gaussian process regression. *BMC Bioinformatics*, 12(180), 2011. [DOI].
- N. D. Lawrence and J. C. Platt. Learning to learn with the informative vector machine. In R. Greiner and D. Schuurmans, editors, *Proceedings of the International Conference in Machine Learning*, volume 21, pages 512–519. Omnipress, 2004. [PDF].
- T. P. Minka and R. W. Picard. Learning how to learn is learning with point sets. Available on-line., 1997. [URL]. Revised 1999, available at <http://www.stat.cmu.edu/~minka/>.
- J. Oakley and A. O'Hagan. Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, 89(4):769–784, 2002.
- C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006. [Google Books].
- M. Seeger and M. I. Jordan. Sparse Gaussian Process Classification With Multiple Classes. Technical Report 661, Department of Statistics, University of California at Berkeley,

References II

- G. Skolidis and G. Sanguinetti. Bayesian multitask classification with Gaussian process priors. *IEEE Transactions on Neural Networks*, 22(12):2011 – 2021, 2011.
- Y. W. Teh, M. Seeger, and M. I. Jordan. Semiparametric latent factor models. In R. G. Cowell and Z. Ghahramani, editors, *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 333–340, Barbados, 6–8 January 2005. Society for Artificial Intelligence and Statistics.
- H. Wackernagel. *Multivariate Geostatistics: An Introduction With Applications*. Springer-Verlag, 3rd edition, 2003. [[Google Books](#)].
- C. K. Williams and D. Barber. Bayesian Classification with Gaussian processes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1342–1351, 1998.
- I. Woodward, M. R. Lomas, and R. A. Betts. Vegetation-climate feedbacks in a greenhouse world. *Philosophical Transactions: Biological Sciences*, 353(1365):29–39, 1998.
- K. Yu, V. Tresp, and A. Schwaighofer. Learning Gaussian processes from multiple tasks. In *Proceedings of the 22nd International Conference on Machine Learning (ICML 2005)*, pages 1012–1019, 2005.