Fast Sparse Gaussian Processes: The Informative Vector Machine

An Overview of Gaussian Processes with a look at the IVM

Neil Lawrence
School of Computer Science
University of Manchester

July 2007

Outline

Online Resources

All source code and slides are available online

- This talk available from my home page (see talks link on side).
- MATLAB examples in the 'oxford' toolbox (vrs 0.13).
  - `http://www.cs.man.ac.uk/~neill/oxford/`.
- And the 'ivm' toolbox (vrs 0.4) and 'mtivm' toolbox (vrs 0.14).
  - `http://www.cs.man.ac.uk/~neill/ivm/`.
  - `http://www.cs.man.ac.uk/~neill/mtivm/`.
- MATLAB commands used for examples given in `typewriter font`.

Introduction to Gaussian Processes

Inference about functions

- Many Machine Learning problems can be reduced to inference about functions.
  - We will see some examples later.
- Gaussian processes (GPs) are probabilistic models for functions. O'Hagan [1978, 1992], Rasmussen and Williams [2006]
- GPs allow inference about functions in the presence of uncertainty.

Defining a Distribution over Functions

Gaussian Process

- What is meant by a distribution over functions?
- Functions are infinite dimensional objects:
  - Defining a distribution over functions seems non-sensical.

Gaussian Distribution

- Start with a standard Gaussian distribution.
- Consider the distribution over a fixed number of instantiations of the function.

Gaussian Distribution

Zero mean Gaussian distribution

- A multi-variate Gaussian distribution is defined by a mean and a covariance matrix.

$$N(\mathbf{f}|\mu, \mathbf{K}) = \frac{1}{(2\pi)^{\frac{N}{2}} |\mathbf{K}|^{\frac{1}{2}}} \exp\left(-\frac{(\mathbf{f}-\mu)^{\mathrm{T}} \mathbf{K}^{-1} (\mathbf{f}-\mu)}{2}\right).$$

- We will consider the special case where the mean is zero,

$$N(\mathbf{f}|\mathbf{0}, \mathbf{K}) = \frac{1}{(2\pi)^{\frac{N}{2}} |\mathbf{K}|^{\frac{1}{2}}} \exp\left(-\frac{\mathbf{f}^{\mathrm{T}} \mathbf{K}^{-1} \mathbf{f}}{2}\right).$$

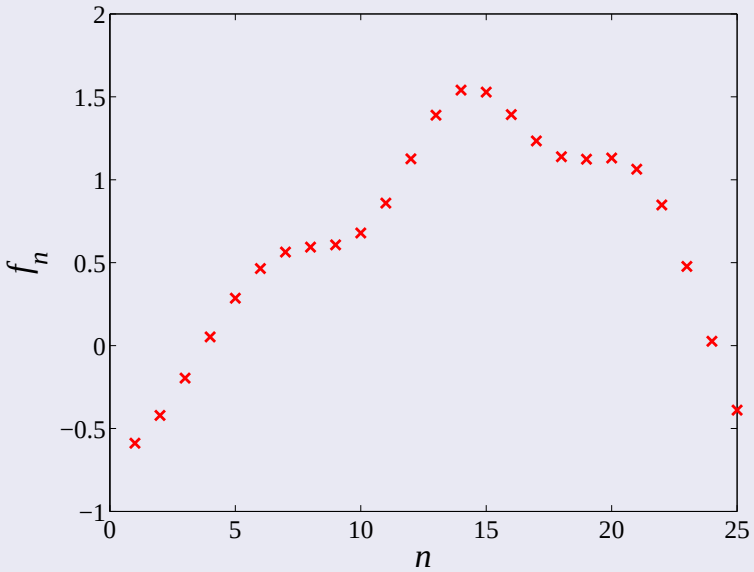
Sampling a Function

Multi-variate Gaussians

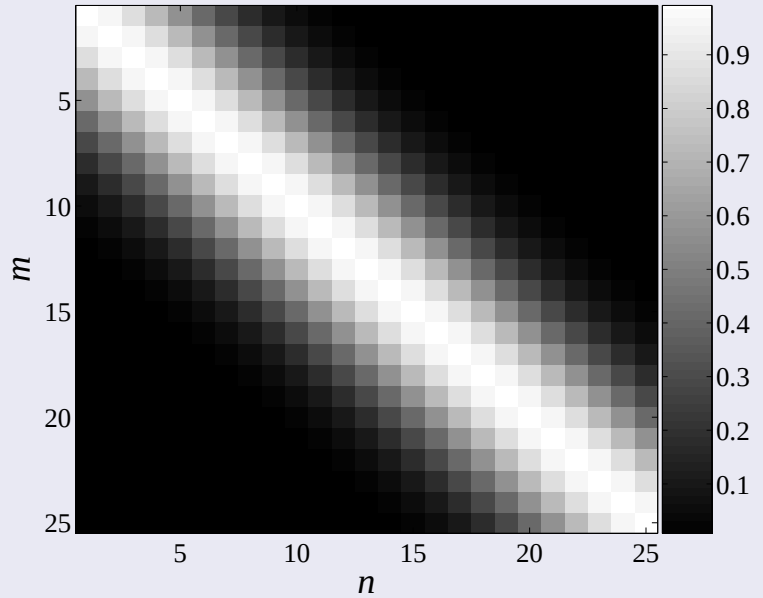
- We will consider a Gaussian with a particular structure of covariance matrix.
- Generate a single sample from this 25 dimensional Gaussian distribution, $\mathbf{f} = [f_1, f_2 \dots f_{25}]$.
- We will plot these points against their index.

Gaussian Distribution Sample

demGPSample



(a)



(b)

Figure: (a) 25 instantiations of a function, f_n, (b) greyscale covariance matrix.

Covariance Function

The covariance matrix

- Covariance matrix shows correlation between points f_m and f_n if n is near to m.
- Less correlation if n is distant from m.
- Our ordering of points means that the *function appears smooth*.
- Let's focus on the joint distribution of two points form the 25.

Covariance Function

The covariance matrix

- Covariance matrix shows correlation between points f_m and f_n if n is near to m.
- Less correlation if n is distant from m.
- Our ordering of points means that the *function appears smooth*.
- Let's focus on the joint distribution of two points form the 25.

Covariance Function

The covariance matrix

- Covariance matrix shows correlation between points f_m and f_n if n is near to m.
- Less correlation if n is distant from m.
- Our ordering of points means that the *function appears smooth*.
- Let's focus on the joint distribution of two points form the 25.

Covariance Function

The covariance matrix

- Covariance matrix shows correlation between points f_m and f_n if n is near to m.
- Less correlation if n is distant from m.
- Our ordering of points means that the *function appears smooth*.
- Let's focus on the joint distribution of two points form the 25.

Prediction of f_2 from f_1

demGPCov2D([1 2])

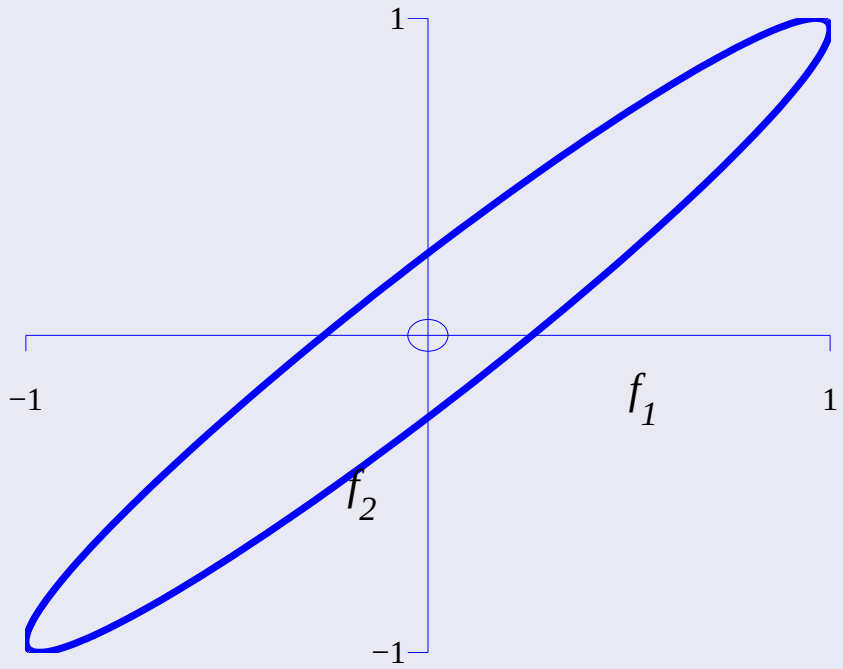


Figure: Covariance for $\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$ is $\mathbf{K}_{12} = \begin{bmatrix} 1 & 0.966 \\ 0.966 & 1 \end{bmatrix}$.

Prediction of f_2 from f_1

demGPCov2D([1 2])

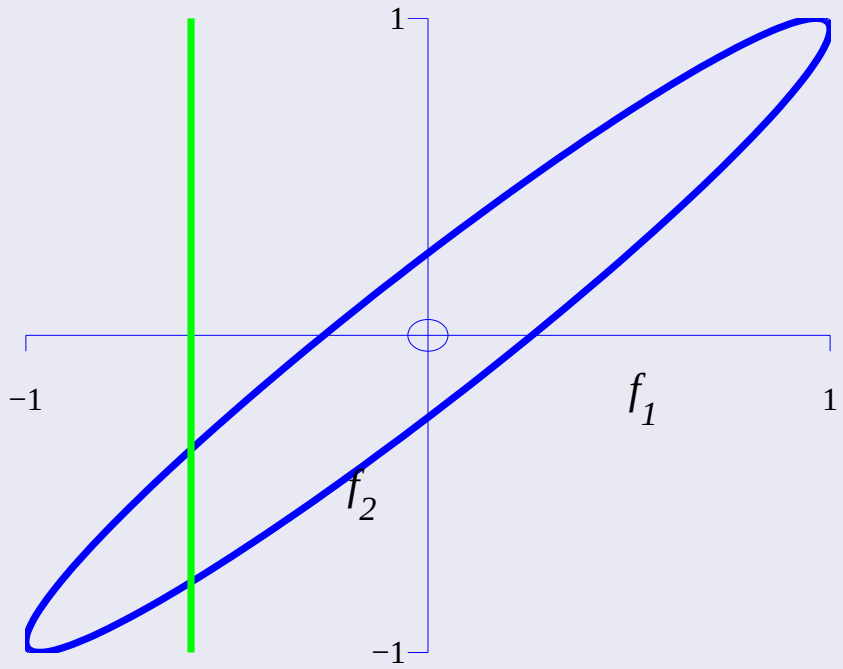


Figure: Covariance for $\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$ is $\mathbf{K}_{12} = \begin{bmatrix} 1 & 0.966 \\ 0.966 & 1 \end{bmatrix}$.

Prediction of f_2 from f_1

demGPCov2D([1 2])

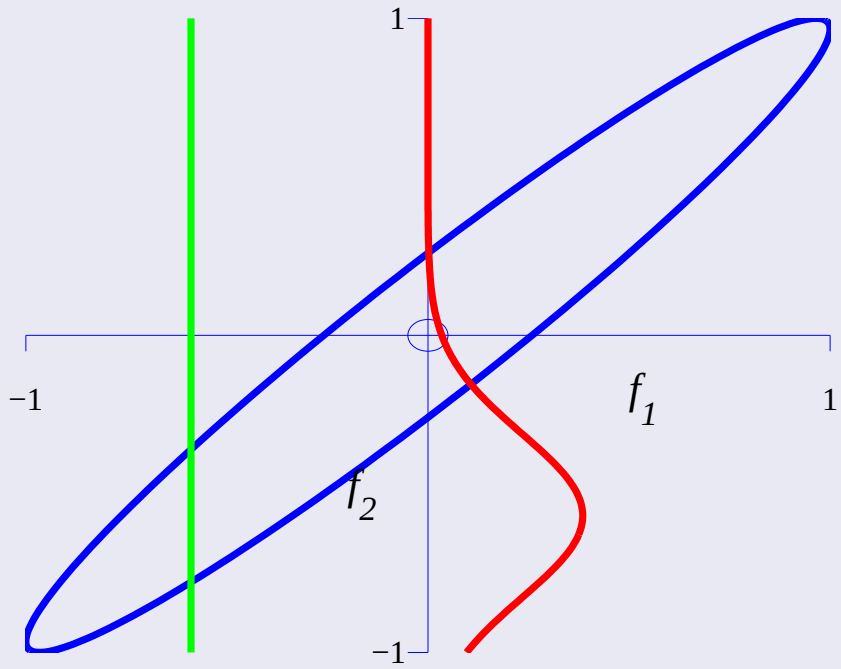


Figure: Covariance for $\begin{bmatrix} f_1 \\ f_2 \end{bmatrix}$ is $\mathbf{K}_{12} = \begin{bmatrix} 1 & 0.966 \\ 0.966 & 1 \end{bmatrix}$.

Prediction of f_5 from f_1

demGPCov2D([1 5])

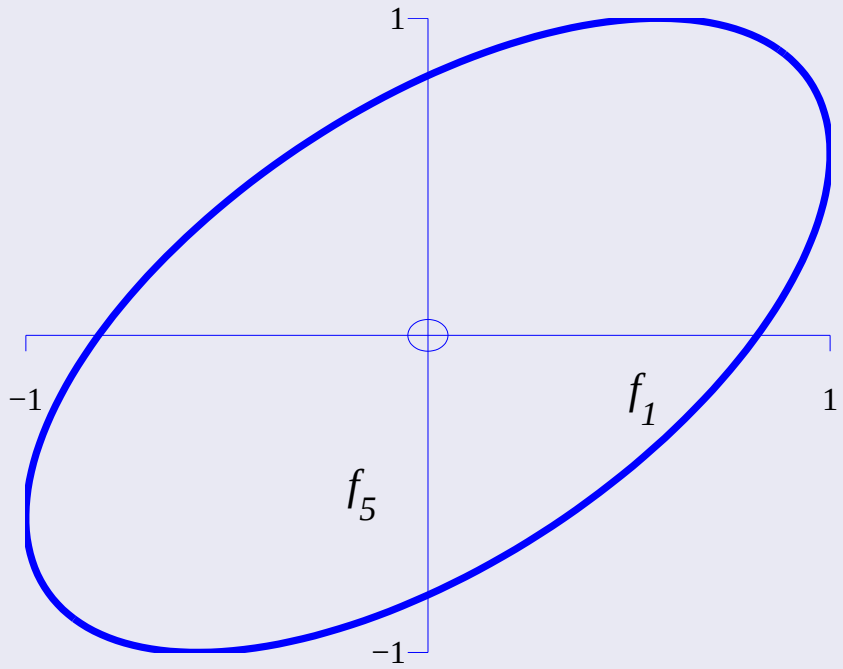


Figure: Covariance for $\begin{bmatrix} f_1 \\ f_5 \end{bmatrix}$ is $\mathbf{K}_{15} = \begin{bmatrix} 1 & 0.574 \\ 0.574 & 1 \end{bmatrix}$.

Prediction of f_5 from f_1

demGPCov2D([1 5])

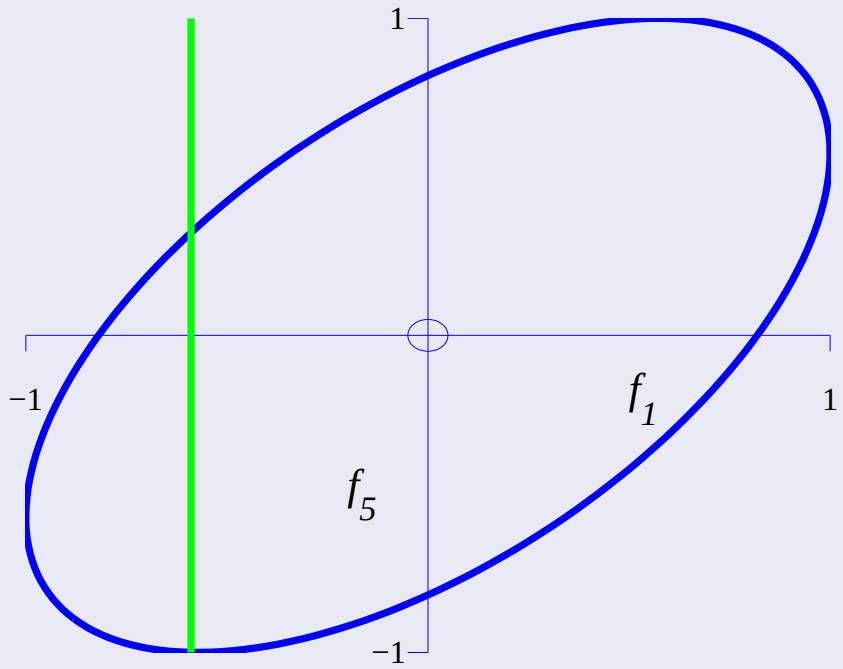


Figure: Covariance for $\begin{bmatrix} f_1 \\ f_5 \end{bmatrix}$ is $\mathbf{K}_{15} = \begin{bmatrix} 1 & 0.574 \\ 0.574 & 1 \end{bmatrix}$.

Prediction of f_5 from f_1

demGPCov2D([1 5])

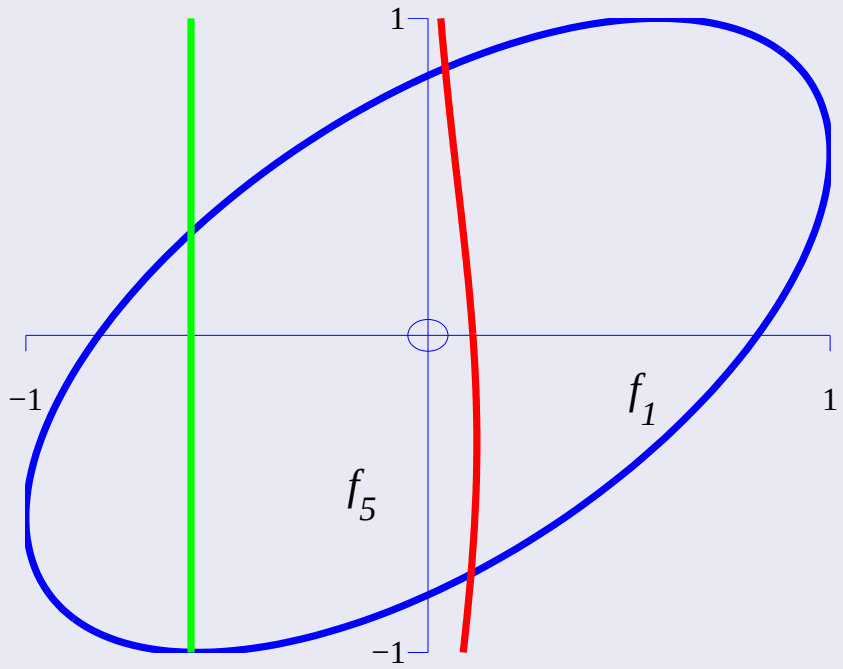


Figure: Covariance for $\begin{bmatrix} f_1 \\ f_5 \end{bmatrix}$ is $\mathbf{K}_{15} = \begin{bmatrix} 1 & 0.574 \\ 0.574 & 1 \end{bmatrix}$.

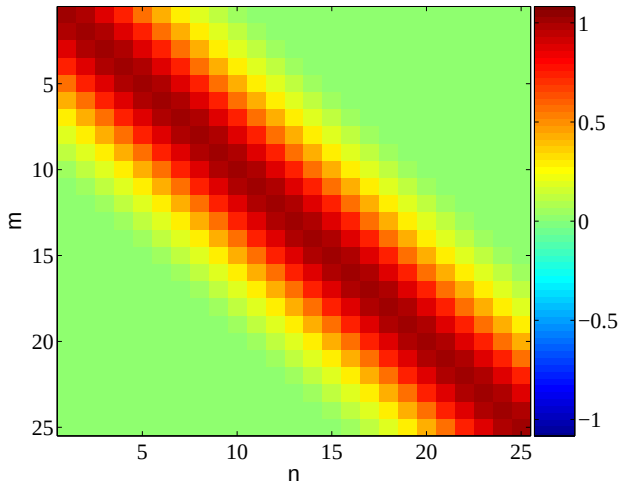
Covariance Functions

Where did this covariance matrix come from?

RBF Kernel Function

$$k(\mathbf{x}_m, \mathbf{x}_n) = \alpha \exp\left(-\frac{||\mathbf{x}_m - \mathbf{x}_n||^2}{2l^2}\right)$$

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

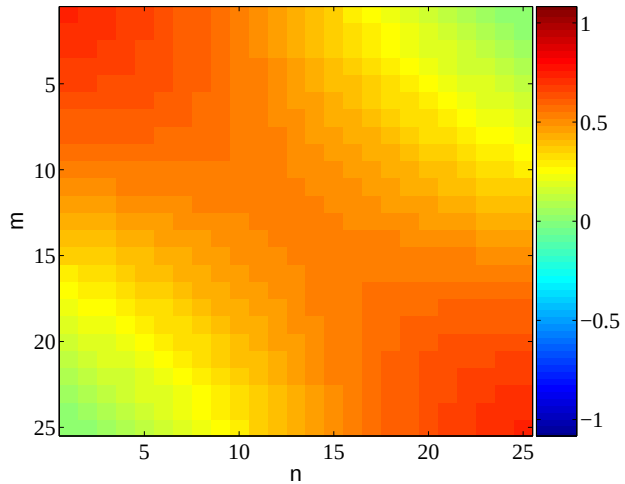


Different Covariance Functions

MLP Kernel Function

$$k\left(\mathbf{x}_m, \mathbf{x}_n\right) = \alpha \sin^{-1}\left(\frac{w\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_n + b}{\sqrt{w\mathbf{x}_m^{\mathrm{T}}\mathbf{x}_m + b + 1}\sqrt{w\mathbf{x}_n^{\mathrm{T}}\mathbf{x}_n + b + 1}}\right)$$

- A non-stationary covariance matrix [Williams, 1997].
- Derived from a multi-layer perceptron (MLP).

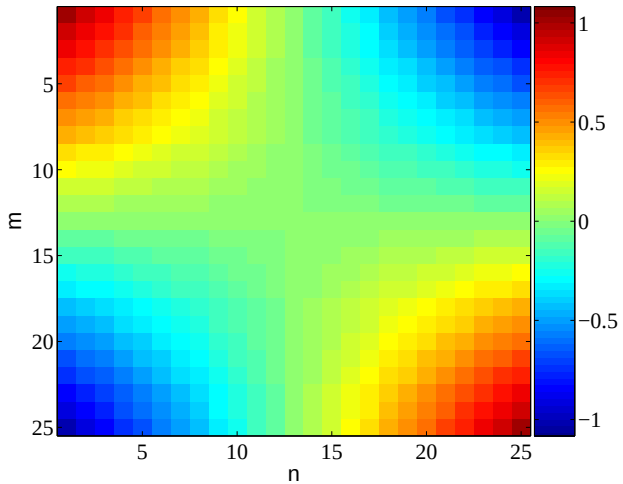


Different Covariance Functions

Linear Kernel Function

$$k(\mathbf{x}_m, \mathbf{x}_n) = \alpha \mathbf{x}_m^{\mathrm{T}} \mathbf{x}_n$$

- Allows for a linear trend.
- Note the anti-correlations in the matrix.

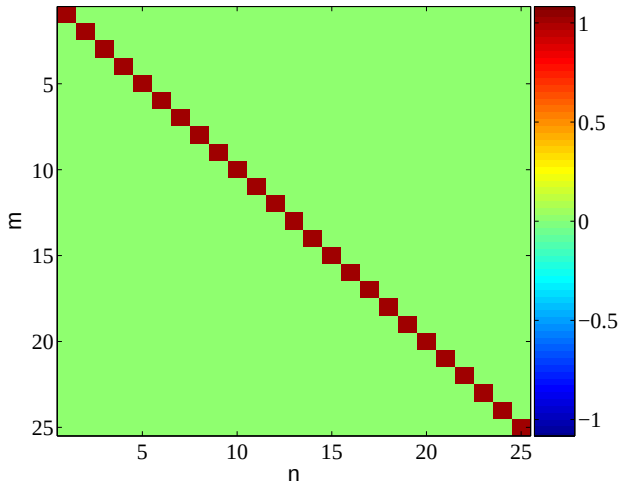


Different Covariance Functions

White noise

$$k(\mathbf{x}_m, \mathbf{x}_n) = \alpha \delta_{mn}$$

- Where δ_{mn} is the Kronecker delta.
- Simply represents uncorrelated independent noise.



Covariance Samples

demCovFuncSample

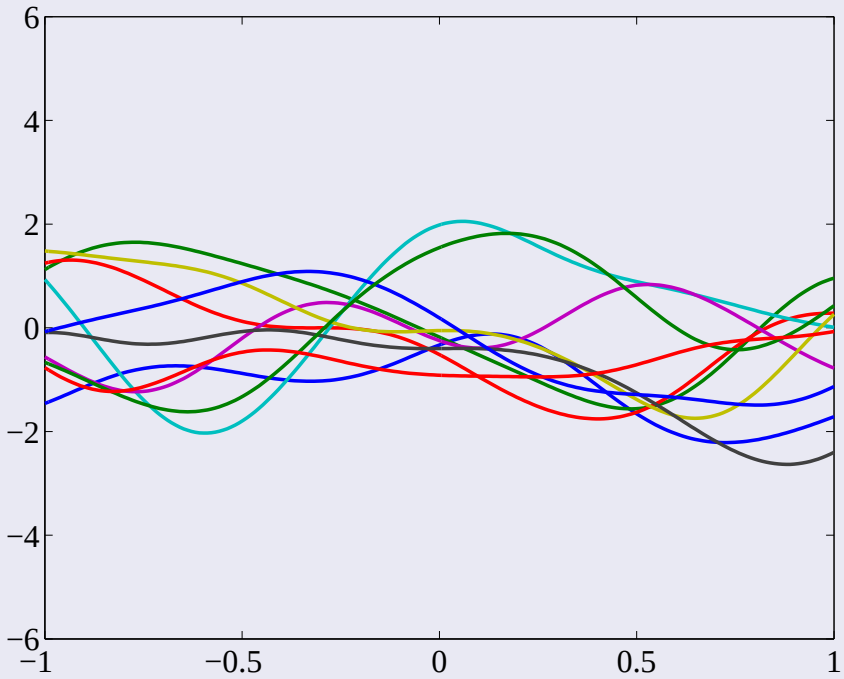


Figure: RBF kernel with $\gamma = 10$, $\alpha = 1$

Covariance Samples

demCovFuncSample

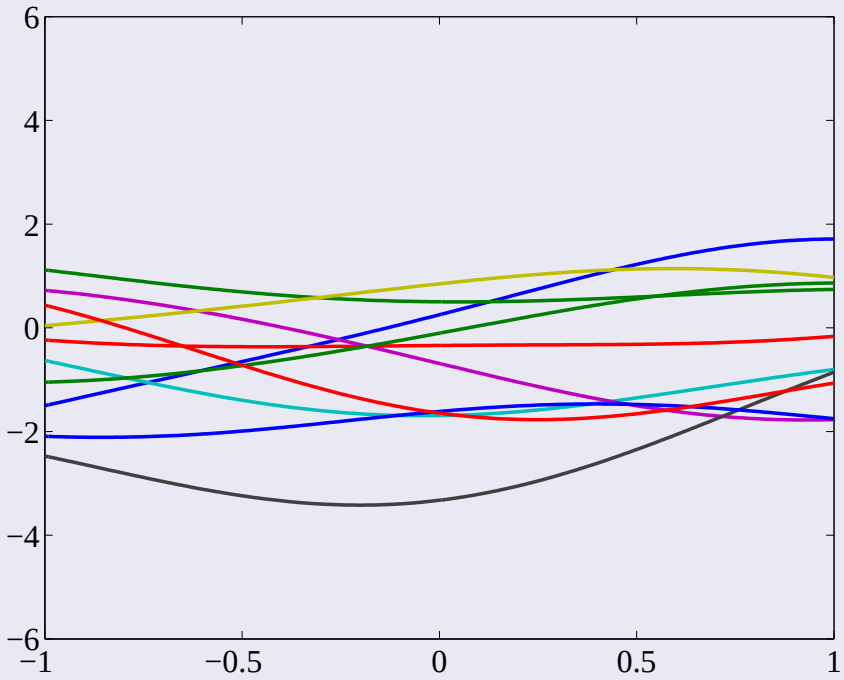


Figure: RBF kernel with $l = 1$, $\alpha = 1$

Covariance Samples

demCovFuncSample

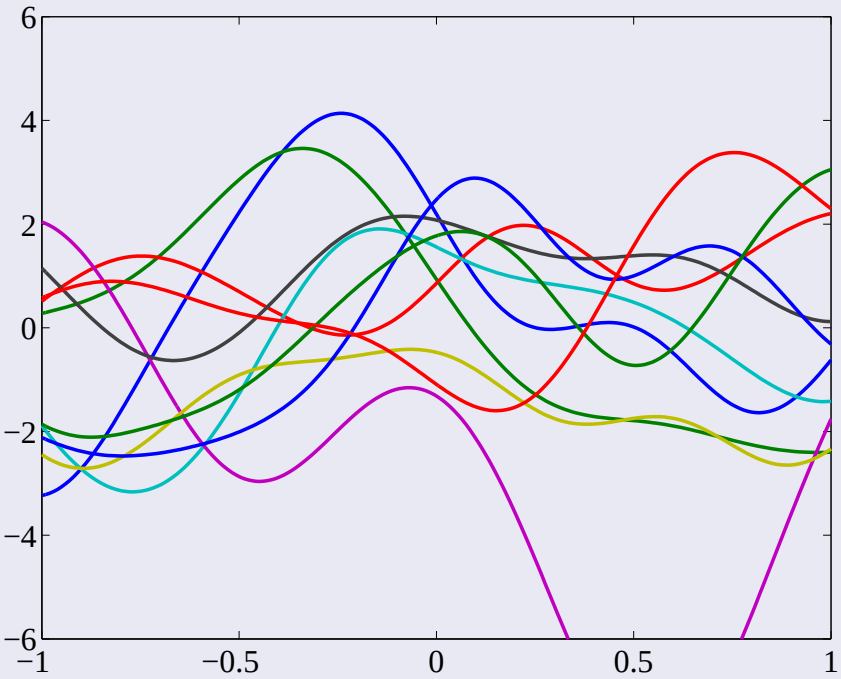


Figure: RBF kernel with $l = 0.3$, $\alpha = 4$

Covariance Samples

demCovFuncSample

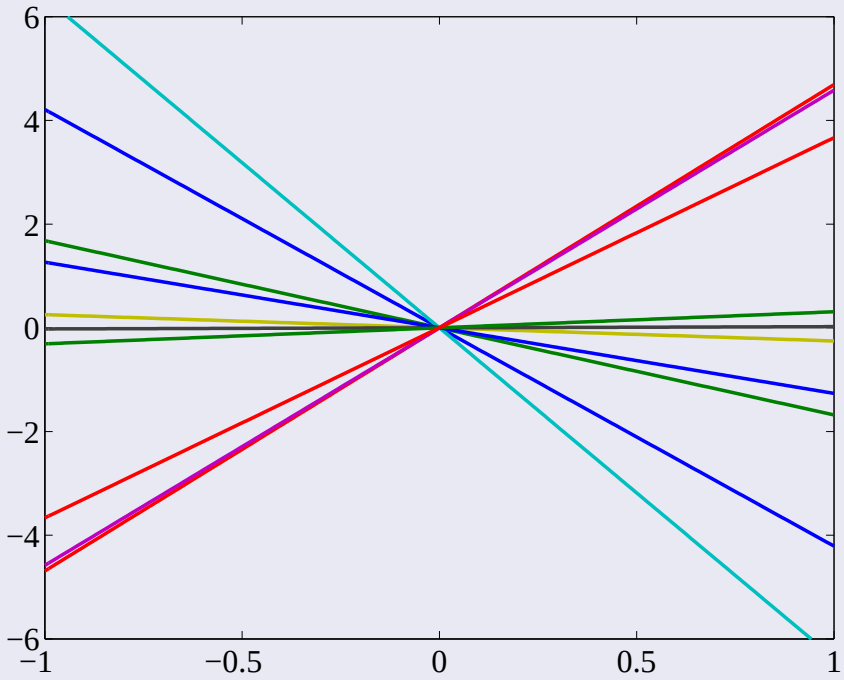


Figure: linear kernel with $\alpha = 16$

Covariance Samples

demCovFuncSample

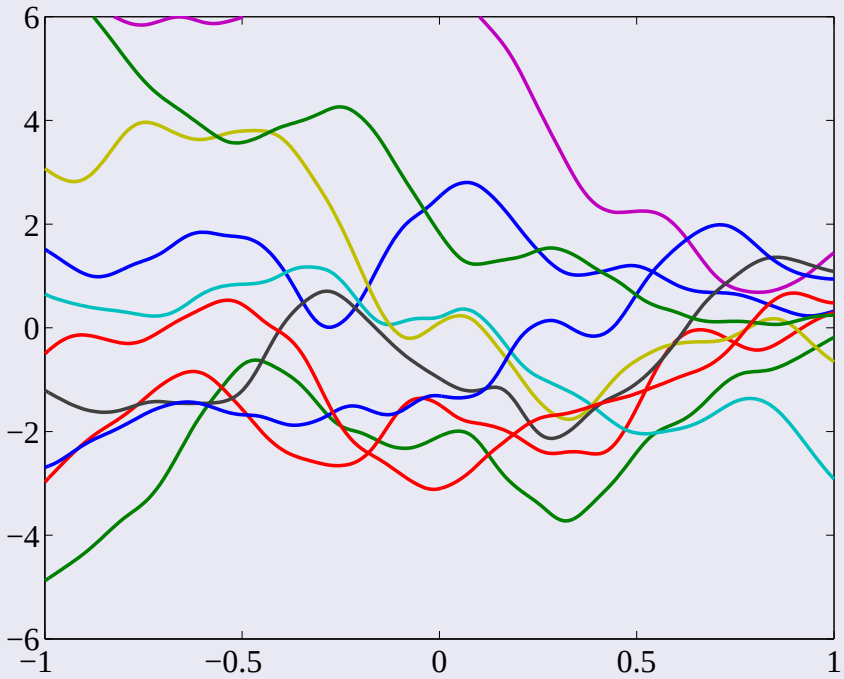


Figure: MLP kernel with $\alpha = 8$, $w = 100$ and $b = 100$

Covariance Samples

demCovFuncSample

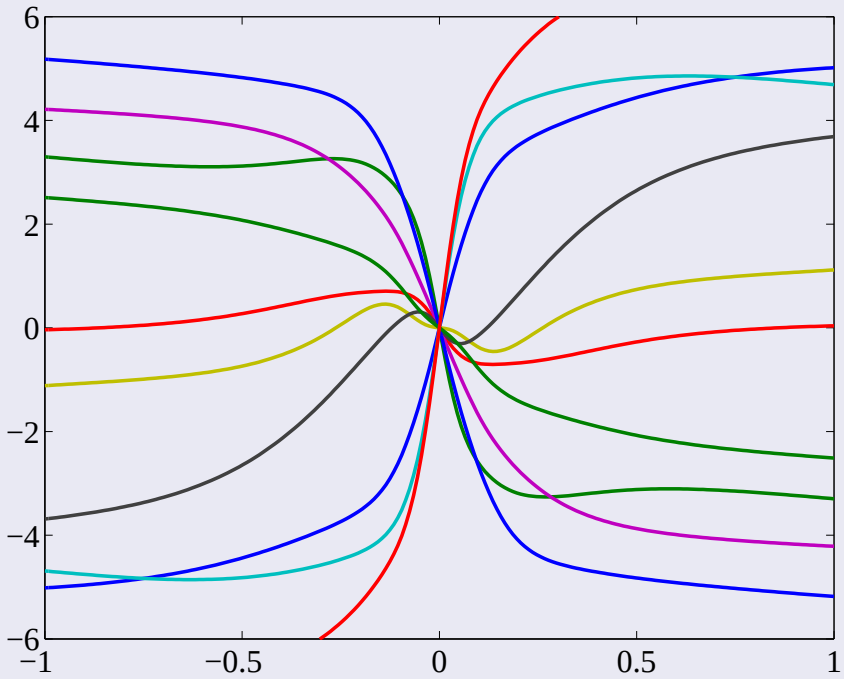


Figure: MLP kernel with $\alpha = 8$, $b = 0$ and $w = 100$

Covariance Samples

demCovFuncSample

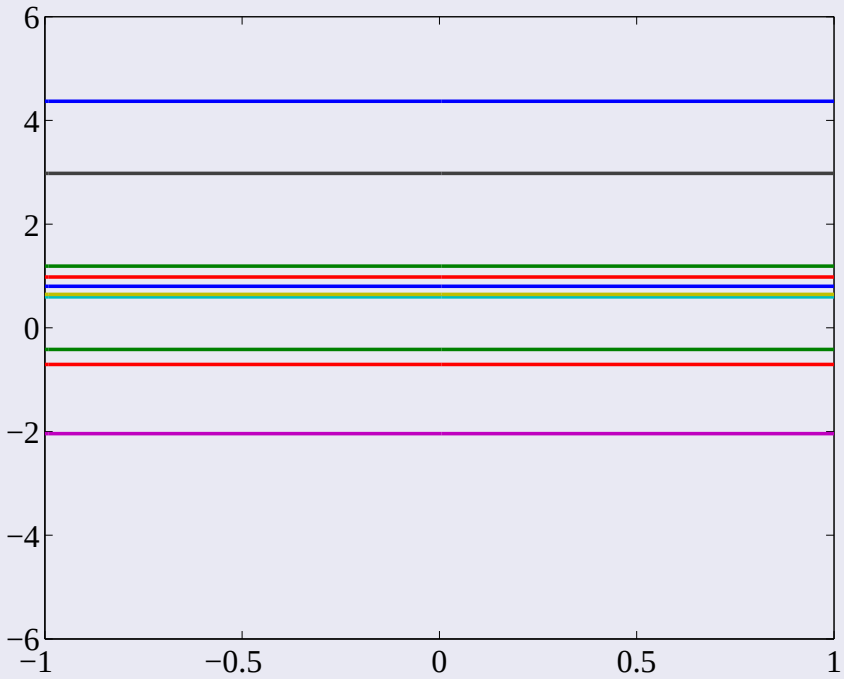


Figure: bias kernel with $\alpha = 1$ and

Covariance Samples

demCovFuncSample

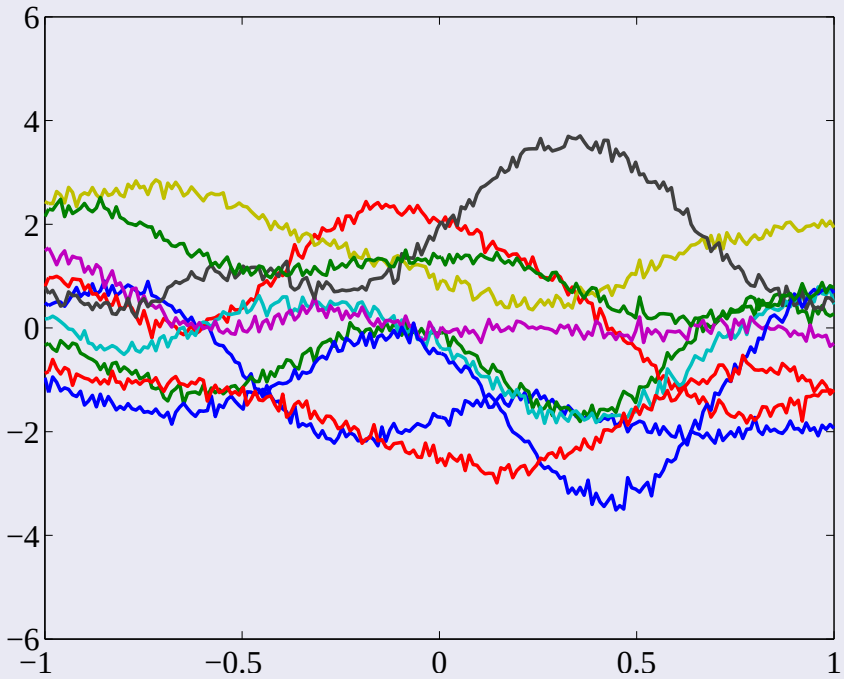


Figure: summed combination of: RBF kernel, $\alpha = 1$, $l = 0.3$; bias kernel, $\alpha = 1$; and white noise kernel, $\beta = 100$

Joint Distribution

Making Predictions

- Covariance function provides the joint distribution over the instantiations.
- Conditional distribution provides predictions.
- Denoting the training set as $\mathbf{f}$ and test set as $\mathbf{f}_*$.
  - Predict using $p(\mathbf{f}_*|\mathbf{f})$.
  - This conditional distribution is also Gaussian.

Gaussian Process Interpolation

demInterpolation

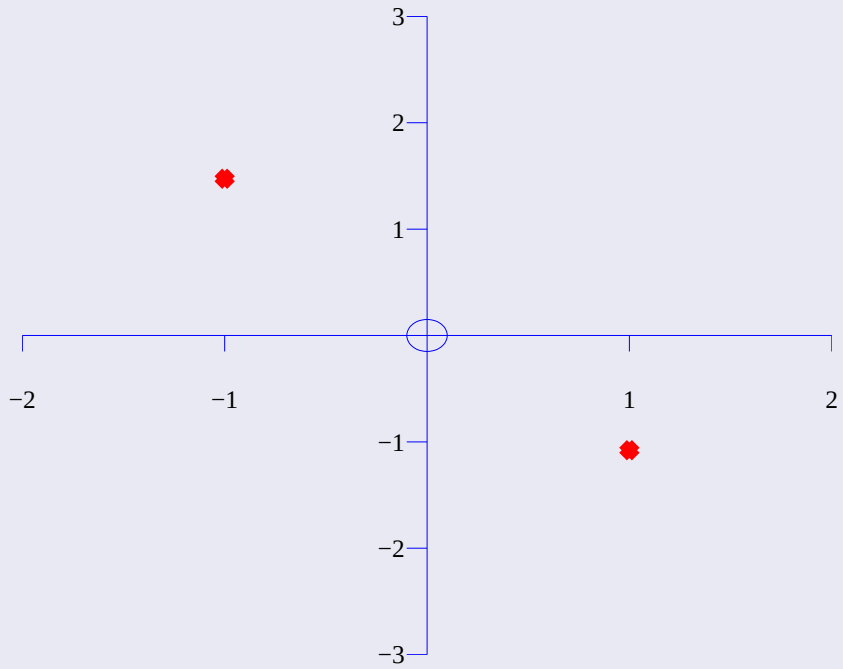


Figure: Real example: BACCO (see *e.g.* [Oakley and O'Hagan, 2002]). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

demInterpolation

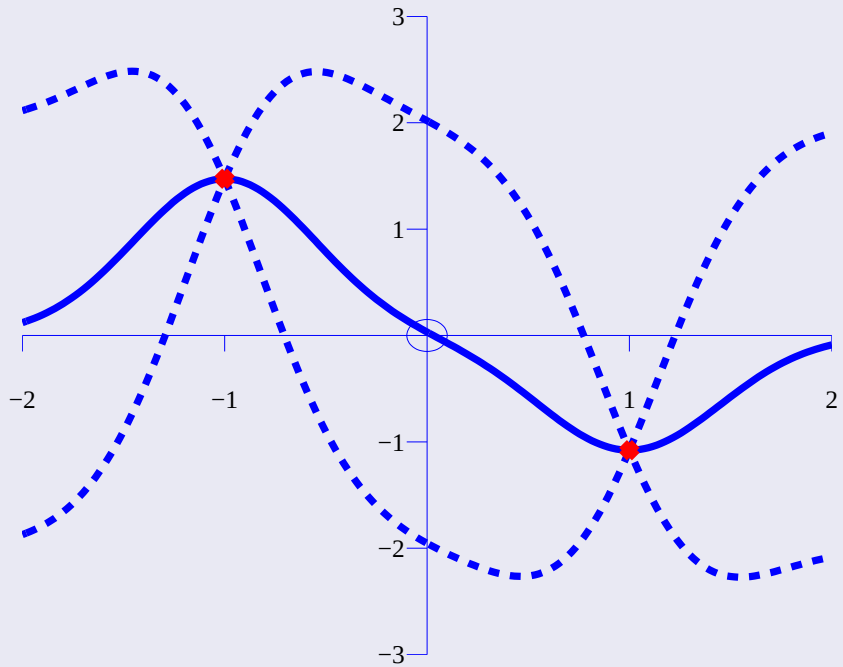


Figure: Real example: BACCO (see *e.g.* [Oakley and O'Hagan, 2002]). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

demInterpolation

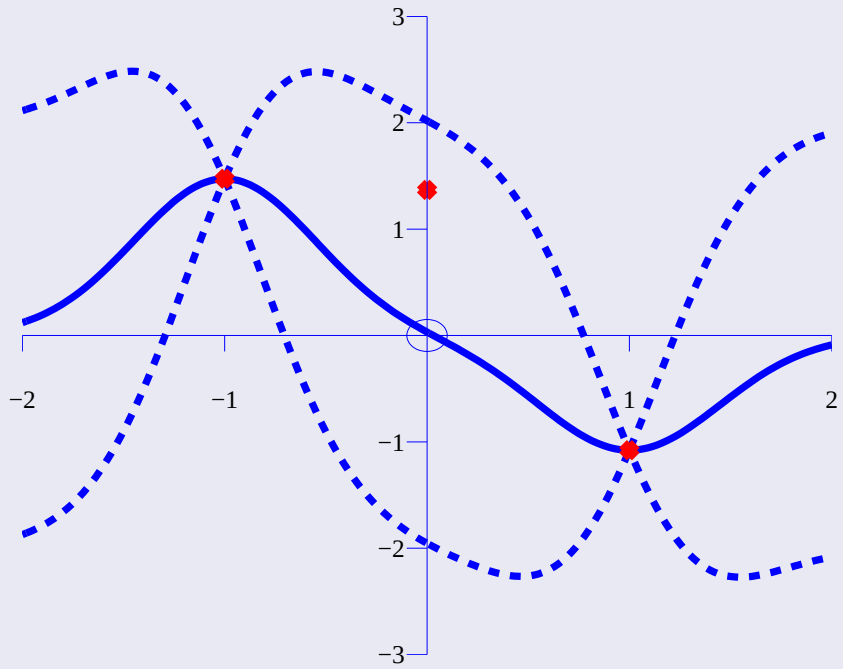


Figure: Real example: BACCO (see *e.g.* [Oakley and O'Hagan, 2002]). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

demInterpolation

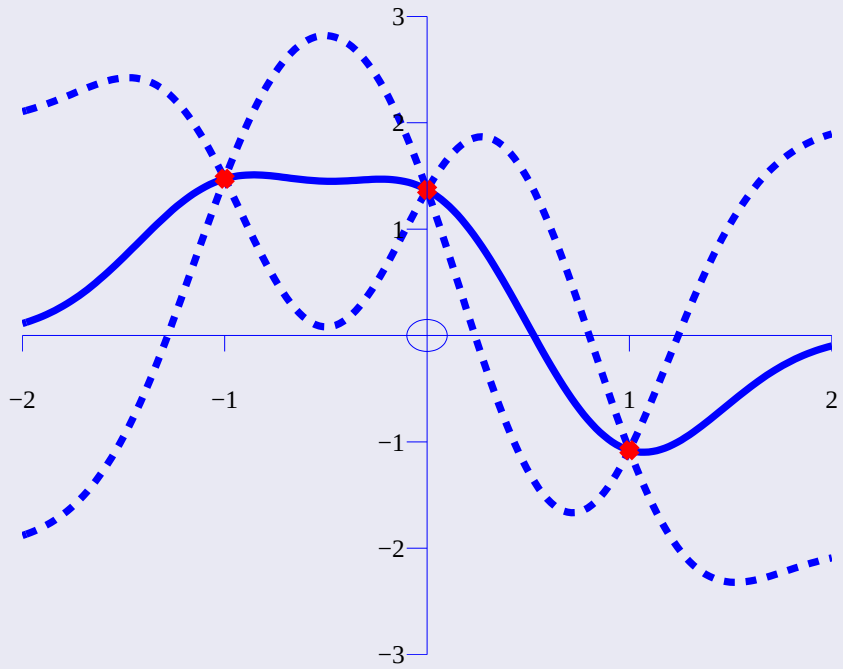


Figure: Real example: BACCO (see *e.g.* [Oakley and O'Hagan, 2002]). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

demInterpolation

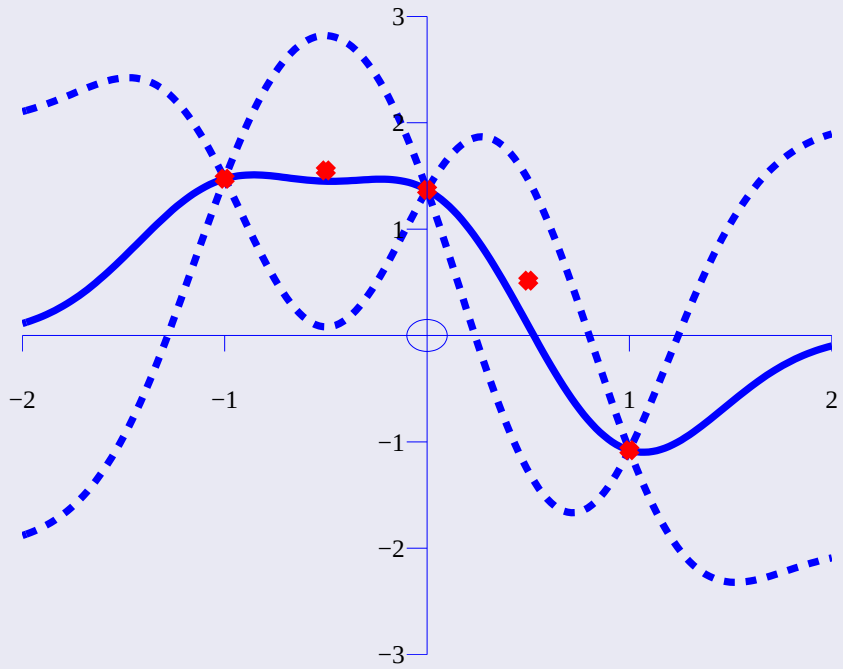


Figure: Real example: BACCO (see *e.g.* [Oakley and O'Hagan, 2002]). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

demInterpolation

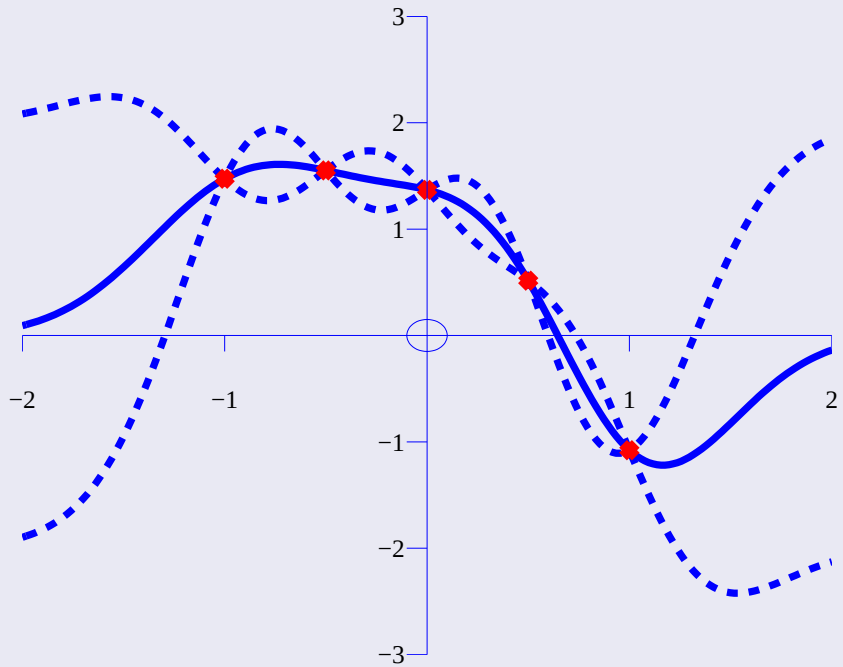


Figure: Real example: BACCO (see *e.g.* [Oakley and O'Hagan, 2002]). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

demInterpolation

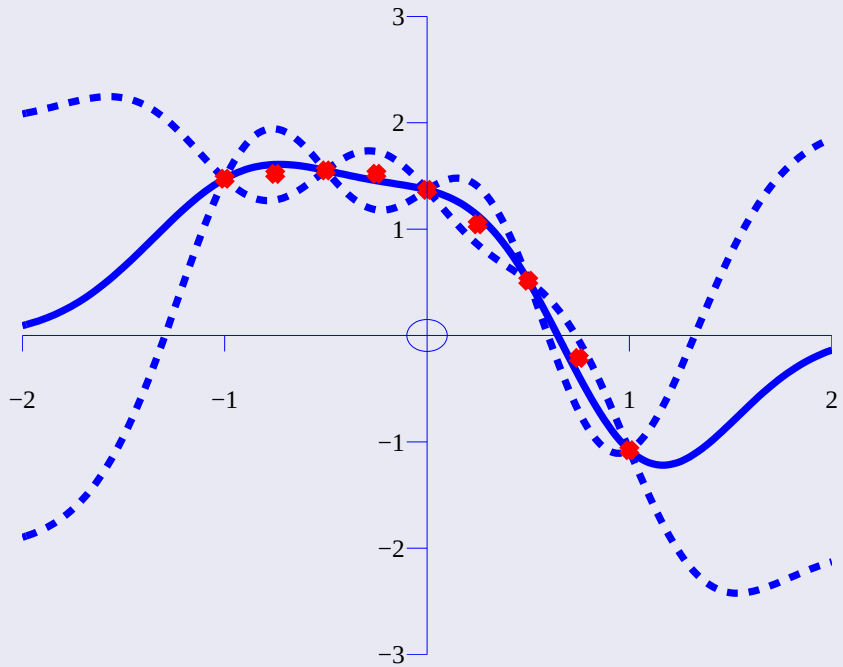


Figure: Real example: BACCO (see *e.g.* [Oakley and O'Hagan, 2002]). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

Gaussian Process Interpolation

demInterpolation

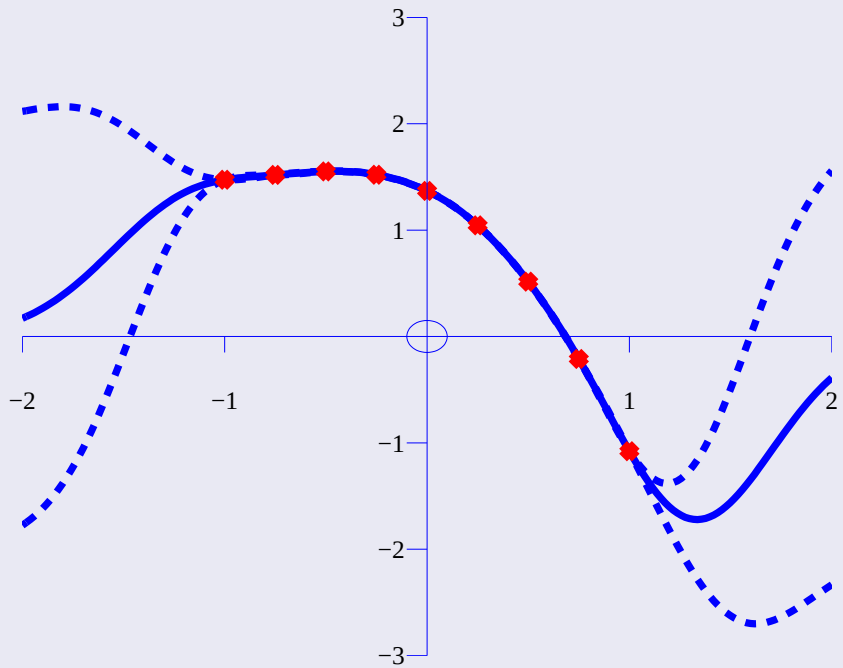


Figure: Real example: BACCO (see *e.g.* [Oakley and O'Hagan, 2002]). Interpolation through outputs from slow computer simulations (*e.g.* atmospheric carbon levels).

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 $y_n|f_n$.
- Noise model, $p(y_n|f_n)$ can take several forms.
- Simplest is Gaussian noise.

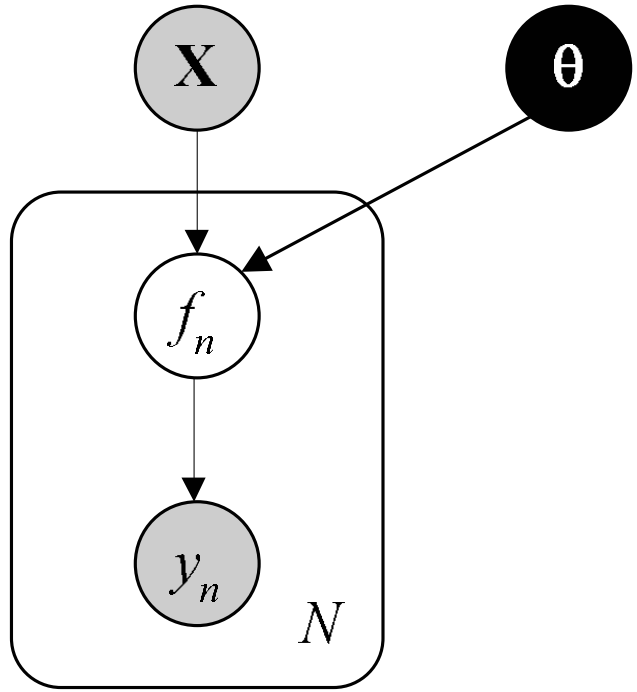


Figure: The Gaussian process depicted graphically.

Gaussian Process Regression

demRegression

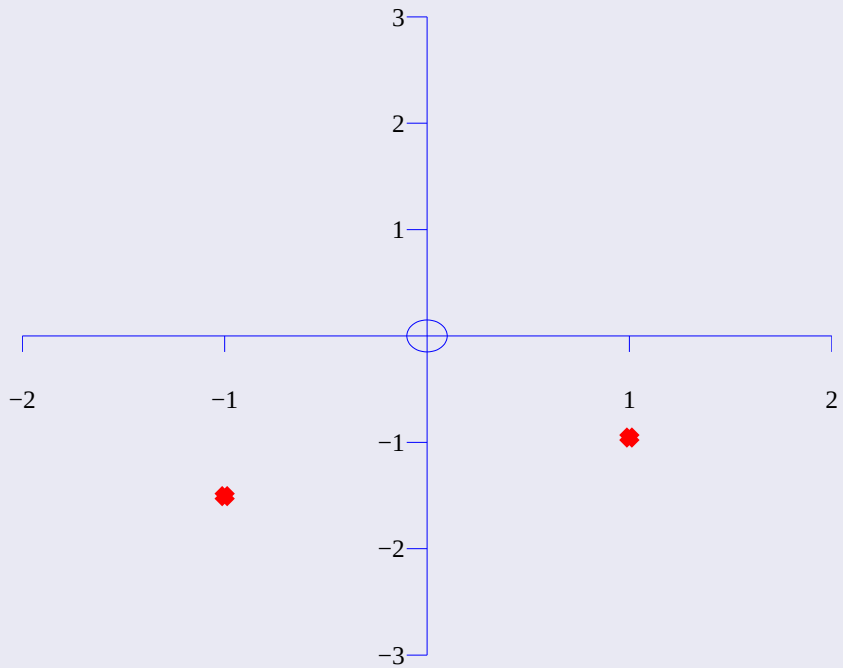


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

demRegression

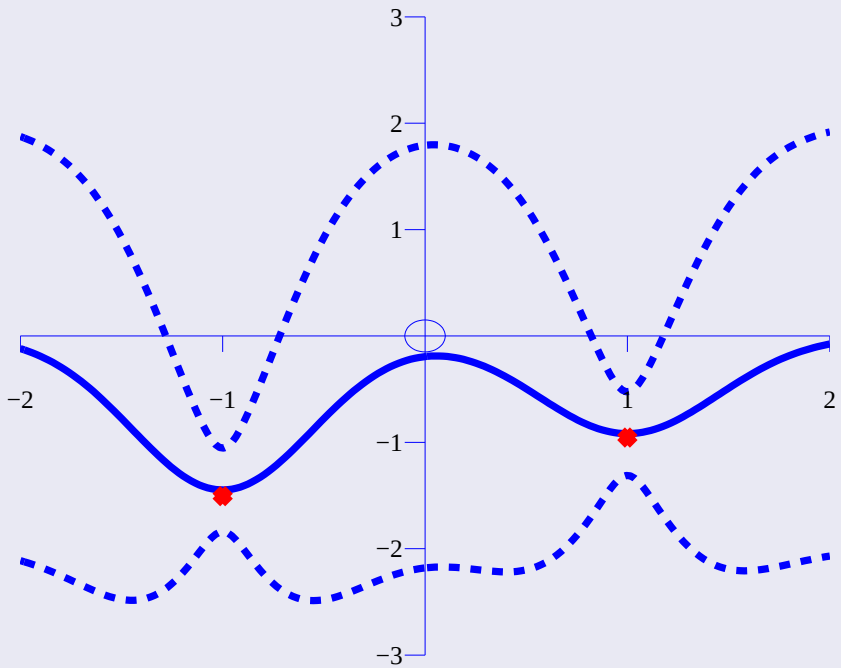


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

demRegression

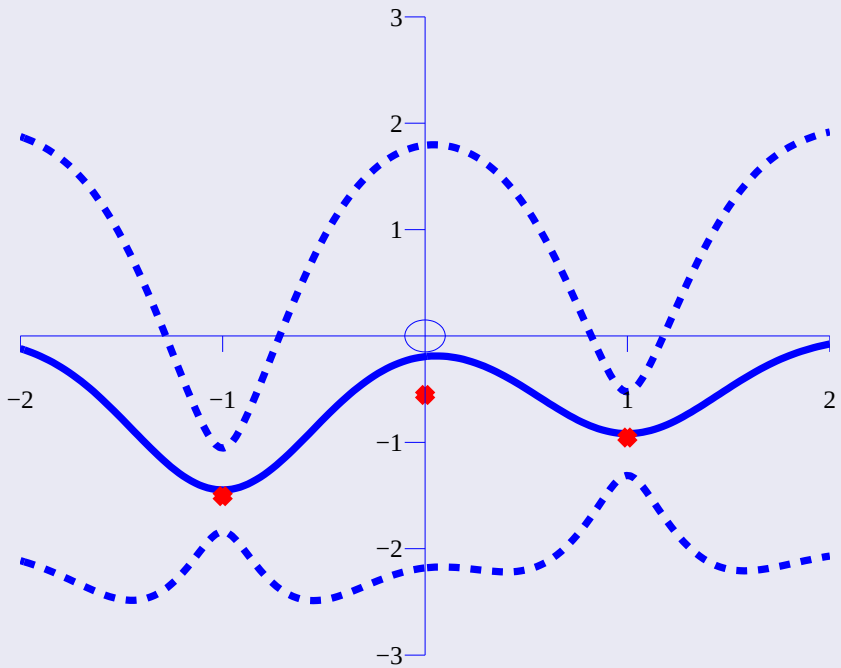


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

demRegression

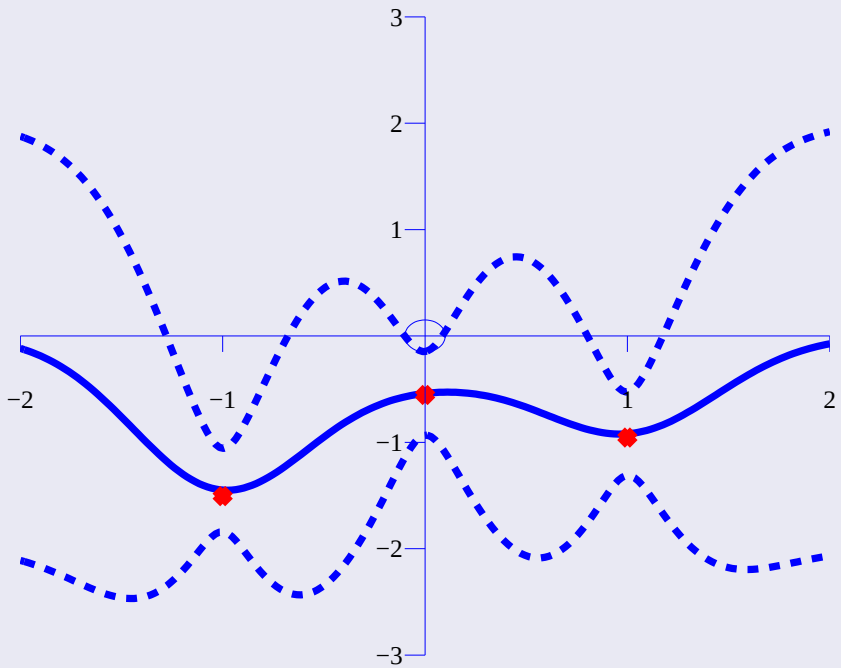


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

demRegression

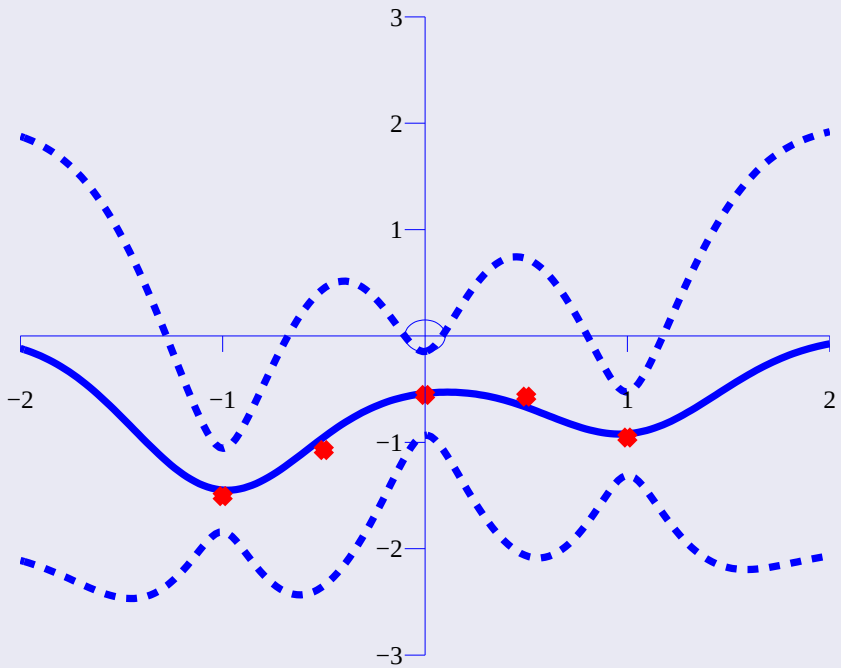


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

demRegression

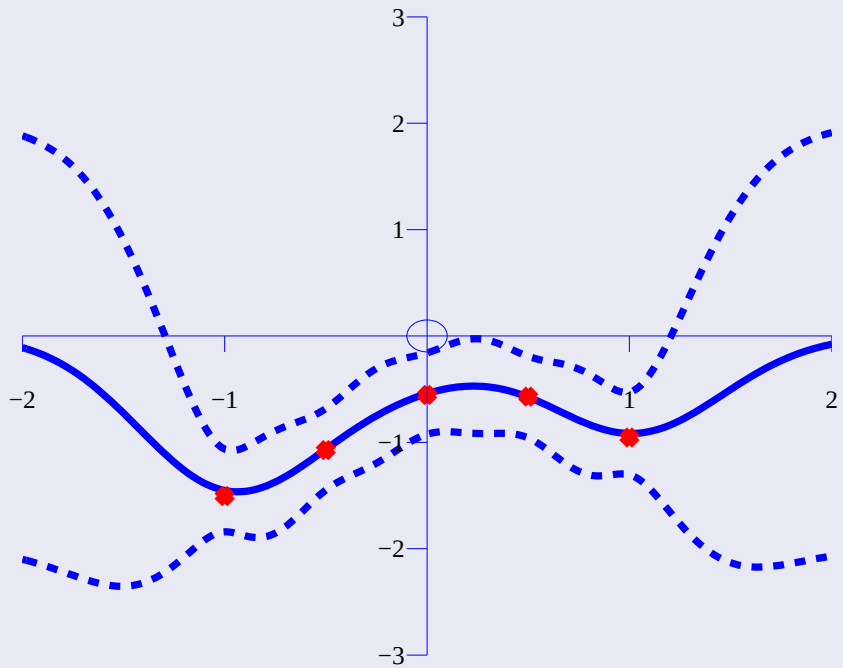


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

demRegression

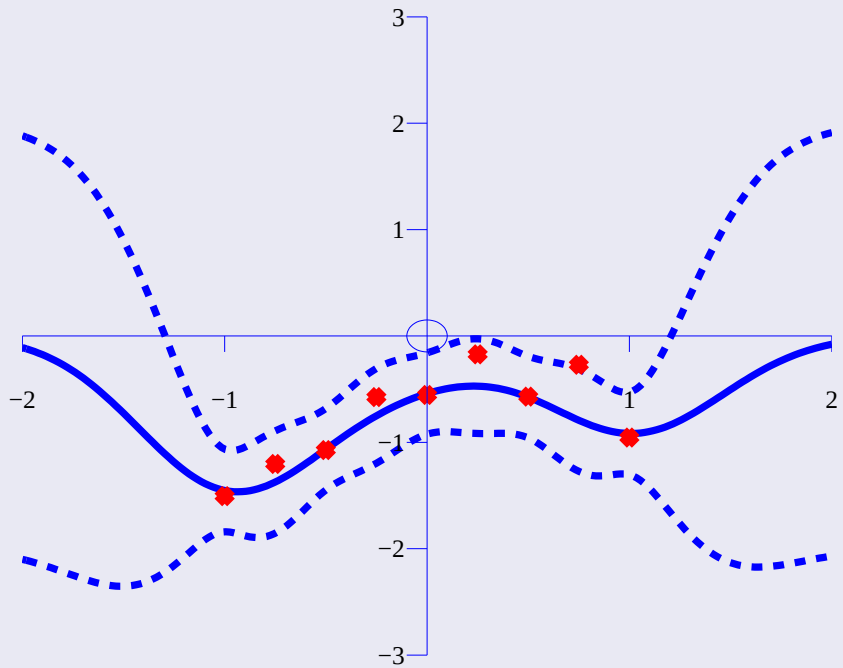


Figure: Examples include WiFi localization, C14 callibration curve.

Gaussian Process Regression

demRegression

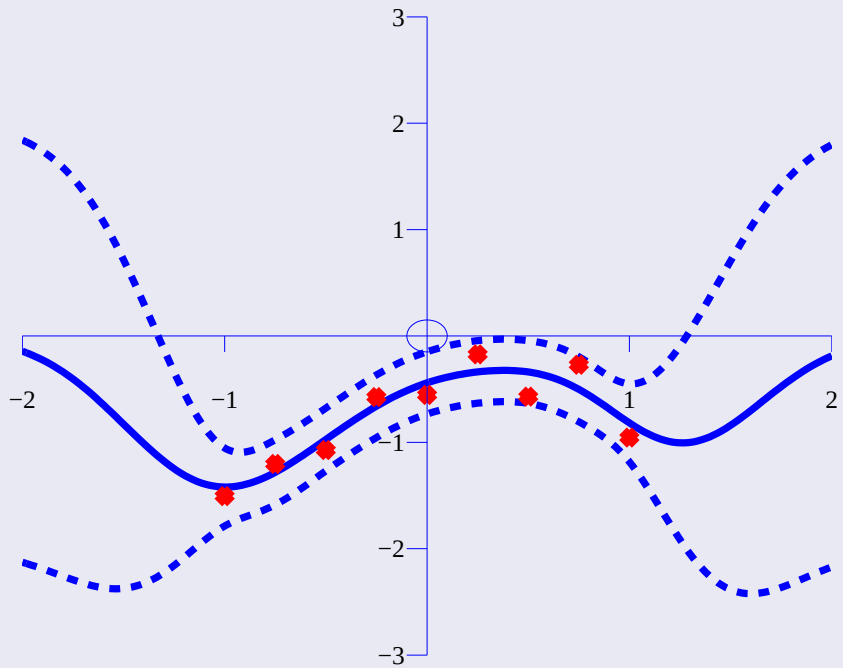


Figure: Examples include WiFi localization, C14 callibration curve.

A Paradigm Shift from i.i.d.

Parameteric Model

$$p\left(y_n|\mathbf{x}_n, \mathbf{w}\right) = N\left(y_n|\mathbf{w}^{\mathrm{T}}\mathbf{x}_n, \sigma^2\right)$$

$$p\left(\mathbf{y}|\mathbf{X}, \mathbf{w}\right) = \prod_{n=1}^{N} p\left(y_n|\mathbf{x}_n, \mathbf{w}\right)$$

Parameteric models normally assume independence given parameters.

A Paradigm Shift from i.i.d.

Gaussian process

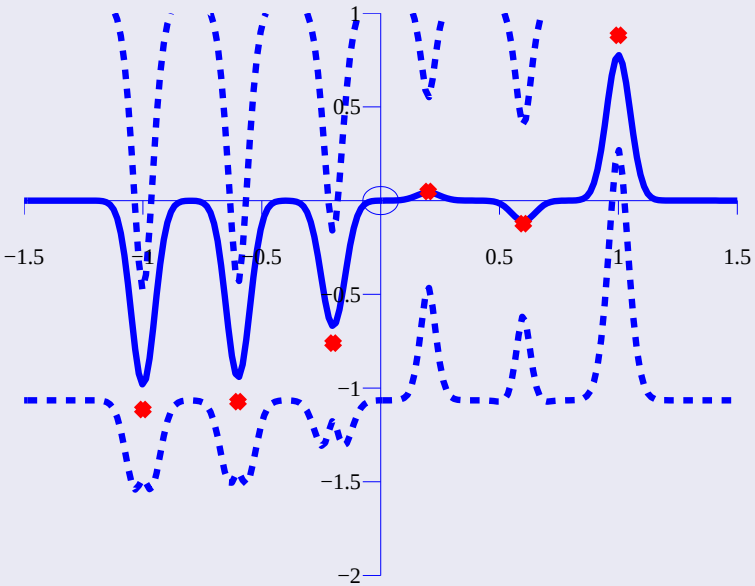
$$p(\mathbf{y}|\mathbf{X}) = N(\mathbf{y}|\mathbf{0}, \mathbf{K})$$

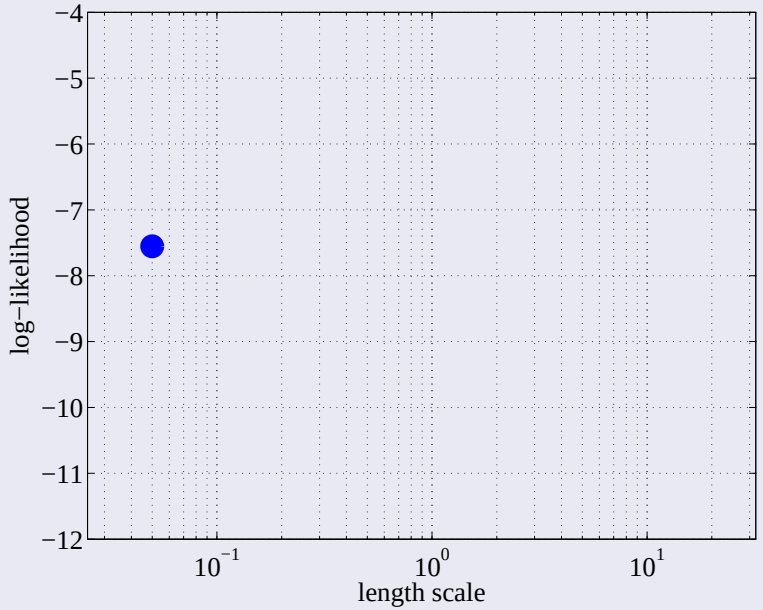
In GPs no i.i.d. assumption is made the kernel expresses correlations.

Learning Kernel Parameters

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

demOptimiseKern

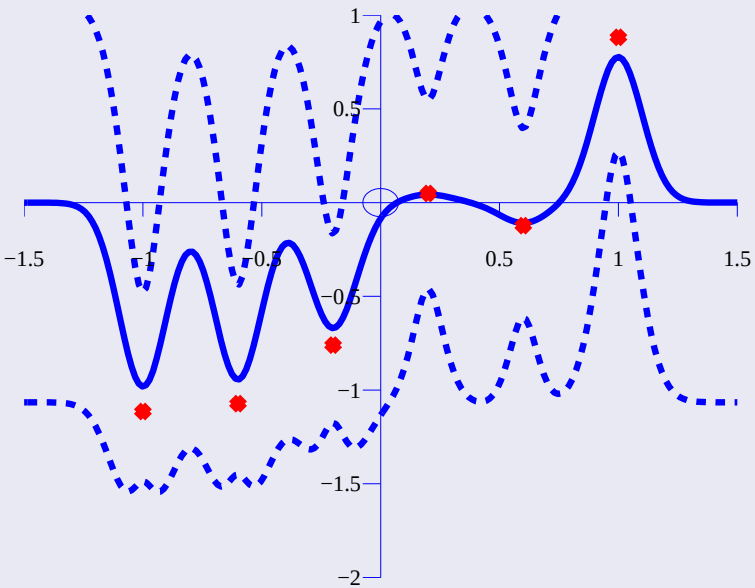


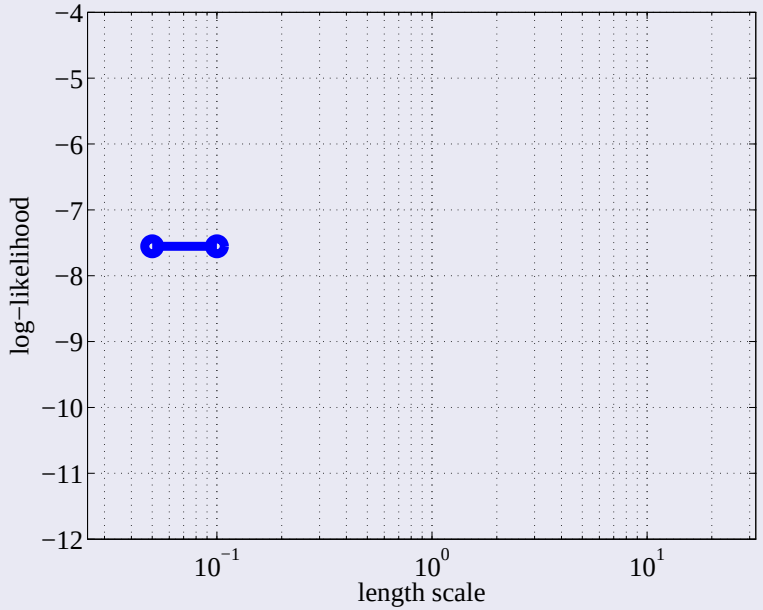


Learning Kernel Parameters

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

demOptimiseKern

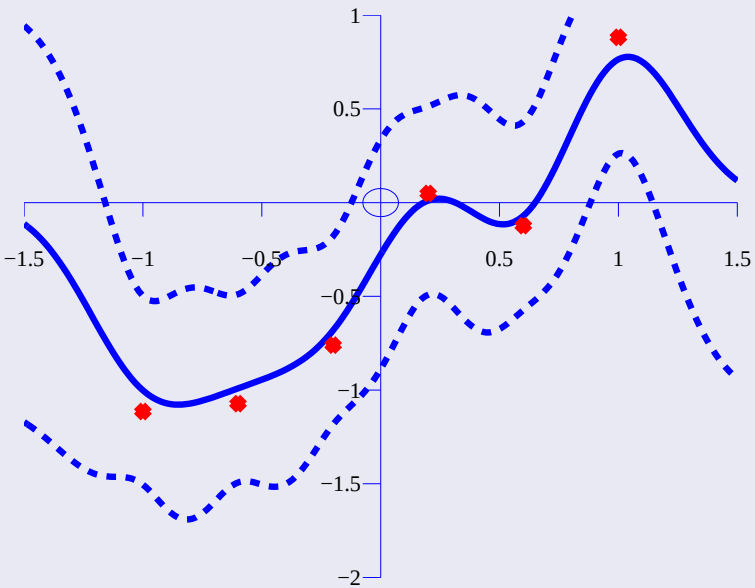


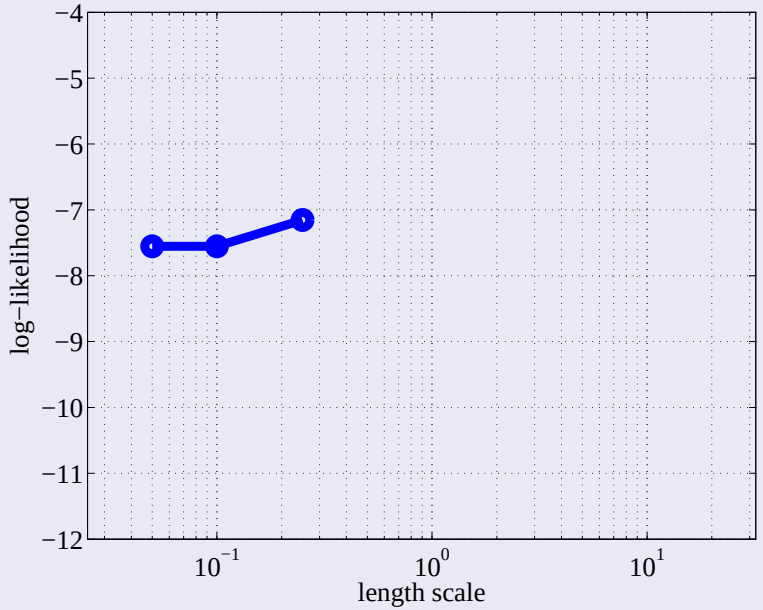


Learning Kernel Parameters

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

demOptimiseKern

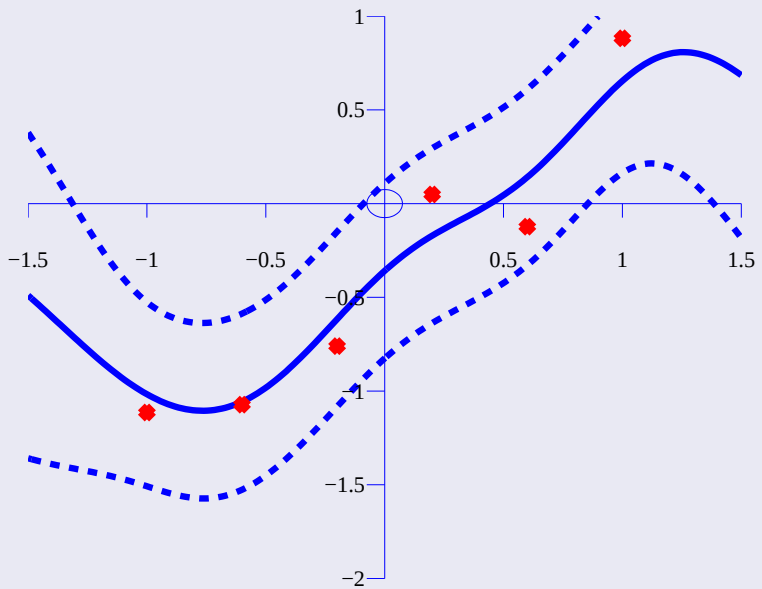


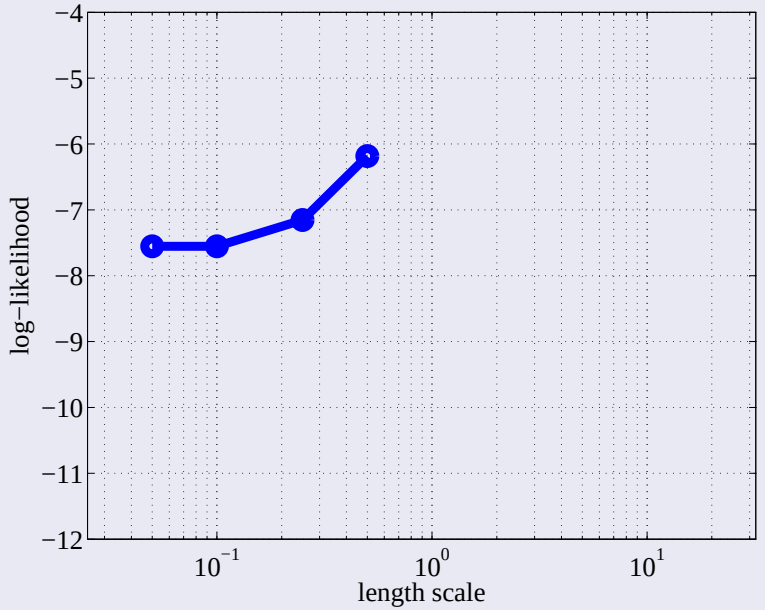


Learning Kernel Parameters

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

demOptimiseKern

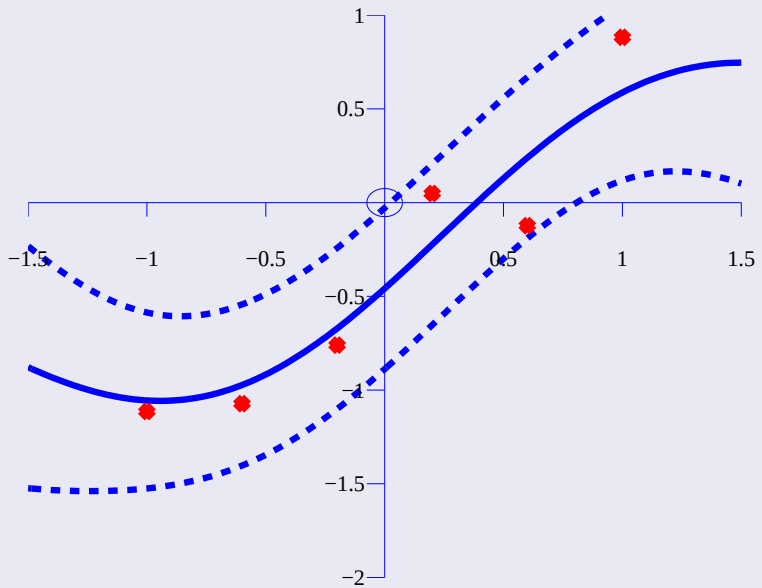


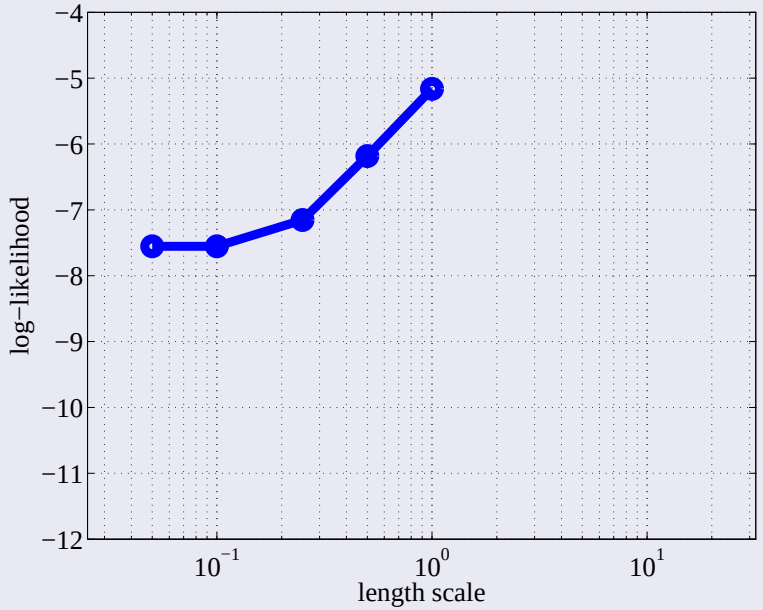


Learning Kernel Parameters

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

demOptimiseKern

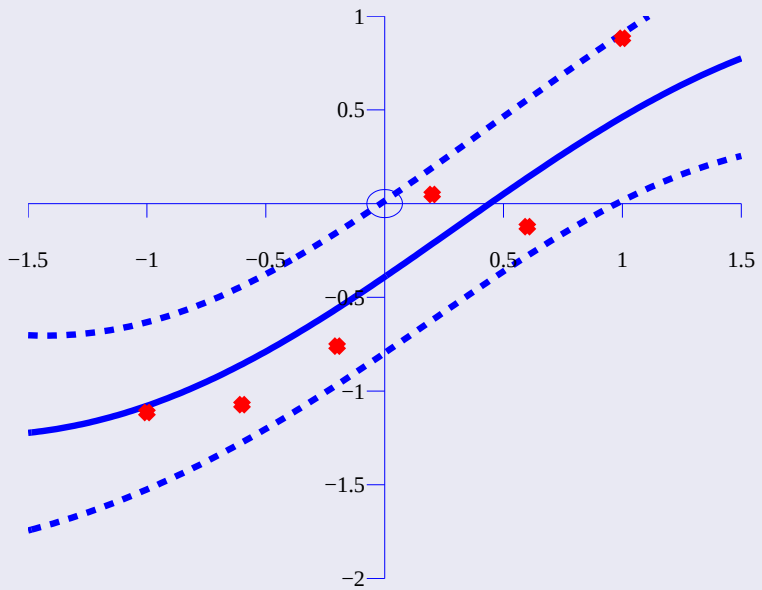


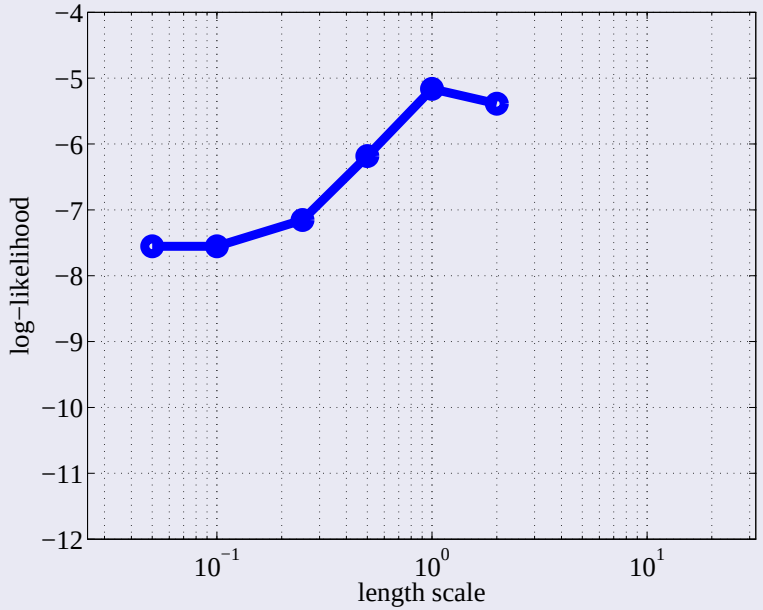


Learning Kernel Parameters

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

demOptimiseKern

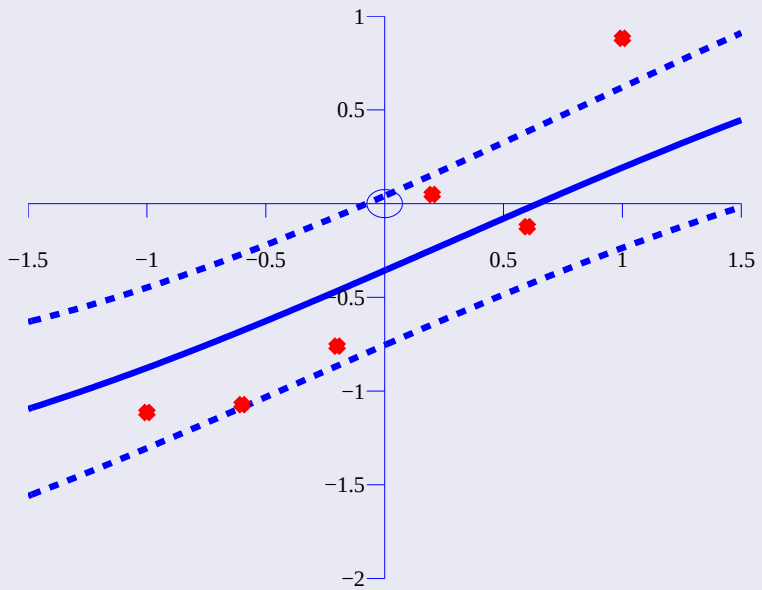


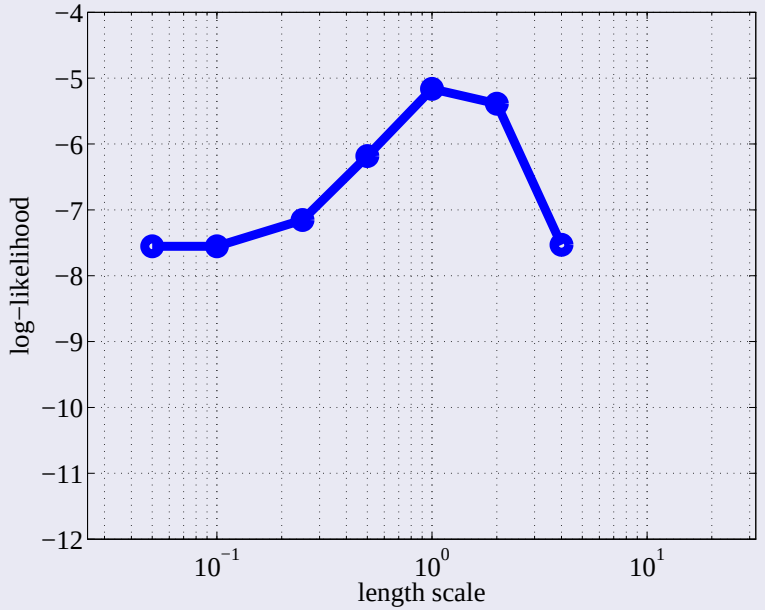


Learning Kernel Parameters

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

demOptimiseKern

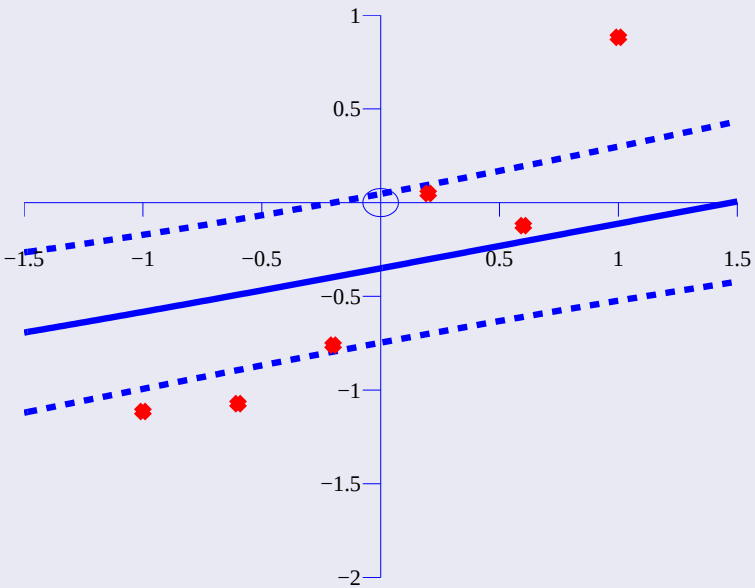


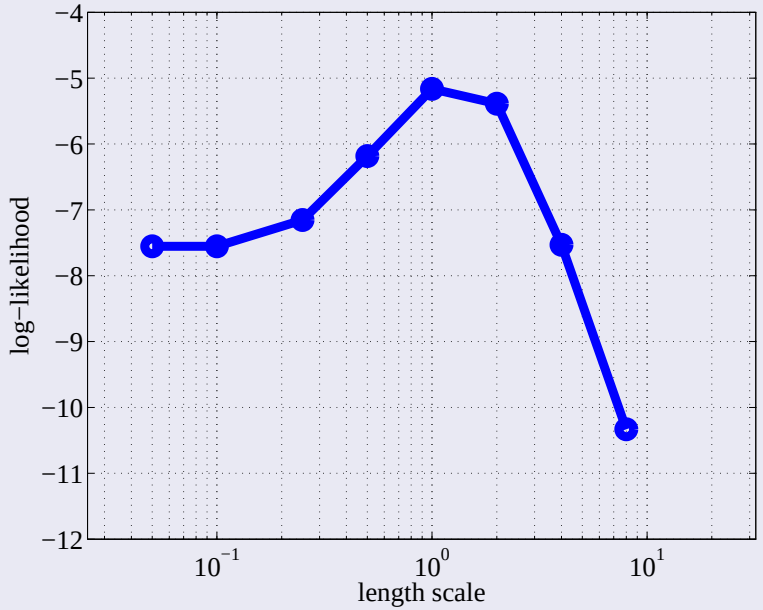


Learning Kernel Parameters

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

demOptimiseKern

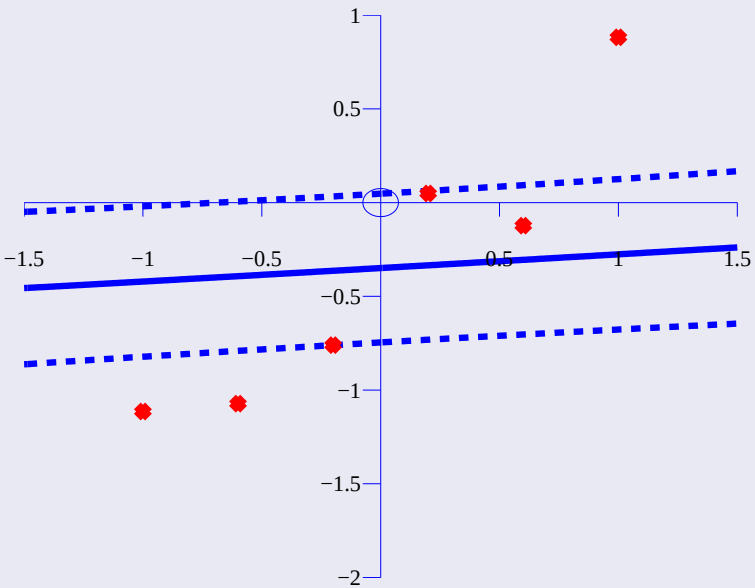


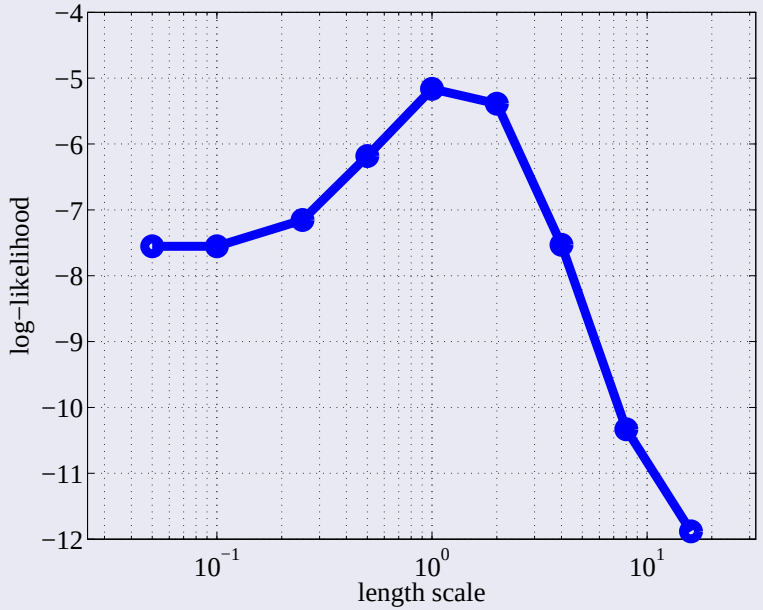


Learning Kernel Parameters

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

demOptimiseKern





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 $y_n|f_n$.
- In general $p(y_n|f_n)$ is non-Gaussian.
- We approximate with Gaussian $p(y_n|f_n) \approx N\left(m_n|f_n, \beta_n^{-1}\right)$.

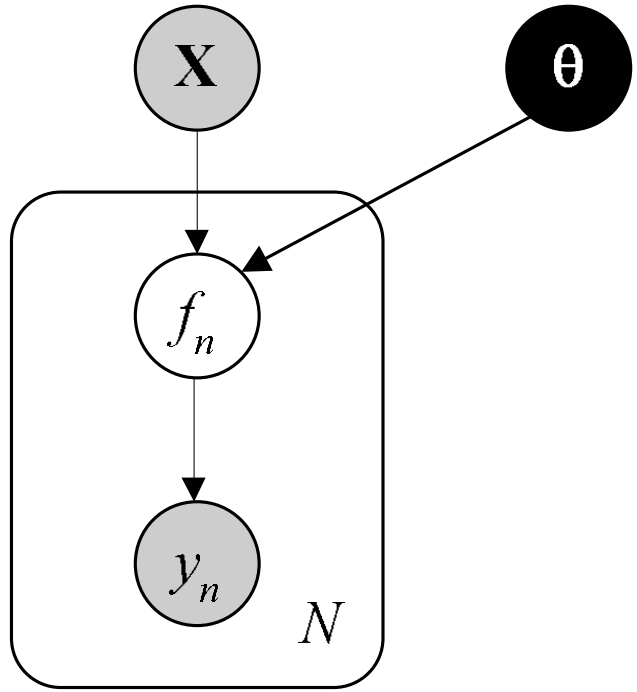


Figure: The Gaussian process depicted graphically.

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

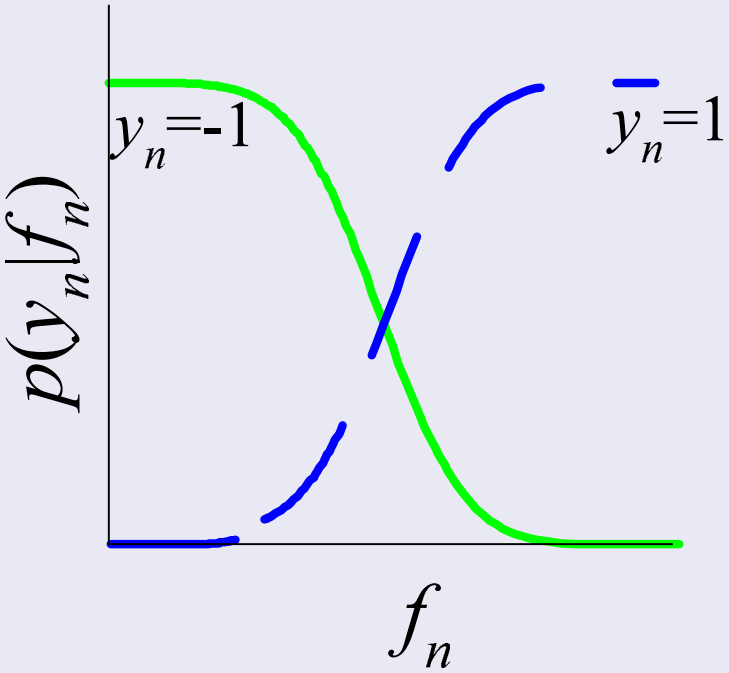


Figure: The probit model (classification). The plot shows $p(y_n|f_n)$ for different values of y_n. For $y_n = 1$ $p(y_n|f_n) = \phi(f_n) = \int_{-\infty}^{f_n} N(z|0,1)\,dz$.

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 $q\left(f_*|\mathbf{y}, y_*, \mathbf{X}, \mathbf{x}_*\right)$ is constrained to be a GP.

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

- This is equivalent to setting

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

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

Expectation Propagation III

Equivalent Gaussian

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

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

becomes

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

Classification

```
epPointUpdate('probit', 1, -1, .1, .6, 1e-2)
```

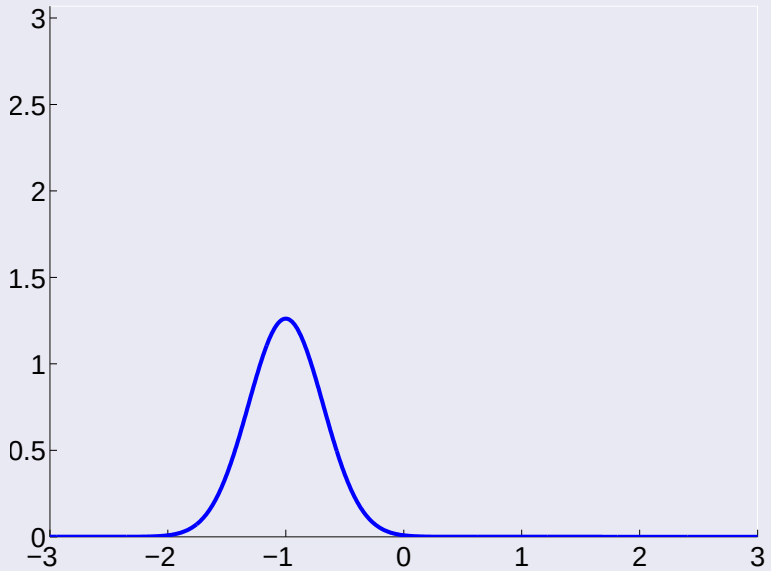


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$.

Classification

epPointUpdate('probit', 1, -1, .1, .6, 1e-2)

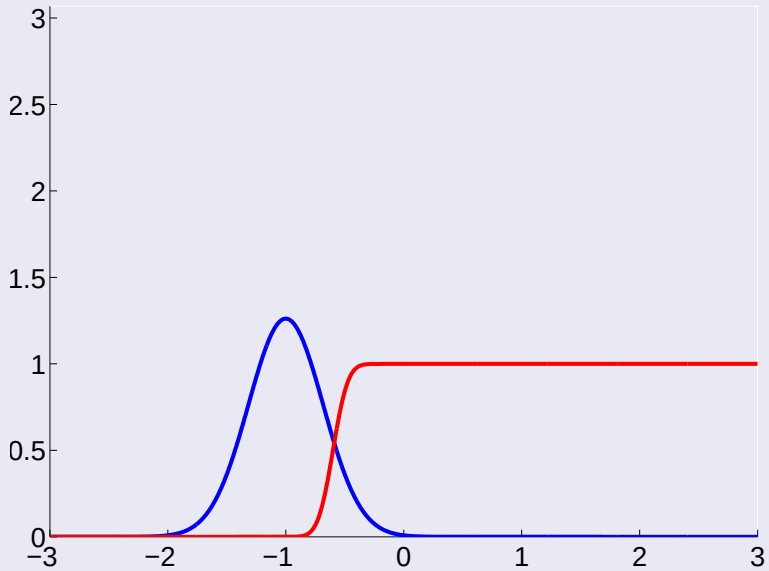


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* = 1|f_*)$.

Classification

`epPointUpdate('probit', 1, -1, .1, .6, 1e-2)`

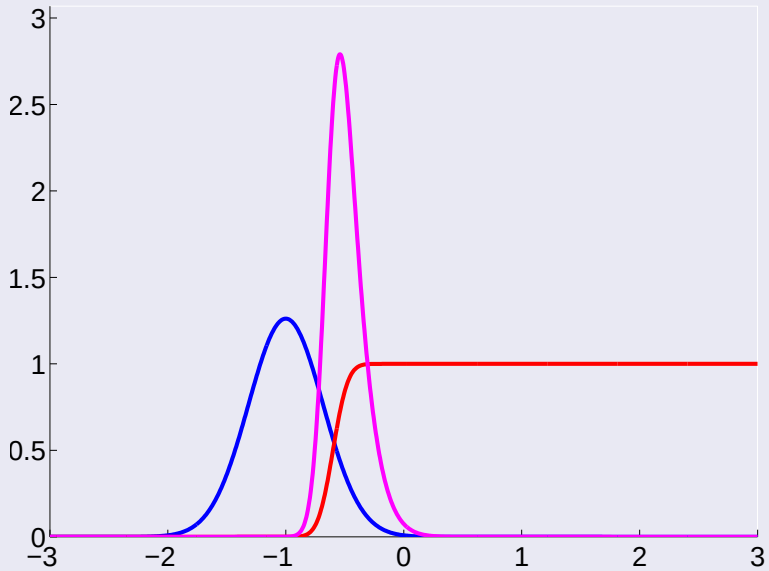


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* = 1|f_*)$, *Magenta:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, y_*)$.

Classification

`epPointUpdate('probit', 1, -1, .1, .6, 1e-2)`

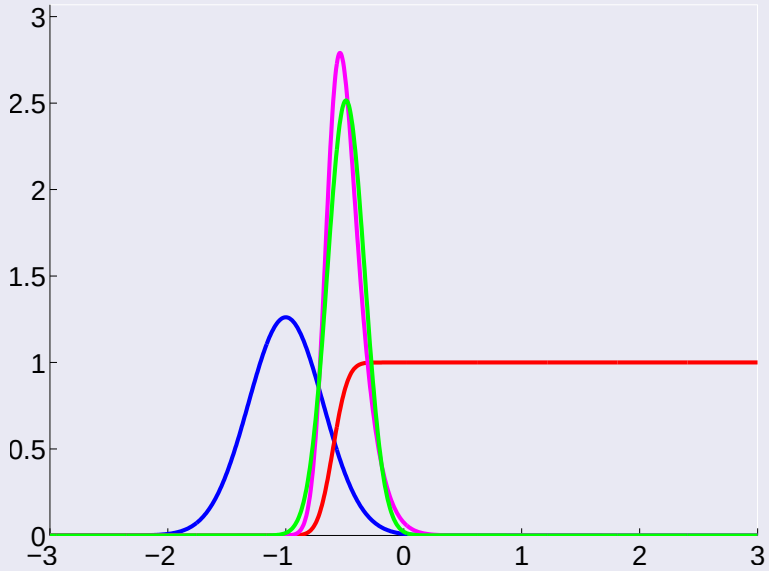


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* = 1|f_*)$, *Magenta:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, y_*)$, *Green:* $q(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$.

Ordinal Noise Model

Ordered Categories

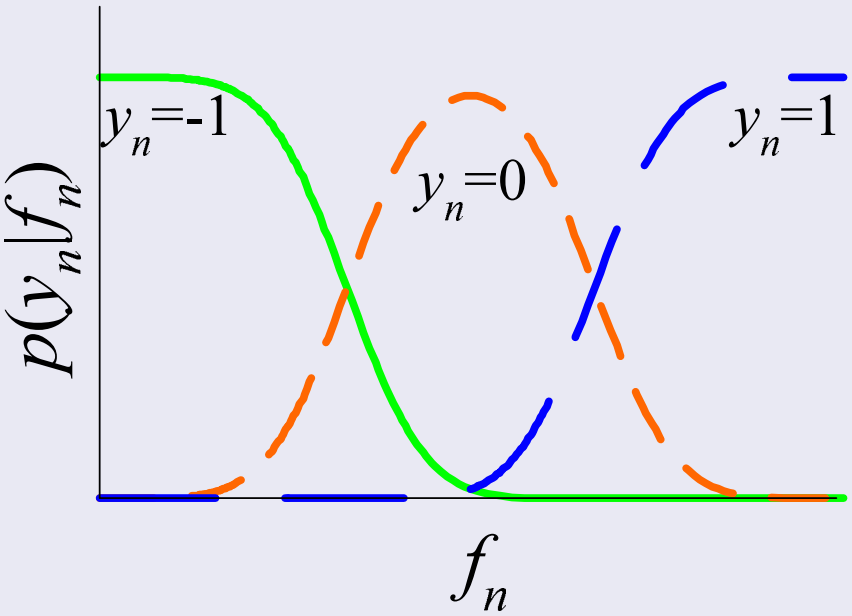


Figure: The ordered categorical noise model (ordinal regression). The plot shows $p(y_n|f_n)$ for different values of y_n. Here we have assumed three categories.

Ordinal Regression

`epPointUpdate('ordered', 1, -1, .1, .6, 1e-3)`

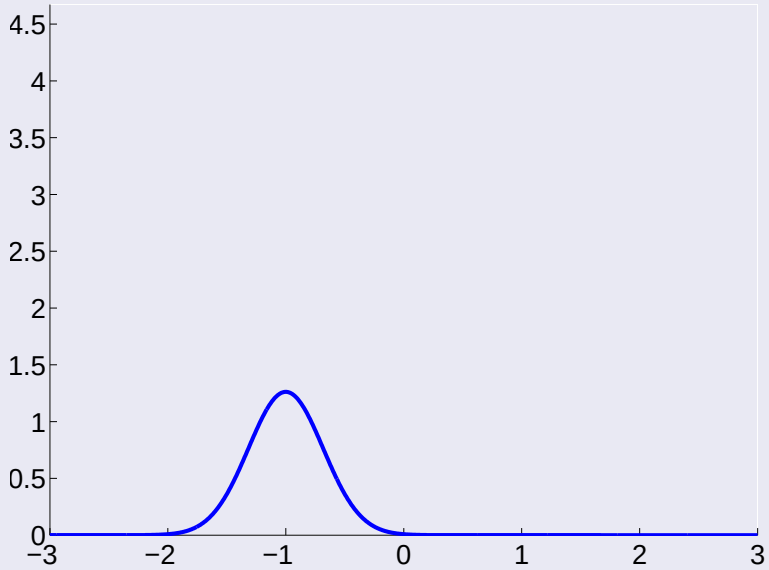


Figure: An EP style update with an ordered category noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$.

Ordinal Regression

epPointUpdate('ordered', 1, -1, .1, .6, 1e-3)

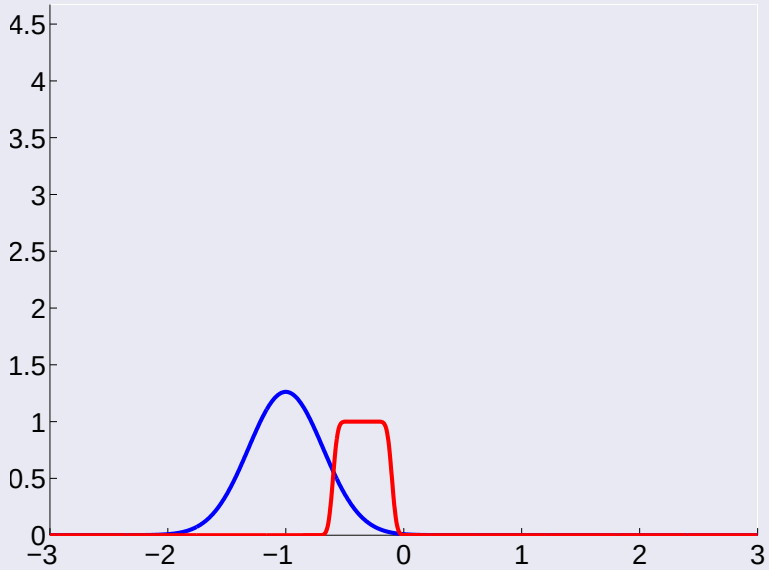


Figure: An EP style update with an ordered category noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* = 0|f_*)$.

Ordinal Regression

`epPointUpdate('ordered', 1, -1, .1, .6, 1e-3)`

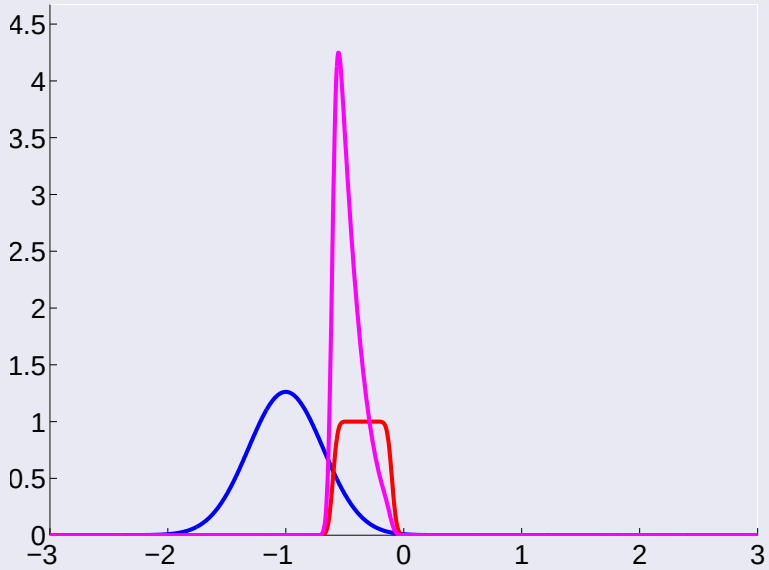


Figure: An EP style update with an ordered category noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* = 0|f_*)$, *Magenta:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, y_*)$.

Ordinal Regression

epPointUpdate('ordered', 1, -1, .1, .6, 1e-3)

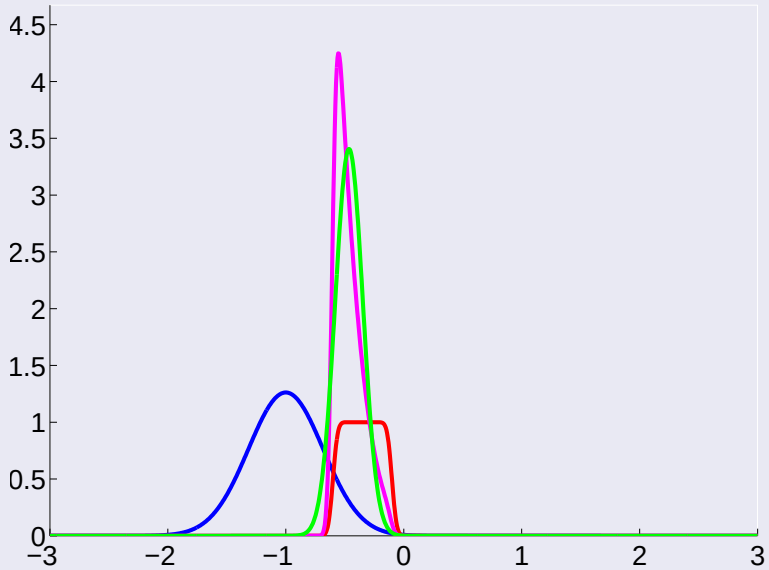


Figure: An EP style update with an ordered category noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* = 0|f_*)$, *Magenta:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, y_*)$, *Green:* $q(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$.

The Informative Vector Machine

Reduce Complexity

- Including N data points through ADF still leads to an $O\left(N^3\right)$ complexity.
- IVM algorithm resolves these problems with a sparse representation for the data set.
- Inspiration: the support vector machine.
- IVM use a simple selection heuristic to incorporate d most informative points [Lawrence et al., 2003, Seeger, 2004, Lawrence et al., 2005]. B
- Computational complexity: $O\left(N^3\right)$ to $O\left(d^2N\right)$.
- Infromation theoretic [Chaloner and Verdinelli, 1995] criteria used to select points.

Data Point Selection

Entropy Criterion

- Original IVM criterion inspired by support vectors being those that reduce the size of the 'version space' most.
- The equivalent Bayesian interpretation is volume of the posterior: measured by *entropy*.
- Entropy change associted with a data point is simple and quick to compute.
- For ith inclusion of nth data point:

$$\begin{aligned}\Delta H_{in} &= -\frac{1}{2}\log|\Sigma_{i,n}| + \frac{1}{2}\log|\Sigma_{i-1}| \\ &= -\frac{1}{2}\log|\mathbf{I} - \Sigma_{i-1}\mathrm{diag}(\boldsymbol{\nu}_i)| \\ &= -\frac{1}{2}\log(1 - \nu_{in}\varsigma_{i-1,n}). \end{aligned} \quad (1)$$

IVM Parameter Updates

Optimising Kernel Parameters

- Need to express the marginal likelihood for optimization.
- Seeger [2004] achieves by expressing the likelihood in terms of both the active and inactive sets.
- We simply express the likelihood in terms of the *active* set only.
- Given the active set, I, and the site parameters, $\mathbf{m}$ and $\boldsymbol{\beta}$, optimise approximation wrt kernel parameters using gradient methods.
- Active set and kernel parameters are interdependent: active set is reselected between optimisations of kernel parameters.

Results

Toy Problems

- Two toy data sets for classification with probit noise. First uses an ARD set up and one irrelevant direction.
- A second demonstation: sampled 500 data points uniformly from a unit square in two dimensions.
  - Sample then made from a GP prior of a function at these points.
  - This function was 'squashed' by a cumulative Gaussian and a class assigned according to this probability.

IVM Classification

Ordered Categorical

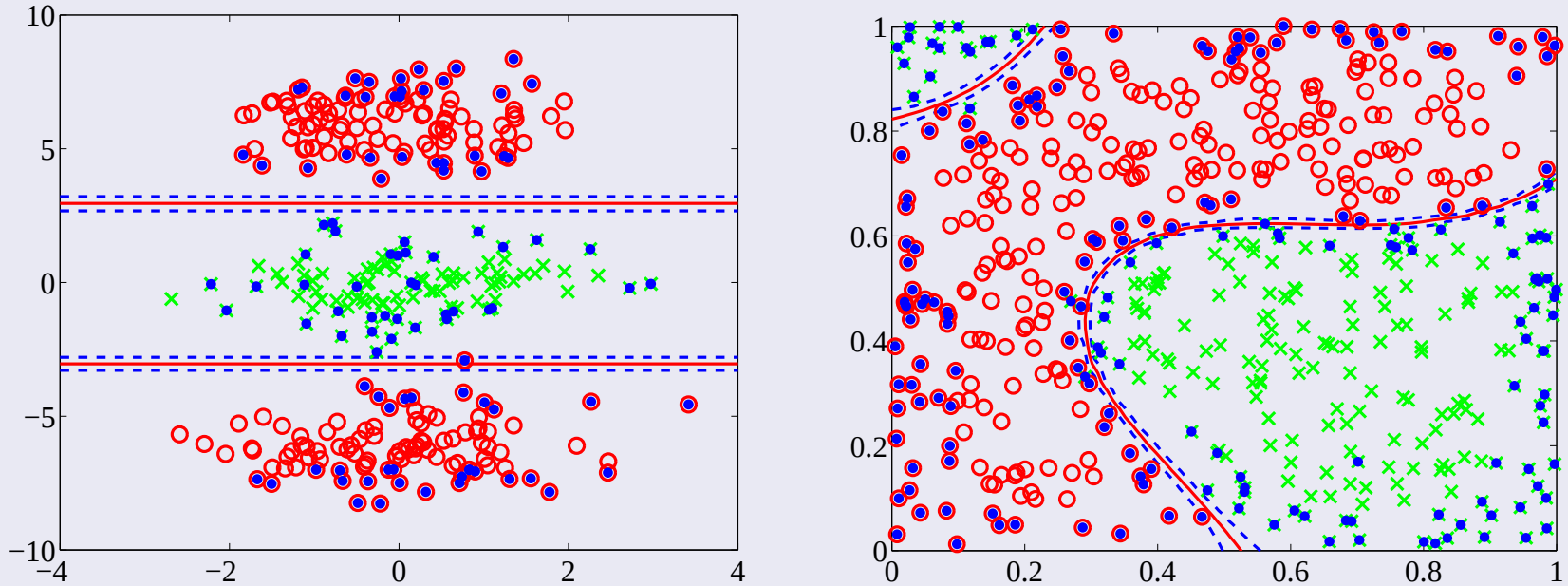


Figure: *Contours*: Red solid line at $p(y|\mathbf{x}) = 0.5$, blue dashed lines at $p(y|\mathbf{x}) = 0.25$ and $p(y|\mathbf{x}) = 0.75$. Active points are blue dots. *Left*: data sampled from from a mixture of Gaussians. *Right*: Data uniformly sampled on the 2–dimensional unit square. Class labels are assigned by sampling from a known Gaussian process prior.

Ordered Categories

Ordered Categories

- Two results from two problems on ordered categorical data.
- First example the categories are separable *linearly*.
- Second example: sampled ordered categorical data in polar co-ordinates.

Ordered Categories

Toy Problems

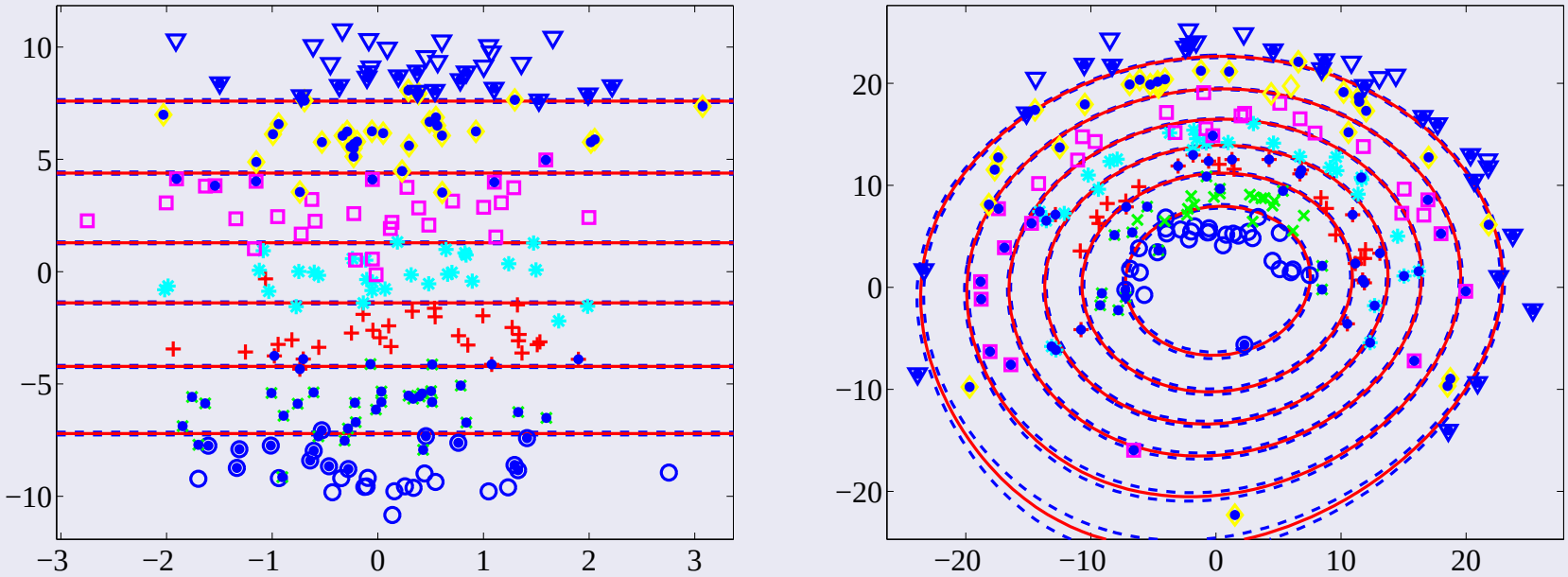


Figure: .*Left*: a linear solution is found. *Right*: this categories in this example were sampled in polar co-ordinates.

USPS digits

Large Data Set

- USPS digit data set of 16×16 greyscale images.
- Contains 7291 training images and 2007 test images.
- Three different kernels with the IVM algorithm.
  - For each data-set we used a 'base kernel' consisting of a linear part, a white noise term and a bias part.
  - Three variations on this base kernel were then used: it was changed by adding first an RBF kernel, then an MLP kernel and finally a variant of the RBF ARD kernel.
  - Set $d = 500$.

USPS digits

Classification error %

	0	1	2	3	4	5	6	7	8	9	Overall
RBF	0.65	0.70	1.40	1.05	1.49	1.25	0.75	0.60	1.20	0.75	4.58
MLP	0.55	0.70	1.49	1.20	1.64	1.25	0.80	0.60	1.20	0.75	4.78
RBF ARD	0.55	0.60	1.49	1.10	1.79	1.20	0.80	0.60	1.20	0.85	4.68

Table: Table of results on the USPS digit data. A comparison with a summary of results on this data-set Schölkopf and Smola [2001, Table 7.4] shows that the IVM is in line with other results on this data. Furthermore these results were achieved with fully automated model selection.

Incorporating Invariances

Virtual Support Vectors

- Invariances present: rotations, translations.
- Could augment the original data set with transformed data points.
- This leads to a rapid expansion in the size of the data set.
- Schölkopf et al. [1996] suggest augmenting only support vectors.
- Augmented points known as 'virtual support vectors'.
- This algorithm gives state-of-the-art performance on the USPS data set.

USPS with Virtual Informative Vectors

Virtual Informative Vectors

- Schölkopf et al. [1996]: biggest improvement using translation invariances.
- Applied standard IVM classification algorithm to the data set using an RBF kernel combined with a linear term.
- Took the active set from these experiments and aumented it:
  - original active set plus four translations: up down lweft and right
  - results in an augmented active set of 2500 points.
- Reselect active set of size $d = 1000$ for final results.

Performance on USPS

Classification Error %

0	**1**	**2**	**3**	**4**	
0.648 ± 0.00	0.389 ± 0.03	0.967 ± 0.06	0.683 ± 0.05	1.06 ± 0.02	
5	**6**	**7**	**8**	**9**	**Overall**
0.747 ± 0.06	0.523 ± 0.03	0.399 ± 0.00	0.638 ± 0.04	0.523 ± 0.04	3.30 ± 0.03

Table: Experiments are summarised by the mean and variance of the % classification error across ten runs with different random seeds. Results match those given by the virtual SVM but model selection was automatic here.

Probabilistic Model

Semi-supervised Noise Model

- New noise model: *the null category noise model*.
- Derives from the general class of *ordered categorical models* (or ordinal regression).

$$p(y_n|f_n) = \begin{cases} \phi\left(-\left(f_n + \frac{w}{2}\right)\right) & \text{for } y_n = -1 \\ \phi\left(f_n + \frac{w}{2}\right) - \phi\left(f_n - \frac{w}{2}\right) & \text{for } y_n = 0 \\ \phi\left(f_n - \frac{w}{2}\right) & \text{for } y_n = 1 \end{cases},$$

Ordinal Noise Model

Ordered Categories

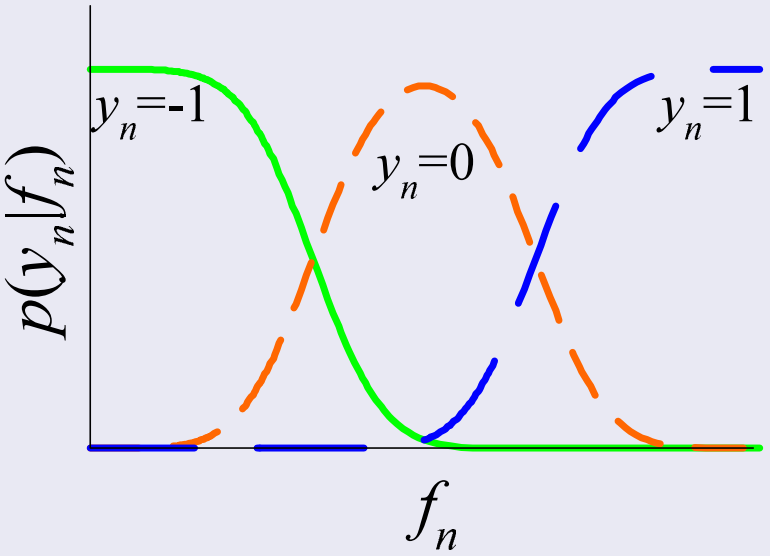
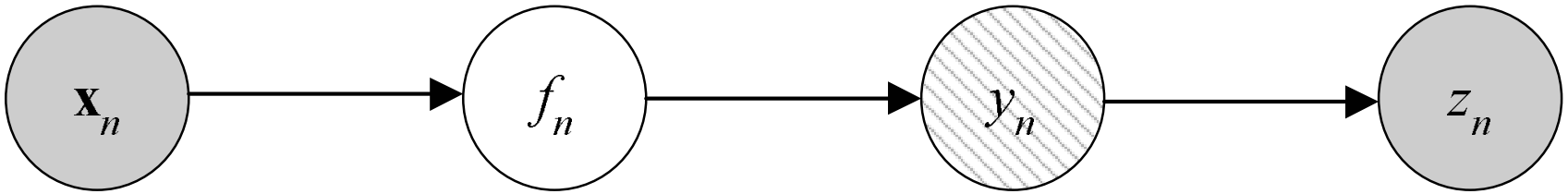


Figure: The ordered categorical noise model (ordinal regression). The plot shows $p(y_n|f_n)$ for different values of y_n. Here we have assumed three categories.

Null Category Noise Model

Noise Model for Semi-supervised Learning

- Indicator variable, $z_n = 1$ if data point is unlabeled.
- We impose the constraint:$p(z_n = 1|y_n = 0) = 0$.
- Assign missing label probabilities $p(z_n = 1|y_n = 1) = \gamma_+$ and $p(z_n = 1|y_n = -1) = \gamma_-$.



Null Category Noise Model

Noise Model for Semi-supervised Learning

- From the graphical representation z_n is d-separated from $\mathbf{x}_n$.
  - When y_n is observed, the posterior process is updated by using $p(y_n|f_n)$.
  - When the data point is unlabeled the posterior process is updated by

$$p(z_n = 1|f_n) = \sum_{y_n} p(y_n|f_n)\, p(z_n = 1|y_n).$$

- The "effective likelihood function" for a single data point, $L(f_n)$, therefore takes one of three forms:

$$L(f_n) = \begin{cases} H\left(-\left(f_n + \frac{1}{2}\right)\right) & \text{for} \quad y_n = -1,\ z_n = 0 \\ \gamma_- H\left(-\left(f_n + \frac{1}{2}\right)\right) + \gamma_+ H\left(f_n - \frac{1}{2}\right) & \text{for} \quad z_n = 1 \\ H\left(f_n - \frac{1}{2}\right) & \text{for} \quad y_n = 1\, z_n = 0 \end{cases}.$$

Null Category Noise Model

Noise Model for Semi-supervised Learning

- The constraint imposed by $p(z_n = 1|y_n = 0) = 0$ implies that:
- An unlabeled data point never comes from the class $y_n = 0$.
  - This is equivalent to a hard assumption that no data comes from the region around the decision boundary.
  - The labeled data only comes from the classes $y_n = 1$ and $y_n = -1$, so we never obtain any evidence for data with $y_n = 0$. We therefore refer to this category as the *null category* and the overall model as a *null category noise model* (NCNM).

Null Category Noise Model

Null Category

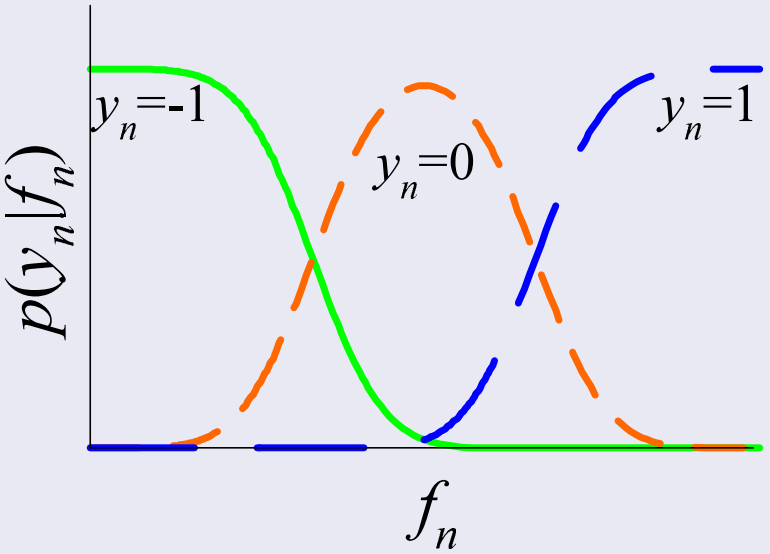


Figure: The null category noise model (semi-supervised classification).Standard noise model for labelled points ($y_n = 0$ is never observed). y_n marginalised for unlabelled points.

Null Category Noise Model

Null Category

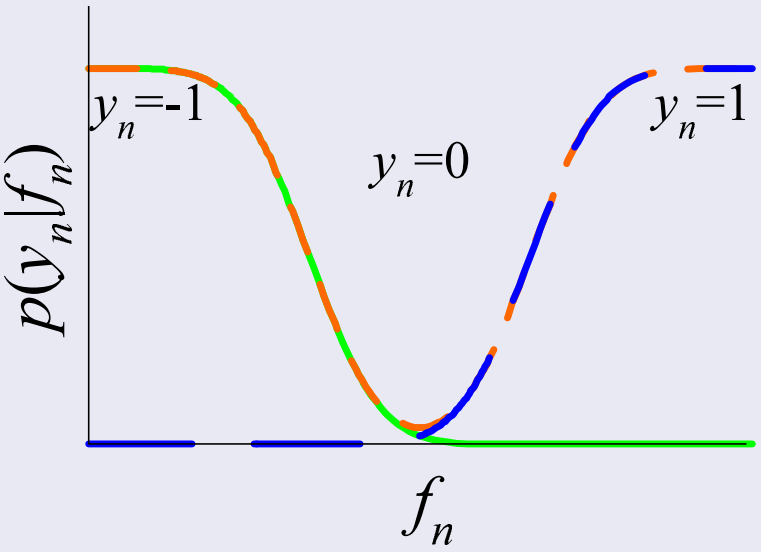


Figure: The null category noise model (semi-supervised classification). Effective noise model with y_n marginalised for unlabelled points.

Sparse Approximations

epPointUpdate('ncnm', NaN, -0.3, .1, 0, 1e-2)

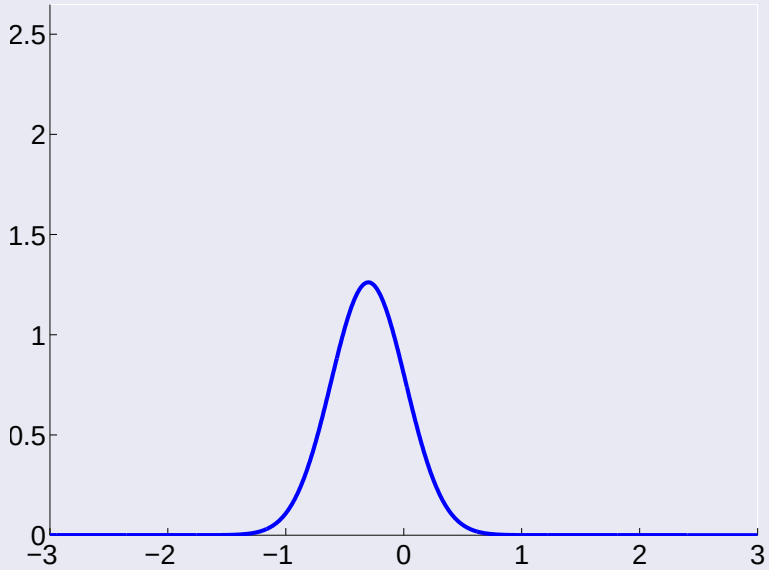


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$.

Sparse Approximations

epPointUpdate('ncnm', NaN, -0.3, .1, 0, 1e-2)

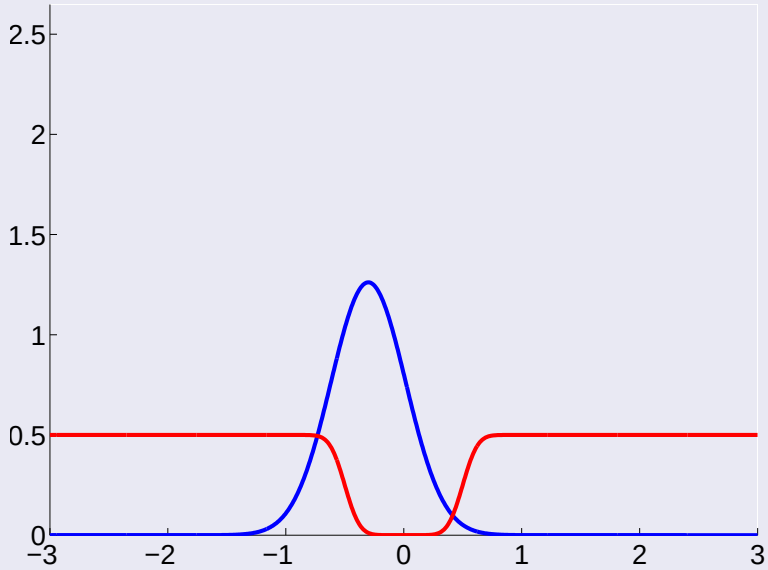


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* \neq 0|f_*)$.

Sparse Approximations

epPointUpdate('ncnm', NaN, -0.3, .1, 0, 1e-2)

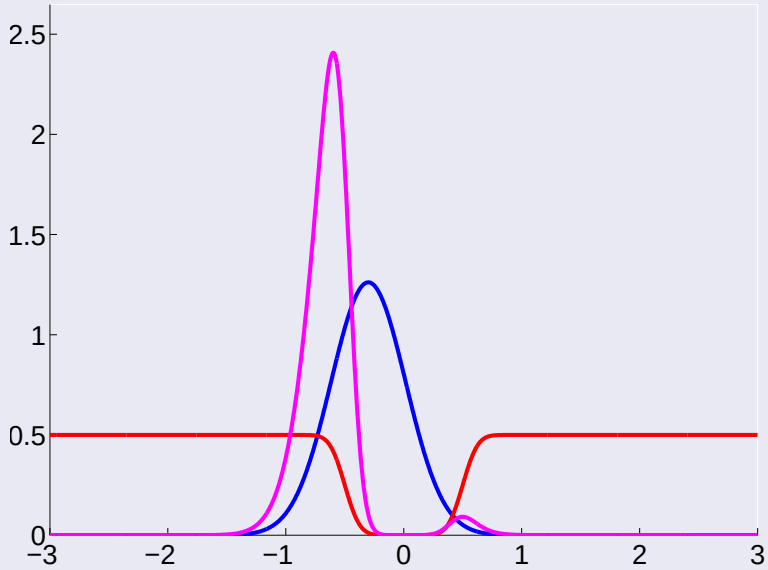


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* \neq 0|f_*)$, *Magenta:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, y_*)$.

Sparse Approximations

epPointUpdate('ncnm', NaN, -0.3, .1, 0, 1e-2)

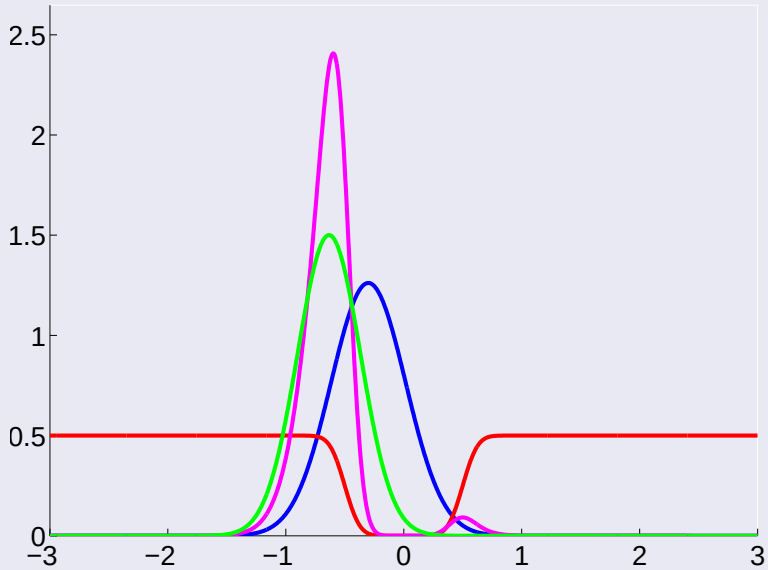


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* \neq 0|f_*)$, *Magenta:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, y_*)$, *Green:* $q(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$.

Sparse Approximations

epPointUpdate('ncnm', NaN, 0, .1, 0, 1e-2)

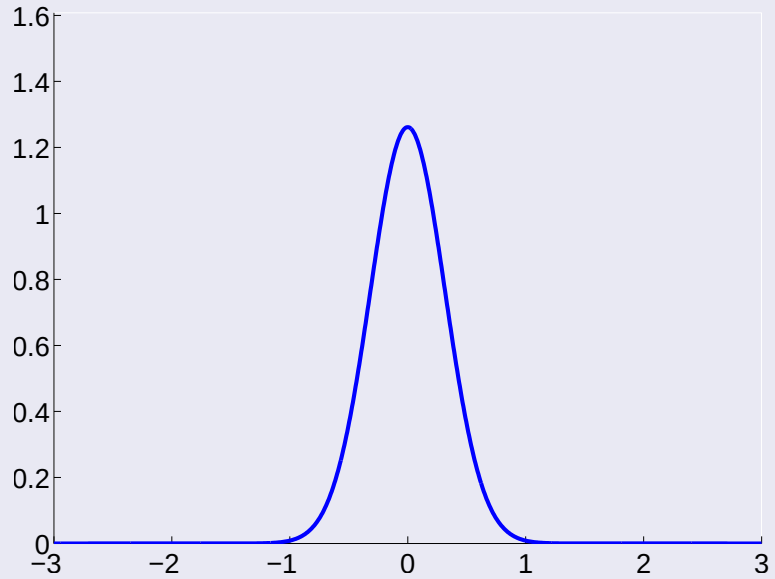


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$.

Sparse Approximations

epPointUpdate('ncnm', NaN, 0, .1, 0, 1e-2)

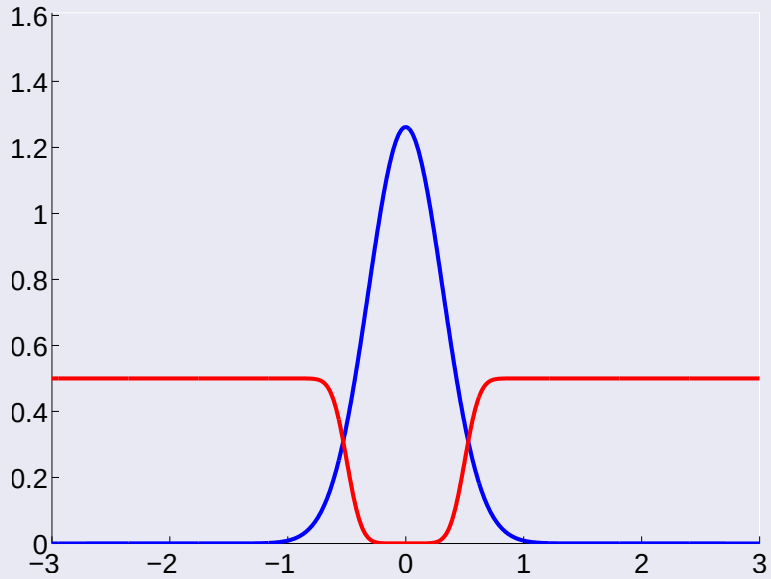


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* \neq 0|f_*)$.

Sparse Approximations

epPointUpdate('ncnm', NaN, 0, .1, 0, 1e-2)

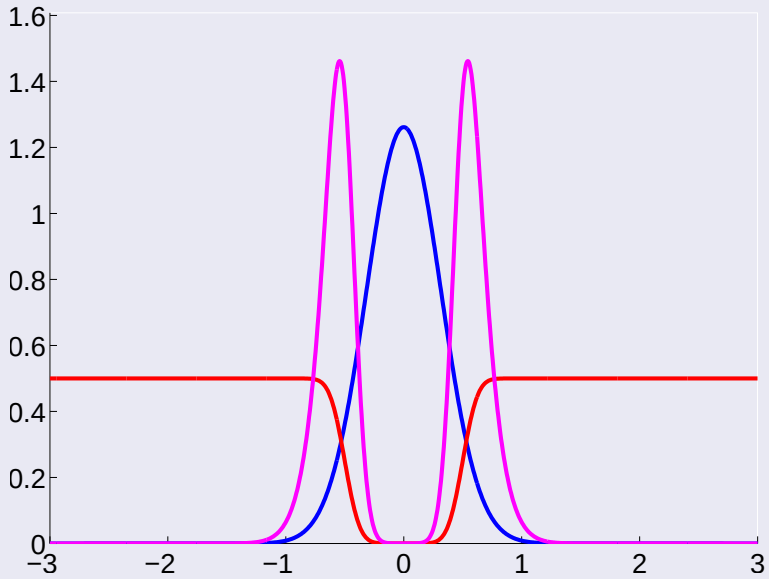


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* \neq 0|f_*)$, *Magenta:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, y_*)$.

Sparse Approximations

epPointUpdate('ncnm', NaN, 0, .1, 0, 1e-2)

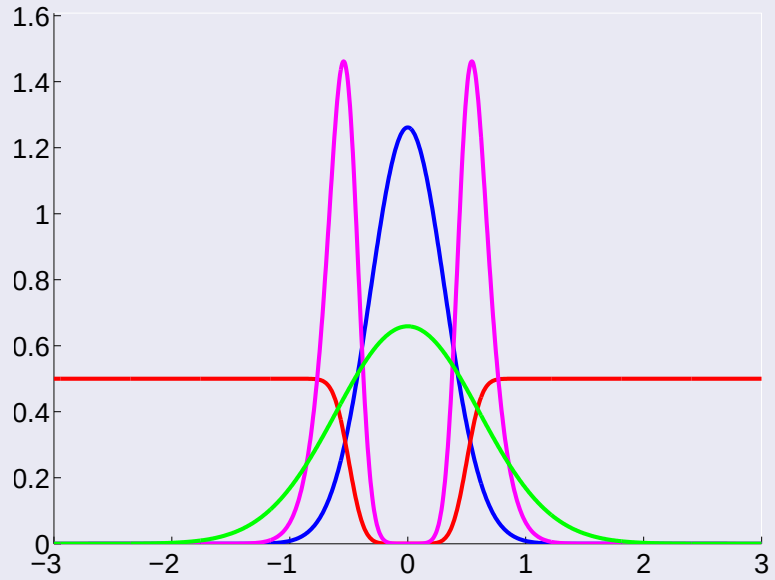


Figure: An EP style update with a classification noise model. *Blue:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$, *Red:* $p(y_* \neq 0|f_*)$, *Magenta:* $p(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y}, y_*)$, *Green:* $q(f_*|\mathbf{X}, \mathbf{x}_*, \mathbf{y})$.

The Null Category

Low Data Density at Decision Boundary

- When a data point is unlabeled the effect will depend on the mean and variance of $p(f_n|\mathbf{x}_n)$.
- If this Gaussian has little mass in the null category region, the posterior will be similar to the prior.
  - If the Gaussian has significant mass in the null category region, the outcome may be loosely described in two ways:
    1. If $p(f_n|\mathbf{x}_n)$ "spans the likelihood", leading to a bimodal posterior: the variance of the posterior will be greater than the variance of the prior.
    2. If $p(f_n|\mathbf{x}_n)$ is "rectified by the likelihood", then the mass of the posterior will be pushed in to one side of the null category.
- Note that the posterior is pushed out to one side or both sides of the null category region.

Toy Problem

Crescent Data

- We considered two-dimensional data in which two class-conditional densities interlock.
- There were 400 points in the original data set. Each point was labeled with probability 0.1, leading to 37 labeled points.
- A standard IVM classifier was trained on the labeled data only.
- We then used the null category approach to train a classifier that incorporates the unlabeled data.
- The resulting decision boundary finds a region of low data density and more accurately reflects the underlying data distribution.

Crescent Data

Standard IVM vs Semi-supervised

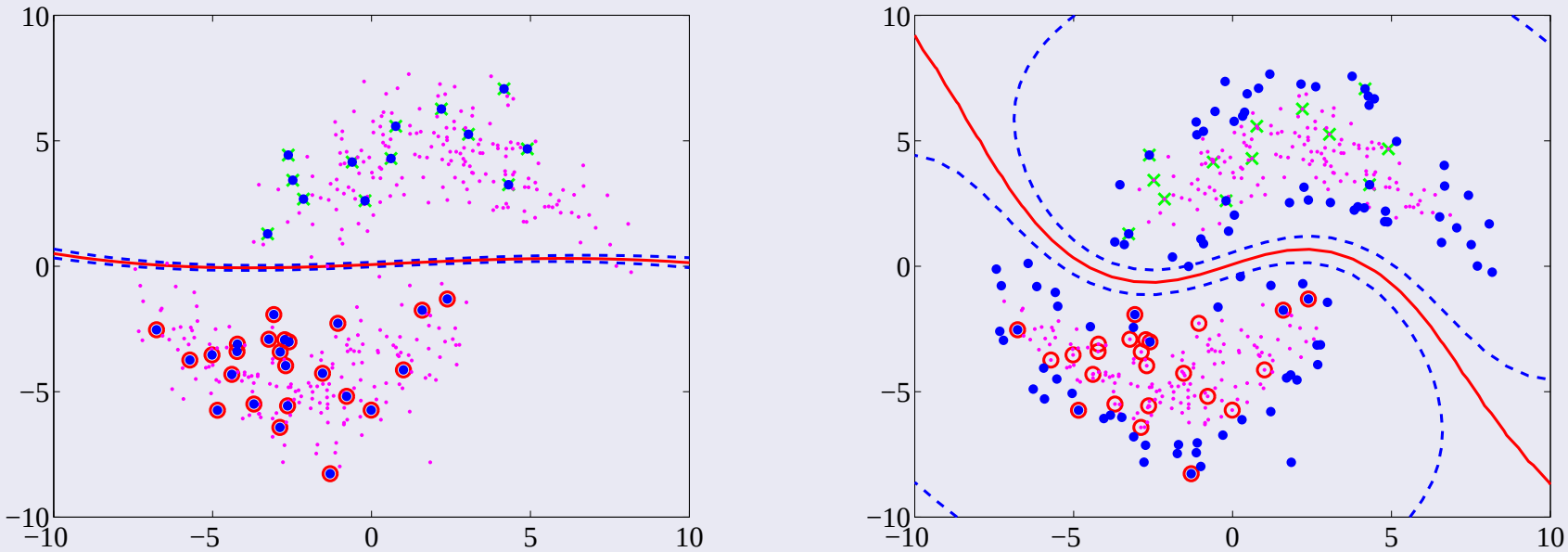


Figure: Data points: small blue dots, are labeled with probability 0.1. Labelled data-points: red circles and green crosses. Active set: large blue dots. *Left*: Learning with standard IVM. *Right*: Learning with the NCNM. Lines show centre and edge of null category.

High-dimensional example

USPS Data 3 vs 5

- As a higher dimensional example we return to the USPS data set.
- Separate the digit 3 from 5: vary probability of unlabelled data between 0.2 and 1.25×10^{-2}.
- Compare four classifiers:
  - standard IVM
  - standard SVM
  - semi-supervised IVM,
  - transductive SVM.
- Each run was completed ten times with different random seeds.

USPS Data

AUC Results

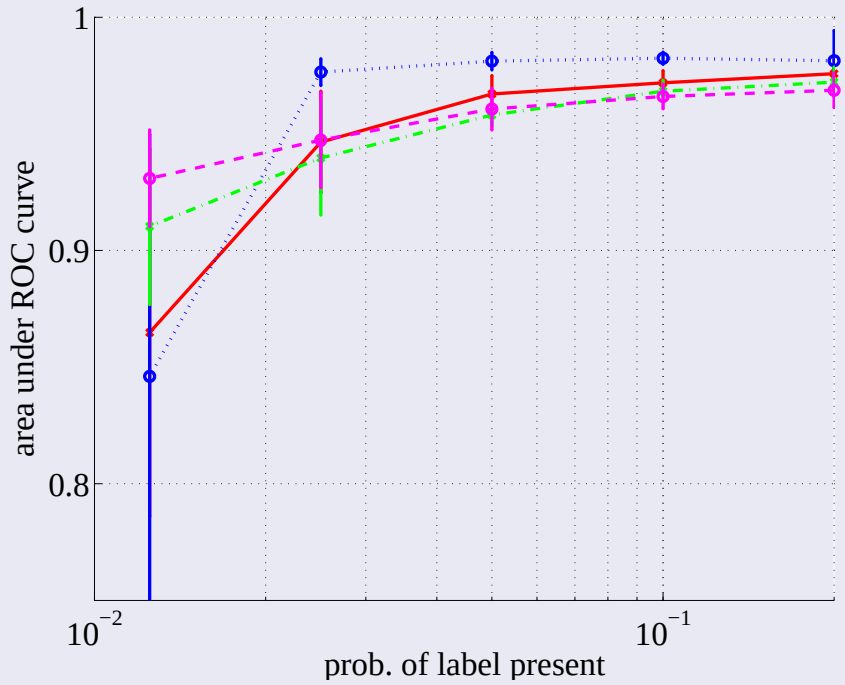


Figure: Mean and standard errors shown. IVM (red solid line), the NCNM (blue dotted line), the SVM (green dash-dot line) and the transductive SVM (pink dashed line).

USPS Data

Digits Results

- Below a label probability of 2.5×10^{-2} both the SVM and transductive SVM outperform the NCNM.
- In this region the estimate θ_1 provided by the NCNM was sometimes very low leading to occasional very poor results (note the large error bar).
- Above 2.5×10^{-2} a clear improvement is obtained for the NCNM over the other models.

Multi-task Learning

Multiple Independent Tasks

- We extend the IVM to handle multiple independent tasks.
- Given M training sets each with input matrix $\mathbf{X}_m$.
- Model the target data for each task, $\mathbf{y}_m$, as a GP

$$p\left(\mathbf{Y}|\mathbf{X},\boldsymbol{\theta}\right) = \prod_{m=1}^{M} p\left(\mathbf{y}_m|\mathbf{X}_m,\boldsymbol{\theta}\right)$$

where each $p\left(\mathbf{y}_m|\mathbf{X}_m,\boldsymbol{\theta}\right)$ is a Gaussian process.

Multi-task Gaussian Process

Graph of a Multi-task GP

- The entire likelihood is a GP over a vector, $\mathbf{y} = \left[\mathbf{y}_1^{\mathrm{T}} \dots \mathbf{y}_M^{\mathrm{T}}\right]^{\mathrm{T}}$

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{K}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{K}_M \end{bmatrix}$$

- IVM: Point selection is now performed *across models*.

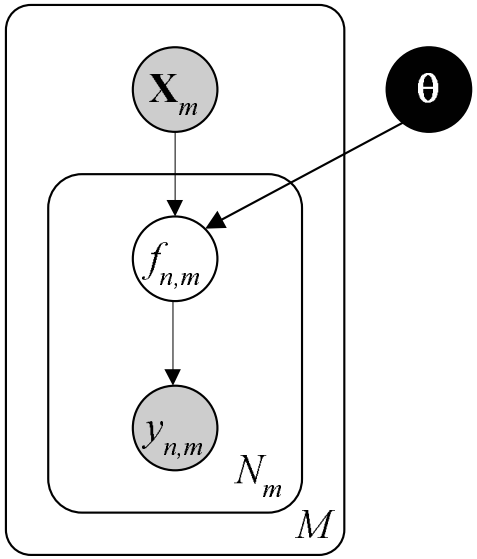


Figure: Plate notation: independence across the M tasks. [Lawrence and Platt, 2004]

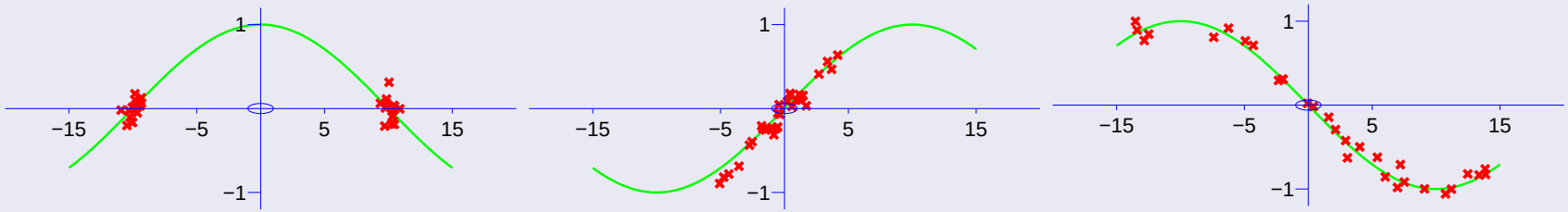
MT-IVM Simple Example

Regression Example

- Three tasks, each contains 30 data-points sampled from sine waves.
- Each task uses a different distributions for the input data.
- Select points using an MT-IVM.

Example Results

demToySine

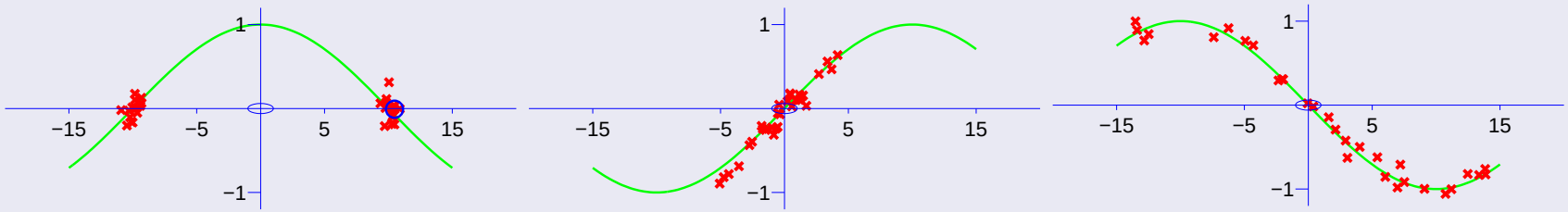


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

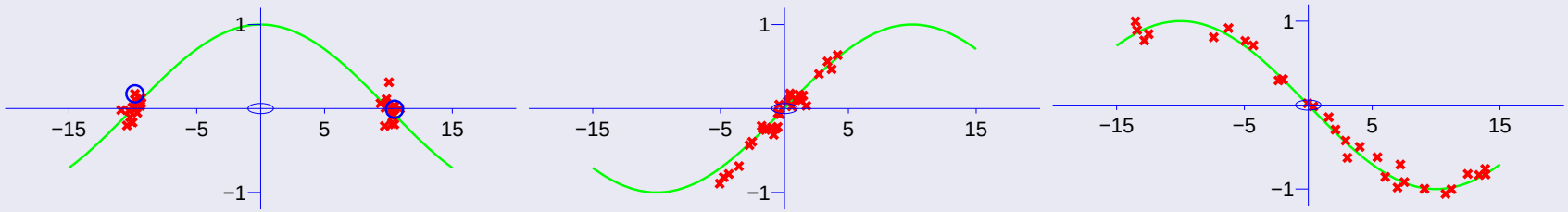


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

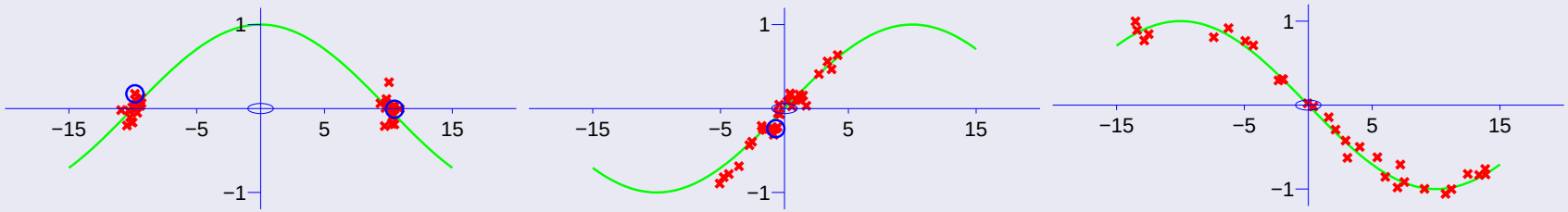


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

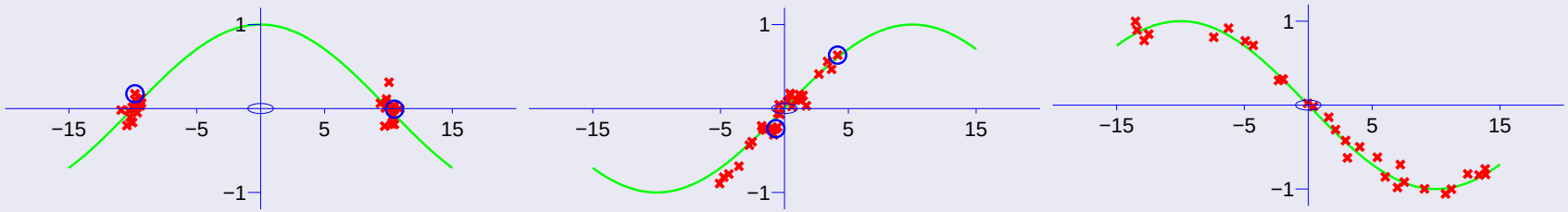


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

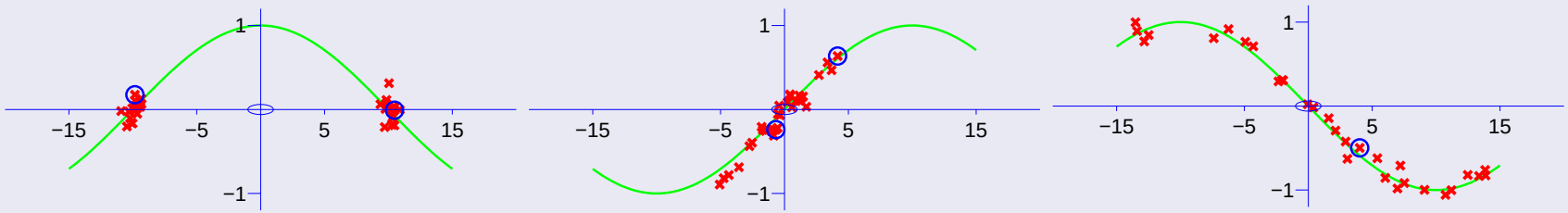


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

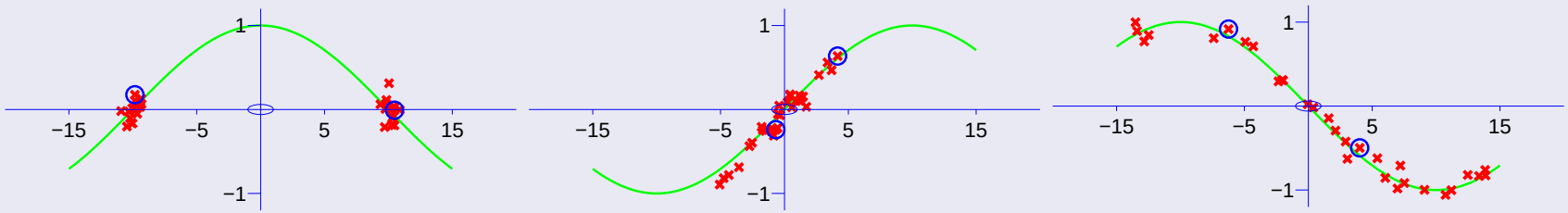


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

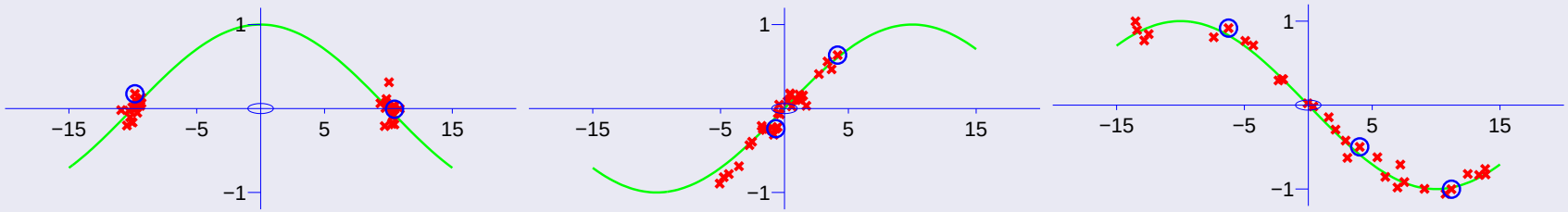


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

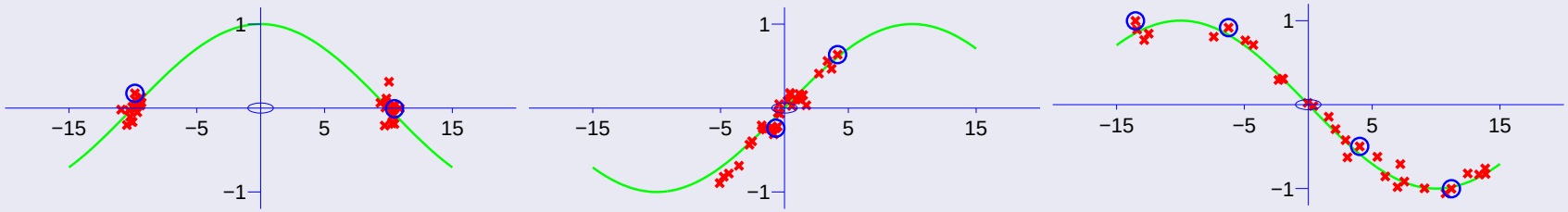


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

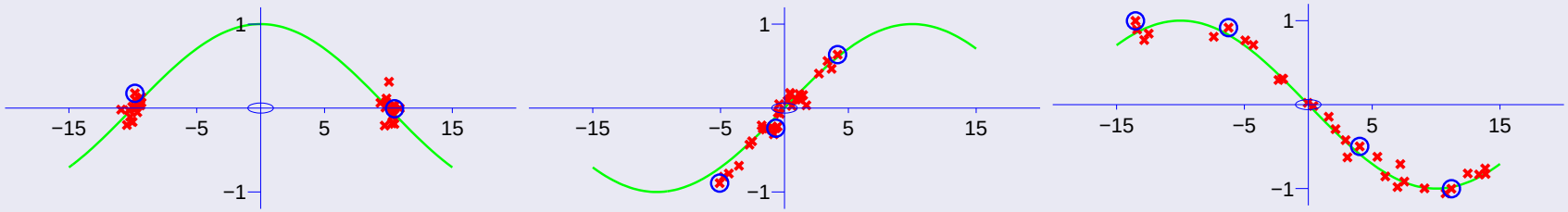


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

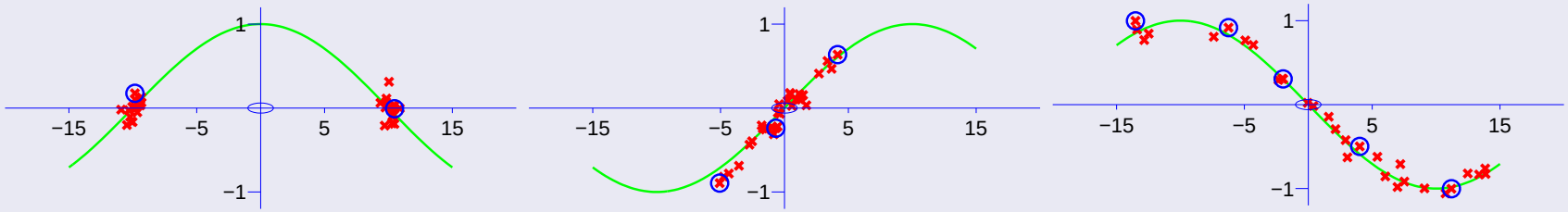


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

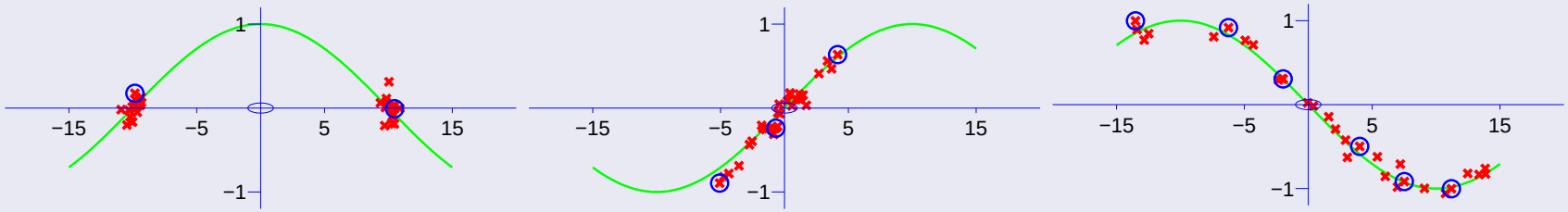


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

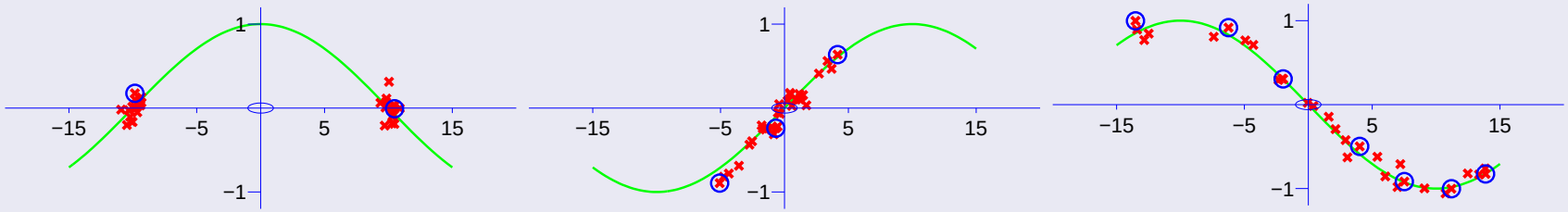


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

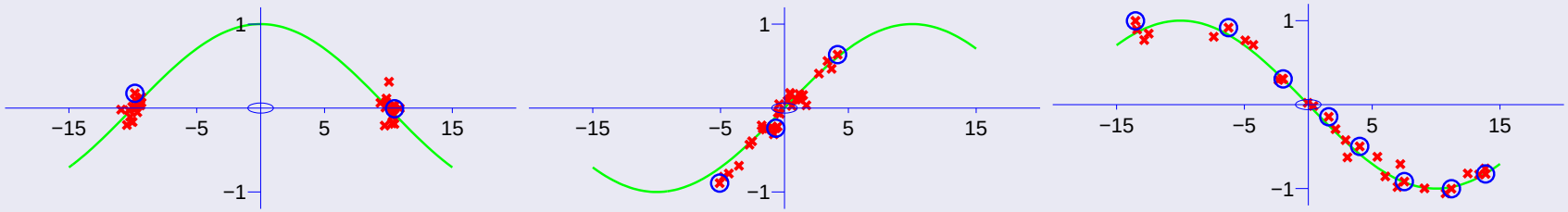


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

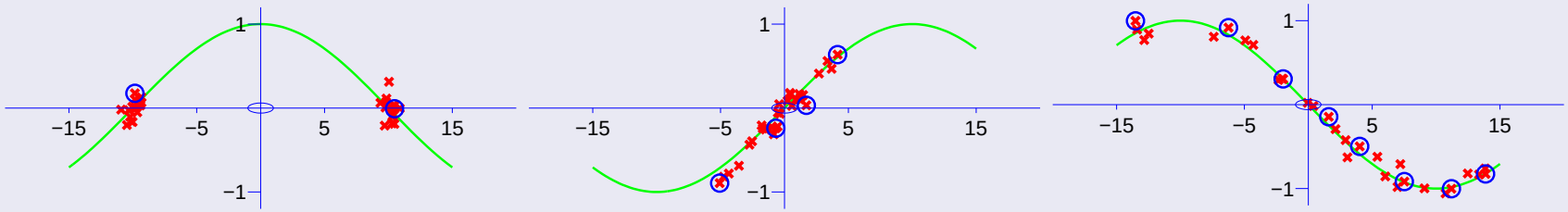


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine

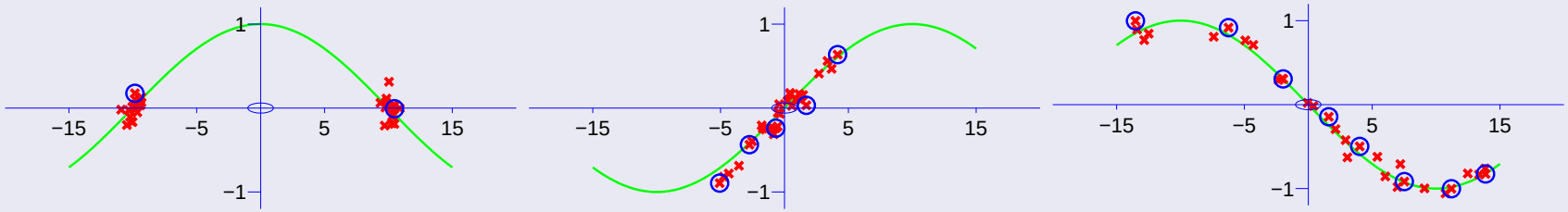


Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Example Results

demToySine



Task 1 Task 2 Task 3

Figure: Three different learning tasks sampled from sine waves. The input distribution for each task is different. Points used by the MT-IVM are circled. Note that more points are taken from tasks which give more information about the function.

Phoneme Classification Example

Classification

- The MT-IVM for phoneme recognition.
- Treat speakers as tasks, independent given $\boldsymbol{\theta}$.
  - Will MT-IVM converge faster than a speaker-independent IVM?

Phoneme Classification

Classification

- UCI repository phoneme example.
- 15 speakers, 11 phonemes: treat each speaker as a separate task.
- Use 14 speakers to learn kernel parameters.
- Evaluate model on remaining speaker.

Phoneme Classification Results

Time vs Error rate

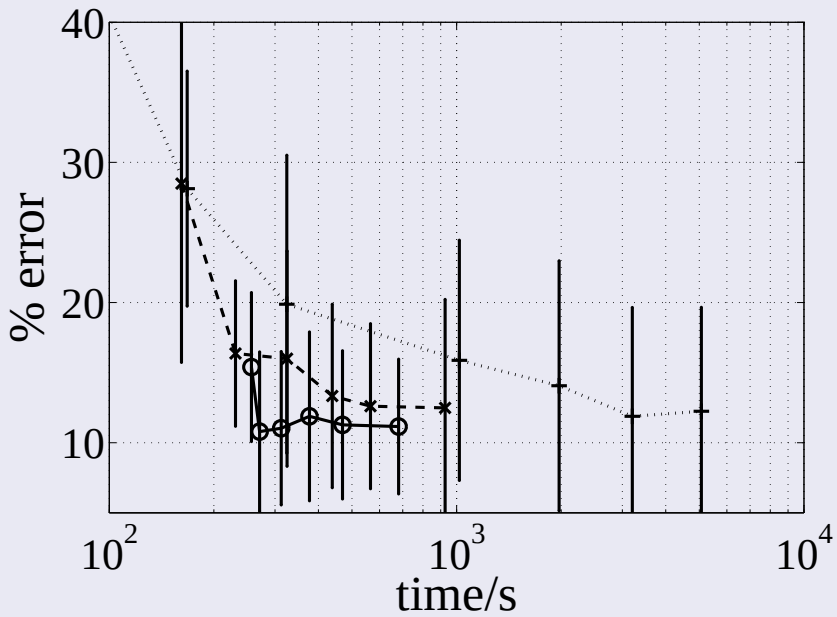


Figure: MT-IVM (solid line with circles) , sub-sampled ADF-GP (dashed line with crosses) — consider each speaker to be an independent task. Standard IVM — consider all points to belong to the same task (dotted line with pluses).

Classification Discussion

Faster Convergence

- MT-IVM reaches $\approx 10\%$ error roughly $10 \times$ faster than IVM.
- This independence structure:
  1. Speeds up training.
  2. Allows for speaker dependent recognisers.

Conclusions

Faster GPs through Sparsity

- We have reviewed GPs and EP briefly.
- We've introduced the IVM for sparsification.
- We've shown how we can:
  - learn invariances
  - do semi-supervised learning
  - do multi-task learning

References I

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3): 273–304, 1995.

Neil D. Lawrence and John C. Platt. Learning to learn with the informative vector machine. In Russell Greiner and Dale Schuurmans, editors, *Proceedings of the International Conference in Machine Learning*, volume 21, pages 512–519. Omnipress, 2004. doi: 10.1145/1015330.1015382.

Neil D. Lawrence, Matthias Seeger, and Ralf Herbrich. Fast sparse Gaussian process methods: The informative vector machine. In Sue Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15, pages 625–632, Cambridge, MA, 2003. MIT Press.

Neil D. Lawrence, John C. Platt, and Michael I. Jordan. Extensions of the informative vector machine. In Joab Winkler, Neil D. Lawrence, and Mahesan Niranjan, editors, *Deterministic and Statistical Methods in Machine Learning*, volume 3635 of *Lecture Notes in Artificial Intelligence*, pages 56–87. Springer-Verlag, Berlin, 2005. ISBN 3-540-29073-7.

Jeremey Oakley and Anthony O'Hagan. Bayesian inference for the uncertainty distribution of computer model outputs. *Biometrika*, 89(4):769–784, 2002.

Anthony O'Hagan. Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society, B*, 40: 1–42, 1978.

Anthony O'Hagan. Some Bayesian numerical analysis. In José M. Bernardo, James O. Berger, A. Phillip Dawid, and Adrian F. M. Smith, editors, *Bayesian Statistics 4*, pages 345–363, Valencia, 1992. Oxford University Press.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006. ISBN 026218253X.

Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, 2001.

Bernhard Schölkopf, Chris J. C. Burges, and Vladimir N. Vapnik. Incorporating invariances in support vector learning machines. In *Artificial Neural Networks — ICANN'96*, volume 1112, pages 47–52, 1996.

Matthias Seeger. *Bayesian Gaussian Process Models: PAC-Bayesian Generalisation Error Bounds and Sparse Approximations*. PhD thesis, The University of Edinburgh, 2004.

Christopher K. I. Williams. Computing with infinite networks. In Michael C. Mozer, Michael I. Jordan, and Thomas Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, Cambridge, MA, 1997. MIT Press.

Consistency

Consistency of a Gaussian Process

- Predictions remain the same regardless of the number and location of the test points.

$$p\left(\mathbf{f}_*|\mathbf{f}\right) = \int p\left(\mathbf{f}_*, \mathbf{f}_+|\mathbf{f}\right) d\mathbf{f}_+,$$

- For the system to be consistent this conditional probability must be independent of the length of $\mathbf{f}_+$.
- In other words.

$$p\left(\mathbf{f}_*|\mathbf{f}\right) = \int p\left(\mathbf{f}_*, \mathbf{f}_+|\mathbf{f}\right) d\mathbf{f}_+ = \int p\left(\mathbf{f}_*, \hat{\mathbf{f}}_+|\mathbf{f}\right) d\hat{\mathbf{f}}_+$$

Joint Distribution

Joint Distribution

- The covariance function provides the joint distribution over the instantiations.
- Write down the conditional distribution provides predictions.
- Denote the training set as $\mathbf{f}$ and test set as $\mathbf{f}_*$.
  - Predict using $p(\mathbf{f}_*|\mathbf{f})$.

The Conditional Distribution

Partioned Inverse

- Use partitioned inverse to find conditional.

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_{\mathbf{f},\mathbf{f}} & \mathbf{K}_{\mathbf{f},*} \\ \mathbf{K}_{*,\mathbf{f}} & \mathbf{K}_{*,*} \end{bmatrix}$$

- Partitioned inverse is then

$$\mathbf{K}^{-1} = \begin{bmatrix} \mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1} + \mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{K}_{\mathbf{f},*}\Sigma^{-1}\mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1} & -\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{K}_{\mathbf{f},*}\Sigma^{-1} \\ -\Sigma^{-1}\mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1} & \mathbf{-1} \end{bmatrix}$$

where

$$\Sigma = \mathbf{K}_{*,*} - \mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{K}_{\mathbf{f},*}.$$

Joint Distribution

Take Log of the Joint

- Logarithm of the joint distribution:

$$\begin{aligned}\log p(\mathbf{f}, \mathbf{f}_*) &= -\frac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{f} - \frac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{K}_{\mathbf{f},*}\Sigma^{-1}\mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{f} \\ &\quad + \mathbf{f}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{K}_{\mathbf{f},*}\Sigma^{-1}\mathbf{f}_* - \frac{1}{2}\mathbf{f}_*^{\mathrm{T}}\Sigma^{-1}\mathbf{f}_* + \mathrm{const}_1\end{aligned}$$

- Conditional is found by dividing joint by the prior, $p(\mathbf{f}) = N(\mathbf{f}|\mathbf{0}, \mathbf{K}_{\mathbf{f},\mathbf{f}})$.

Conditional Distribution

Deriving the Conditional

- In log space this is equivalent to subtraction of

$$\log p(\mathbf{f}) = -\frac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{f} + \text{const}_2$$

giving

$$\log p(\mathbf{f}_*|\mathbf{f}) = \log p(\mathbf{f}_*, \mathbf{f}) - \log p(\mathbf{f}) = \log N(\mathbf{f}_*|\bar{\mathbf{f}}_*, \Sigma).$$

where $\bar{\mathbf{f}} = \mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{f}$ and $\Sigma = \mathbf{K}_{*,*} - \mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{K}_{\mathbf{f},*}$.

Making Predictions

- If we observe points from the function, $\mathbf{f}$.
- We can predict the locations of functions at as yet unseen locations.
- The prediction is also a Gaussian process, with mean $\bar{\mathbf{f}}$ and covariance Σ.
- Often observe corrupted version of function.