Functional modelling with Gaussian Processes

Lehel Csató

Faculty of Mathematics and Informatics
Babeş–Bolyai University, Cluj-Napoca,

Matematika és Informatika Tanszék
Babeş–Bolyai Tudományegyetem, Kolozsvár

September 2008

Table of Contents

Modelling

Motivation

D. Donoho

Data, Data, Data!
Challenges and opportunities of the coming **Data Deluge**

Several data types:

- classification problem – needed in decision systems: frequently there are data of very high dimension and several hundred classes to take into account;
- regression/prediction problems.

Machine learning

Historical background / Motivation:

- Huge amount of data, that should automatically be processed,
- Mathematics provides general solutions, solutions are i.e. not for a given problem,
- Need for “science”, that uses mathematics machinery for solving practical problems.

Definition of Machine Learning

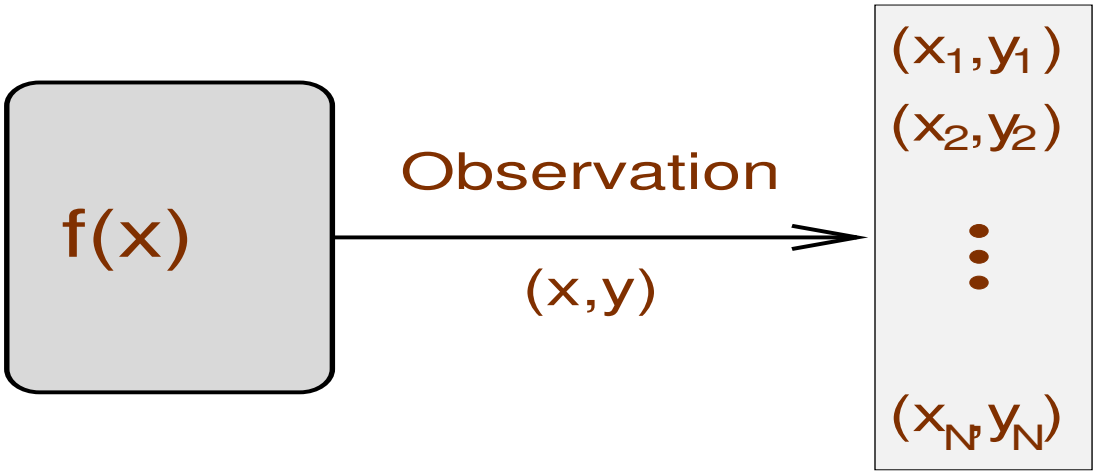
Machine learning

Collection of methods (from statistics, probability theory) to solve problems met in practice.

- noise filtering for
  - non-linear regression and/or
  - non-Gaussian noise
- Classification:
  - binary,
  - multiclass,
  - partially labeled
- Clustering,
- Inversion problems,
- density estimation, novelty detection.

Generally, we need to model the data,

Modelling Data

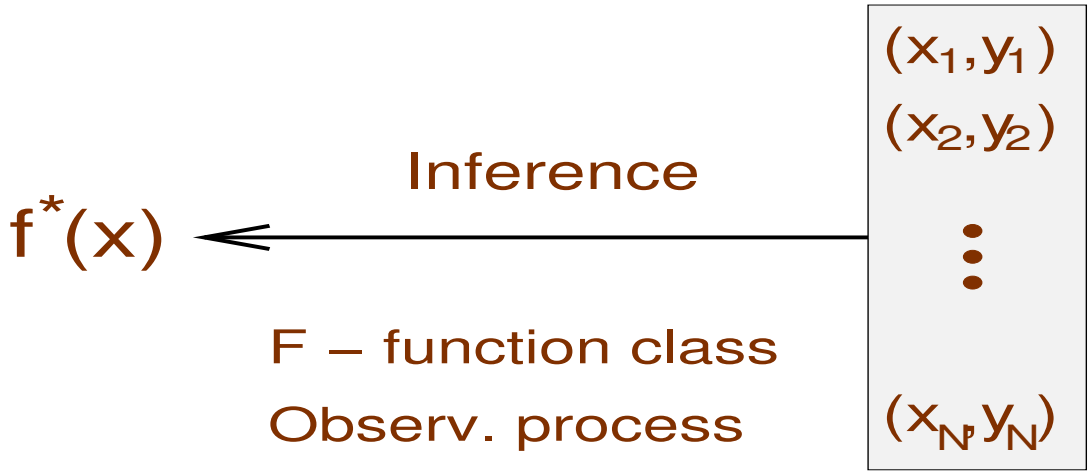


- Real world: there "is" a function $y = f(x)$
- Observation process: a corrupted datum is collected for a sample x_n:

$$\begin{aligned} t_n &= y_n + \epsilon && \text{additive noise} \\ t_n &= h(y_n, \epsilon) && h \text{ distortion function} \end{aligned}$$

- **Problem: find function $y = f(x)$**

Latent variable models



- **Data set** – collected.
- Assume a function class.
  - polynomial,
  - Fourier expansion,
  - Wavelet;
- Observation process – encodes the noise;
- Find the optimal function from the class.

Latent variable models II

- We have the **data set** $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$.
- Consider a function class:

$$(1) \quad \mathcal{F} = \left\{ \boldsymbol{w}^T \boldsymbol{x} + b | \boldsymbol{w} \in \mathbb{R}^d,\ b \in \mathbb{R} \right\}$$

$$(2) \quad \mathcal{F} = \left\{ a_0 + \sum_{k=1}^{K} a_k \sin(2\pi k x) + \sum_{k=1}^{K} b_k \cos(2\pi k x) \right.$$
$$\left. | \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^K,\ a_0 \in \mathbb{R} \right\}$$

- Assume an observation process:

$$y_n = f(\boldsymbol{x}_n) + \epsilon \quad \text{with } \epsilon \sim N(0, \sigma^2).$$

Latent variable models III

1. The **data set:** $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_N, y_N)\}$.

2. Assume a function class:

$$\mathcal{F} = \left\{ f(\boldsymbol{x}, \boldsymbol{\theta}) | \boldsymbol{\theta} \in \mathbb{R}^p \right\}$$

   $\mathcal{F}$ – polynomial, etc.

3. Assume an observation process. Define a loss function:

$$L\left(y_n, f(\boldsymbol{x}_n, \boldsymbol{\theta})\right)$$

   For the Gaussian noise:

$$L(y_n, f(\boldsymbol{x}_n, \boldsymbol{\theta})) = (y_n - f(\boldsymbol{x}_n, \boldsymbol{\theta}))^2.$$

Parameter estimation

Estimating parameters:

Finding the **optimal value to $\boldsymbol{\theta}$**:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Omega} L(\mathcal{D}, \boldsymbol{\theta})$$

where

- Ω is the domain of the parameters.
- $L(\mathcal{D}, \boldsymbol{\theta})$ is a "loss function" for the data set. Example:

$$L(\mathcal{D}, \boldsymbol{\theta}) = \sum_{n=1}^{N} L(y_n, f(\boldsymbol{x}_n, \boldsymbol{\theta}))$$

Parameter estimation – M.L.

$L(\mathcal{D}, \boldsymbol{\theta})$ – (log)likelihood function.

Maximum likelihood estimation of the model:

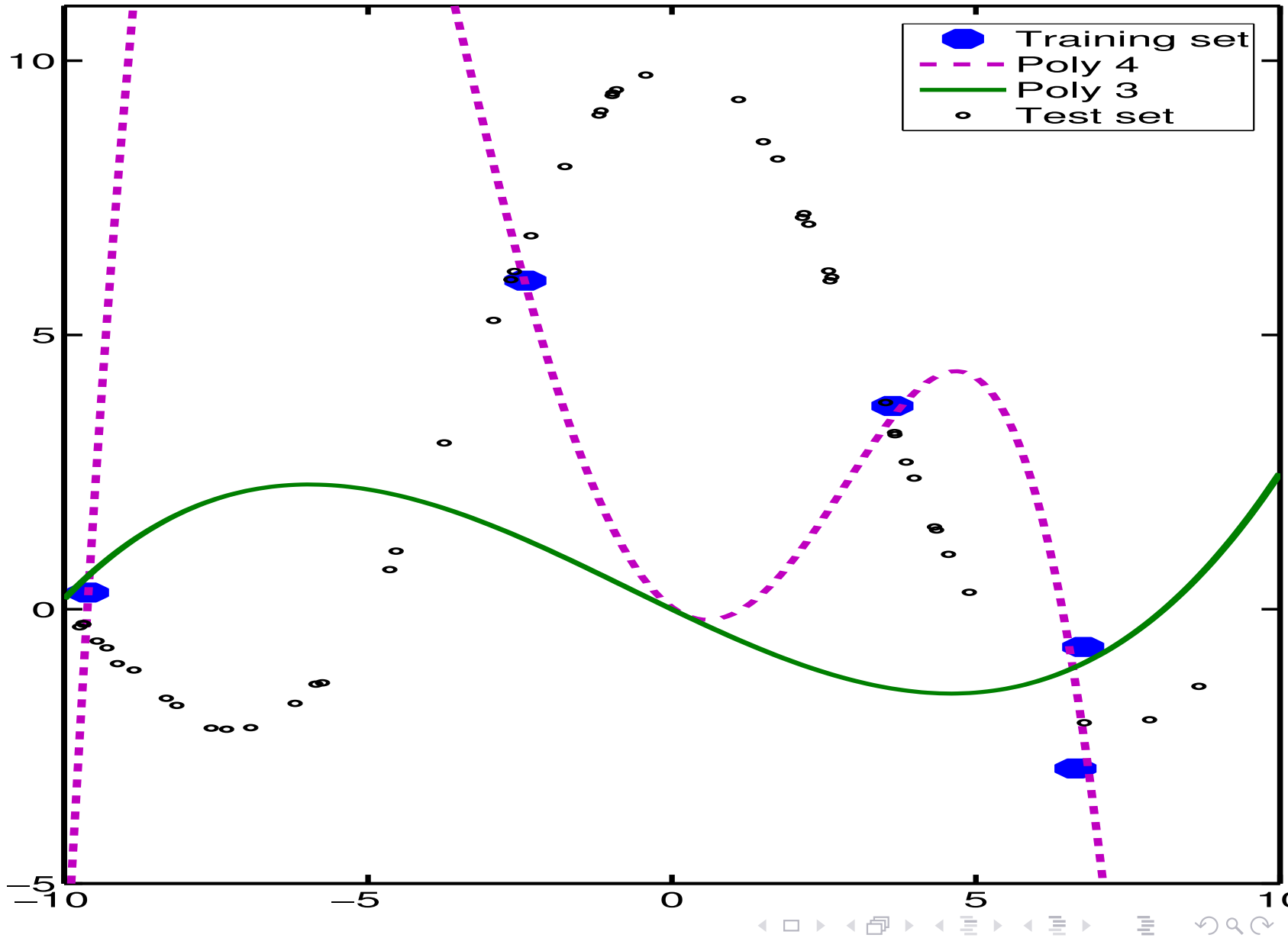
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta} \in \Omega} L(\mathcal{D}, \boldsymbol{\theta})$$

Example – quadratic regression:

$$L(\mathcal{D}, \boldsymbol{\theta}) = \sum_{n=1}^{N} (y_n - f(\boldsymbol{x}_n, \boldsymbol{\theta}))^2 \quad \text{– factorisation}$$

Drawback: produces perfect fit to the data – **over-fitting**.

Maximum Likelihood – Over-fitting



Maximum a–posteriori estimation

M.A.P. – assigning probabilities to the

- Data $\mathcal{D}$: log–likelihood function: the probability of the data drawn using $\boldsymbol{\theta}$

$$P(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}, \mathcal{F}) \propto \exp\left[-L(y_n, f(\boldsymbol{x}_n, \boldsymbol{\theta}))\right]$$

$\propto$ – a normalisation constant missing.

- Parameters $\boldsymbol{\theta}$: probability that $\boldsymbol{\theta}$ could have had a given value

$$p_0(\boldsymbol{\theta}) \propto \exp\left[-\frac{\|\boldsymbol{\theta}\|^2}{2}\right]$$

prior probability.

Maximum a–posteriori estimation

M.A.P. – assigning probabilities to the

- Data $\mathcal{D}$: log–likelihood function: the probability of the data drawn using $\boldsymbol{\theta}$

$$P(y_n|\boldsymbol{x}_n, \boldsymbol{\theta}, \mathcal{F}) \propto \exp\left[-L(y_n, f(\boldsymbol{x}_n, \boldsymbol{\theta}))\right]$$

$\propto$ – a normalisation constant missing.

- Parameters $\boldsymbol{\theta}$: probability that $\boldsymbol{\theta}$ could have had a given value

$$p_0(\boldsymbol{\theta}) \propto \exp\left[-\frac{\|\boldsymbol{\theta}\|^2}{2}\right]$$

prior probability.

M.A.P. estimation II

Combining prior with observation – likelihood – using Bayes' rule:

A–posteriori probability of the parameters:

$$p(\boldsymbol{\theta}|\mathcal{D},\boldsymbol{\mathcal{F}}) = \frac{P(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta})}{p(\mathcal{D}|\boldsymbol{\mathcal{F}})}$$

$$p(\mathcal{D}|\boldsymbol{\mathcal{F}}) = \int_\Omega d\boldsymbol{\theta}\ \ P(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta})$$

$p(\mathcal{D}|\boldsymbol{\mathcal{F}})$ – probability of the data set under a given family of models.

M.A.P. estimation III

M.A.P. estimation – aims at finding $\boldsymbol{\theta}$ with the largest probability:

$$\boldsymbol{\theta}^*_{MAP} = \arg\max_{\boldsymbol{\theta}\in\Omega} p(\boldsymbol{\theta}|\mathcal{D}, \boldsymbol{\mathcal{F}})$$

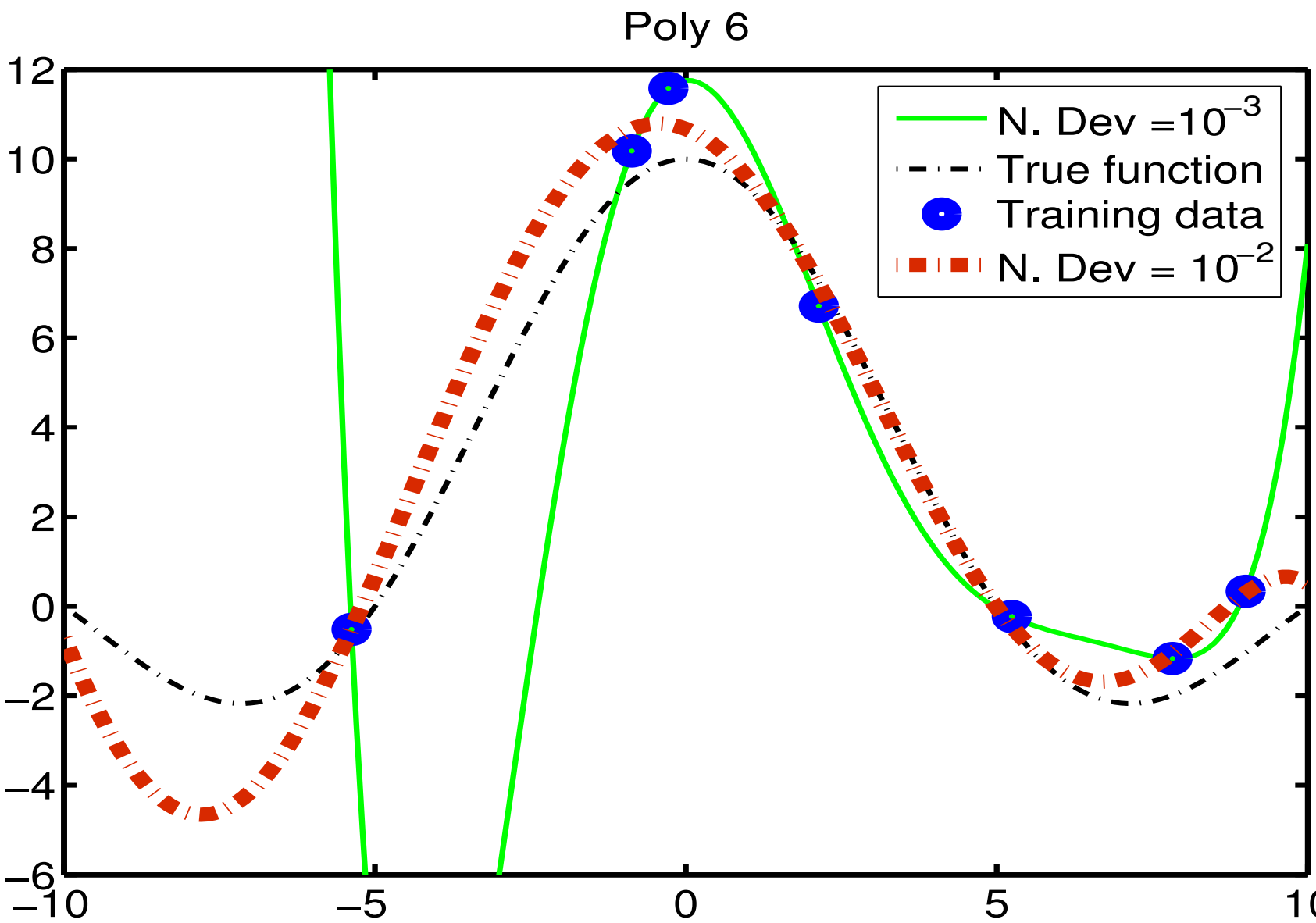
Example:
Using $L(y_n, f(\boldsymbol{x}_n, \boldsymbol{\theta}))$ in defining the likelihood and Gaussian prior:

$$\boldsymbol{\theta}^*_{MAP} = \arg\max_{\boldsymbol{\theta}\in\Omega} K - \sum_n L(y_n, f(\boldsymbol{x}_n, \boldsymbol{\theta})) - \frac{\|\boldsymbol{\theta}\|^2}{2\sigma_o^2}$$

For $\sigma_o^2 \to \infty$ we have maximum likelihood

after a change of sign and max $\to$ min

M.A.P. – Example


Poly 6
N. Dev =10^−3
True function
Training data
N. Dev = 10^−2
12
10
8
6
4
2
0
−2
−4
−6
−10
−5
0
5
10

Parameter estimation – Bayes

We use Bayes' rule:

$$p(\boldsymbol{\theta}|\mathcal{D},\mathcal{F}) = \frac{P(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta})}{p(\mathcal{D}|\mathcal{F})}$$

$$p(\mathcal{D}|\mathcal{F}) = \int_\Omega d\boldsymbol{\theta}\ \ P(\mathcal{D}|\boldsymbol{\theta})p_0(\boldsymbol{\theta})$$

and try to store the whole distribution of the possible values.

We operate therefore with

$$p_{\text{post}}(\boldsymbol{\theta}|\mathcal{D},\mathcal{F})$$

Bayes estimation II

When computing $p_{\text{post}}(\boldsymbol{\theta}|\mathcal{D}, \mathcal{F})$ we assumed that the posterior can be represented analytically.

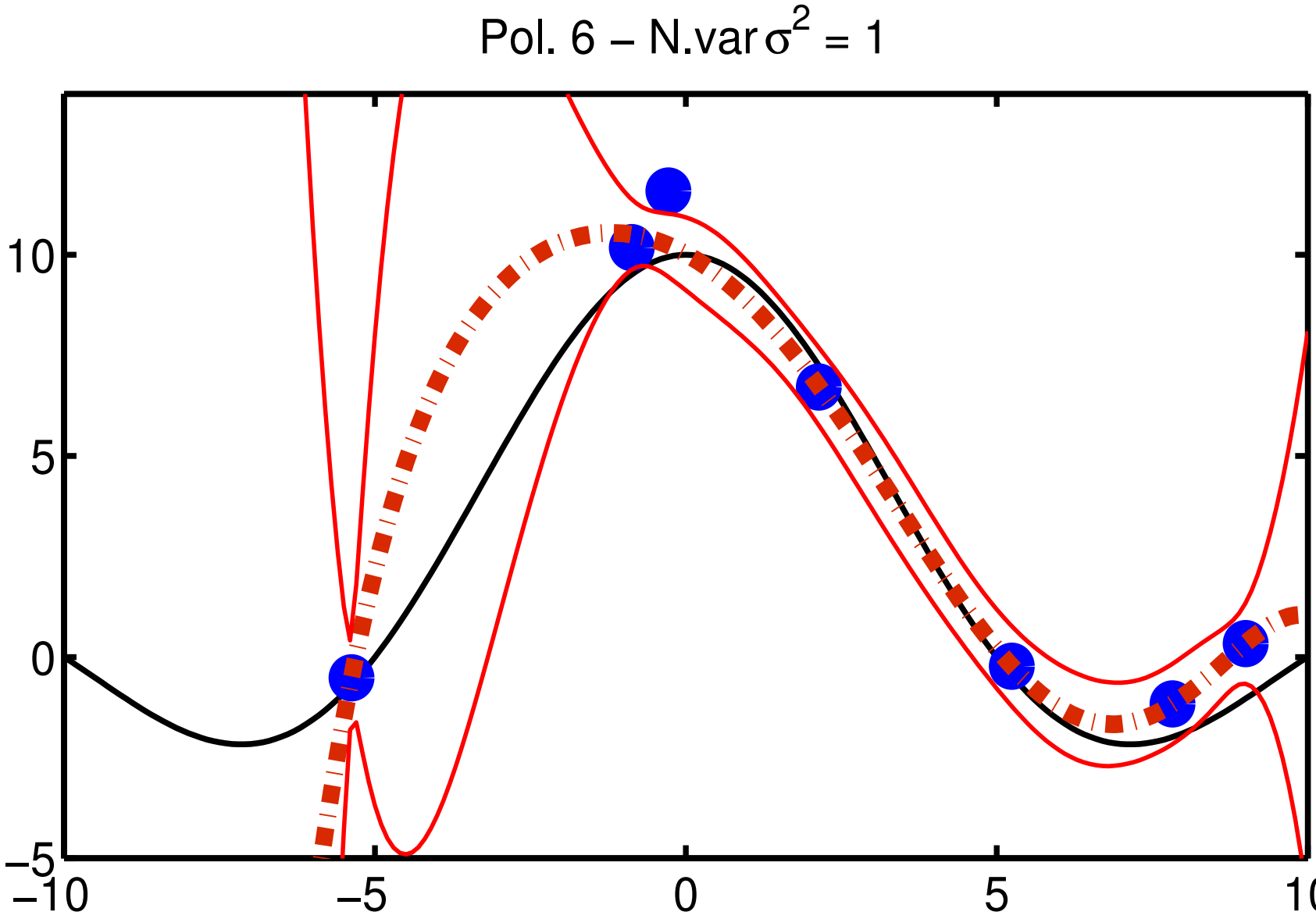
This is not the case.

Approximations are needed for the

- posterior distribution
- predictive distribution

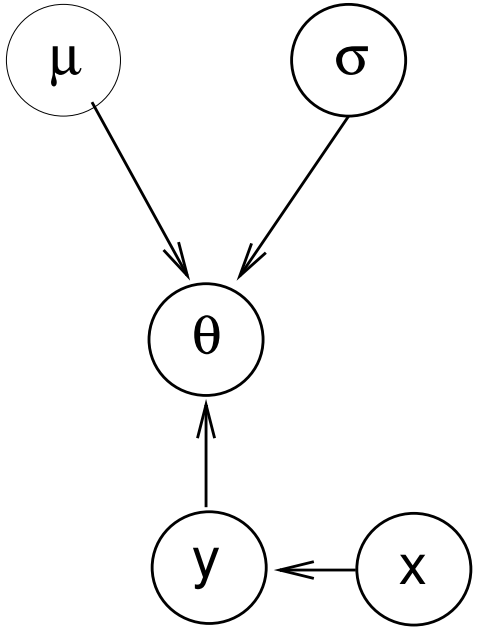
In Bayesian modelling an important issue is how we approximate the posterior distribution.

Bayes – Example



Graphical Models

- M.L. estimation: no prior
- M.A.P. estimation: no distribution
- Bayes est.



Bayes' models – if approximations used – we have Level II Maximum Likelihood.

Drawbacks of Parametric Models

Model complexity – is an important "parameter":

- choosing $\mathcal{F}$ is an important decision in modelling the data:
- Example – for polynomial functions:
  - linear – too simple
  - quadratic – "good" for medium-sized data
  - ...
- The model complexity should be changed if we have more data available.

Non-parametric models

Non-parametric Models

Non-parametric models: use the available data for prediction.

1. Density estimation:
   - Histogram method;
   - Parzen window;
2. Regression/Classification:
   - K-Nearest Neighbour Rule;
   - Support Vector Machine;
   - Gaussian Processes;

Non-parametric Models

Non-parametric models: use the available data for prediction.

1. Density estimation:
   - Histogram method;
   - Parzen window;
2. Regression/Classification:
   - K-Nearest Neighbour Rule;
   - Support Vector Machine;
   - Gaussian Processes;

Density Estimation

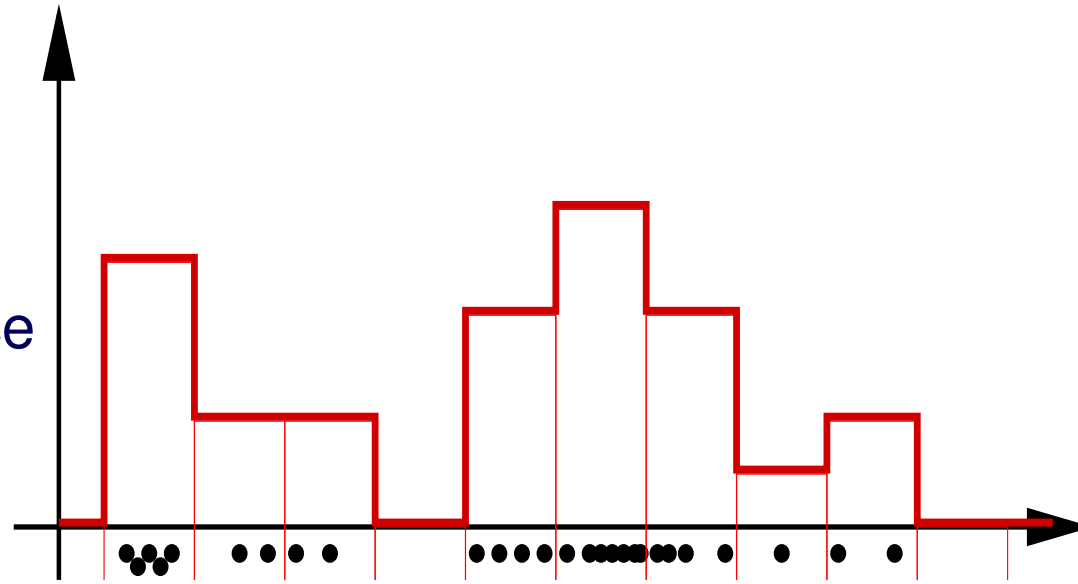
Histogram method:

"Parameters"

- bin width;
- locations;

Sensitive to the choice of parameters.

Not usable for higher dimensions



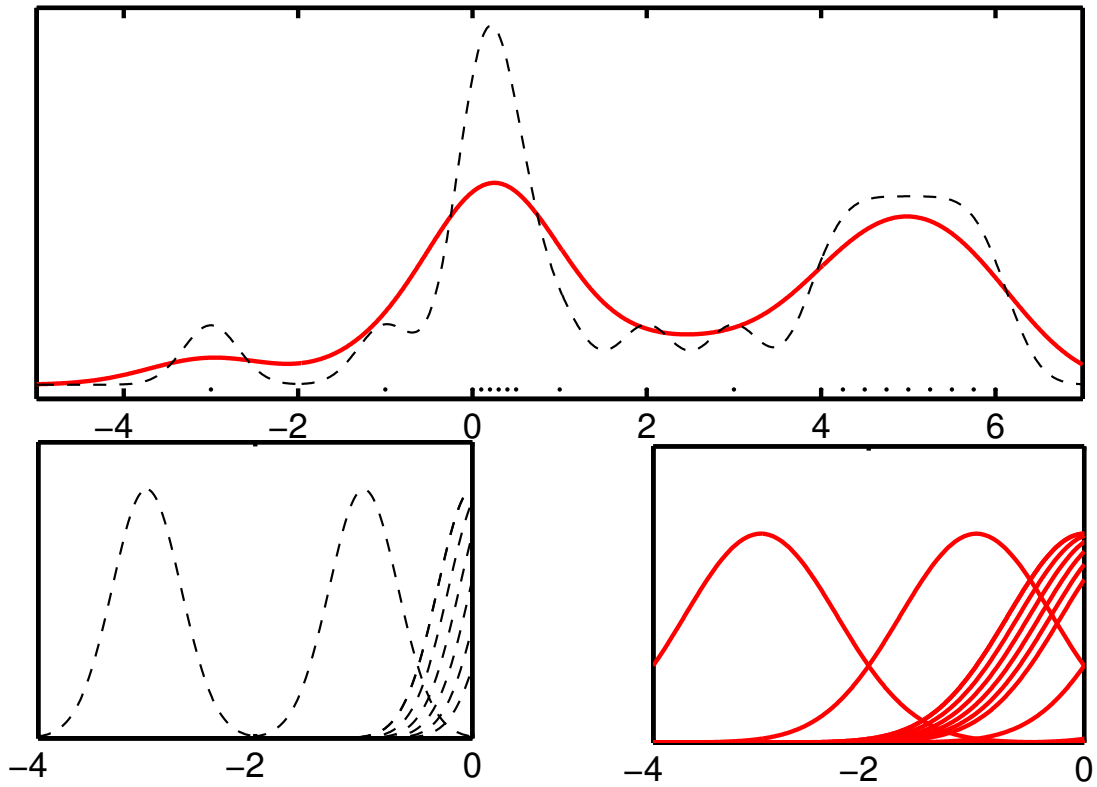
Density Estimation II

Parzen window:

"Parameters"

- function width;
- data locations

Not usable for large data-sets.



$$h(\boldsymbol{x}) = \frac{1}{N} \sum_{n=1}^{N} k(\boldsymbol{x}_n, \boldsymbol{x})$$

Nonparametric: summation scales with N.

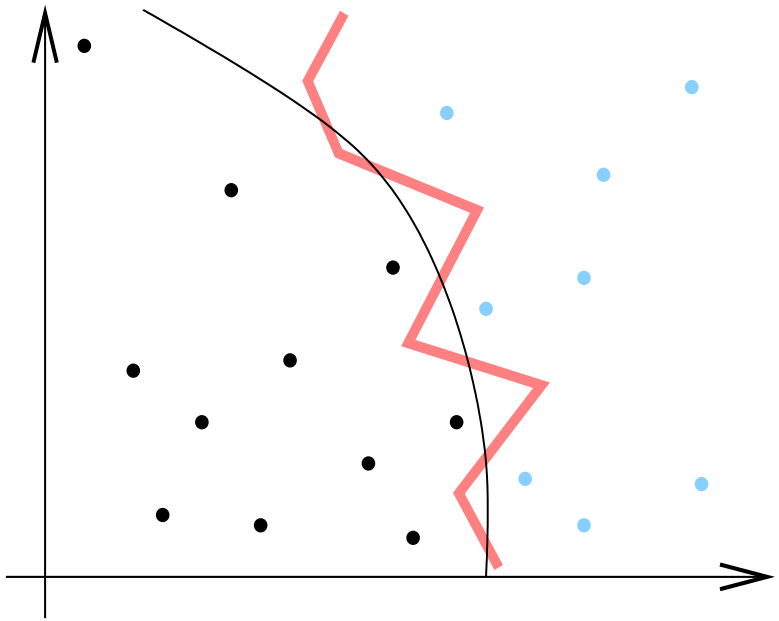
Non–parametric Classification

K–Nearest Neighbour: (Knn)

"Parameters"

- # of neighbours

Slow for large data.



Nonparametric: all data taken into account.

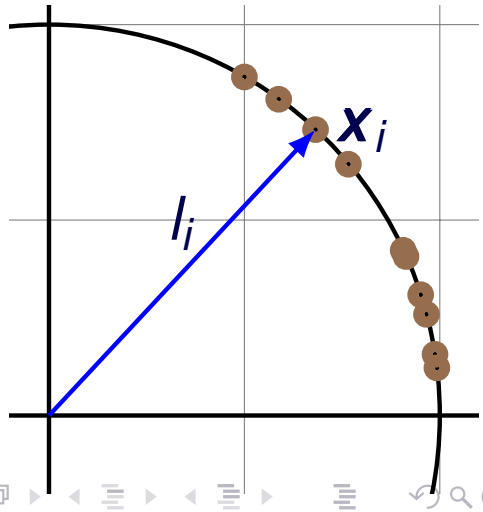
We generally:

- normalise and center the data:

$$\begin{aligned} \boldsymbol{x}_i &\Leftarrow \boldsymbol{x}_i - \overline{\boldsymbol{x}} & \overline{\boldsymbol{x}} &- \text{mean} \\ \boldsymbol{x}_{ij} &\Leftarrow \boldsymbol{x}_{ij}/\sigma_j & \sigma_j^2 &\text{ is var. for } j\text{-th comp.} \end{aligned}$$

- Each x_{ij} has zero mean and unit variance.
- the length of the random vector $\boldsymbol{x}_i$ is

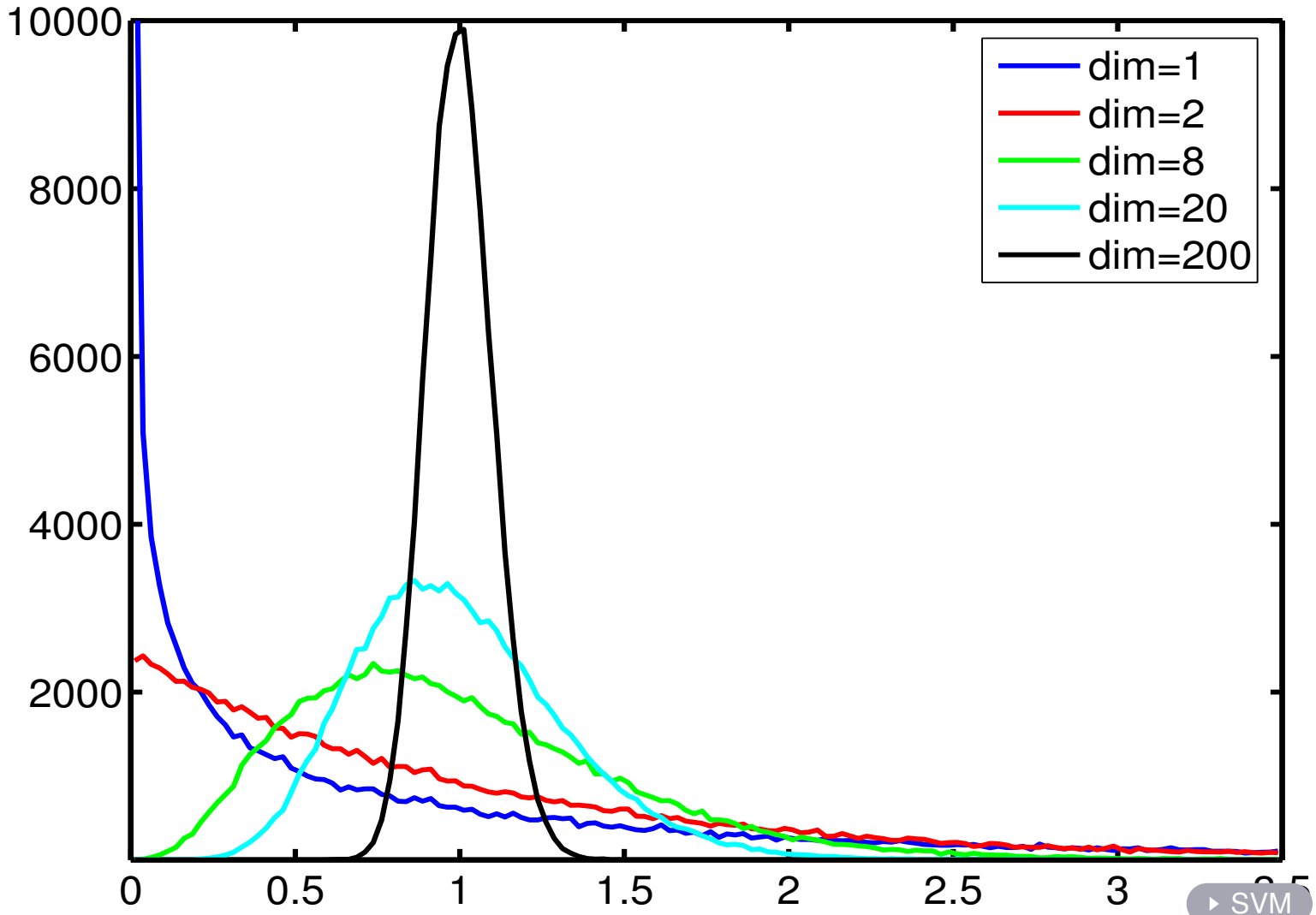
$$l_i^2 = \sum_j x_{ij}^2$$



cf. central limit theorem

the larger the dimension, the more concentrated the average length is around the mean.

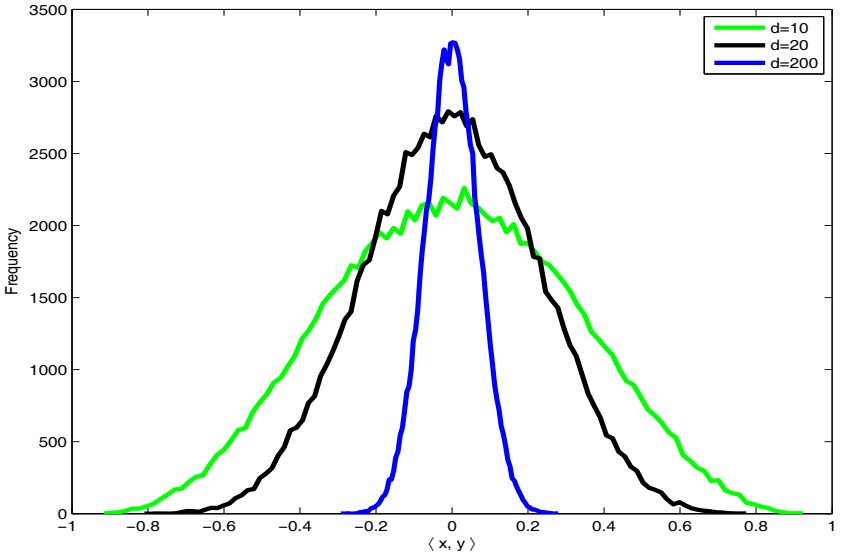
Example: dim=20;x=randn(100000,dim);s2=sum(x.^2,2)/dim;g=histc(s2,bin)



The angle:

$$\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle = \sum_{\ell=1}^{d} x_{i\ell} x_{j\ell}$$

with average value 0.



Central limit theorem

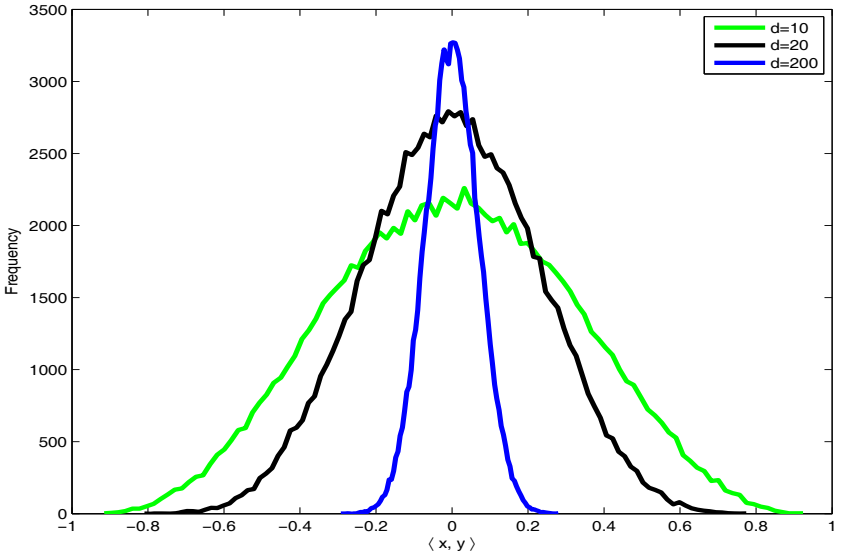
The larger the dimension

- the more orthogonal random vectors are;
- the more difficult is to select a representative.

The angle:

$$\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle = \sum_{\ell=1}^{d} x_{i\ell} x_{j\ell}$$

with average value 0.



Central limit theorem

The larger the dimension

- the more orthogonal random vectors are;
- the more difficult is to select a representative.

Non-parametric models

Motivation:
Knn

- is too demanding for large datasets;
- works only in low-dimensions;

▸ HD

Algorithm that works "well" in high-dimensions

$$f(\boldsymbol{x}) = \operatorname*{argmin}_{f \in \mathcal{F}} \left\{ \sum_n L\left(y_n, f(\boldsymbol{x}_n)\right) + \|\mathbf{P} f(\cdot)\|^2 \right\}$$

Learning Algorithm

Design of a method that simultaneously minimises the empirical error (first term) and selects "cleverly" (second term) from a large family $\mathcal{F}$ of available functions.

Details of the elements: ...

▸ Skip HighDim

Algorithm that works "well" in high-dimensions

$$f(\boldsymbol{x}) = \operatorname*{argmin}_{f \in \mathcal{F}} \left\{ \sum_n L\left(y_n, f(\boldsymbol{x}_n)\right) + \|\mathbf{P} f(\cdot)\|^2 \right\}$$

Learning Algorithm

Design of a method that simultaneously minimises the empirical error (first term) and selects "cleverly" (second term) from a large family $\mathcal{F}$ of available functions.

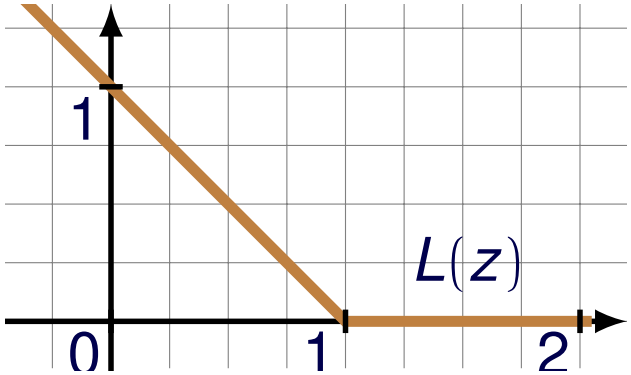
Within the **SVM framework** we look for candidates

- in a large family of functions; and
- we penalise the complexity of the functions;

Explain loss function and $\mathcal{F}$.

Hinge Loss:

$$L(z) = \begin{cases} 0 & \text{if } z \geq 1 \\ 1 - z & \text{if } z < 1 \end{cases}$$



For classification with labels $y_n = \pm 1$ the loss function for the data-set $\mathcal{D}$:

$$L(\mathcal{D}) \stackrel{\text{def}}{=} \sum_n L\left(y_n f(\boldsymbol{x}_n)\right)$$

Minimality

$L(\mathcal{D})$ is minimal if all elements separated with margin ≥ 1.

Loss functions:

Hinge loss – in classification – penalises data away from boundary;

- assuming class labels $y_i = \pm 1$, we have

$$L\left(y_n, f(\boldsymbol{x}_n)\right) = H_+\left(1 - y_n f(\boldsymbol{x}_n)\right)$$

Logit loss – in binary classification, returns the log-probabilities:

$$L_{\text{logit}}(y_n, f(\boldsymbol{x}_n)) = -\log\left(1 + \exp\left(-y_n f(\boldsymbol{x})\right)\right)$$

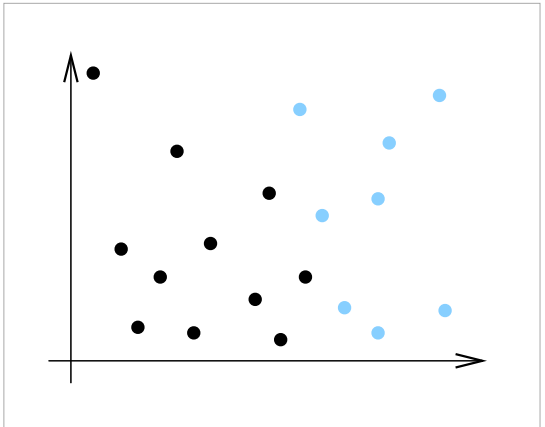
- when $y_n f(\boldsymbol{x}_n) \to \infty$, $L_{\text{logit}} \to 0$
- in probit model $(1 + \exp(\cdot))^{-1}$ replaced with $\Phi(\cdot)$.

Quadratic loss – quadratic error:

$$L_{\text{mse}}(y_n, f(\boldsymbol{x}_n)) = (y_n - f(\boldsymbol{x}_n))^2 \equiv (1 - y_n f(\boldsymbol{x}_n))^2$$

- can also be applied to regression problems.

Which family of functions? – we have only data $\mathcal{D}$.



Possibilities:

- **linear – might be too simple;**
- **complex – might be too complex;**

In general we

- want a flexible function class, but
- do not want a too complex solution.

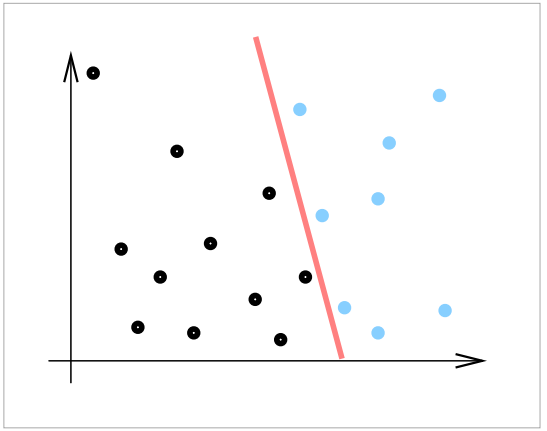
Solution:

use a large function class $\mathcal{F}$ **and** define penalties on the complexity of the solution.

Which family of functions? – we have only data $\mathcal{D}$.

Possibilities:

- **linear – might be too simple;**
- **complex – might be too complex;**



In general we

- want a flexible function class, but
- do not want a too complex solution.

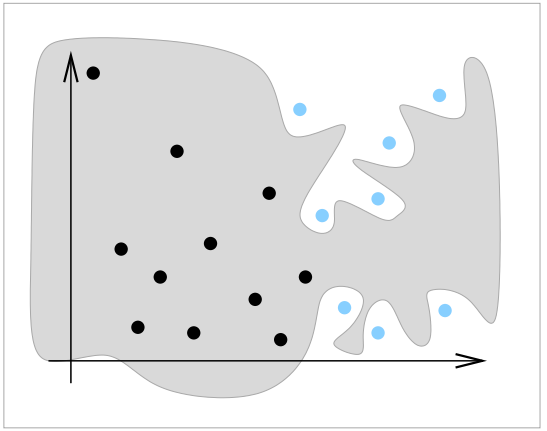
Solution:

use a large function class $\mathcal{F}$ **and** define penalties on the complexity of the solution.

Which family of functions? – we have only data $\mathcal{D}$.

Possibilities:

- linear – might be too simple;
- **complex – might be too complex;**



In general we

- want a flexible function class, but
- do not want a too complex solution.

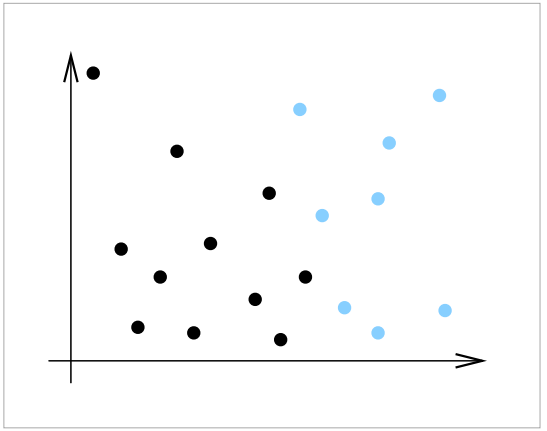
Solution:

use a large function class $\mathcal{F}$ **and** define penalties on the complexity of the solution.

Which family of functions? – we have only data $\mathcal{D}$.

Possibilities:

- **linear – might be too simple;**
- **complex – might be too complex;**



In general we

- want a flexible function class, but
- do not want a too complex solution.

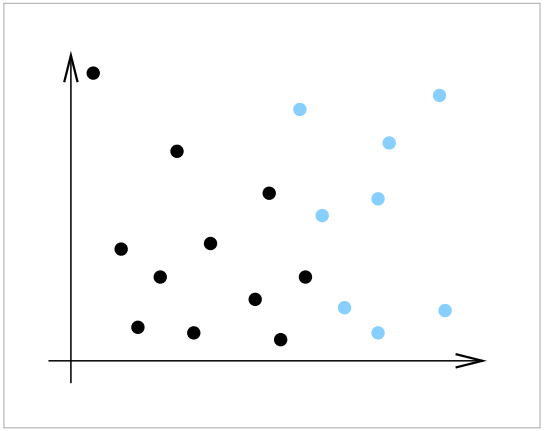
Solution:

use a large function class $\mathcal{F}$ **and** define penalties on the complexity of the solution.

Which family of functions? – we have only data $\mathcal{D}$.

Possibilities:

- **linear – might be too simple;**
- **complex – might be too complex;**



In general we

- want a flexible function class, but
- do not want a too complex solution.

Solution:

use a large function class $\mathcal{F}$ **and** define penalties on the complexity of the solution.

- Loss functions help us find the best function
- from a family of functions.

The family of functions is important

It should be flexible enough

- for large data–sets
- and to allow for different length–scales.

Solution is of the form ($\mathbf{P}$ penalty):

$$f(\boldsymbol{x}) = \operatorname*{argmin}_{f \in \mathcal{F}} \left\{ \sum_n L\left(y_n, f(\boldsymbol{x}_n)\right) + \|\mathbf{P} f(\cdot)\|^2 \right\}$$

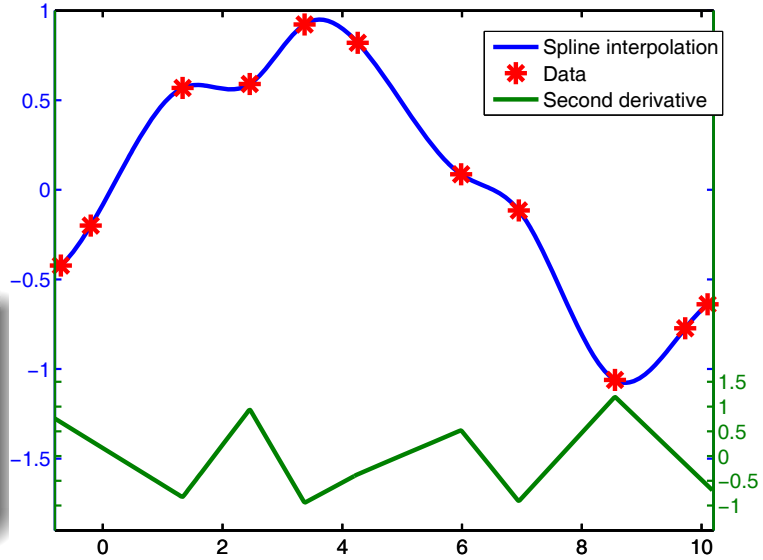
$\|\mathbf{P} f(\cdot)\|^2 = 0$ $\Rightarrow$ null–space, gives possible solutions.

Example:

$\mathcal{F}$ – twice diff. functions with cont. second derivatives;

$\mathbf{P} \stackrel{\text{def}}{=} \left(\partial_x^2 f(\cdot)\right)^2$ – the sum of the second derivatives.

Result: interpolating splines with knots at the data points.



Message

Functional optimisation plausible with reliable results.

Presented in Wahba'90: Splines Models for Observational Data

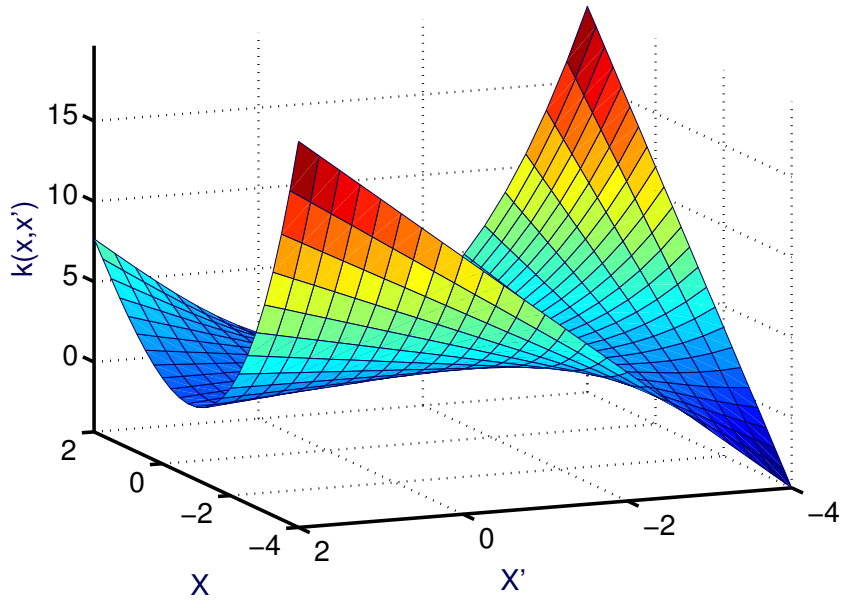
Result of the optimisation problem:

$$K_s(\boldsymbol{x}, \boldsymbol{x}') = \frac{1}{3}\min\left(\boldsymbol{x}, \boldsymbol{x}'\right)^3 + \frac{1}{2}\left|\boldsymbol{x} - \boldsymbol{x}'\right|\min\left(\boldsymbol{x}, \boldsymbol{x}'\right)^2 + \boldsymbol{x}\boldsymbol{x}' + 1$$

Solution of the argmin:

$$\hat{f}(\boldsymbol{x}) = \sum_n \alpha_n K_s(\boldsymbol{x}, \boldsymbol{x}_n)$$

Linear combination of polynomials.



Reproducing Kernel Hilbert space:
Assume $\mathcal{X}$ an index set, $\mathcal{H} = \{f \mid f : \mathcal{X} \to \mathbb{R}\}$ with $\langle \cdot, \cdot \rangle$ the dot product s.t. $\|f\|^2 = \langle f, f \rangle$.
If there exists $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that

1. k has the reproducing property:

$$\forall\, f \in \mathcal{H} \qquad \langle f, k(x, \cdot) \rangle = f(x);$$

2. and k spans $\mathcal{H}$.

Not a feature map

RKHS is not equivalent with a feature map.
More than a single feature map for the same $k(\cdot, \cdot)$.

$k(\cdot,\cdot)$ spans $\mathcal{H}$ $\Rightarrow$ every function $f \in \mathcal{H}$ is a linear combination of $\{k(\boldsymbol{x}_n,\cdot)\}_{n=1}^{N_\mathcal{H}}$:

$$f(\boldsymbol{x}) = \sum_{n=1}^{N_\mathcal{H}} w_i\, k(\boldsymbol{x},\boldsymbol{x}_n) \quad (N_\mathcal{H} = \infty \text{ allowed})$$

Note:

- we choose the *support set*:
  - the location of the points and
  - the set size;
- we choose the *weights*.

Empirical kernel map

The support set $\mathcal{X}$ is the training data set:

$$\mathcal{H}_e = \left\{ \sum\nolimits_n \alpha_n\, k(\boldsymbol{x},\boldsymbol{x}_n) \mid \alpha_n \in \mathbb{R} \right\}$$

$k(\cdot,\cdot)$ spans $\mathcal{H}$ $\Rightarrow$ every function $f \in \mathcal{H}$ is a linear combination of $\{k(\boldsymbol{x}_n,\cdot)\}_{n=1}^{N_\mathcal{H}}$:

$$f(\boldsymbol{x}) = \sum_{n=1}^{N_\mathcal{H}} w_i\, k(\boldsymbol{x},\boldsymbol{x}_n) \quad (N_\mathcal{H} = \infty \text{ allowed})$$

Note:

- we choose the *support set*:
  - the location of the points and
  - the set size;
- we choose the *weights*.

Empirical kernel map

The support set $\mathcal{X}$ is the training data set:

$$\mathcal{H}_e = \left\{ \sum\nolimits_n \alpha_n\, k(\boldsymbol{x},\boldsymbol{x}_n) \mid \alpha_n \in \mathbb{R} \right\}$$

Mercer kernel

For a positive-definite $k(\cdot,\cdot)$ there exists a set $\{\phi_j\}$ of orthogonal functions and $\{\lambda_j\}$ positive constants s.t.

$$k(\boldsymbol{x},\boldsymbol{x}') = \sum_j \lambda_j \phi_j(\boldsymbol{x})\phi_j(\boldsymbol{x}')$$

Note:

- The function $k(\cdot,\cdot)$ defines:
  - the eigen-functions $\{\phi_j\}$
  - the corresponding eigen-values $\{\lambda_j\}$;
- independent of the data-set we are using;
- convergence guarantee: $\sum_j \lambda_j^2 < \infty$.

Mercer kernel

For a positive-definite $k(\cdot,\cdot)$ there exists a set $\{\phi_j\}$ of orthogonal functions and $\{\lambda_j\}$ positive constants s.t.

$$k(\boldsymbol{x},\boldsymbol{x}') = \sum_j \lambda_j \phi_j(\boldsymbol{x})\phi_j(\boldsymbol{x}')$$

Note:

- The function $k(\cdot,\cdot)$ defines:
  - the eigen-functions $\{\phi_j\}$
  - the corresponding eigen-values $\{\lambda_j\}$;
- independent of the data-set we are using;
- convergence guarantee: $\sum_j \lambda_j^2 < \infty$.

Kernel functions: Let $\Phi : \mathbb{R} \to \mathbb{R}^3$ be given by:

$$\Phi(x) = \begin{bmatrix} 1 \\ \sqrt{2}x \\ x^2 \end{bmatrix}$$

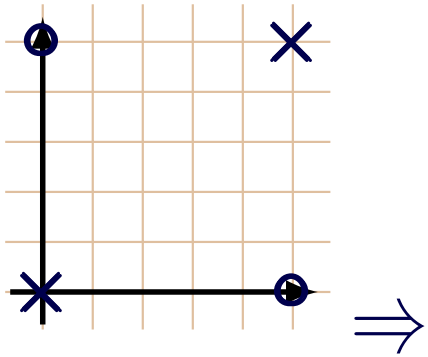
and compute

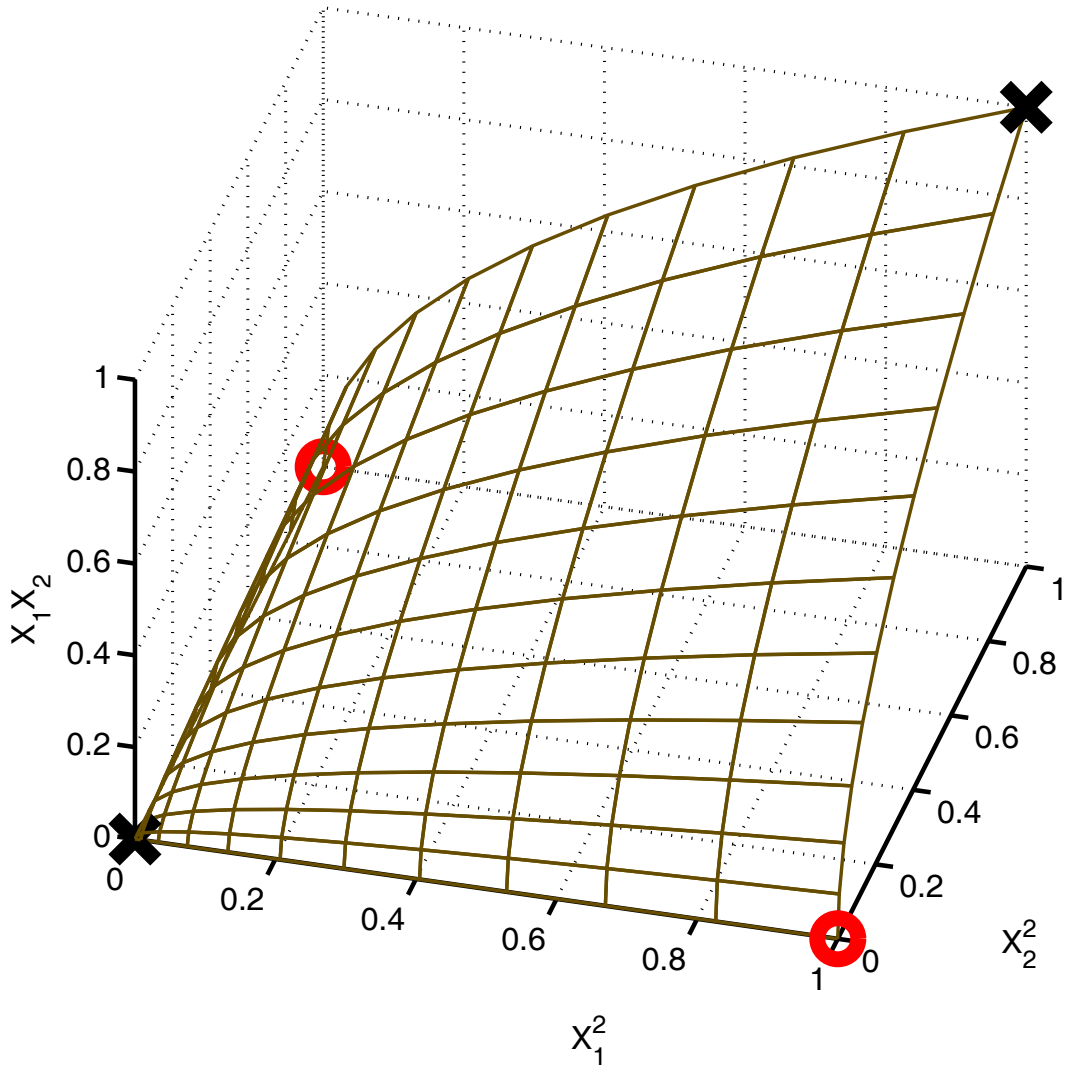
$$\begin{aligned} \Phi(x)^T \Phi(x') &= 1 \cdot 1 + \sqrt{2}x \cdot \sqrt{2}y + x^2 y^2 \\ &= (1 + xy)^2 \quad \stackrel{\text{def}}{=} \quad K(x, y) \end{aligned}$$

Kernel trick

We can translate linear algorithms into nonlinear ones using a **kernel** function – represented as a scalar product.

The XOR problem





Kernels – two-argument functions that are the generalisation of a matrix to non–countable index sets.

$$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$$

Valid kernel functions are positive definite: for any $\boldsymbol{a} = [a_1, a_2, \ldots, a_L]$ and $\mathcal{D} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L]^T$:

$$\sum_{k=1}^{L} \sum_{l=1}^{L} a_k K(\boldsymbol{x}_k, \boldsymbol{x}_l) a_l \geq 0$$

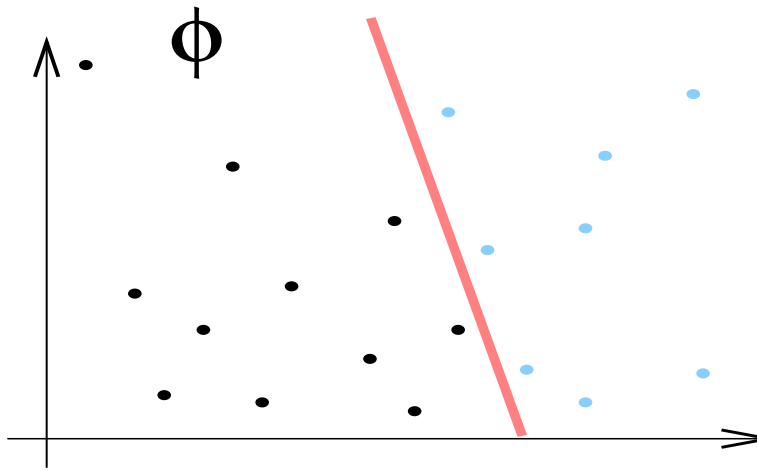
Proof idea: kernel function is $K(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{x})^T \Phi(\boldsymbol{x}')$
and we have
$\sum_{k,l=1}^{L} a_k \Phi(\boldsymbol{x}_k)^T \Phi(\boldsymbol{x}_l) a_l = \left(\sum\ a_i \Phi(\boldsymbol{x}_i)\right)^T \left(\sum\ a_i \Phi(\boldsymbol{x}_i)\right) = \boldsymbol{s}^T \boldsymbol{s} \geq 0$
where
$\boldsymbol{s} = \sum_{i=1}^{L} a_i \Phi(\boldsymbol{x}_i)$

What is the dimensionality of $\Phi(\boldsymbol{x})$.

Developed for classification.

Idea – **KERNEL TRICK**:

- use linear algorithms;
- project to Φ;
- solution in Φ;
- back-project;
- NON-linear solution;



Solution to the problem is of the form:

$$f(\boldsymbol{x}) = \Phi_{\boldsymbol{x}}^T \left(\sum_{i \in SV} \alpha_i \Phi_{\boldsymbol{x}_i} \right) = \sum_{i \in SV} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i)$$

Nonparametric: number of parameters is not fixed.

Kernel Algorithms

The projection "trick" is exploited.

- Based on the success of the S.V.M.-s;
- General recipe:
  - Find/Construct a linear algorithm;
  - Re-express it in Φ – the space of "features";
  - Write the – non–linear – solution in the space of inputs and use $K(\boldsymbol{x}, \boldsymbol{x}')$.
- Algorithms: Kernel . . .
  - . . . regression, . . . ridge regression;
  - . . . Principal/Independent Components;
  - . . . Fisher Discriminants;

Advantage / Price

Definition: non-parametric methods $\equiv$ number of parameters/complexity might change with data.

- Advantages:
  - $\uparrow$ Greater complexity;
  - $\uparrow$ Built-in regularisation;
  - $\uparrow$ Performant algorithms.
- Disadvantages:
  - $\downarrow$ Small data sets only;
  - $\downarrow$ No selection of parameters;

Advantage / Price

Definition: non-parametric methods $\equiv$ number of parameters/complexity might change with data.

- Advantages:
  - $\uparrow$ Greater complexity;
  - $\uparrow$ Built-in regularisation;
  - $\uparrow$ Performant algorithms.
- Disadvantages:
  - $\downarrow$ Small data sets only;
  - $\downarrow$ No selection of parameters;

Advantage / Price

Definition: non-parametric methods $\equiv$ number of parameters/complexity might change with data.

- Advantages:
  - $\uparrow$ Greater complexity;
  - $\uparrow$ Built-in regularisation;
  - $\uparrow$ Performant algorithms.
- Disadvantages:
  - $\downarrow$ Small data sets only;
  - $\downarrow$ No selection of parameters;

Non-parametric models

- $\mathcal{GP}$ models
  $\equiv$ functional latent variable models.
  $\equiv$ "simple" random functions.

- Appear in the likelihood:

$$P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}}) = \prod_i P(y_i|\boldsymbol{x}_i, \boldsymbol{f}) = \prod_i P(y_i|\boldsymbol{x}_i, f_{\boldsymbol{x}_i})$$

Local dependencies only: $\boldsymbol{f} \longrightarrow f_{\boldsymbol{x}}$

Gaussian processes I

Gaussian process: generalisation of a Gaussian.

- $\boldsymbol{f}$ Gaussian random function.

$$\boldsymbol{f} = [f_{\boldsymbol{x}_1}, f_{\boldsymbol{x}_2}, \ldots, f_{\boldsymbol{x}_N}, \ldots]^T, \quad \boldsymbol{x}_n \in \text{domain}.$$

- $\mathcal{GP}$ prior $p_0(\boldsymbol{f})$ characterised with
  - mean function $\langle f_{\boldsymbol{x}} \rangle_0$,
  - covariance kernel $K_0(\boldsymbol{x}, \boldsymbol{x}')$.

Property - for any sample set $\mathcal{D}$, a joint Gaussian r.v.:

$$\boldsymbol{f}_{\mathcal{D}} = [f_{\boldsymbol{x}_1}, \ldots, f_{\boldsymbol{x}_N}] \sim \mathcal{N}(\boldsymbol{f}_{\mathcal{D}} | \langle f_{\mathcal{D}} \rangle_0, \boldsymbol{K}_0)$$

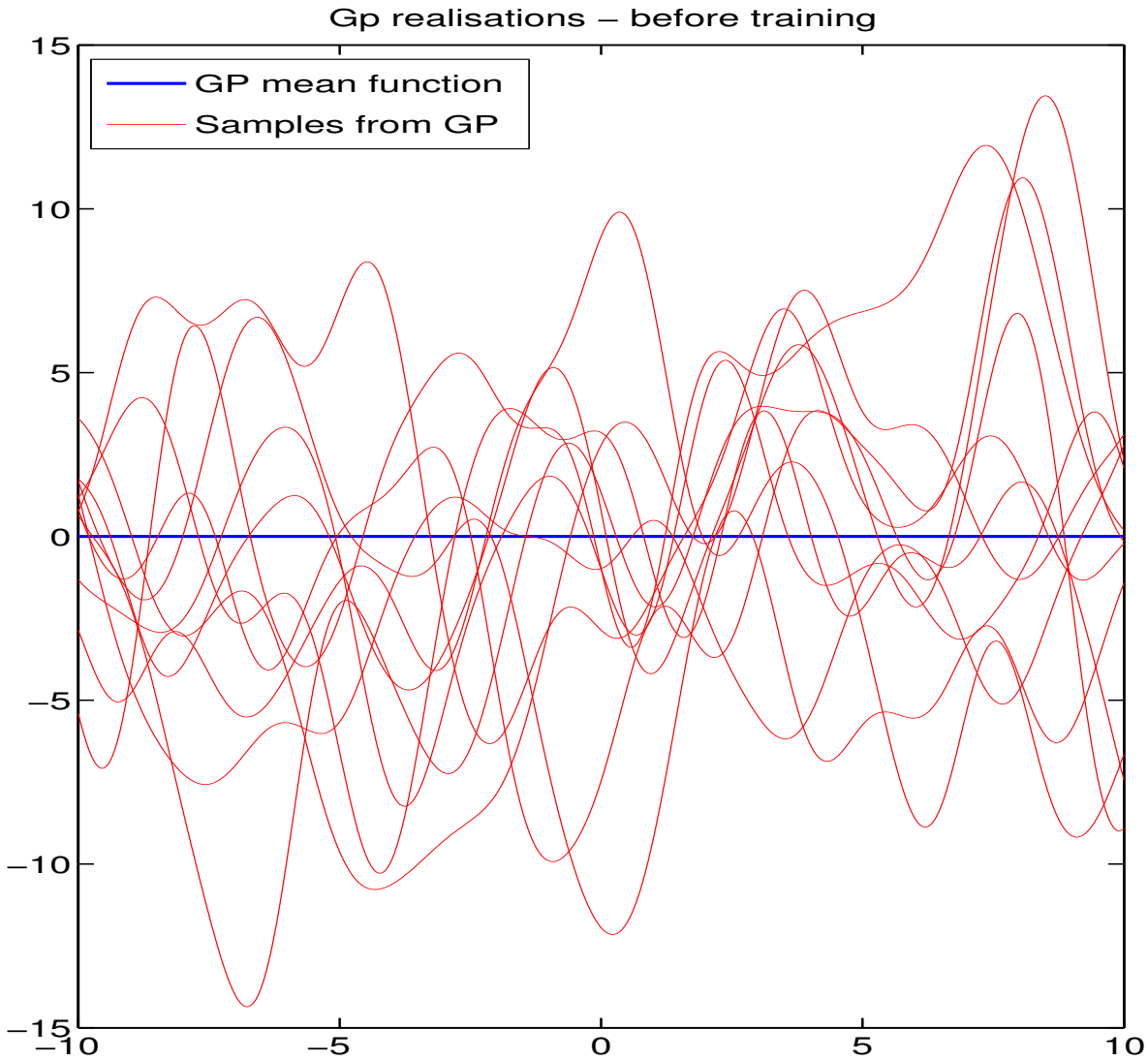
Gaussian processes II

Gaussian process: random function

"Parameters"

- mean function
- covariance kernel

DEMO



$\mathcal{GP}$ parameters

Gaussian process parameters

- mean function – usually is 0.
- **parameter:** the class of the kernel function
- parameters hidden into the kernel function. Example:

$$K(\boldsymbol{x}, \boldsymbol{x}'|\boldsymbol{\theta}) = \exp\left[\theta_0 - \frac{1}{2}\sum_{i=1}^{d}\theta_i\left(x_i - x_i'\right)^2\right]$$

$\boldsymbol{\theta} = [\theta_0, \theta_1, \ldots, \theta_d]^T$ – parameter vector.

Kernel Functions

Kernel functions:

- Generate the covariance matrix.
- Need to be positive definite functions/matrices

$$\forall \mathcal{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T\}$$

$$\boldsymbol{K}_{\mathcal{X}} = \begin{bmatrix} K(\boldsymbol{x}_1, \boldsymbol{x}_1) & \ldots & K(\boldsymbol{x}_1, \boldsymbol{x}_T) \\ \vdots & \ddots & \vdots \\ K(\boldsymbol{x}_T, \boldsymbol{x}_1) & \ldots & K(\boldsymbol{x}_T, \boldsymbol{x}_T) \end{bmatrix}$$

  must be a positive definite matrix.

- A construction of kernels as covariances:

Kernel constructions I

Let $\{x_k = k\Delta_N\}_{k=1}^N$ be the **location** of Gaussian independent r.v.-s

$$v_k \sim \mathcal{N}(0, \sigma_0^2 \Delta_N)$$

where $\Delta_N = 1/N$. Let t denote the the index where $t = k_t \Delta_N$.

If $b_t \stackrel{\text{def}}{=} \sum_{i=1}^t v_i$, then $\langle b_t \rangle = 0$ and the covariance:

$$\langle b_s b_t \rangle = \left\langle \sum_{i_s=1}^{s} \sum_{i_t=1}^{t} v_{i_s} v_{i_t} \right\rangle = \sum_{i_s=1}^{\min(s,t)} \langle v_{i_s}^2 \rangle$$

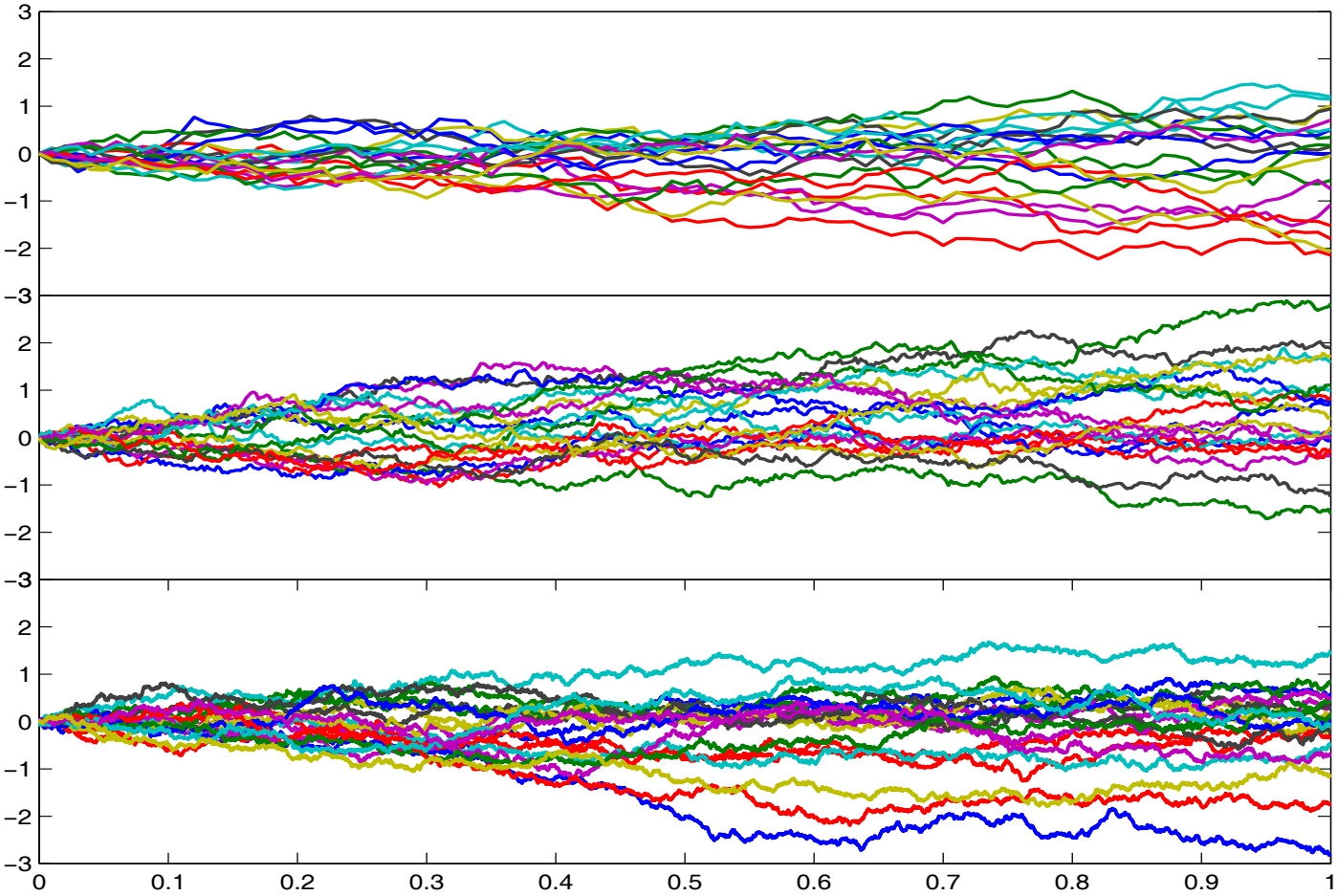
$$= \sum_{i_s=1}^{\min(s,t)} \sigma_0^2 \Delta_N = \min(s,t)\sigma_0^2$$

in the following we take $\sigma_0^2 = 1$.

Brownian motion

A stochastic process with covariance kernel $K(s,t) = \min(s,t)$ is a Brownian motion.
Its derivative – v_t – is the Wiener process.

Images of the Brownian motion at different resolutions (different N-s).



Let us integrate (sum) the Brownian motion.

Define

$$s_t \stackrel{\text{def}}{=} \sum_{i_t=1}^{t} b_{i_t}$$

leading to $\langle s_t \rangle = 0$ and

$$K(s,t) = \langle s_s s_t \rangle = \left\langle \sum_{i_s=1}^{s} \sum_{i_t=1}^{t} s_{i_s} s_{i_t} \right\rangle$$
$$= \sum_{i_s=1}^{s} \sum_{i_t=1}^{t} \langle s_{i_s} s_{i_t} \rangle = \int_0^s dz_s \int_0^t dz_t \, \min(z_s, z_t)$$

For the integration we assume

$$s < t \quad \Longrightarrow \quad z_s < t \quad \Longrightarrow \quad [0,t] = [0,z_s] \cup [z_s,t] \,.$$

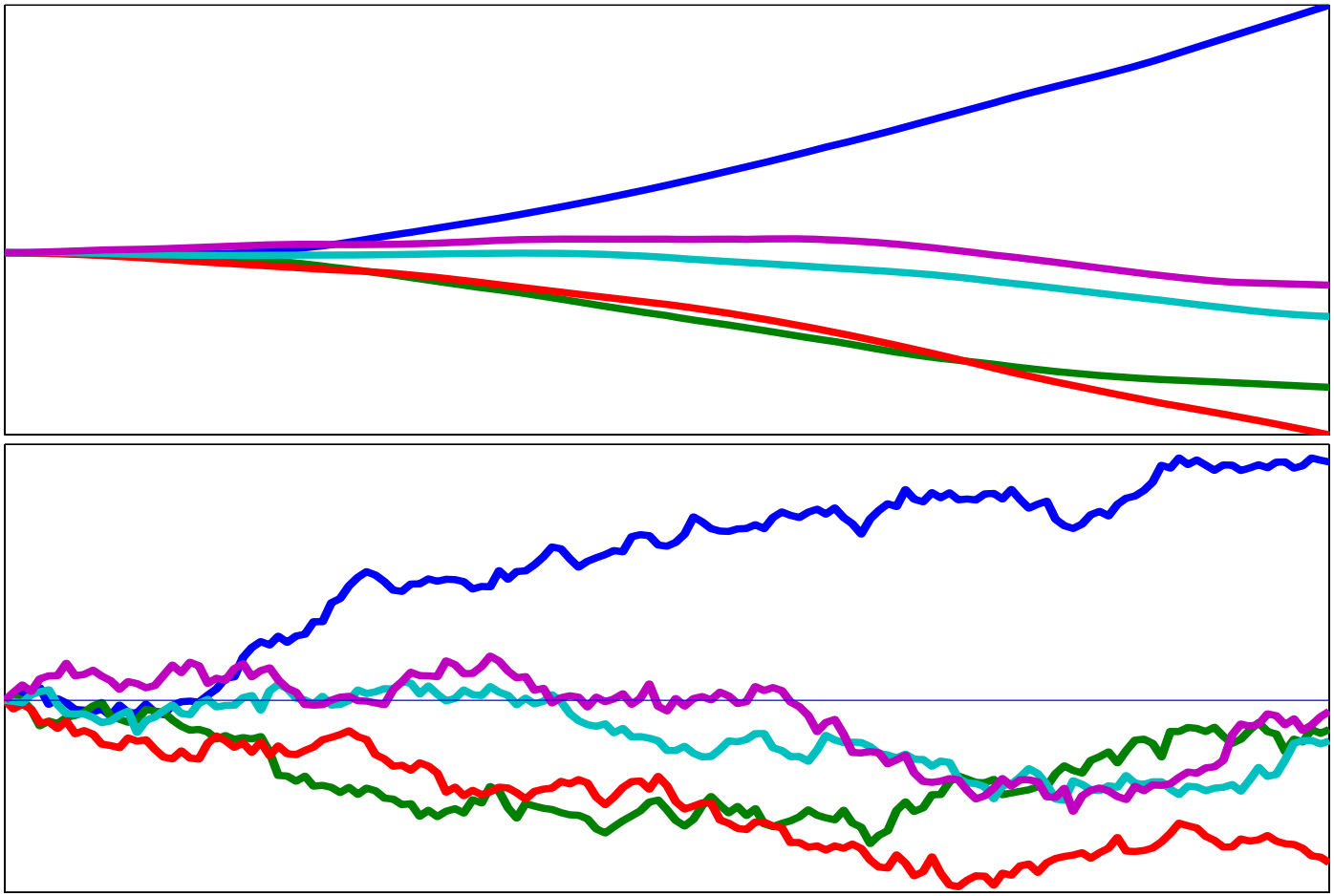
Using $[0, t] = [0, z_s] \cup [z_s, t]$

$$K(s,t) = \int_0^s dz_s \left(\int_0^{z_s} dz_t \, z_t + \int_{z_s}^t dz_t \, z_s \right) = \int_0^s dz_s \left(\frac{z_s^2}{2} + z_s(t - z_s) \right)$$
$$= \int_0^s dz_s \left(z_s t - \frac{z_s^2}{2} \right)$$
$$= \frac{s^2 t}{2} - \frac{s^3}{6} \quad \text{assuming } s < t$$

After symmetrization (writing the $s > t$ case and unifying)

$$K(s,t) = \frac{st \min(s,t)}{2} - \frac{\min(s,t)^3}{6}$$
$$= \frac{1}{2} \min(s,t)^2 |s - t| + \frac{1}{3} \min(s,t)^3$$

Samples from the integrated Brownian motion (Brownian motion on the bottom).



- The integrated Brownian motion: $s(0) = s'(0) = 0$.
- For generality we add a *constant* and a *linear* term:

$$s_2(x) = w_0 + w_1 x + s(x)$$

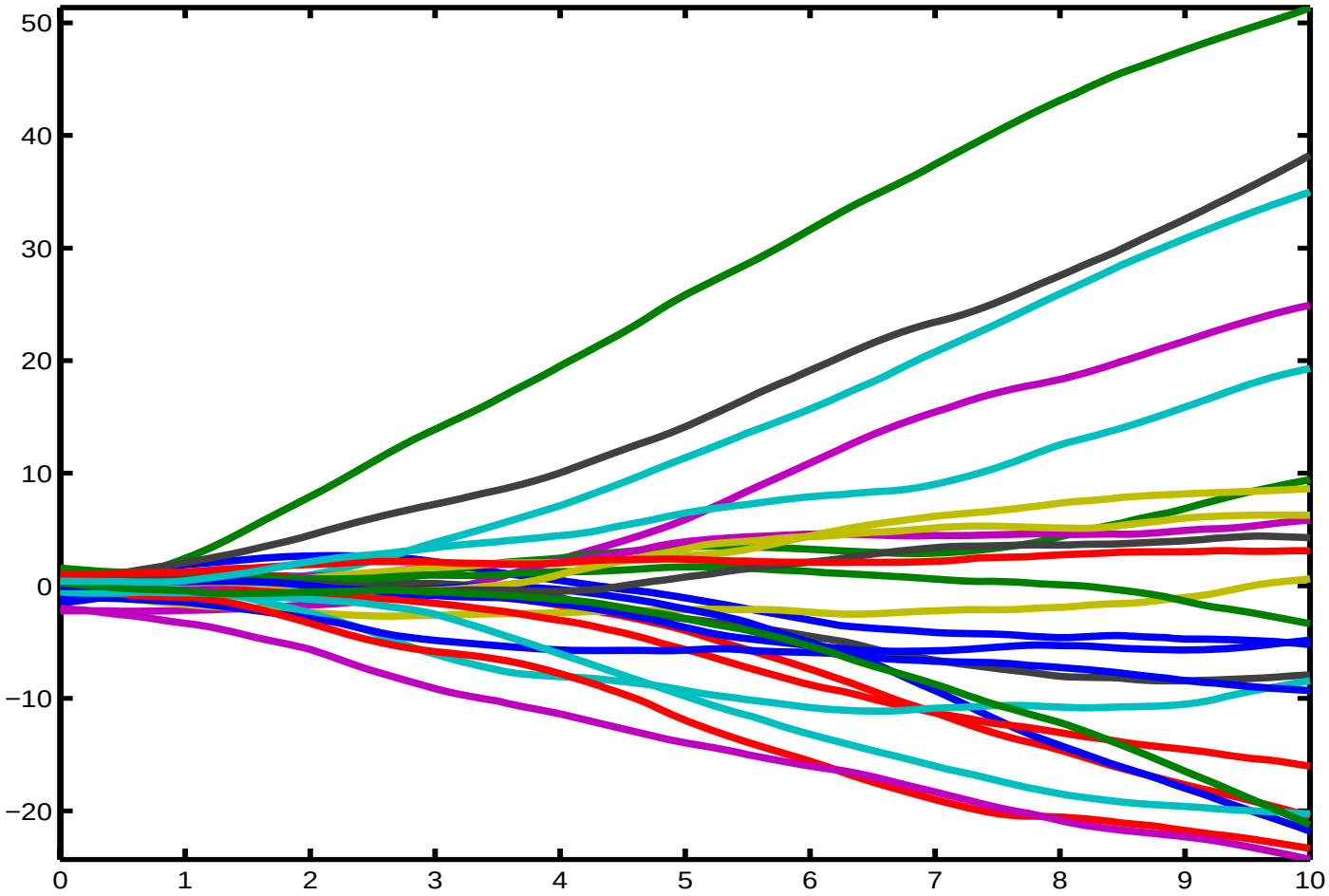
where w_0, w_1 are i.i.d. Gaussian r.v-s.

- Means that the kernel is:

$$K_2(s,t) = \langle s_2(s) s_2(t) \rangle = 1 + st + K_s(s,t)$$

$$K_2(s,t) = 1 + st + \frac{1}{2}\min(s,t)^2 |s-t| + \frac{1}{3}\min(s,t)^3$$

Random splines:



- Consider the Brownian motion: $K_b(s,t) = \min(s,t)$.
- We generate random functions with $x_1 = 0$.

  Called Brownian bridge
- To calculate the covariance, we have to condition the r.v.-s x_s and x_t on $x_1 = 0$.

We identify the kernel from the conditioned **joint** Gaussian distribution

$$p(x_s, x_t | x_1 = 0) \propto \exp\left\{ -\frac{1}{2} \begin{bmatrix} x_s \\ x_t \\ 0 \end{bmatrix}^T \begin{bmatrix} K_b(s,s) & K_b(s,t) & s \\ K_b(t,s) & K_b(t,t) & t \\ s & t & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_s \\ x_t \\ 0 \end{bmatrix} \right\}$$

$$\propto \exp\left\{ -\frac{1}{2} \begin{bmatrix} x_s \\ x_t \\ 0 \end{bmatrix}^T \begin{bmatrix} \left(\begin{bmatrix} K_b(s,s) & K_b(s,t) \\ K_b(t,s) & K_b(t,t) \end{bmatrix} - \begin{bmatrix} s \\ t \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix}^T \right)^{-1} & E \\ E^T & F \end{bmatrix} \begin{bmatrix} x_s \\ x_t \\ 0 \end{bmatrix} \right\}$$

where we used the matrix inversion lemma

$$\Rightarrow \qquad K_0(s,t) = K_b(s,t) - st$$

Kernels from conditioning VIII

- Consider the Brownian motion: $K_b(s,t) = \min(s,t)$.
- We generate random functions with $x_1 = 0$.
  Called Brownian bridge
- To calculate the covariance, we have to condition the r.v.-s x_s and x_t on $x_1 = 0$.

We identify the kernel from the conditioned **joint** Gaussian distribution

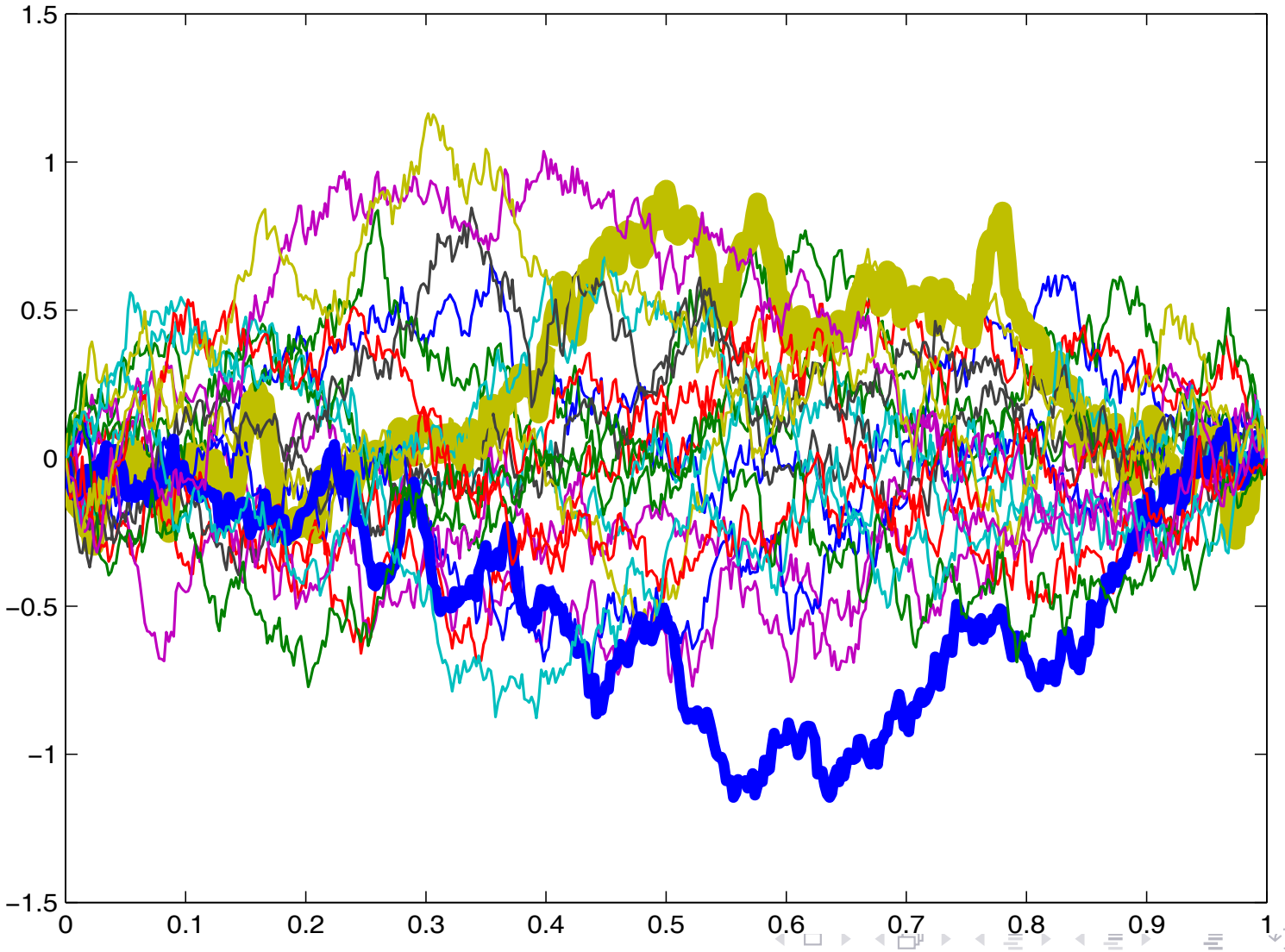
$$p(x_s, x_t | x_1 = 0) \propto \exp\left\{ -\frac{1}{2} \begin{bmatrix} x_s \\ x_t \\ 0 \end{bmatrix}^T \begin{bmatrix} K_b(s,s) & K_b(s,t) & s \\ K_b(t,s) & K_b(t,t) & t \\ s & t & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_s \\ x_t \\ 0 \end{bmatrix} \right\}$$

$$\propto \exp\left\{ -\frac{1}{2} \begin{bmatrix} x_s \\ x_t \\ 0 \end{bmatrix}^T \begin{bmatrix} \left(\begin{bmatrix} K_b(s,s) & K_b(s,t) \\ K_b(t,s) & K_b(t,t) \end{bmatrix} - \begin{bmatrix} s \\ t \end{bmatrix} \begin{bmatrix} s \\ t \end{bmatrix}^T \right)^{-1} & E \\ E^T & F \end{bmatrix} \begin{bmatrix} x_s \\ x_t \\ 0 \end{bmatrix} \right\}$$

where we used the matrix inversion lemma

$$\Rightarrow \qquad K_0(s,t) = K_b(s,t) - st$$

Samples from a Brownian bridge:



Exercise:

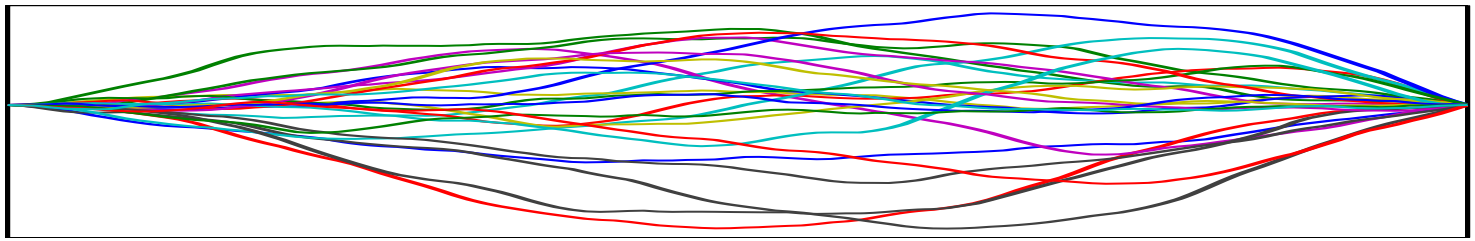
- Find the mean and kernel functions corresponding to the Brownian bridge where $x_1 = 1$.

- Consider the spline kernel

$$K_s(s,t) = \frac{1}{2}\min(s,t)^2|s-t| + \frac{1}{3}\min(s,t)^3$$

Similarly to the Brownian bridge, find the kernel function for the splines conditioned on $x_1 = 0$.

Samples from the second family look like this:



Generating random samples


```
clear all;
N = 100;  T=25;  D = 0.99;
t = linspace(0,D,N+1);
t = t(2:end);

% put covariance function here
k_bb= inline('min(s,t)-s*t','s','t');

Ks = zeros(N,N);
for ii=1:N;
    for jj=1:N;
        Ks(ii,jj) = k_sb(t(ii),t(jj));
    end;
end;

kks = chol(Ks);
yr = randn(T,N);
ys = zeros(N+2,T);
ys(2:N+1,:) = (yr*kks)';

t0=[0,t,1];
figure(1); cla; box on; hold on;
plot(t0,ys);
```

Gaussian Process Inference I

- $\mathcal{GP}$ inference: application of Bayes' rule.

$$p_{\text{post}}(\boldsymbol{f}, \boldsymbol{f}_{\mathcal{D}}) \propto P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}})\; p_0(\boldsymbol{f}_{\mathcal{D}}, \boldsymbol{f})$$

- For any collection of indexes $\mathcal{X}$ the posterior:

$$p_{\text{post}}(\boldsymbol{f}_{\mathcal{X}}) = \frac{1}{Z_{\mathcal{D}}} \int d\boldsymbol{f}_{\mathcal{D}}\; P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}})\; p_0(\boldsymbol{f}_{\mathcal{D}}, \boldsymbol{f}_{\mathcal{X}})$$

where

$$Z_{\mathcal{D}} = \int d\boldsymbol{f}_{\mathcal{D}}\; P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}})\; p_0(\boldsymbol{f}_{\mathcal{D}})$$

Data probability – conditioned on the model.

A.R.D.

obs: NO specific $P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}})$

Problems with the representation

$$p_{\text{post}}(\boldsymbol{f}_{\mathcal{X}}) \propto \int d\boldsymbol{f}_{\mathcal{D}}\; P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}})\; p_0(\boldsymbol{f}_{\mathcal{D}}, \boldsymbol{f}_{\mathcal{X}})$$

Integral evaluation necessary for posterior distribution.

- Representation – How to represent the posterior?
  - Finite representation of the posterior process;
  - Non–Gaussian posterior processes: approximations to them

Property of Gaussian averages:

$$\langle f_{\boldsymbol{x}} \rangle_{\text{post}} = \langle f_{\boldsymbol{x}} \rangle_0 + \sum_i K_0(\boldsymbol{x}, \boldsymbol{x}_i)\alpha(i)$$

Where coefficients:

$$\alpha(i) = \frac{\partial}{\partial \langle f_i \rangle_0} \ln \left\langle P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}}) \right\rangle_0$$

Provide **parametrisation** (see Kimeldorf-Wahba).

For the posterior kernel:

$$K_{\text{post}}(\boldsymbol{x}, \boldsymbol{x}') = K_0(\boldsymbol{x}, \boldsymbol{x}') + \sum_{i,j} K_0(\boldsymbol{x}, \boldsymbol{x}_i) C(ij) K_0(\boldsymbol{x}_j, \boldsymbol{x}')$$

Where coefficients:

$$C(ij) = \frac{\partial^2}{\partial \langle f_i \rangle_0 \partial \langle f_j \rangle_0} \ln \left\langle P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}}) \right\rangle_0$$

$\mathcal{GP}$s in Feature space I

$K_0(\boldsymbol{x}, \boldsymbol{x}')$ defines a *feature space* $\mathcal{F}$:

$$\phi_{\boldsymbol{x}}, \phi_{\boldsymbol{x}'} \in \mathcal{F} \quad \text{and} \quad K_0(\boldsymbol{x}, \boldsymbol{x}') = \phi_{\boldsymbol{x}}^T \phi_{\boldsymbol{x}'}$$

Using $\mathcal{F}$ and the scalar product:

$$\langle f_{\boldsymbol{x}} \rangle_{\text{post}} = \phi_{\boldsymbol{x}}^T \sum_{i=1}^{N} \alpha(i) \phi_i = \phi_{\boldsymbol{x}}^T \boldsymbol{\mu}_{\text{post}}$$

$$K_{\text{post}}(\boldsymbol{x}, \boldsymbol{x}') = \phi_{\boldsymbol{x}}^T \left(\boldsymbol{I}_{\mathcal{F}} + \sum_{i,j=1}^{N} \phi_i C(ij) \phi_j^T \right) \phi_{\boldsymbol{x}'} = \phi_{\boldsymbol{x}}^T \boldsymbol{\Sigma}_{\text{post}} \phi_{\boldsymbol{x}'}$$

$\mathcal{GP}$s in Feature space II

$K_0(\boldsymbol{x}, \boldsymbol{x}')$ defines a *feature space* $\mathcal{F}$:

$$\phi_{\boldsymbol{x}}, \phi_{\boldsymbol{x}'} \in \mathcal{F} \quad \text{and} \quad K_0(\boldsymbol{x}, \boldsymbol{x}') = \phi_{\boldsymbol{x}}^T \phi_{\boldsymbol{x}'}$$

$$\begin{aligned} \langle f_{\boldsymbol{x}} \rangle_{\text{post}} &\Longleftrightarrow \boldsymbol{\mu}_{\text{post}} \\ K_{\text{post}}(\boldsymbol{x}, \boldsymbol{x}') &\Longleftrightarrow \boldsymbol{\Sigma}_{\text{post}} \end{aligned}$$

$\mathcal{GP}$ inference:

Estimating a Gaussian distribution in $\mathcal{F}$

Prediction with Gaussian processes

Given: $\boldsymbol{x}^*$ - for which we require answer y^*.

$$
\begin{aligned}
p(y^*|\boldsymbol{x}^*, \mathcal{D}) &= \int df^* \int d\boldsymbol{f}_{\mathcal{D}}\; p(y^*, \boldsymbol{f}_{\mathcal{D}}, f^*|\boldsymbol{x}^*, \mathcal{D}) \\
&= \int df^*\; P(y^*|\boldsymbol{x}^*, f^*) \int d\boldsymbol{f}_{\mathcal{D}}\; p_{\text{post}}(\boldsymbol{f}_{\mathcal{D}}, f^*|\mathcal{D}) \\
&= \int df^*\; P(y^*|\boldsymbol{x}^*, f^*) p_{\text{post}}(f^*|\mathcal{D})
\end{aligned}
$$

where $f^* = f_{\boldsymbol{x}^*}$ – random variable associated to $\boldsymbol{x}^*$.

We use posterior process:

- irrespective of the likelihood;
- if not Gaussian, we approximate.

Regression with Gaussian noise

- Gaussian noise:

$$P(\boldsymbol{y}_{\mathcal{D}}|\boldsymbol{f}_{\mathcal{D}}) \propto \exp\left[-\frac{1}{2\sigma_o^2}\left\|\boldsymbol{y}_{\mathcal{D}} - \boldsymbol{f}_{\mathcal{D}}\right\|^2\right]$$

- Gaussian latent variables:

$$P(\boldsymbol{f}_{\mathcal{D}}|K(\cdot,\cdot|\boldsymbol{\theta})) \propto \exp\left[-\frac{1}{2}\left(\boldsymbol{f}_{\mathcal{D}} - \boldsymbol{\mu}_{\mathcal{D}}\right)^T \boldsymbol{K}_{\mathcal{D}}{}^{-1}\left(\boldsymbol{f}_{\mathcal{D}} - \boldsymbol{\mu}_{\mathcal{D}}\right)\right]$$

- Combining: product quadratic $\Longrightarrow$ Gaussian

Posterior distribution – Gaussian noise

Gaussian distribution – reading off the coefficients:

$$\boldsymbol{\mu}_{\text{post}} = \left(\boldsymbol{K}_{\mathcal{D}}{}^{-1} + \frac{1}{\sigma_o^2}\boldsymbol{I}_{\mathcal{N}}\right)^{-1}\left[\boldsymbol{K}_{\mathcal{D}}{}^{-1}\mu_D + \frac{1}{\sigma_o^2}\boldsymbol{y}_{\mathcal{D}}\right]$$

$$\boldsymbol{\Sigma}_{\text{post}} = \left(\boldsymbol{K}_{\mathcal{D}}{}^{-1} + \frac{1}{\sigma_o^2}\boldsymbol{I}_{\mathcal{N}}\right)^{-1}$$

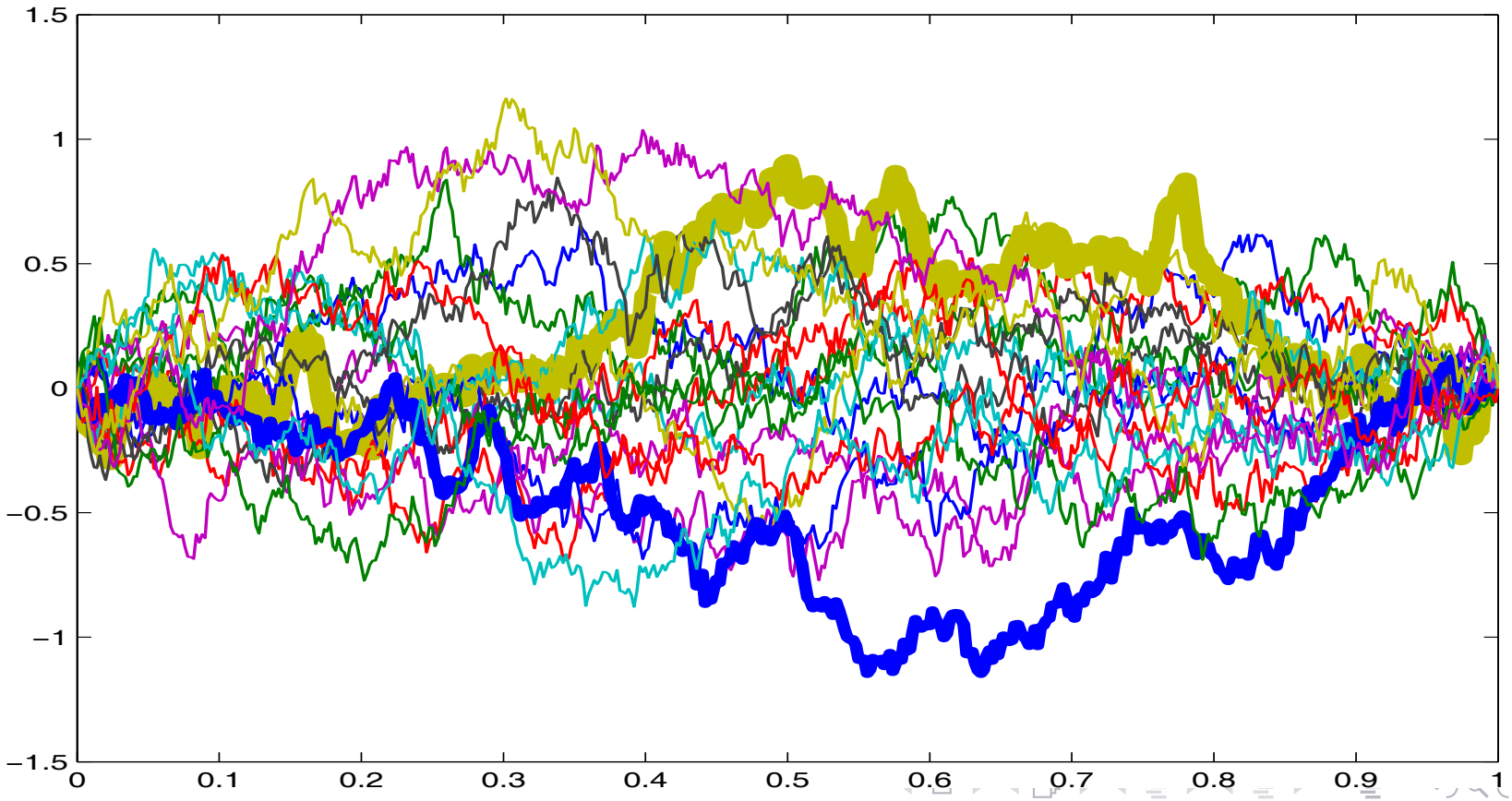
The **joint** distribution of all r.v.-s is a Gaussian:

$$\boldsymbol{f}_{\mathcal{D}} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{post}}, \boldsymbol{\Sigma}_{\text{post}})$$

The distribution of the r.v.-s at **training locations**.

Posterior distribution — Brownian bridge

- **Assume** that there was an observation for the Brownian motion $k(\boldsymbol{x}, \boldsymbol{x}') = \min(\boldsymbol{x}, \boldsymbol{x}')$;
- at 1 the value of the process is 0.
- It generates a new process: $k_B(\boldsymbol{x}, \boldsymbol{x}') = \min(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{x}\boldsymbol{x}'$



Predictive distributions – Gaussian noise

For test location x^*:

$$\mu^* = \boldsymbol{\alpha}_{\mathcal{D}}{}^T \boldsymbol{k}_*$$
$$\sigma^* = K(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}_*^T \boldsymbol{C}_{\mathcal{D}} \boldsymbol{k}_*$$

with

$$\boldsymbol{\alpha}_{\mathcal{D}} = \boldsymbol{C}_{\mathcal{D}} * \boldsymbol{y}_{\mathcal{D}}$$
$$\boldsymbol{C}_{\mathcal{D}} = \left(\boldsymbol{K}_{\mathcal{D}} + \sigma_o^2 \boldsymbol{I}_N\right)^{-1}$$

where $\boldsymbol{k}_* = [K(\boldsymbol{x}^*, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}^*, \boldsymbol{x}_N)]^T$

Posterior mean and covariance functions:

$$\mu(\boldsymbol{x}) = \boldsymbol{\alpha}_{\mathcal{D}}{}^T \boldsymbol{k}_{\boldsymbol{x}}$$
$$K_{\text{post}}(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}_{\boldsymbol{x}}^T \boldsymbol{C}_{\mathcal{D}} \boldsymbol{k}_{\boldsymbol{x}'}$$

where $\boldsymbol{k}_{\boldsymbol{x}} = [K(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, K(\boldsymbol{x}, \boldsymbol{x}_N)]^T$.

Non-computable Posteriors

- **If likelihood non–Gaussian** $\Rightarrow$ posterior does not have analytical form.
  (No "summarising" statistics)

- Methods to obtain posterior:
  - Sampling;
  - Analytic approximations;

Sampling from the posterior I

Sampling

$$p_{\text{post}}(\boldsymbol{f}_{\mathcal{X}}) = \frac{1}{Z_{\mathcal{D}}} \int d\boldsymbol{f}_{\mathcal{D}}\; P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}}) p_0(\boldsymbol{f}_{\mathcal{D}}, \boldsymbol{f}_{\mathcal{X}})$$

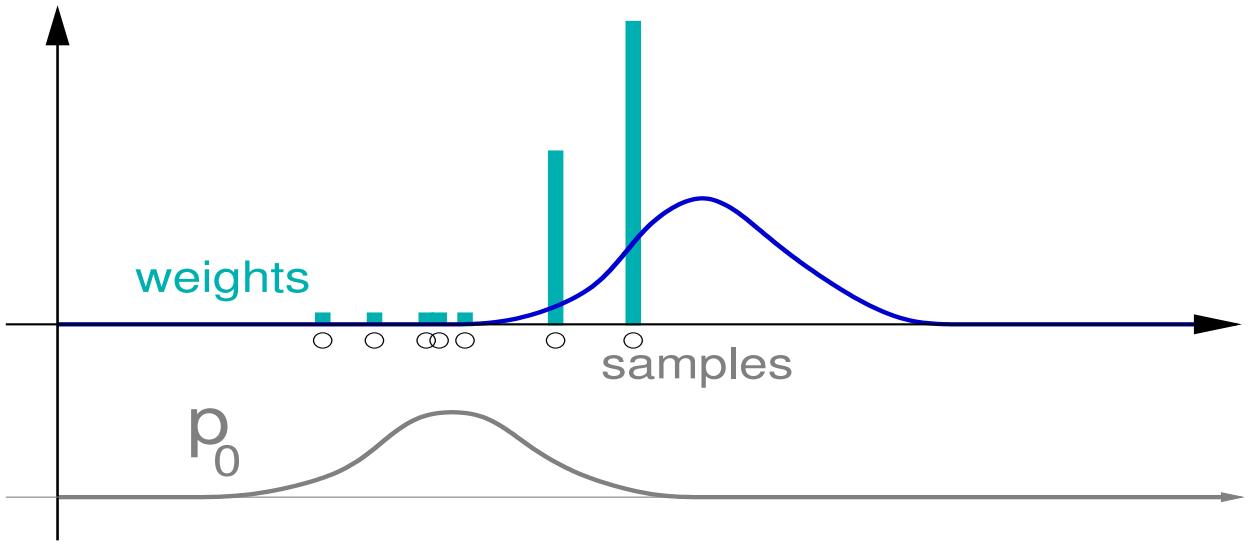
In practise: – joint sampling from $p_{\text{post}}(\boldsymbol{f}_{\mathcal{X}}, \boldsymbol{f}_{\mathcal{D}})$,
– keeping only $\boldsymbol{f}_{\mathcal{X}}$.

Implementation: sampling from $p_0(\boldsymbol{f}_{\mathcal{D}}, \boldsymbol{f}_{\mathcal{X}})$ + weighting:

$$p_{\text{post}}(\boldsymbol{f}_{\mathcal{X}}) \approx \frac{1}{C_T} \sum_{t=1}^{T} P(\boldsymbol{y}_N|\boldsymbol{f}_{\mathcal{D}}^{(i)})\; \delta(\boldsymbol{f}_{\mathcal{X}} - \boldsymbol{f}_{\mathcal{X}}^{(i)})$$

with $C_T = \sum_{t=1}^{T} P(\boldsymbol{y}_N|\boldsymbol{f}_{\mathcal{D}}^{(i)})$

Sampling from the posterior II



Sampling methods:

- powerful *i.e.* allow flexibility in modelling
- Hard to assess convergence
- Sampling algorithms suited for different models.
- Can be incredibly slow (tempering, MCMC)

Laplace Approximation

Log–Posterior:

$$\log p_{\text{post}}(\boldsymbol{f}_{\mathcal{X}}, \boldsymbol{f}_{\mathcal{D}}) = K + \log P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}}) + \log p_0(\boldsymbol{f}_{\mathcal{D}}, \boldsymbol{f}_{\mathcal{X}})$$
$$= \underbrace{\log P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}}) + \log p_0(\boldsymbol{f}_{\mathcal{D}})}_{g_{\mathcal{D}}(\boldsymbol{f}_{\mathcal{D}})} + \underbrace{\log p_0(\boldsymbol{f}_{\mathcal{X}}|\boldsymbol{f}_{\mathcal{D}})}_{g_{\mathcal{X}}(\boldsymbol{f}_{\mathcal{X}})}$$

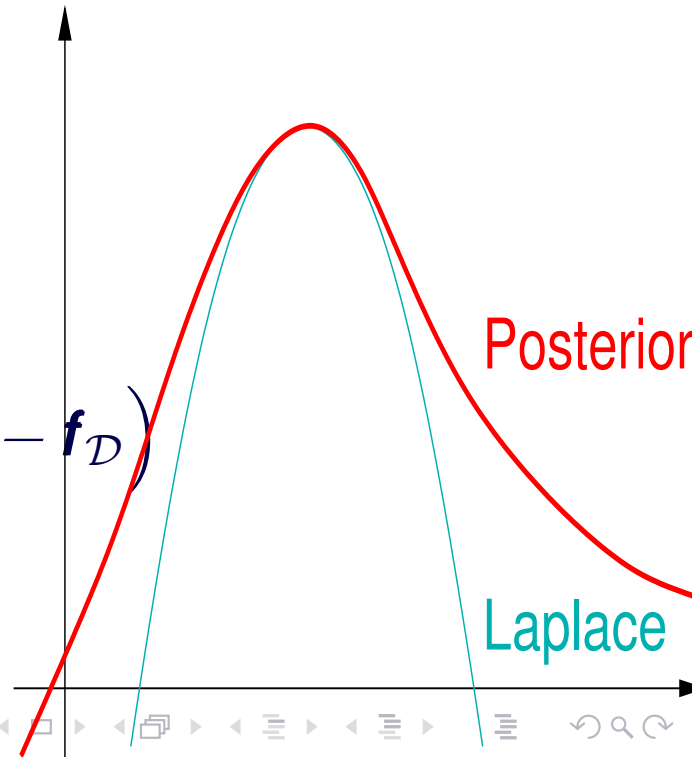
Finding maximum of $\boldsymbol{f}_{\mathcal{X}}$ and $\boldsymbol{f}_{\mathcal{D}}$:

$$\hat{\boldsymbol{f}}_{\mathcal{X}} = \boldsymbol{P}_{\mathcal{X}\mathcal{D}}\, \boldsymbol{f}_{\mathcal{D}}$$
$$\hat{\boldsymbol{f}}_{\mathcal{D}} = \arg\max g_{\mathcal{D}}(\boldsymbol{f}_{\mathcal{D}})$$

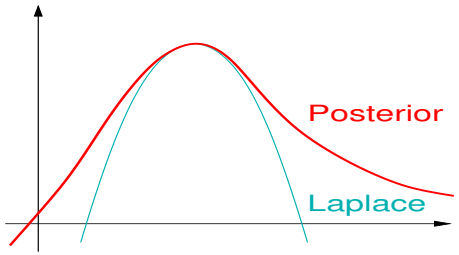
Taylor expansion around $\hat{\boldsymbol{f}}_{\mathcal{D}}$:

$$g_{\mathcal{D}}(\boldsymbol{f}_{\mathcal{D}}) \approx \left(\hat{\boldsymbol{f}}_{\mathcal{D}} - \boldsymbol{f}_{\mathcal{D}}\right)^T \left[H_g(\hat{\boldsymbol{f}}_{\mathcal{D}})\right] \left(\hat{\boldsymbol{f}}_{\mathcal{D}} - \boldsymbol{f}_{\mathcal{D}}\right)$$
$$+ \mathbf{0}\left(\hat{\boldsymbol{f}}_{\mathcal{D}} - \boldsymbol{f}_{\mathcal{D}}\right) + g_{\mathcal{D}}(\hat{\boldsymbol{f}}_{\mathcal{D}})$$

$\Rightarrow$ **Gaussian**



Gaussian approximation:



$$\hat{p}_{\text{post}}(\boldsymbol{f}_{\mathcal{X}}, \boldsymbol{f}_{\mathcal{D}}) \propto p_0(\boldsymbol{f}_{\mathcal{X}}, \boldsymbol{f}_{\mathcal{D}}) \underbrace{\frac{\mathcal{N}_L\left(\boldsymbol{f}_{\mathcal{D}} \middle| \hat{\boldsymbol{f}}_{\mathcal{D}}, \left[H_g(\hat{\boldsymbol{f}}_{\mathcal{D}})\right]^{-1}\right)}{p_0(\boldsymbol{f}_{\mathcal{D}})}}_{\hat{P}(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}})}$$

Defines an approximation to the **likelihood**:

$$\hat{P}(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}}) \propto \frac{\mathcal{N}_L\left(\boldsymbol{f}_{\mathcal{D}} \middle| \hat{\boldsymbol{f}}_{\mathcal{D}}, \left[H_g(\hat{\boldsymbol{f}}_{\mathcal{D}})\right]^{-1}\right)}{p_0(\boldsymbol{f}_{\mathcal{D}})}$$

The Laplace approximation:

- \+ generates an approximation to the *likelihood*;
- − applicable only for differentiable likelihood functions;
- \+ defines an approximation to the *whole* process;
- − the Hessian has to be positive definite and **"smaller" than the prior**

The Laplace approximation:

- \+ generates an approximation to the *likelihood*;
- − applicable only for differentiable likelihood functions;
- \+ defines an approximation to the *whole* process;
- − the Hessian has to be positive definite and **"smaller" than the prior**

The Laplace approximation:

- \+ generates an approximation to the *likelihood*;
- − applicable only for differentiable likelihood functions;
- \+ defines an approximation to the *whole* process;
- − the Hessian has to be positive definite and **"smaller" than the prior**

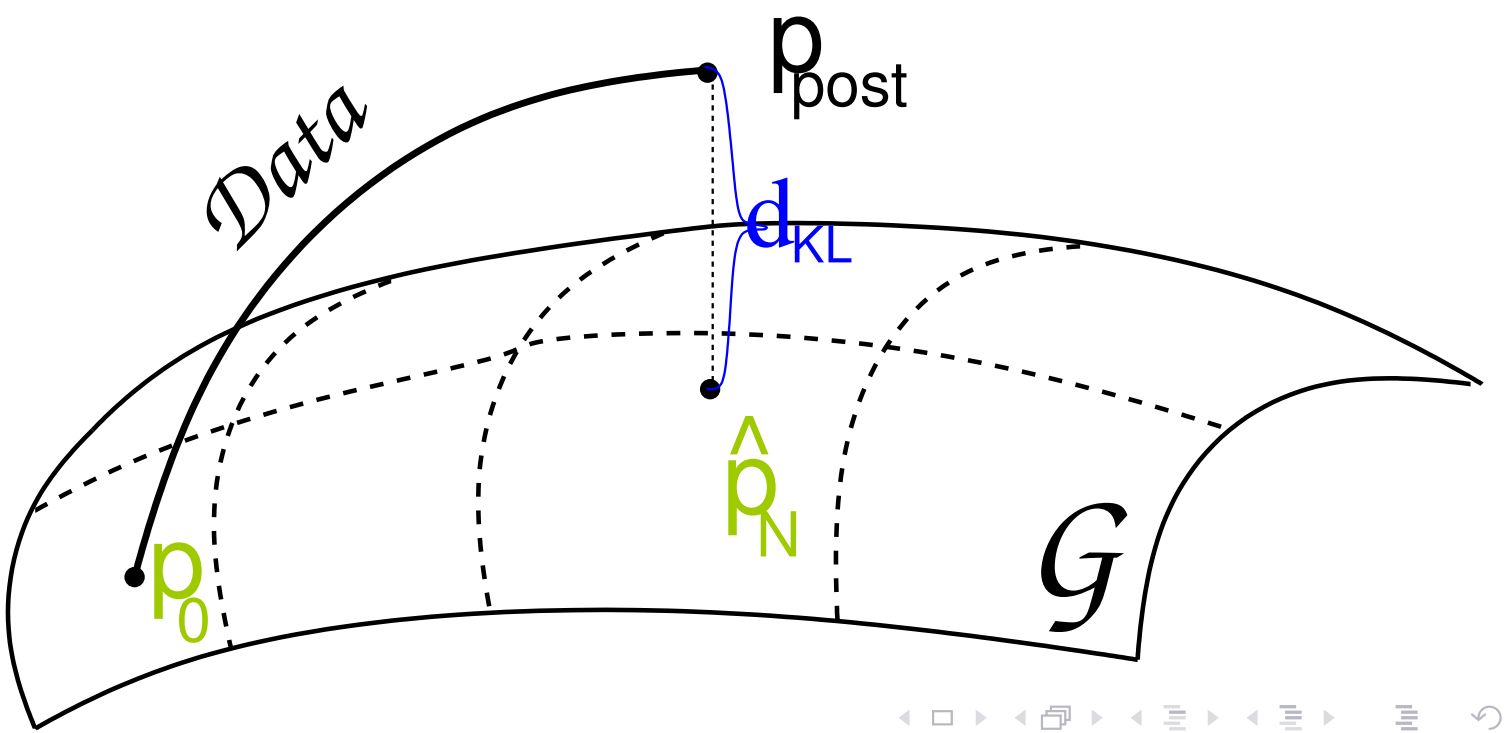
The Laplace approximation:

- \+ generates an approximation to the *likelihood*;
- − applicable only for differentiable likelihood functions;
- \+ defines an approximation to the *whole* process;
- − the Hessian has to be positive definite and **"smaller" than the prior**

Analytic approximations – I

Aim: approximate the posterior distribution – or the posterior process.

$\mathcal{GP}$ **prior** $\longrightarrow$ $\mathcal{GP}$ **approximation to posterior.**
projection – closest $\mathcal{GP}$.



Analytic approximations – II

Choice of projection: Kullback-Leibler divergence

$$\mathrm{KL}(\mathcal{GP}_{\text{post}} \| \mathcal{GP}) = \int d\mathcal{GP}_{\text{post}}(\boldsymbol{f}) \; \log \frac{d\mathcal{GP}_{\text{post}}(\boldsymbol{f})}{d\mathcal{GP}(\boldsymbol{f})}$$

$$\mathcal{GP}^* = \arg\min_{\mathcal{GP}} \mathrm{KL}(\mathcal{GP}_{\text{post}} \| \mathcal{GP})$$

The minimiser:

$$\begin{aligned} \langle f_x \rangle_{\mathcal{GP}^*} &\stackrel{\text{def}}{=} \langle f_x \rangle_{\text{post}} \\ K_{\mathcal{GP}^*}(\boldsymbol{x}, \boldsymbol{x}') &\stackrel{\text{def}}{=} K_{\text{post}}(\boldsymbol{x}, \boldsymbol{x}') \end{aligned}$$

Implies that the KL-approximation the $\mathcal{GP} \Leftrightarrow (\boldsymbol{\alpha}_{\mathcal{D}}, \boldsymbol{C}_{\mathcal{D}})$.

Computing KL-distances

Between $\mathcal{GP}$s with the same prior:
$\mathcal{GP}_1 = \mathcal{N}(\boldsymbol{\mu}_1, \Sigma_1)$ and $\mathcal{GP}_2 = \mathcal{N}(\boldsymbol{\mu}_2, \Sigma_2)$, the KL-distance is:

$$2\mathrm{KL}(\mathcal{GP}_1 \| \mathcal{GP}_2) = (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^T \Sigma_2^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) + \operatorname{tr}\left(\Sigma_1 \Sigma_2^{-1} - \boldsymbol{I}_{\mathcal{F}}\right) - \ln \left|\Sigma_1 \Sigma_2^{-1}\right|$$

With parameters ($\boldsymbol{Q}_{\mathcal{BV}} = \boldsymbol{K}_{\mathcal{BV}}^{-1}$):

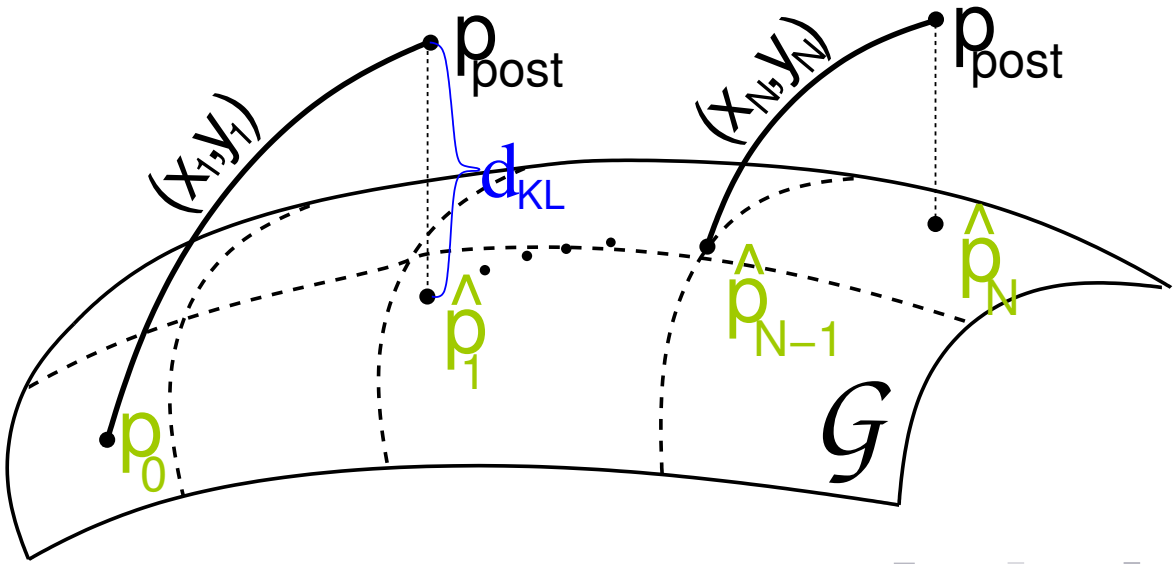
$$(\boldsymbol{\alpha}_2 - \boldsymbol{\alpha}_1) (\boldsymbol{C}_2 + \boldsymbol{Q}_{\mathcal{BV}})^{-1} (\boldsymbol{\alpha}_2 - \boldsymbol{\alpha}_1)$$
$$\operatorname{tr}\left[(\boldsymbol{C}_1 - \boldsymbol{C}_2)(\boldsymbol{C}_2 + \boldsymbol{Q}_{\mathcal{BV}})^{-1}\right] - \ln \left|(\boldsymbol{C}_1 + \boldsymbol{Q}_{\mathcal{BV}}) (\boldsymbol{C}_2 + \boldsymbol{Q}_{\mathcal{BV}})^{-1}\right|$$

Assumptions: the kernel matrix on the $\mathcal{BV}$ set is non-singular $|\boldsymbol{K}_{\mathcal{BV}}| \neq 0$.

Bayesian Online Learning (recursive)

- Instead of $|\mathcal{D}| = N$ uses $\mathcal{D} = (\boldsymbol{x}_{t+1}, y_{t+1})$ and;
- For prior process $\langle f_{\boldsymbol{x}} \rangle_t$, $K_t(\boldsymbol{x}, \boldsymbol{x}')$.

$$\mathrm{KL}(\mathcal{GP}^{t+1}_{\text{post}} \| \mathcal{GP}^*) \quad \text{smaller approximation}$$



Bayesian Online Learning

Learning: propagating the mean and the kernel:

$$\langle f_{\boldsymbol{x}} \rangle_{t+1} = \langle f_{\boldsymbol{x}} \rangle_t + q\, K_t(\boldsymbol{x}, \boldsymbol{x}_{t+1})$$
$$K_{t+1}(\boldsymbol{x}, \boldsymbol{x}') = K_t(\boldsymbol{x}, \boldsymbol{x}') + r\, K_t(\boldsymbol{x}, \boldsymbol{x}_{t+1}) K_t(\boldsymbol{x}_{t+1}, \boldsymbol{x}')$$

q, r functions of the **single** likelihood:

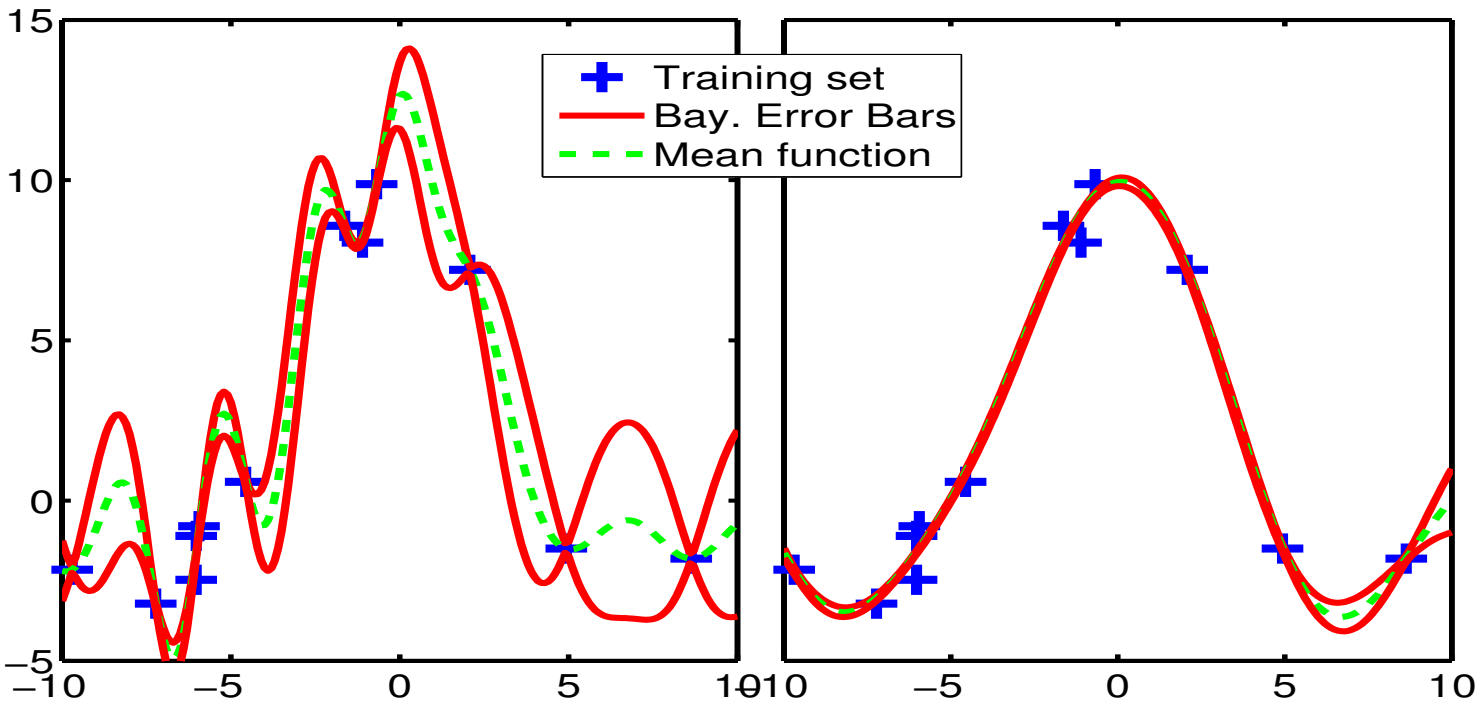
$$q = q^{(t+1)} = \frac{\partial}{\partial \langle f_{t+1} \rangle_t} \ln \langle P(y_{t+1} | f_{t+1}) \rangle_t$$

where $\langle \cdot \rangle_t$ average w.r.to $f_{t+1} \sim \mathcal{N}_t(\langle f_{t+1} \rangle, \sigma^2_{t+1})$.

Optimising hyper-parameters I

$\mathcal{GP}$ **kernel** parameters $\Leftrightarrow$ model choice.
Exemplu:

RBF kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = A \exp\left[-\sum (x - x')^2 \beta\right]$



$\beta = 0.1$ $\beta = 1.$

Optimising hyper-parameters II

Model evidence:

$$Z_{\mathcal{D}}(\boldsymbol{\theta}) = P(\mathcal{D}|\boldsymbol{\theta}) = \int d\boldsymbol{f}_{\mathcal{D}}\; P(\mathcal{D}|\boldsymbol{f}_{\mathcal{D}})\; p_0(\boldsymbol{f}_{\mathcal{D}}|\theta)$$

Maximum Likelihood II inference

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Omega} P(\boldsymbol{\theta}|\mathcal{D}) = \arg\min_{\boldsymbol{\theta}\in\Omega} \frac{P(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{M})}{p(\mathcal{D}|\mathcal{M})}$$

if $p(\boldsymbol{\theta}|\mathcal{M})$ "flat"

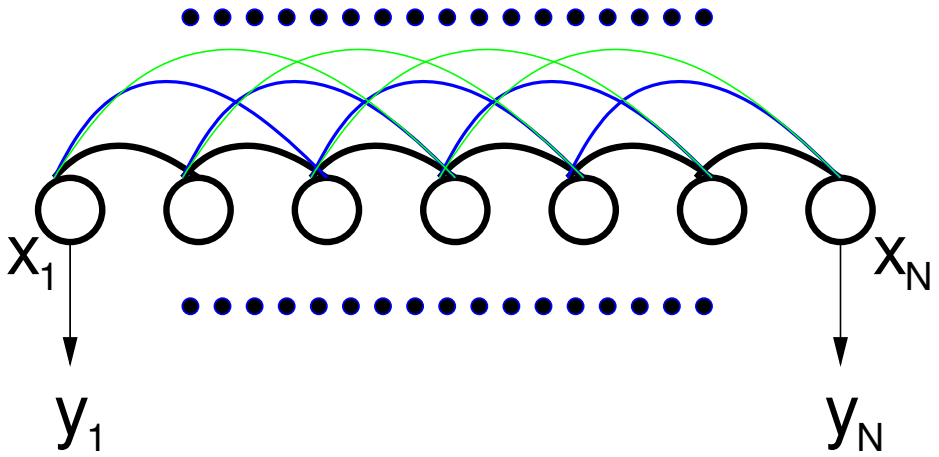
$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}\in\Omega} Z_{\mathcal{D}}(\boldsymbol{\theta})$$

Evidence maximisation.

Gradient/conj.grad. methods are used.

Sparse representations – Motivation

Gaussian Processes are fully connected graphical models.



$\Longrightarrow$ Computing estimates is difficult. E.g for the posterior mean:

$$\langle f_{\boldsymbol{x}} \rangle_{\text{post}} = \boldsymbol{y}^T \left(\boldsymbol{K}_N + \sigma_o^2 \boldsymbol{I}_N \right)^{-1} \boldsymbol{k}_{\boldsymbol{x}}$$

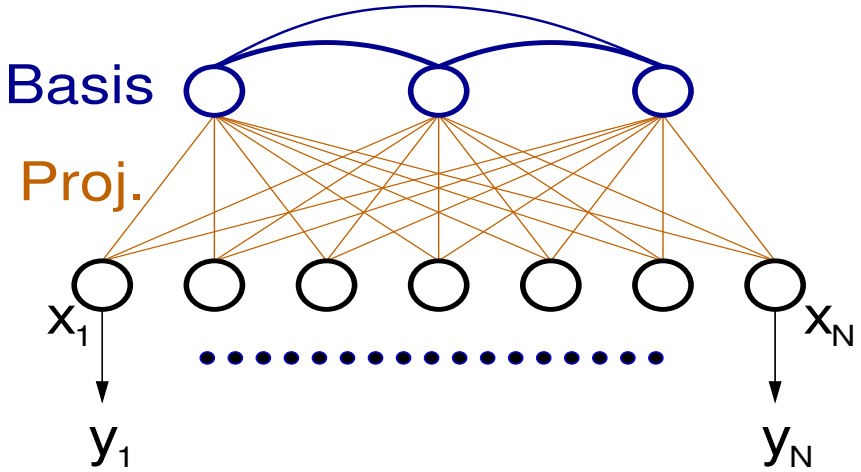
inversion requires $\mathcal{O}(N^3)$ time.

Regr

Sparse representations – a solution

Condition all training locations on a set of basis locations.

$$\boldsymbol{f}_{\mathcal{BV}} \sim \mathcal{N}\left(\mu_{\mathcal{BV}}, \Sigma_{\mathcal{BV}}\right)$$



The pseudo-latents $\boldsymbol{f}_{\mathcal{X}}$ are conditioned on $\boldsymbol{f}_{\mathcal{BV}}$:

$$\boldsymbol{f}_{\mathcal{X}} | \boldsymbol{f}_{\mathcal{BV}} \sim \mathcal{N}\left(\boldsymbol{P}\ \mu_{\mathcal{BV}}, \boldsymbol{P}\ \Sigma_{\mathcal{BV}}\ \boldsymbol{P}^T\right)$$

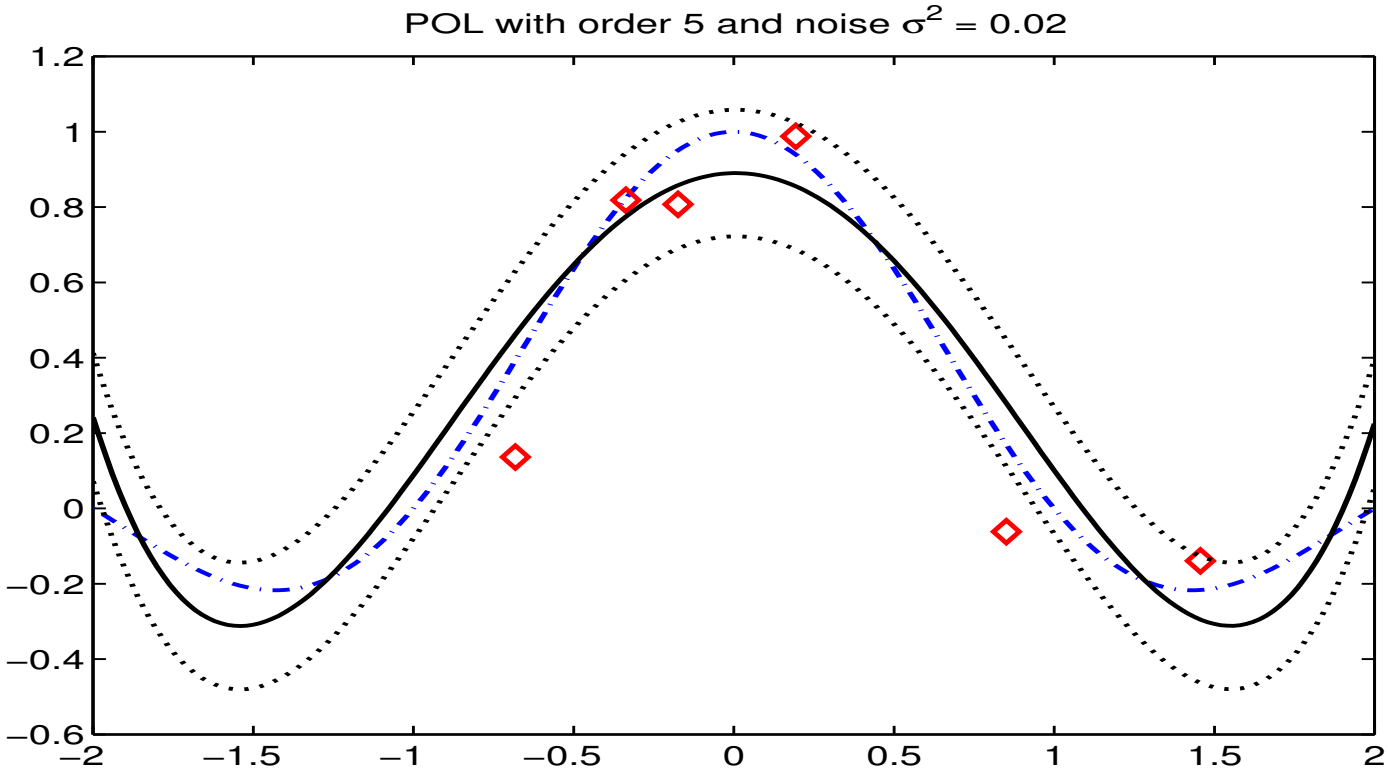
where $\boldsymbol{P}$ is the projection matrix:

$$\boldsymbol{P} = \boldsymbol{P}_{\boldsymbol{X},\mathcal{BV}} = \boldsymbol{K}_{\mathcal{X},\mathcal{BV}}\ \boldsymbol{K}_{\mathcal{BV}}^{-1}$$

Gaussian Regression

Artificial data: $y = \sin(x)/x$ and polynomial kernel $K_0(x, x') = (1 + \boldsymbol{x}^T \boldsymbol{x}')^k$.

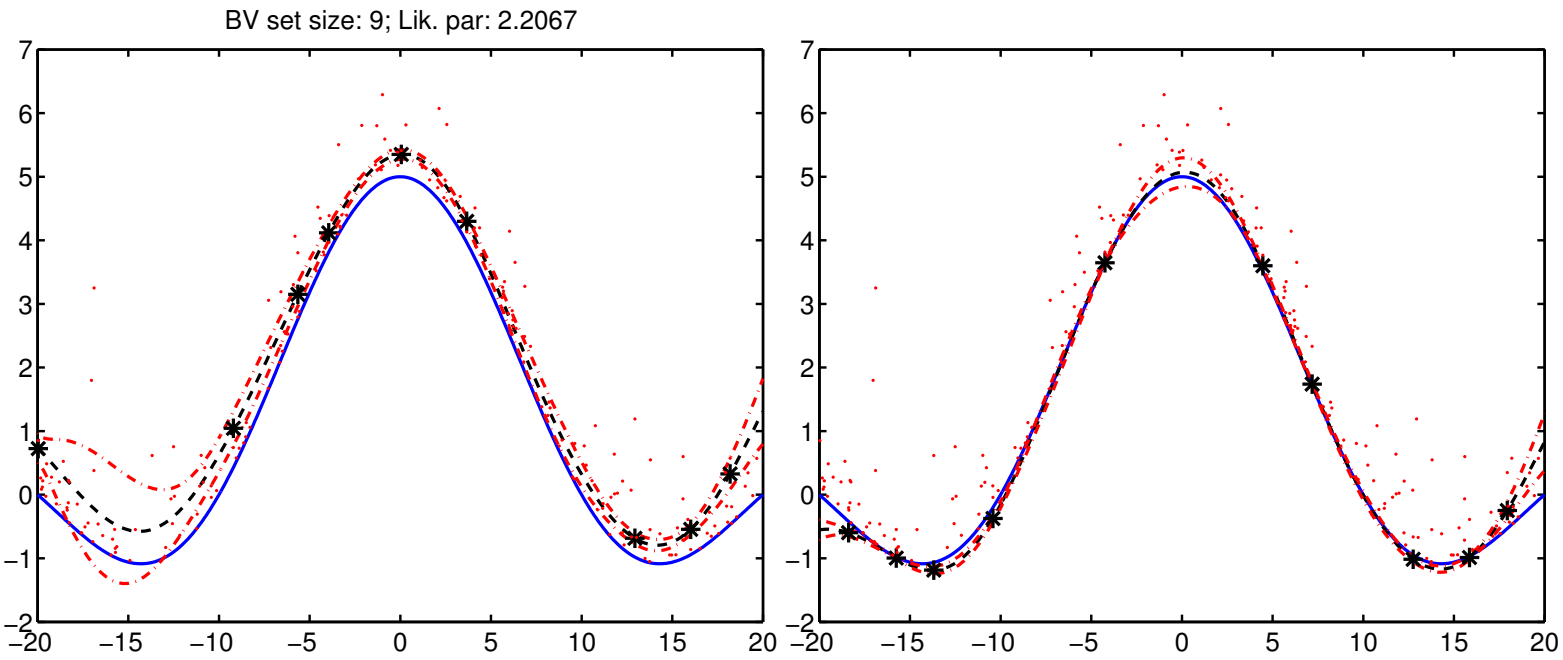
Number of training points: 1000 with added Gaussian noise $\sigma^2 = 0.02$



Robust one-sided regression

Exponential, one-sided, additive noise.

$$P(y|f_x) = \begin{cases} \lambda \exp\left[-\lambda(y - f_x)\right] & \text{if } y > f_x. \\ 0 & \text{otherwise.} \end{cases}$$



Classification

For each location $\boldsymbol{x}$ we have ± 1. The likelihood function for this model is:

$$P\left(y|f(\boldsymbol{x})\right) = \mathrm{Erf}\left(\frac{y\ f_{\boldsymbol{x}}}{\sigma_0}\right)$$

Erf the incomplete Gaussian ($\sim$ sigmoid):

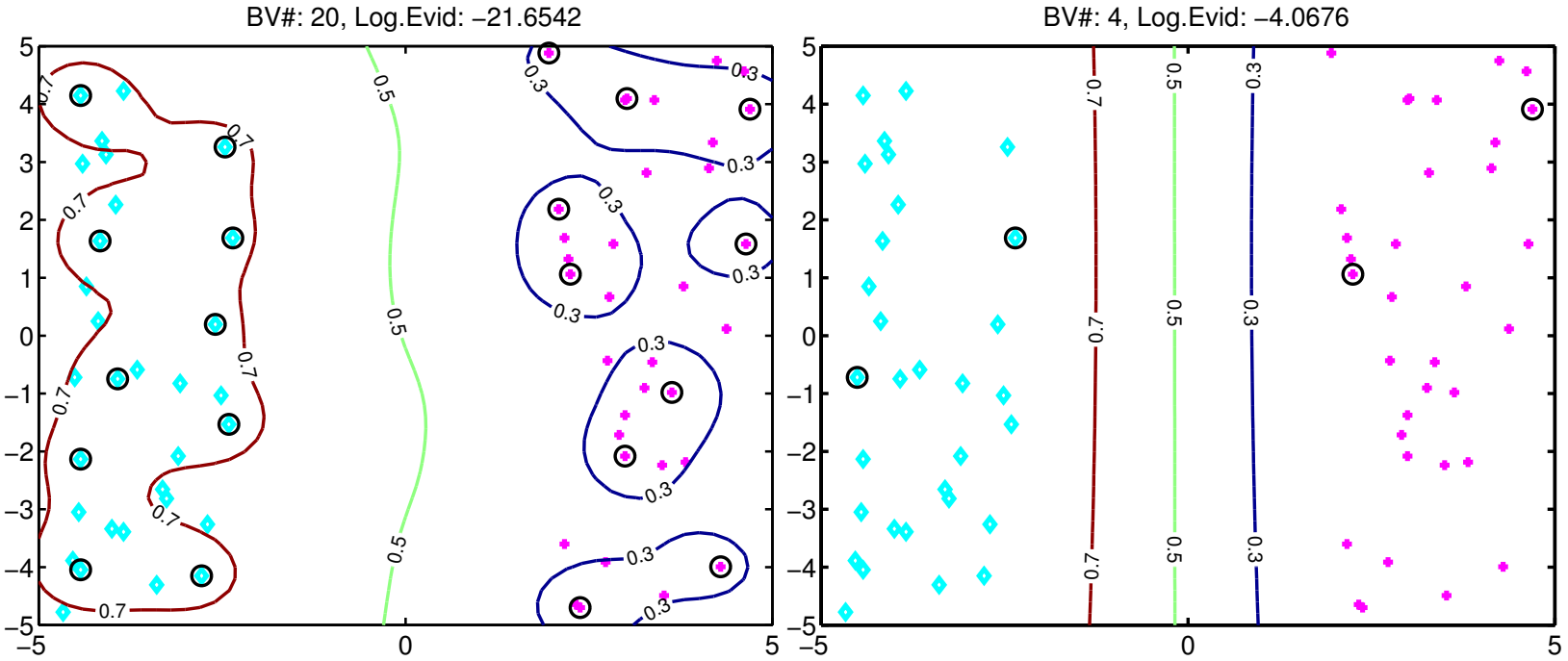
$$\mathrm{Erf}(x) = \int_{-\infty}^{x} dt\ \exp(-t^2/2)/\sqrt{2\pi}$$

- Posterior is not Gaussian.
- For single data, mean-var computable $\Rightarrow$ iterative methods can be used.

Toy Classification

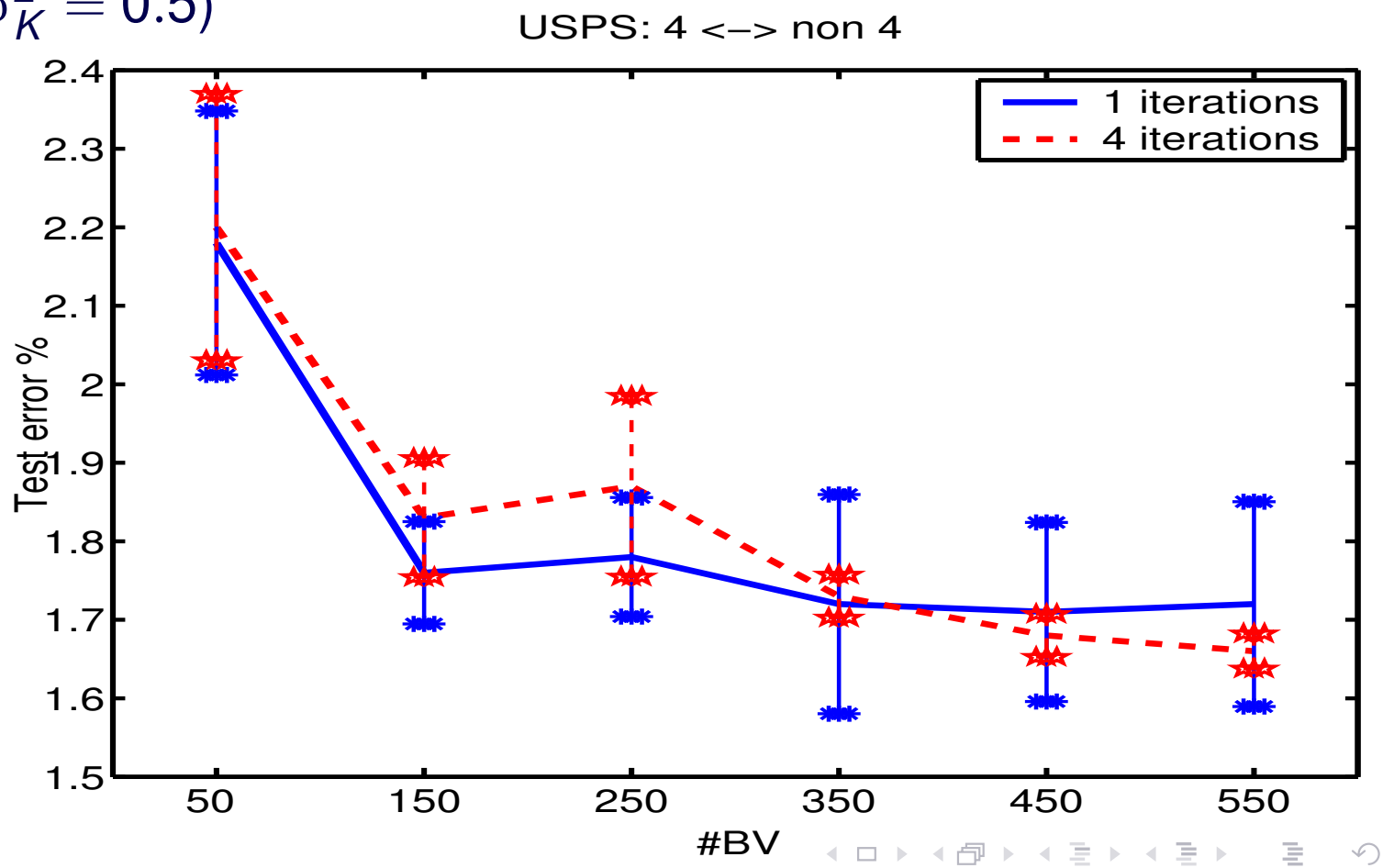
RBF kernel: $K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left[-b - \sum_{i=1}^{d} (x_i - x_i')^2 \beta_i\right]$

behaviour of the ARD parameters β_i



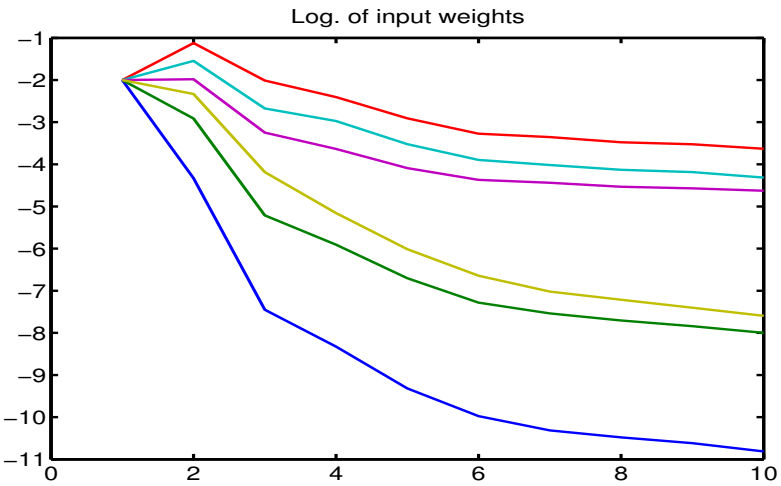
USPS data-set

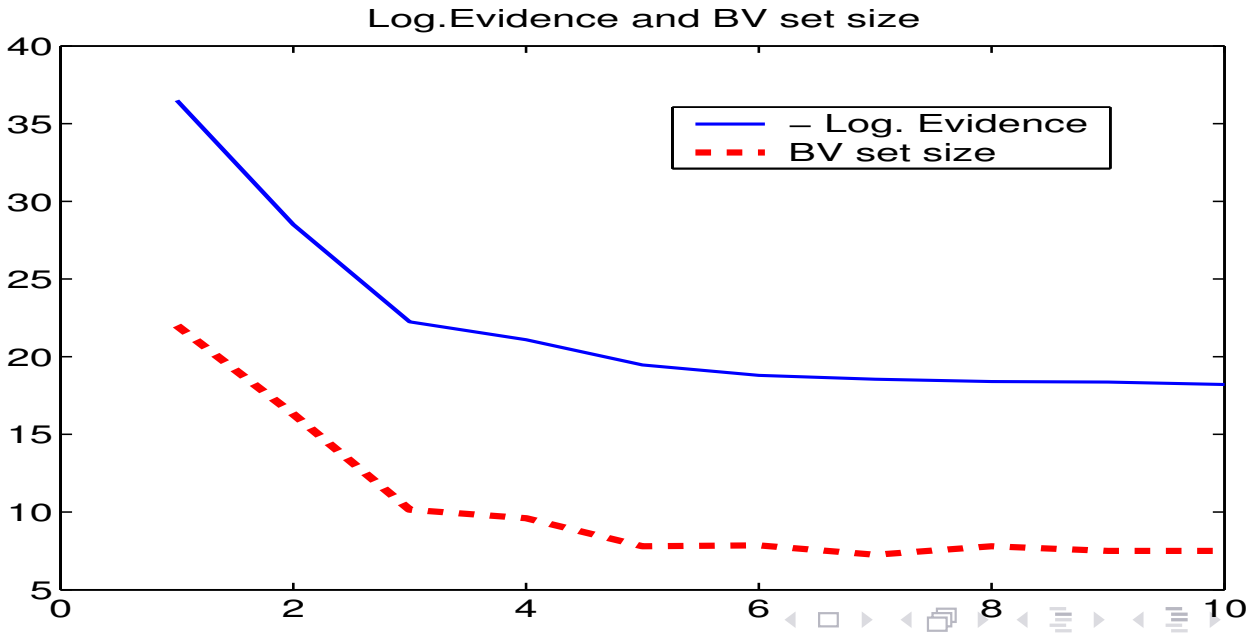
Handwritten image data-set of gray-scale images with 7291 training and 2007 test patterns. (RBF kernel with $\sigma_K^2 = 0.5$)



Crab data-set

Using RBF kernels





Multiclass Classification

Problem setup:

- For each location $\boldsymbol{x}$ we have $y \in \{1, \ldots, K\}$.
- Transforming it into $y \in \{0, 1\}^K$ Coding:

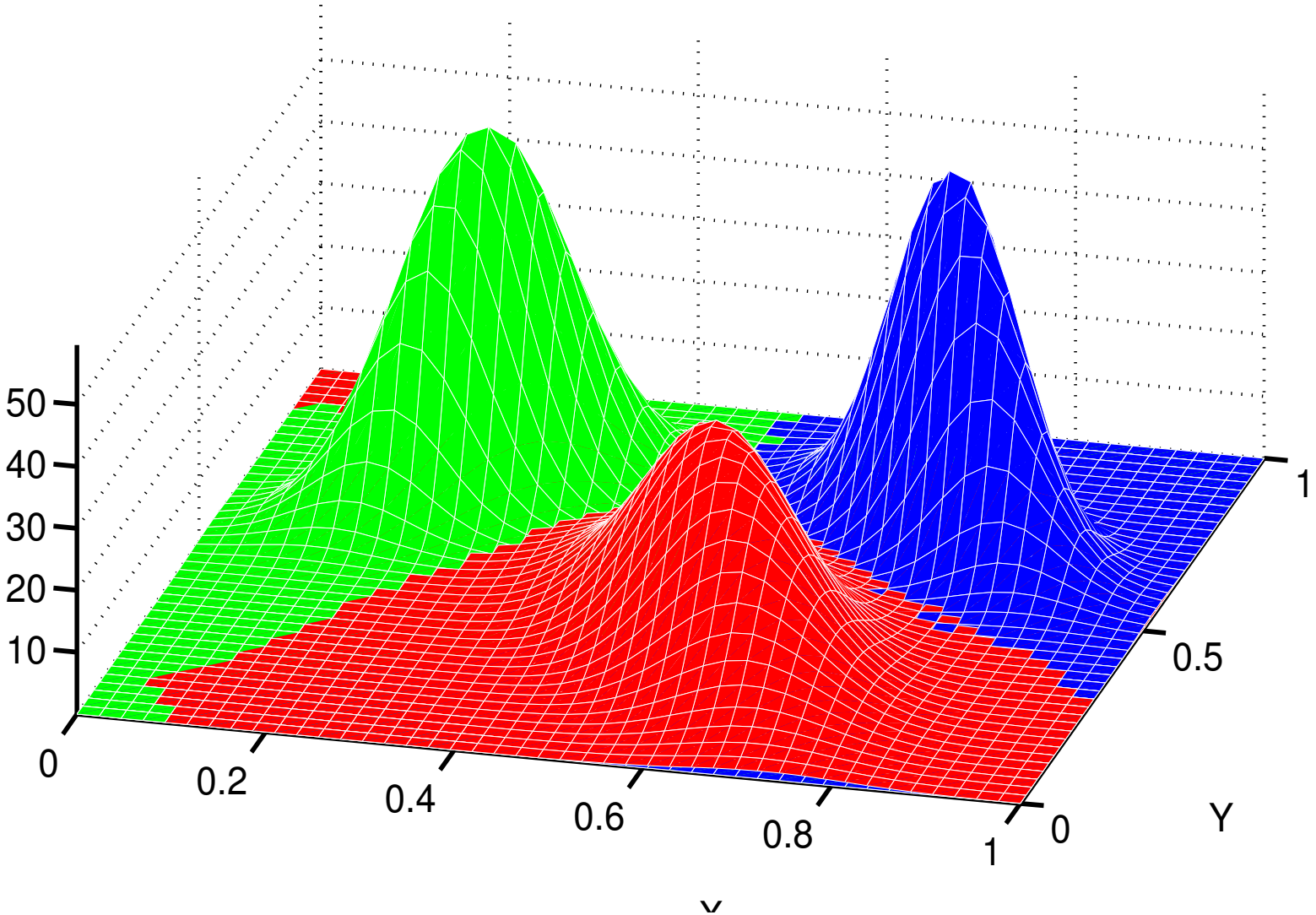
$$y = [0, \ldots, 0, 1, 0, \ldots]^T \quad \text{on the } k\text{-th position}$$

- K independent GP's are used. Indep. is a-priori.
- The likelihood function is:

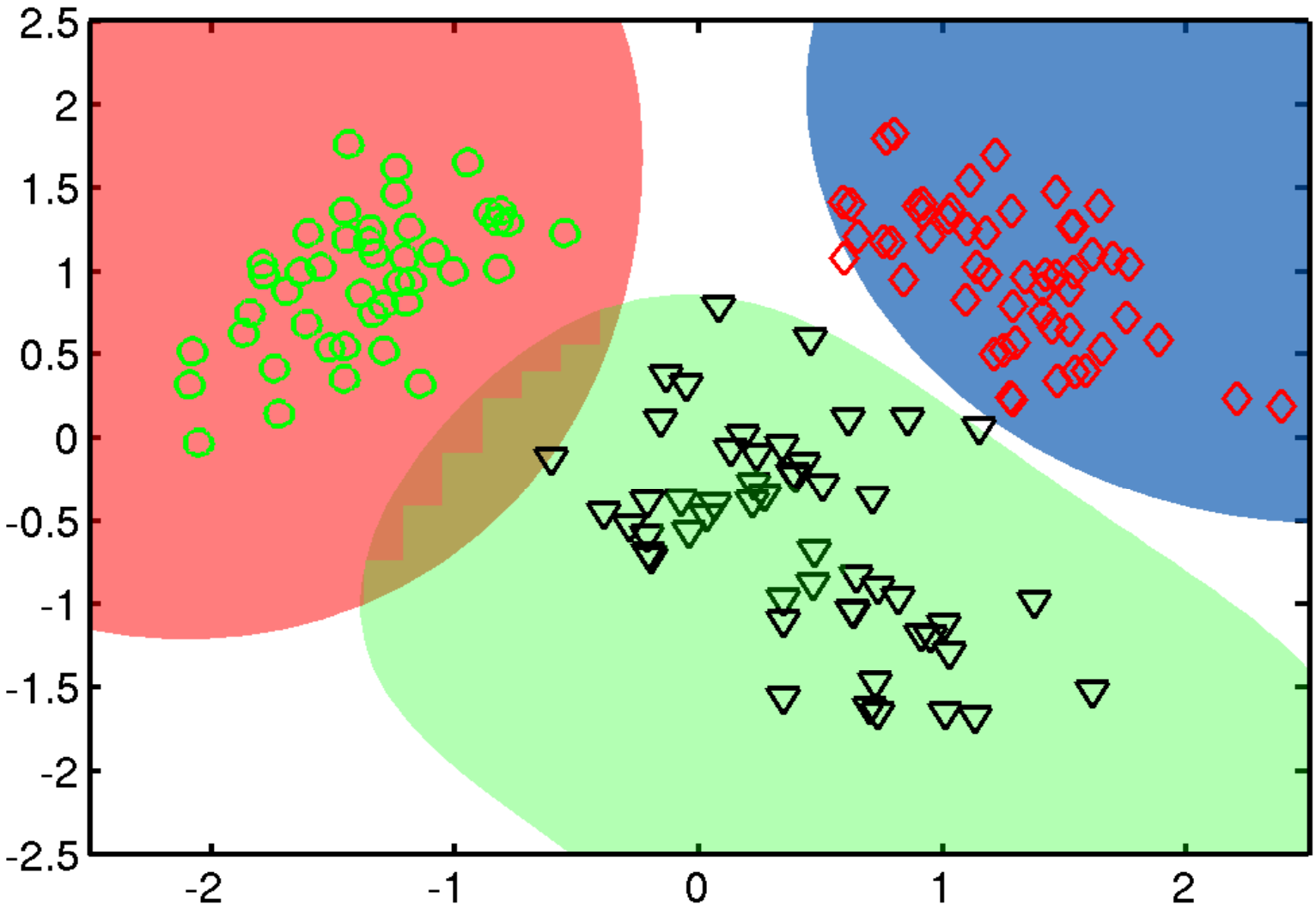
$$P(y|f(\boldsymbol{x})) = \frac{y^T \boldsymbol{s}}{\mathbf{1}^T \boldsymbol{s}} \text{ where } \boldsymbol{s} = \exp\left([f_1(\boldsymbol{x}), \ldots, f_K(\boldsymbol{x})]^T\right).$$

- The posterior processes are not independent.

Two-dimensional demo: class-conditional distributions



Multiclass Classification



Modelling Inverse Problems

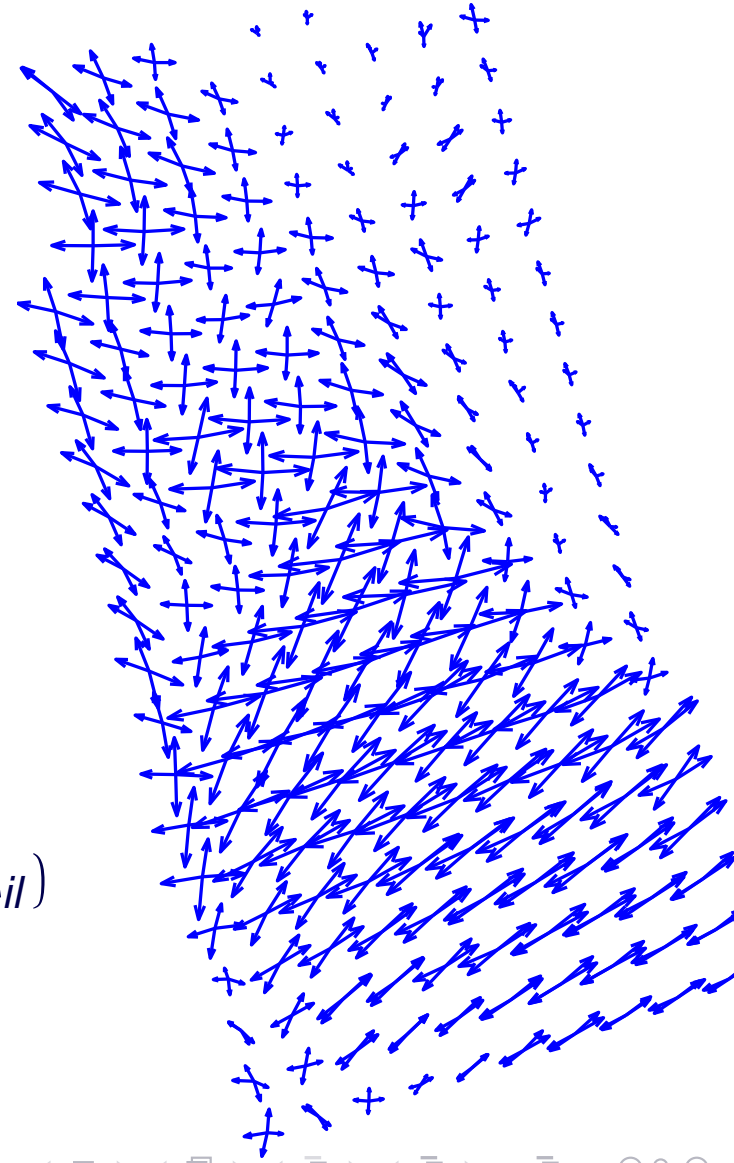
Likelihood:
local observations of global wind-fields (u_i, v_i).

Probabilistic framework preferred due to lack of direct observations.

Uncertainty captured in Mixture density networks:

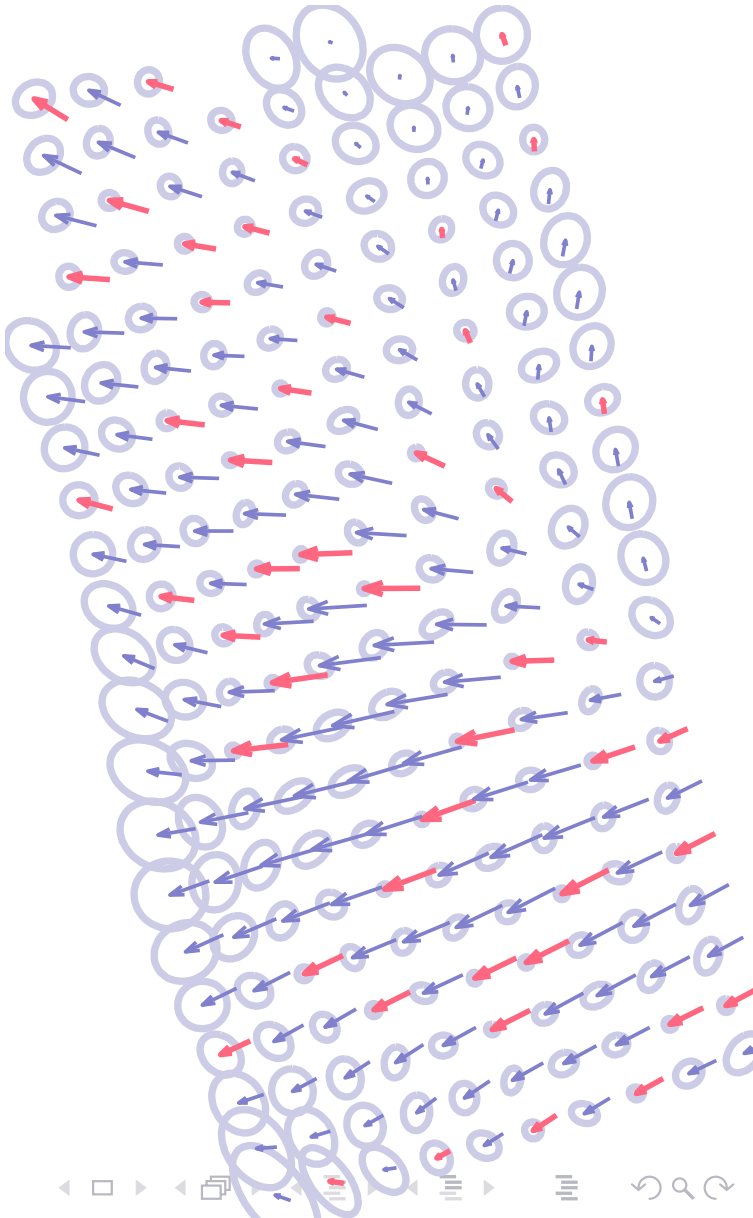
$$P(u_i, v_i|\mathrm{obs}) = \sum_{k=1}^{4} \beta_{il} \mathcal{N}(u_i, v_i|\mu_{il}, A_{il})$$

$\beta_{il}, \mu_{il}, A_{il}$ local parameters.



Modelling Inverse Problems

- Few Basis Vectors retained
- Approximation preserves information about local uncertainty
- The inference process is fast



Bibliography I

- D.J.C. MacKay: *The evidence framework ...*

  `http://wol.ra.phy.cam.ac.uk/mackay/class.nc.ps.gz`

  H: `http://wol.ra.phy.cam.ac.uk/mackay/`

- C. Rasmussen: *Gaussian Processes for Regression.*

  `http://www.kyb.mpg.de/publications/pss/ps2468.ps`

  H: `http://www.kyb.mpg.de/~carl`

- C.K.I. Williams: *Prediction with Gaussian Processes: ...*

  `http://www.dai.ed.ac.uk/homes/ckiw/postscript/NCRG_97_012.ps.gz`

  H: `http://www.dai.ed.ac.uk/homes/ckiw`

▸ Bib 2

Bibliography II

- R. Neal: *Priors for infinite networks*

  `http://www.cs.utoronto.ca/~radford/ftp/pin.ps`

  H: `http://www.cs.utoronto.ca/~radford/`

- M. Seeger: *Bayesian Gaussian Process Models: ...*

  `http://www.kyb.tuebingen.mpg.de/bs/people/seeger/papers/thesis.pdf`

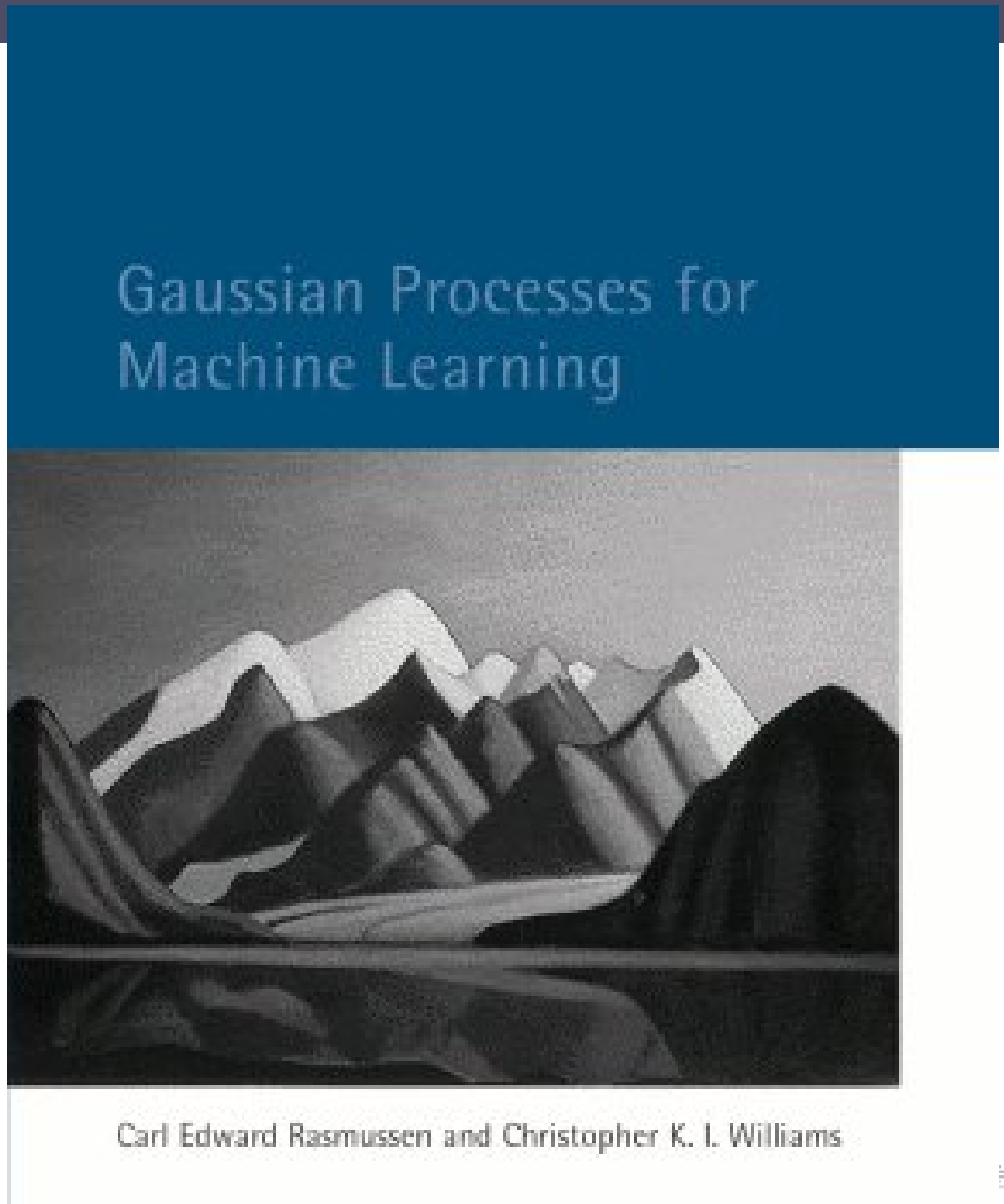
  H: `http://www.kyb.tuebingen.mpg.de/~seeger`

- L. Csato: *Gaussian Processes: ... Sparse Approximations*

  `http://www.ncrg.aston.ac.uk/Publications`

  H: `http://www.cs.ubbcluj.ro/~csatol`

Bibliography III



NIPS'05 I

NIPS'05 GP-related articles:

- J. Murillo-Fuentes, F. Perez-Cruz: Gaussian Processes for Multiuser Detection in CDMA Receivers
- Y. Shen, A. Ng, M. Seeger: Fast Gaussian Process Regression Using KD-Trees
- A. Shon, K. Grochow, A. Hertzmann, R. Rao: Gaussian Process CCA for Image Synthesis and Robotic Imitation
- R. Der, D. Lee: Beyond Gaussian Processes: On the Distributions of Infinite Networks
- D. Fleet, J. Wang, A. Hertzmann: Gaussian Process Dynamical Models

NIPS'05 II

NIPS'05 GP-related articles:

- Y. Engel, P. Szabo, D. Volkinshtein: Learning With Gaussian Process Temporal Difference Methods
- M. Kuss, C. Rasmussen: Assessing Approximations for Gaussian Process Classification
- E. Snelson, Z. Ghahramani: Sparse Parametric Gaussian Processes
- S. Kakade, M. Seeger, D. Foster: Worst-Case Bounds for Gaussian Process Models
- S. Keerthi, W. Chu: A Matching Pursuit Approach to Sparse Gaussian Process Regression
- E. Meeds, S. Osindero: An Alternative Infinite Mixture Of Gaussian Process Experts