Multiple Output Processes

Neil D. Lawrence

GPRS
25th–27th February 2015

Outline

Outline

Multiple Output Gaussian Processes

- In this section we will study Gaussian processes with multiple outputs.
- they have various names, vector valued functions, multiple outputs, multidimensional GPs, multi-task learning.
- Key idea, we want to relate several different functions.
- Sounds more complex, but actually it's a special case of a normal GP where one input is discrete.
- Question: how to embed covariation between the functions.
- Start by introducing *Kalman filter/smoother*.

Simple Markov Chain

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

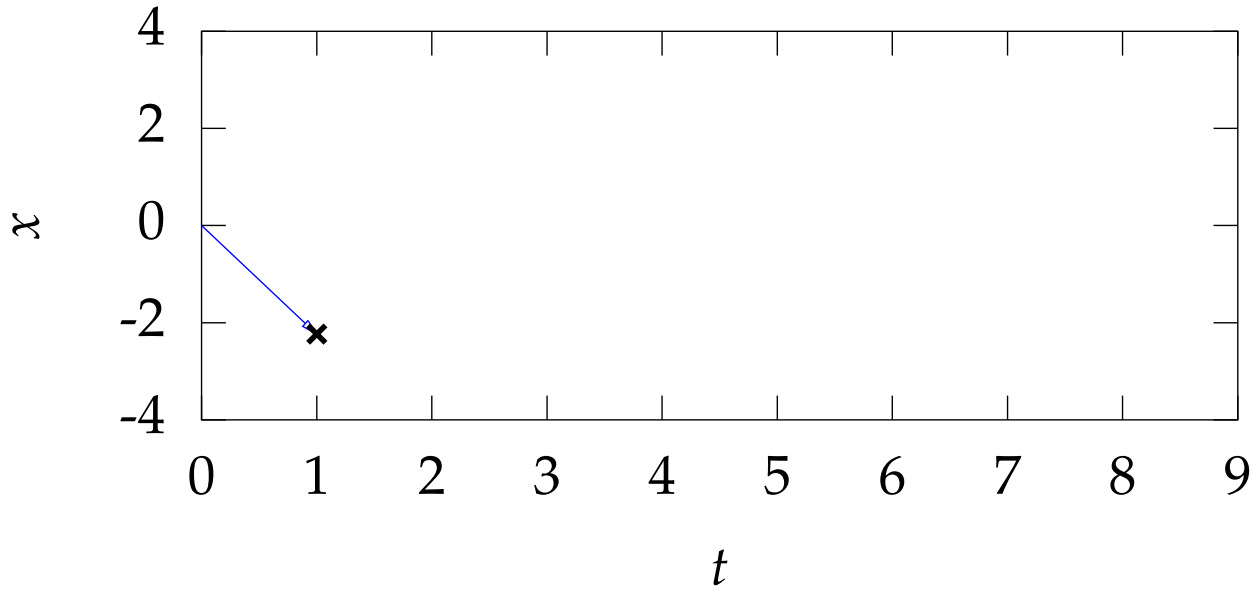
$$\begin{aligned} x_i &= x_{i-1} + \epsilon_i, \\ \epsilon_i &\sim \mathcal{N}(0, \alpha) \\ \implies x_i &\sim \mathcal{N}(x_{i-1}, \alpha) \end{aligned}$$

- Initial state,

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

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

Gauss Markov Chain

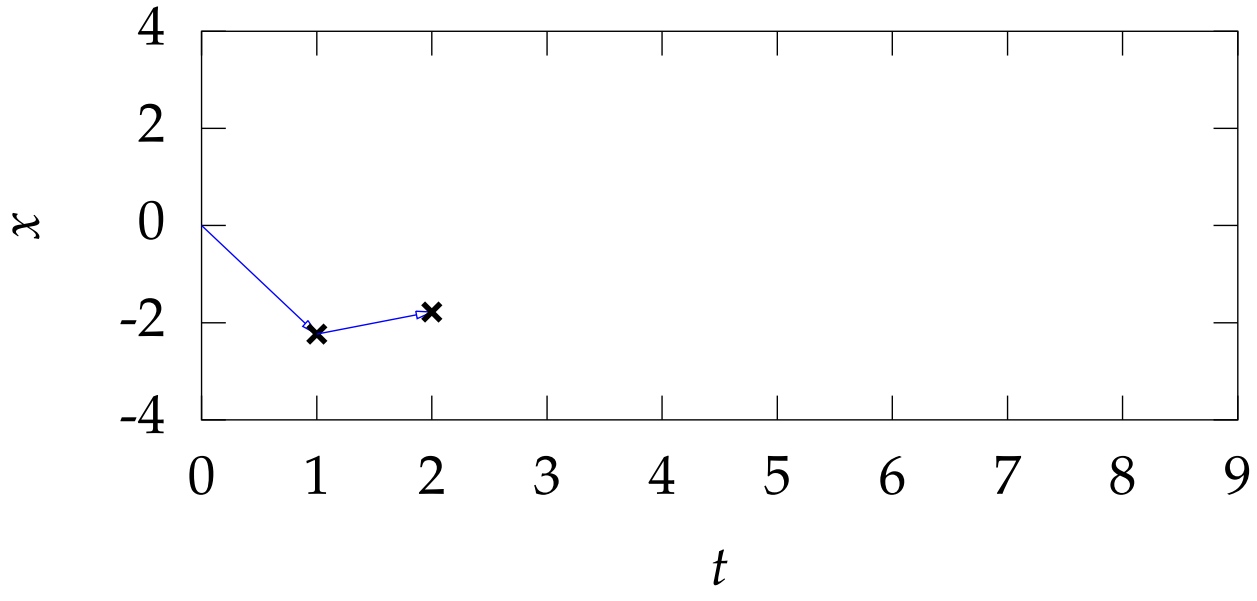


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0,1)$$

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

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

Gauss Markov Chain

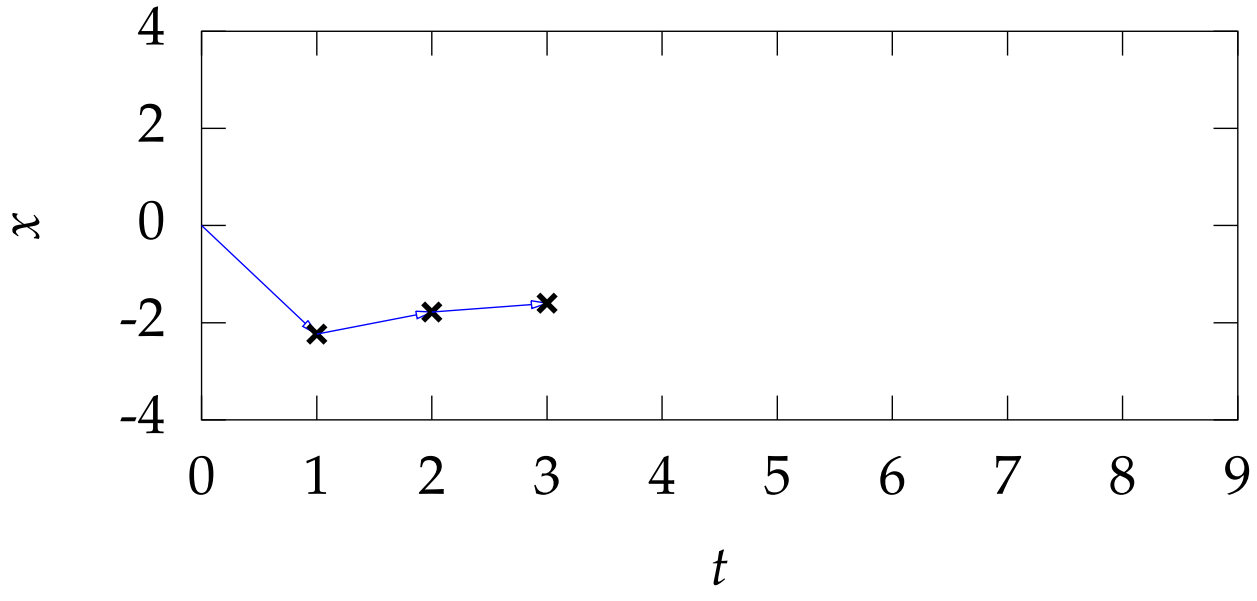


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0,1)$$

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

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

Gauss Markov Chain

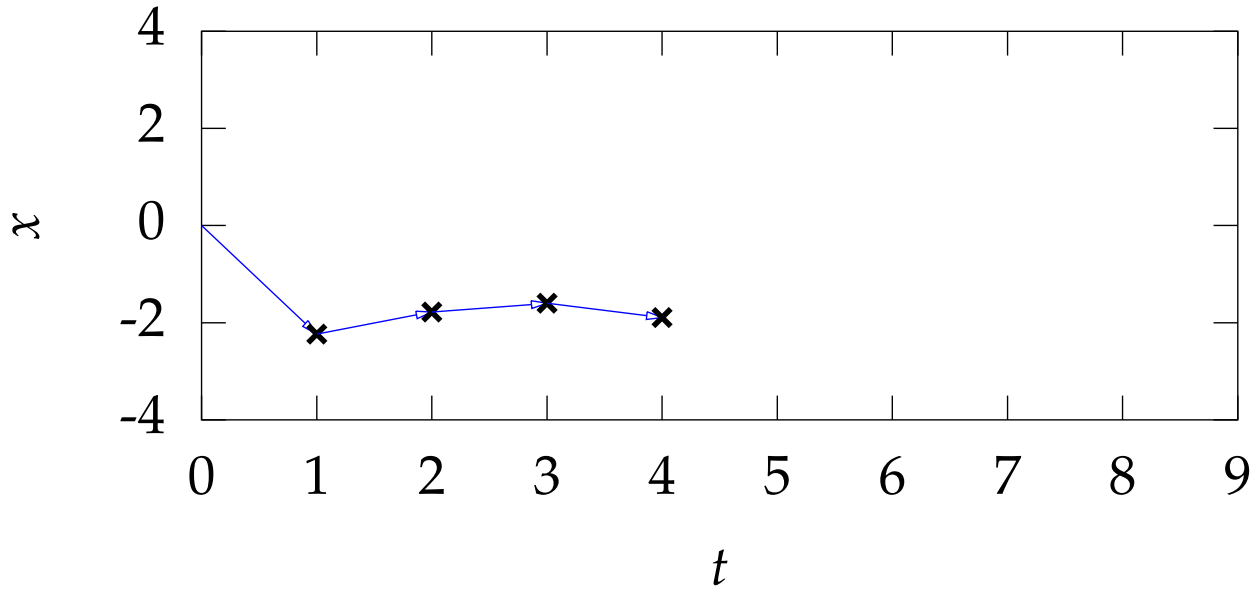


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0,1)$$

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

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

Gauss Markov Chain

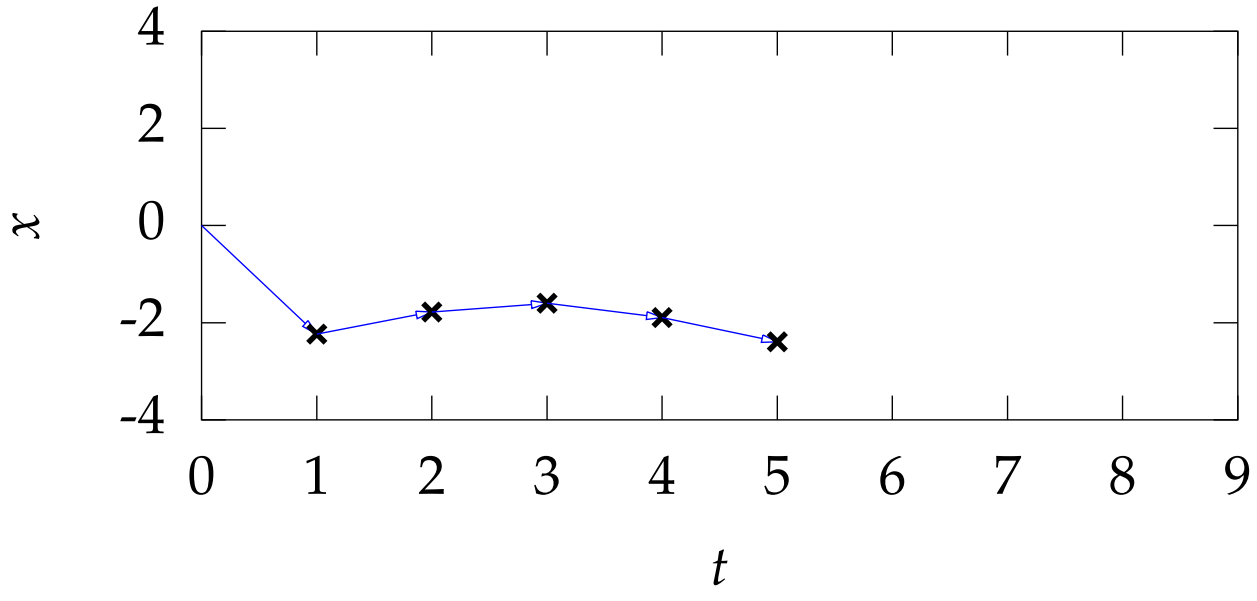


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0,1)$$

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

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

Gauss Markov Chain

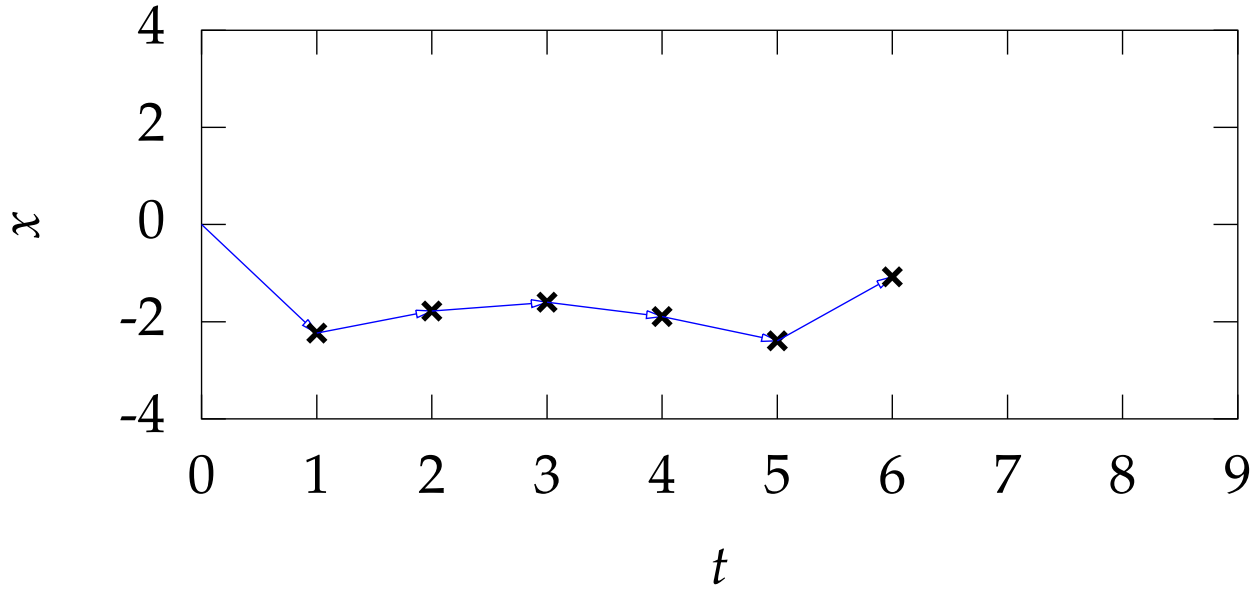


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0,1)$$

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

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

Gauss Markov Chain

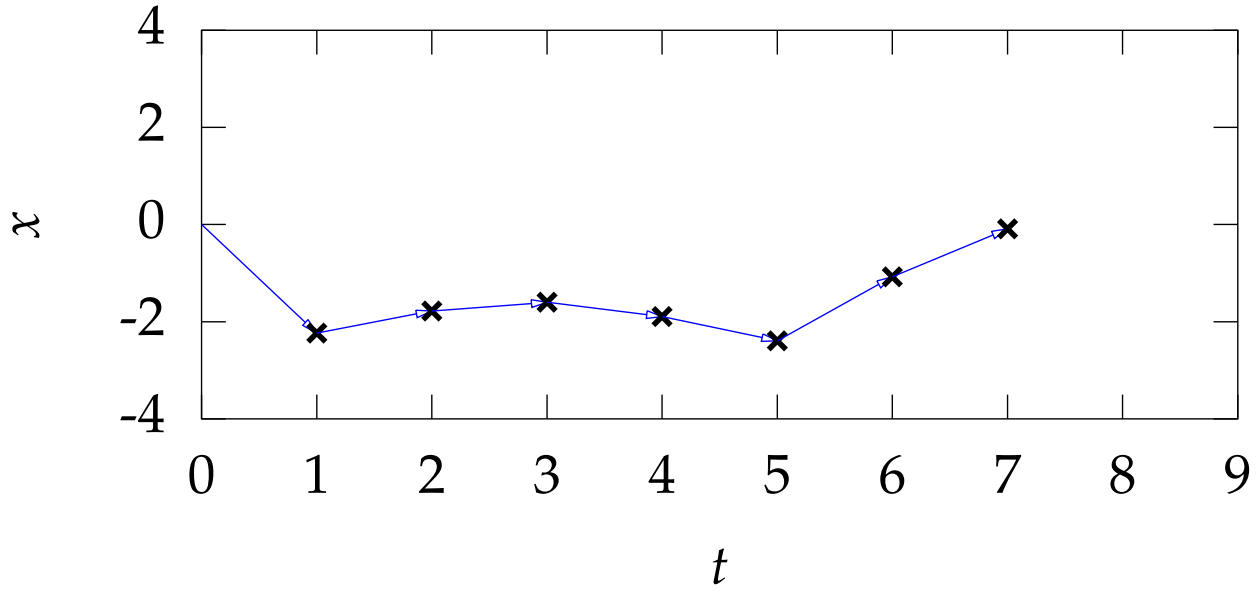


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0,1)$$

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

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

Gauss Markov Chain

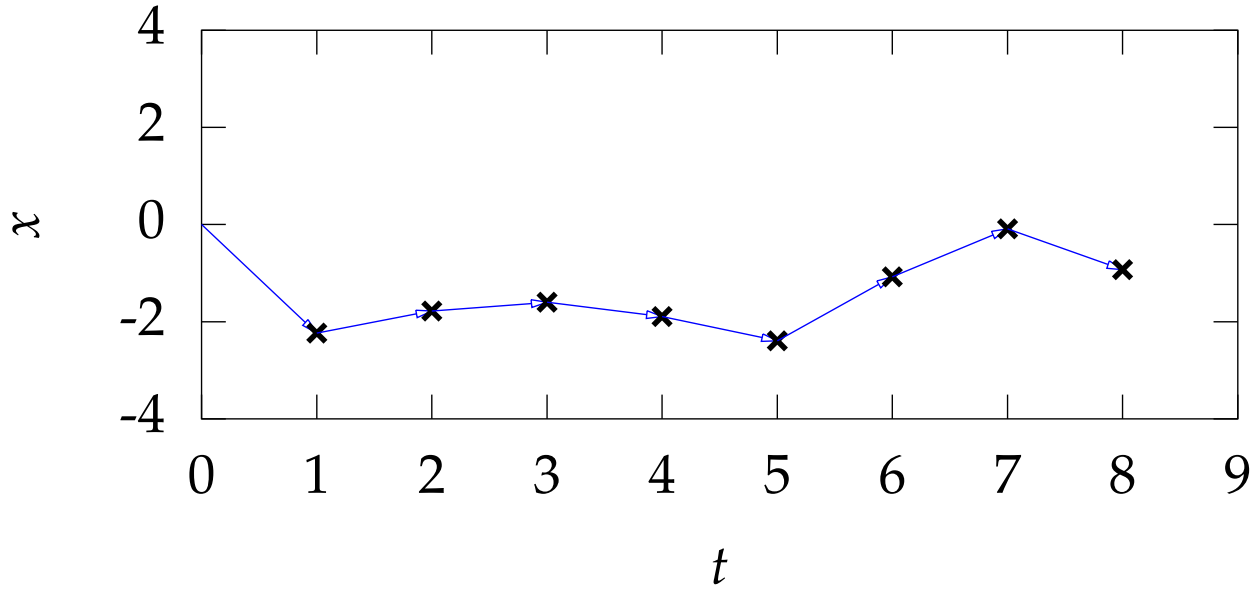


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0,1)$$

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

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

Gauss Markov Chain

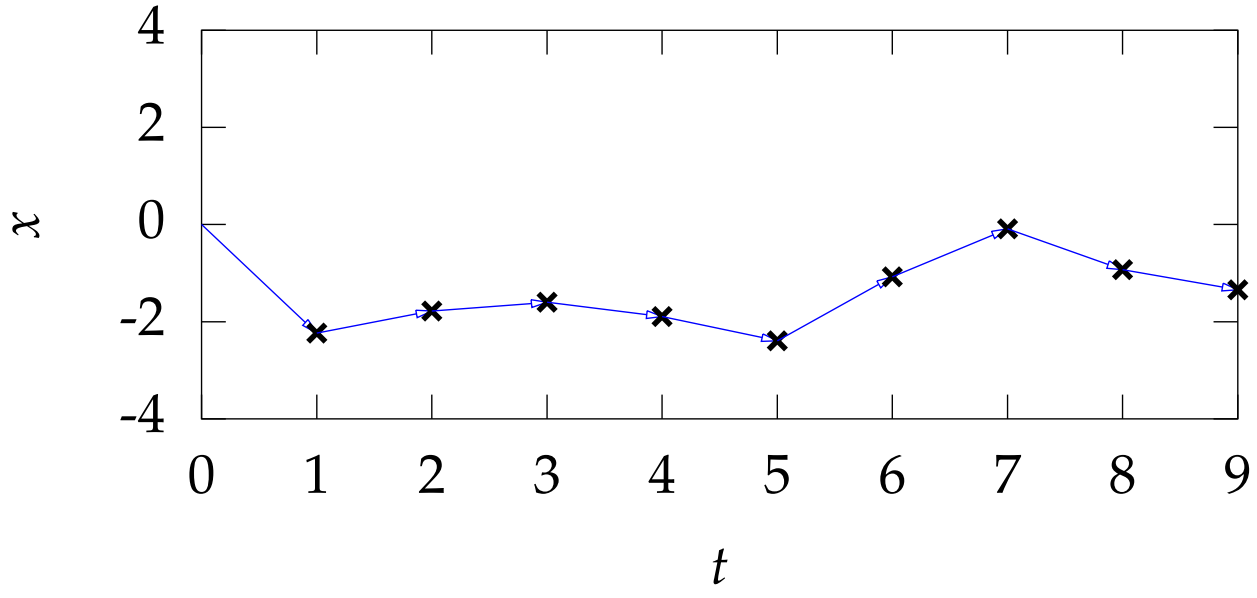


$$x_0 = 0, \quad \epsilon_i \sim \mathcal{N}(0,1)$$

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

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

Gauss Markov Chain



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

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

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

Multivariate Gaussian Properties: Reminder

If

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

and

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

then

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

Multivariate Gaussian Properties: Reminder

Simplified: If

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

and

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

then

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

Matrix Representation of Latent Variables

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

$$x_1 = \epsilon_1$$

Matrix Representation of Latent Variables

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

$$x_2 = \epsilon_1 + \epsilon_2$$

Matrix Representation of Latent Variables

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

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

Matrix Representation of Latent Variables

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

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

Matrix Representation of Latent Variables

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

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

Matrix Representation of Latent Variables

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

Multivariate Process

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

Latent Process Mean

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

Latent Process Mean

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

Latent Process Mean

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

Latent Process Mean

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

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

Latent Process Mean

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

Latent Process Mean

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

Latent Process Covariance

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

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

Latent Process Covariance

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

Latent Process Covariance

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

Latent Process Covariance

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

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

Latent Process Covariance

$$\left\langle \mathbf{x}\mathbf{x}^{\top} \right\rangle = \alpha \mathbf{L}_1 \mathbf{L}_1^{\top}$$

Latent Process

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

Latent Process

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

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

Latent Process

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

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

$$\Longrightarrow$$

Latent Process

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

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

$$\Longrightarrow$$

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

Covariance for Latent Process II

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

Covariance for Latent Process II

- Given
$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \alpha\mathbf{I}) \Longrightarrow \boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \alpha\mathbf{L_1}\mathbf{L_1}^\top\right).$$
Then
$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \Delta t\alpha\mathbf{I}) \Longrightarrow \boldsymbol{\epsilon} \sim \mathcal{N}\left(\mathbf{0}, \Delta t\alpha\mathbf{L_1}\mathbf{L_1}^\top\right).$$
where Δt is the time interval between observations.

Covariance for Latent Process II

$$\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \alpha \Delta t \mathbf{I}\right), \quad \mathbf{x} \sim \mathcal{N}\left(0, \alpha \Delta t \mathbf{L_1} \mathbf{L_1}^\top\right)$$

Covariance for Latent Process II

$$\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \alpha \Delta t \mathbf{I}\right), \quad \mathbf{x} \sim \mathcal{N}\left(0, \alpha \Delta t \mathbf{L_1} \mathbf{L_1}^\top\right)$$

$$\mathbf{K} = \alpha \Delta t \mathbf{L_1} \mathbf{L_1}^\top$$

Covariance for Latent Process II

$$\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \alpha\Delta t \mathbf{I}\right), \quad \mathbf{x} \sim \mathcal{N}\left(0, \alpha\Delta t \mathbf{L_1}\mathbf{L_1}^\top\right)$$

$$\mathbf{K} = \alpha\Delta t \mathbf{L_1}\mathbf{L_1}^\top$$

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

where $\mathbf{l}_{:,k}$ is a vector from the kth row of $\mathbf{L_1}$: the first k elements are one, the next $T - k$ are zero.

Covariance for Latent Process II

$$\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \alpha\Delta t \mathbf{I}\right), \quad \mathbf{x} \sim \mathcal{N}\left(0, \alpha\Delta t \mathbf{L_1}\mathbf{L_1}^\top\right)$$

$$\mathbf{K} = \alpha\Delta t \mathbf{L_1}\mathbf{L_1}^\top$$

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

where $\mathbf{l}_{:,k}$ is a vector from the kth row of $\mathbf{L_1}$: the first k elements are one, the next $T - k$ are zero.

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

define $\Delta t i = t_i$ so

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

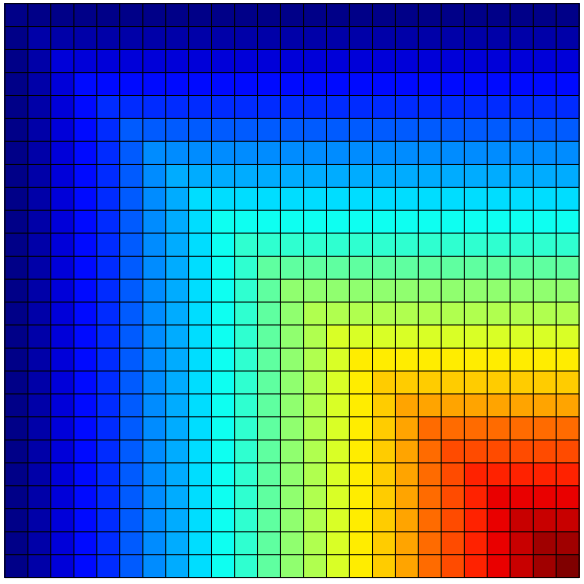
Covariance Functions

Where did this covariance matrix come from?

Markov Process

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

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



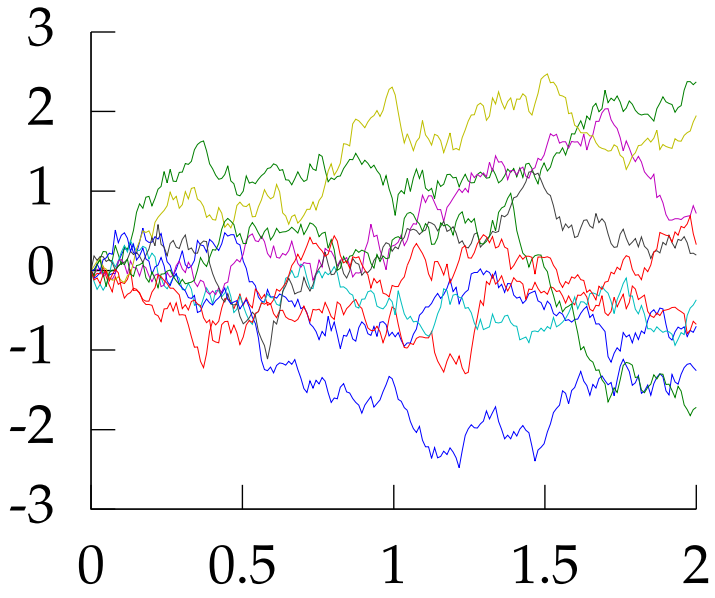
Covariance Functions

Where did this covariance matrix come from?

Markov Process

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

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



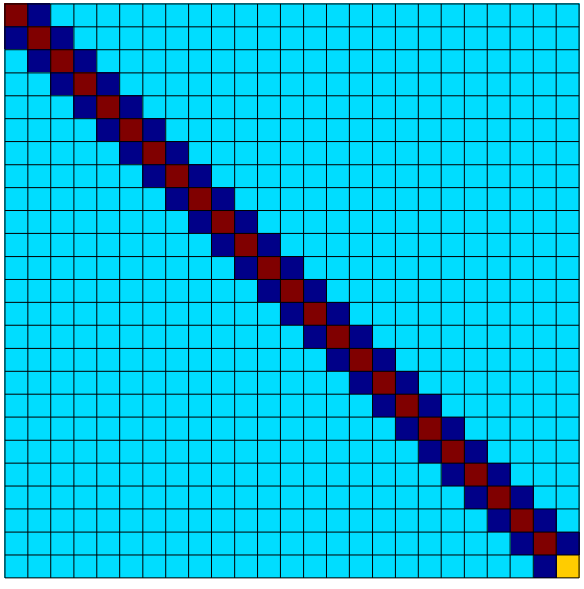
Covariance Functions

Where did this covariance matrix come from?

Markov Process

Visualization of inverse covariance (precision).

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



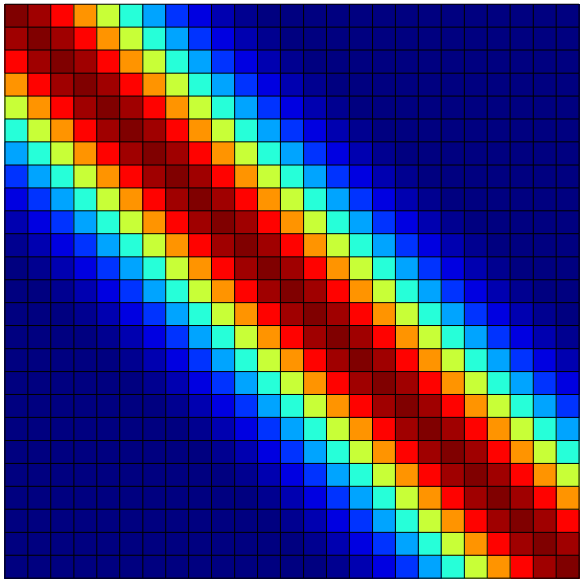
Covariance Functions

Where did this covariance matrix come from?

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

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

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



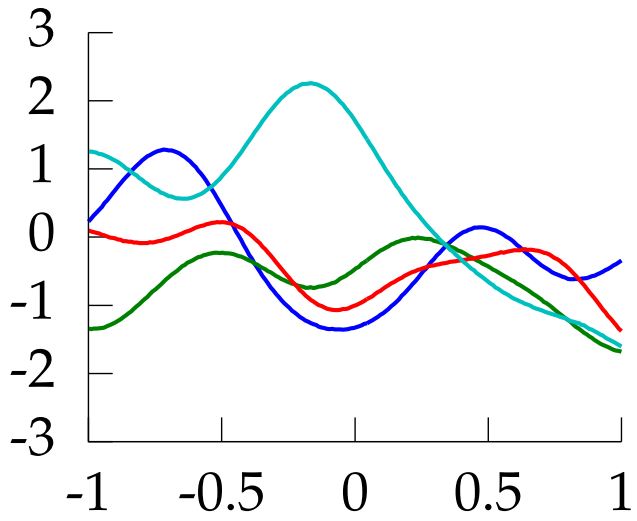
Covariance Functions

Where did this covariance matrix come from?

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

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

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



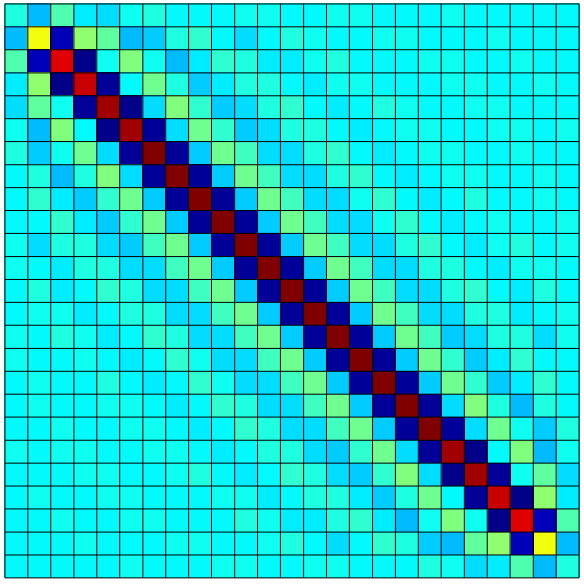
Covariance Functions

Where did this covariance matrix come from?

Exponentiated Quadratic

Visualization of inverse covariance (precision).

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



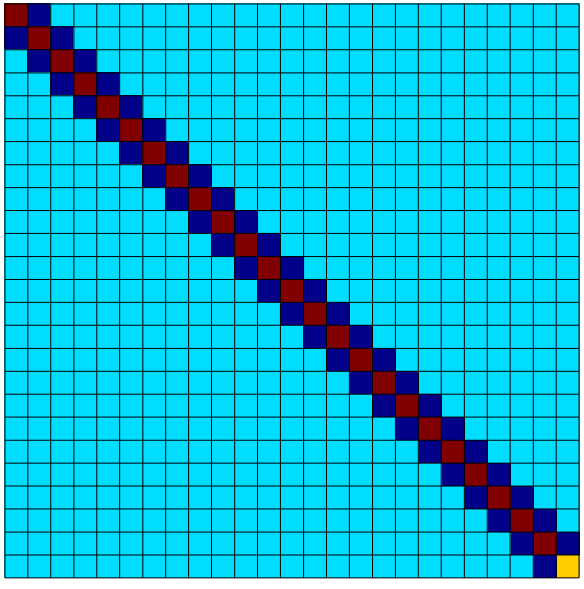
Covariance Functions

Where did this covariance matrix come from?

Markov Process

Visualization of inverse covariance (precision).

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



Simple Kalman Filter I

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

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

- We want to obtain outputs through:

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

Stacking and Kronecker Products I

- Represent with a 'stacked' system:

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

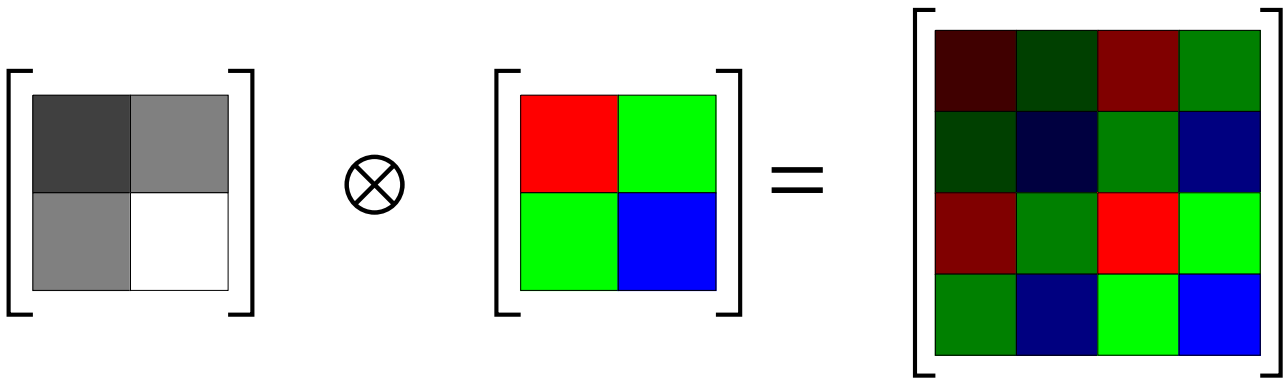
where the stacking is placing each column of $\mathbf{X}$ one on top of another as

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

Kronecker Product

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

Kronecker Product



Stacking and Kronecker Products I

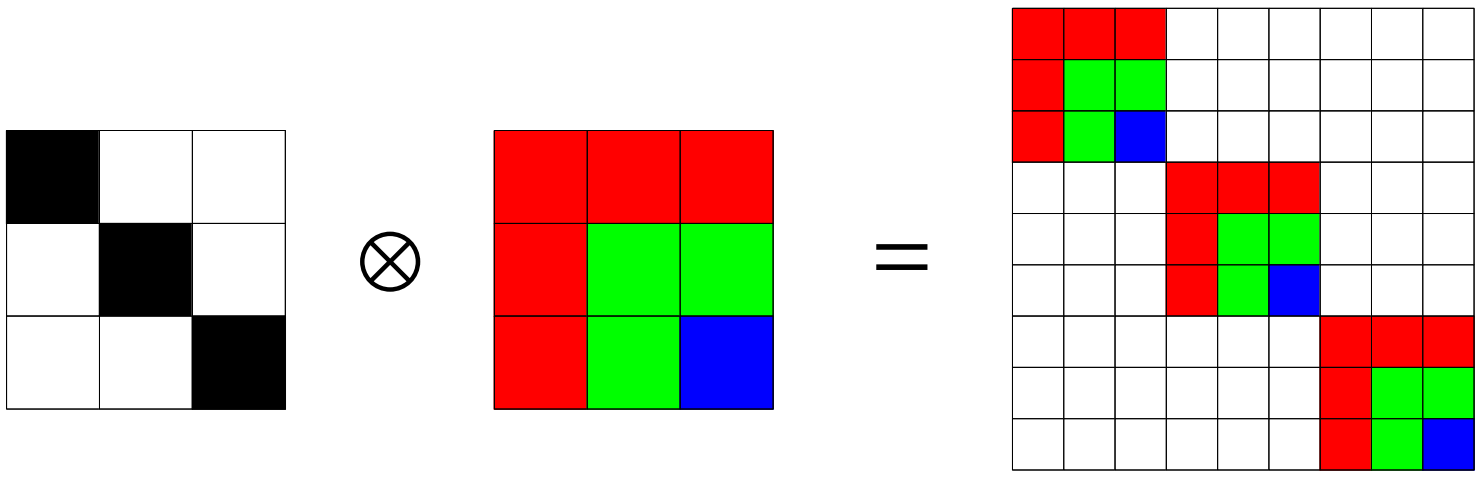
- Represent with a 'stacked' system:

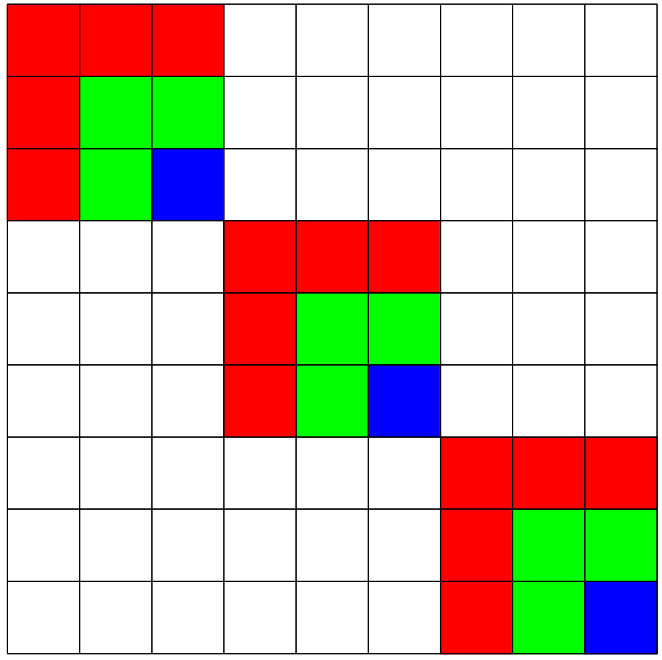
$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{I} \otimes \mathbf{K})$$

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

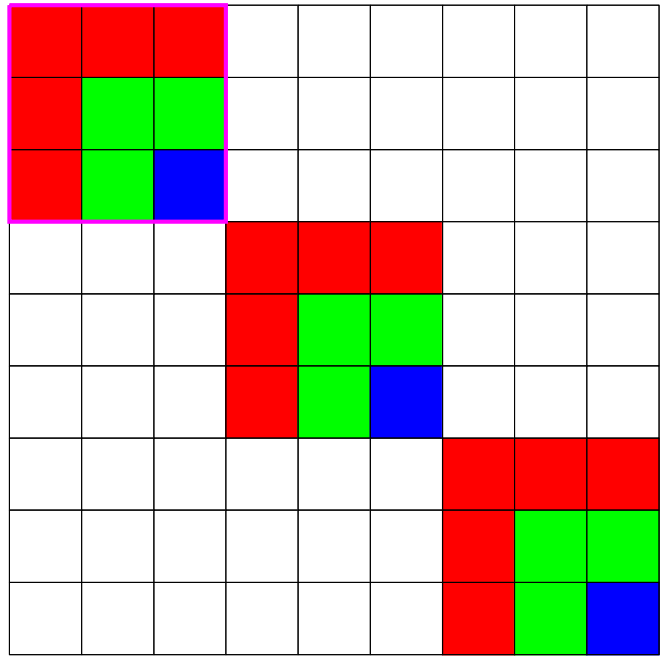
$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{:,1} \\ \mathbf{x}_{:,2} \\ \vdots \\ \mathbf{x}_{:,q} \end{bmatrix}$$

Column Stacking

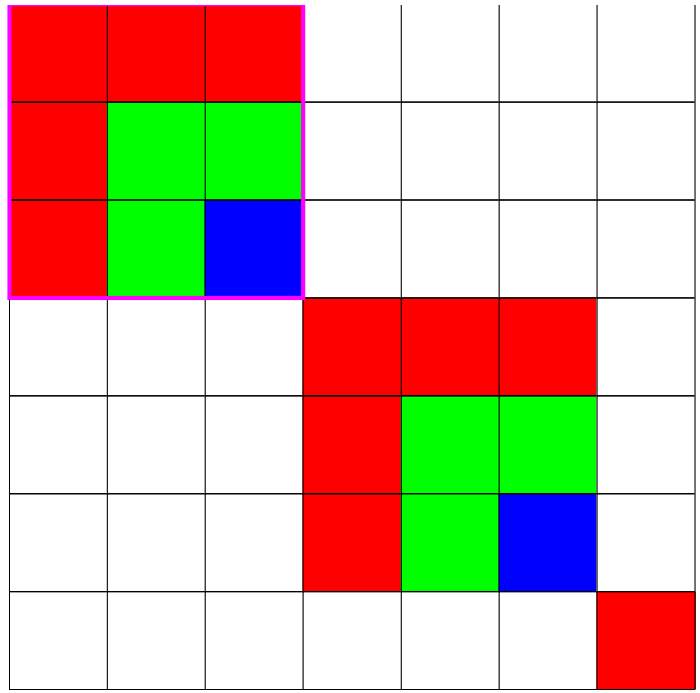




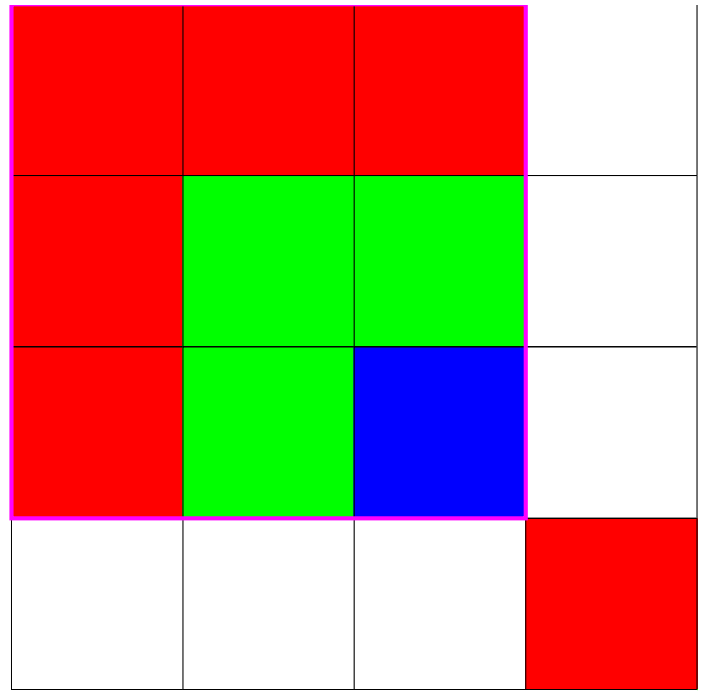
For this stacking the marginal distribution over *time* is given by the block diagonals.



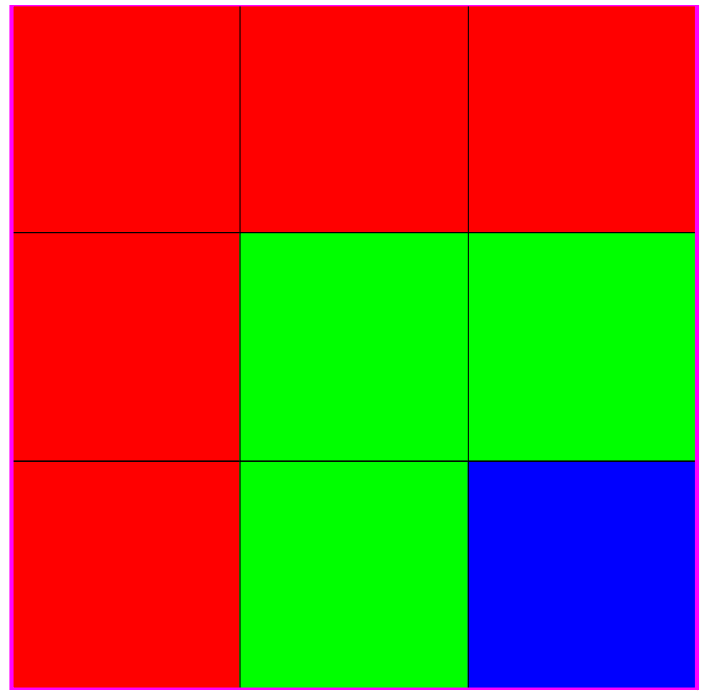
For this stacking the marginal distribution over *time* is given by the block diagonals.



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



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



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

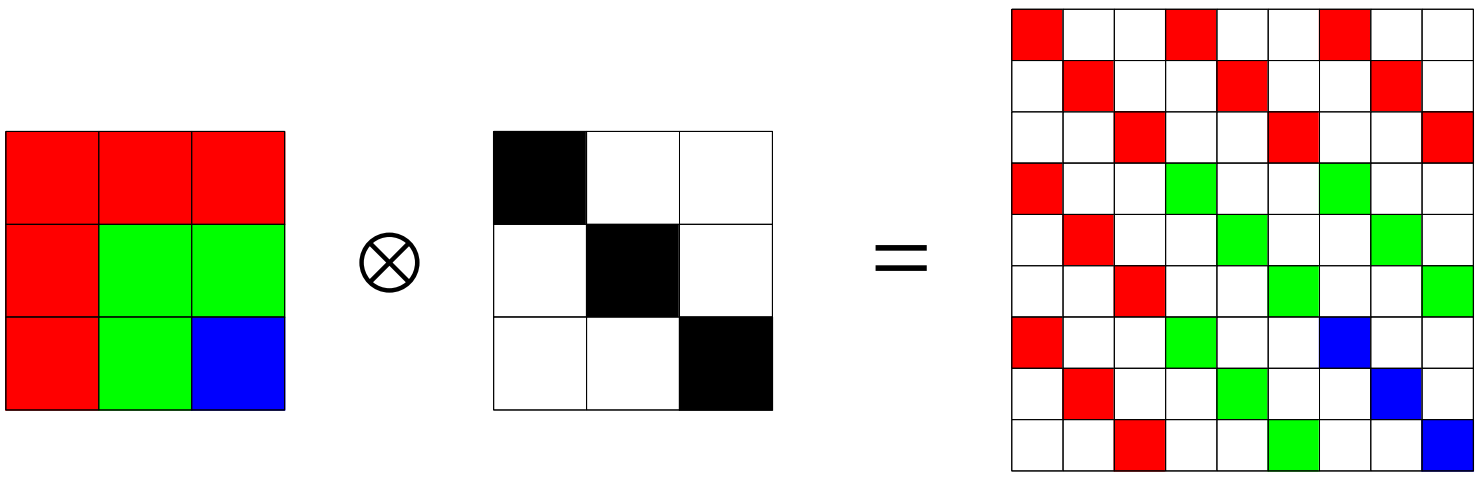
Two Ways of Stacking

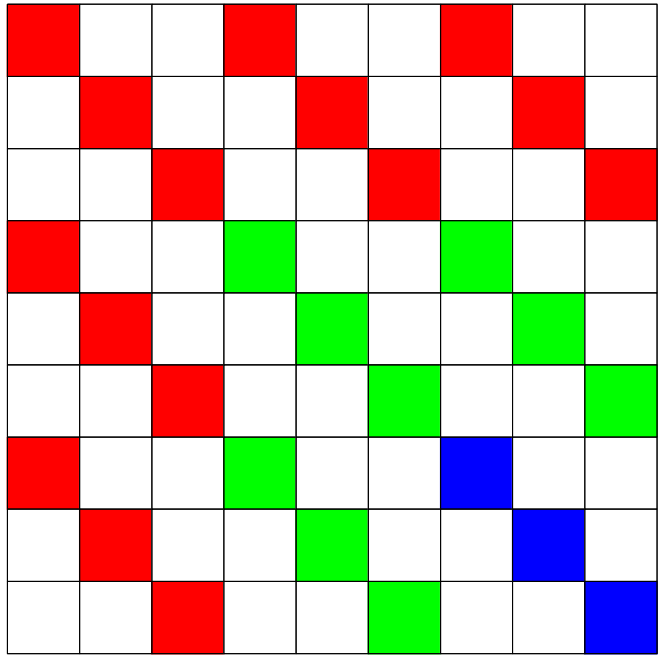
Can also stack each row of $\mathbf{X}$ to form column vector:

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

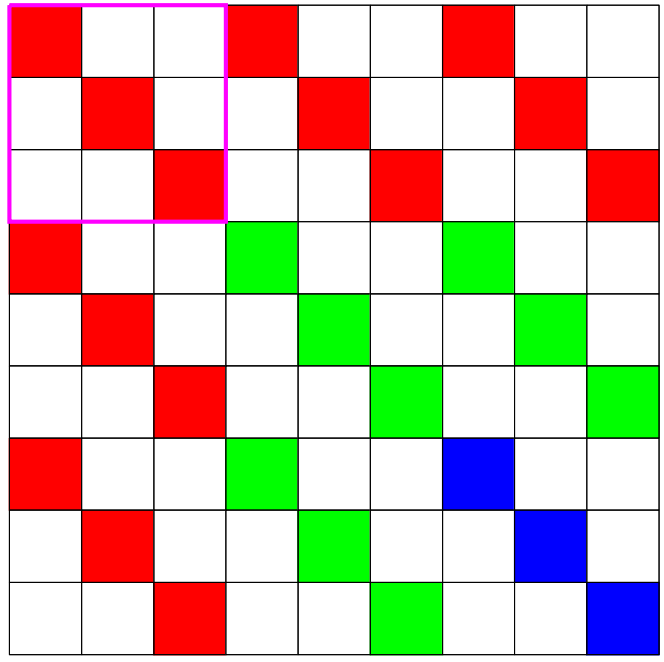
$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x}|\mathbf{0}, \mathbf{K} \otimes \mathbf{I}\right)$$

Row Stacking

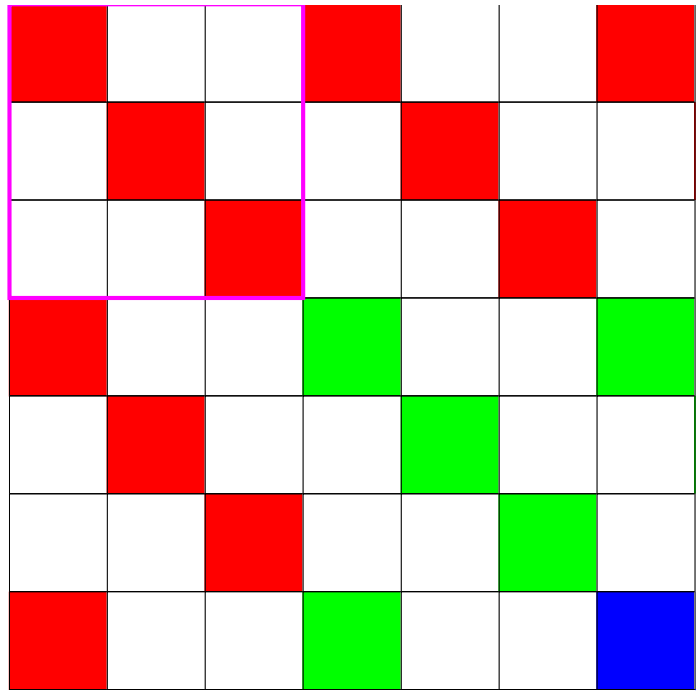




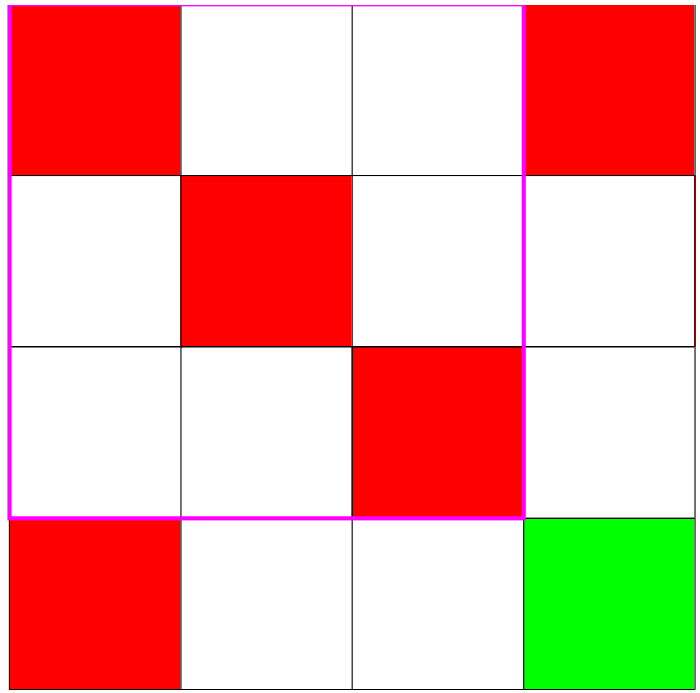
For this stacking the marginal distribution over the latent *dimensions* is given by the block diagonals.



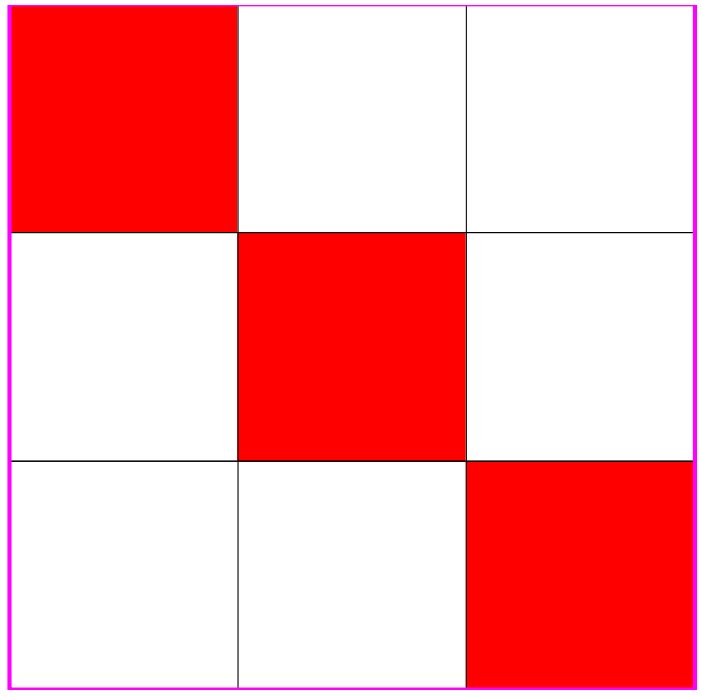
For this stacking the marginal distribution over the latent *dimensions* is given by the block diagonals.



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



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



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

Observed Process

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

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

$$\boldsymbol{\epsilon} \sim \mathcal{N}\left(0, \sigma^2\mathbf{I}\right)$$

Mapping from Latent Process to Observed

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

Output Covariance

This leads to a covariance of the form

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

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

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

or

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

Foundations and Trends® in
Machine Learning
Vol. 4, No. 3 (2011) 195–266

DOI: 10.1561/2200000036

now
the essence of knowledge

Kernels for Vector-Valued Functions: A Review

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

Kronecker Structure GPs

- This Kronecker structure leads to several published models.

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

  where k has $\mathbf{x}$ and k_T has i as inputs.
- Can think of multiple output covariance functions as covariances with augmented input.
- Alongside $\mathbf{x}$ we also input the j associated with the *output* of interest.

Separable Covariance Functions

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

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

  where $\mathbf{B}$ is a $p \times p$ symmetric and positive semi-definite matrix.
- $\mathbf{B}$ is called the *coregionalization* matrix.
- We call this class of covariance functions *separable* due to their product structure.

Sum of Separable Covariance Functions

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

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

- This can also be written as

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

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

Geostatistics

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

Weighted sum of Latent Functions

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

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

  where the latent functions are independent and have covariance functions $k_j(\mathbf{x}, \mathbf{x}')$.
- The processes $\{f_j(\mathbf{x})\}_{j=1}^{q}$ are independent for $q \neq j'$.

Kalman Filter Special Case

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

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

Intrinsic Coregionalization Model

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

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

Autokrigeability

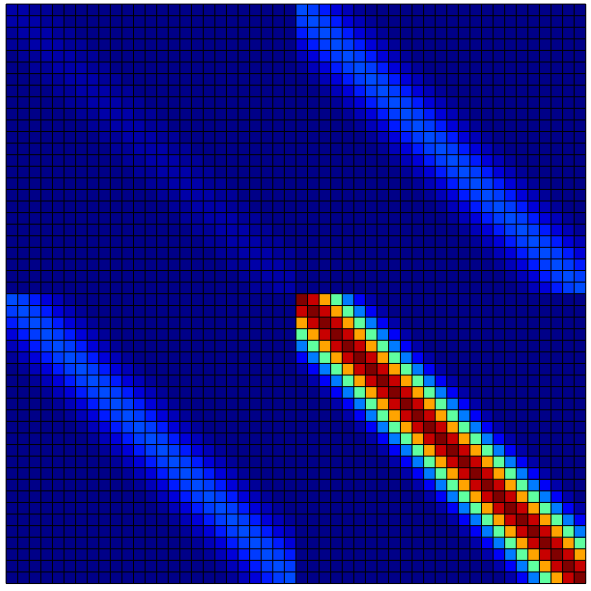
- If outputs are noise-free, maximum likelihood is equivalent to independent fits of $\mathbf{B}$ and $k(\mathbf{x}, \mathbf{x}')$ (Helterbrand and Cressie, 1994).
- In geostatistics this is known as autokrigeability (Wackernagel, 2003).
- In multitask learning its the cancellation of intertask transfer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

Intrinsic Coregionalization Model

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

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

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

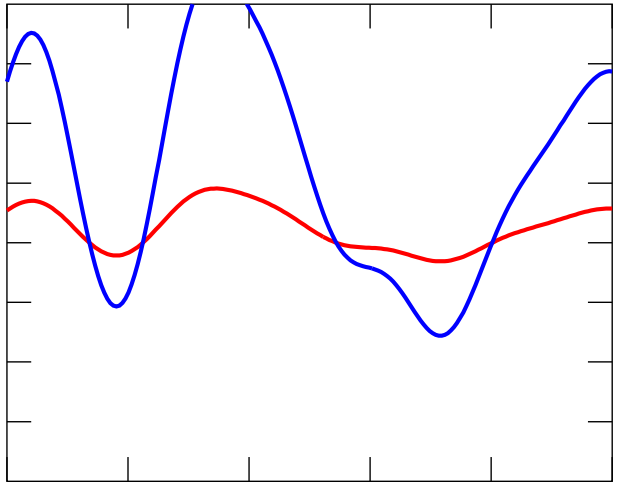


Intrinsic Coregionalization Model

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

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

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

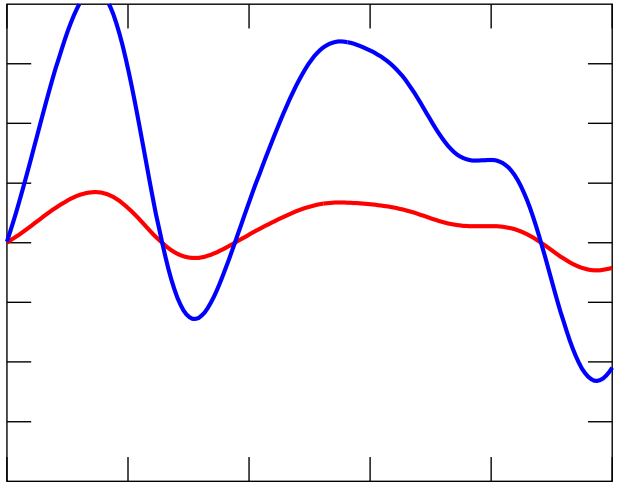


Intrinsic Coregionalization Model

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

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

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

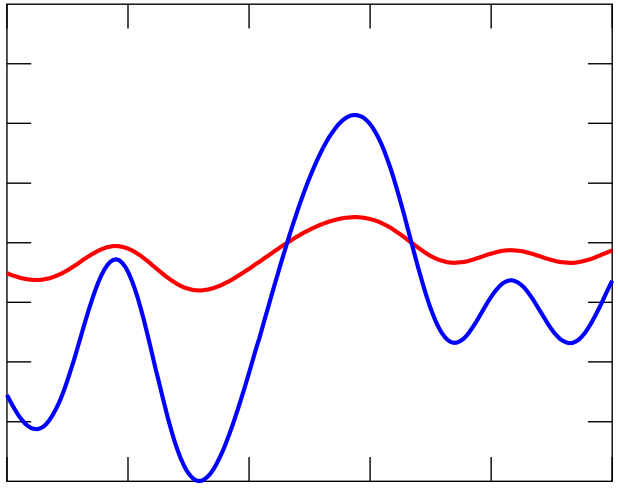


Intrinsic Coregionalization Model

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

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

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

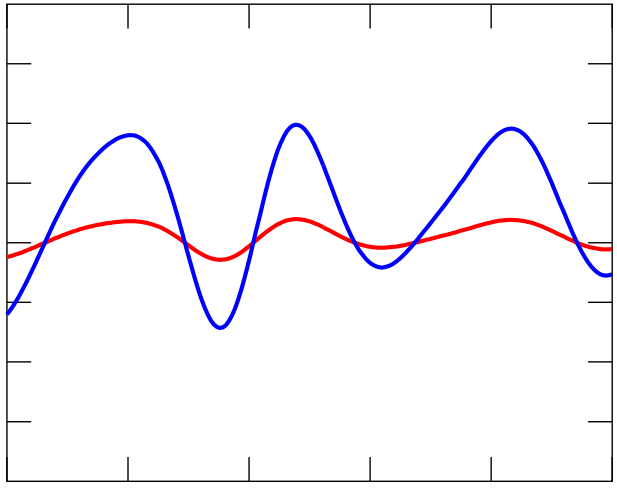


Intrinsic Coregionalization Model

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

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

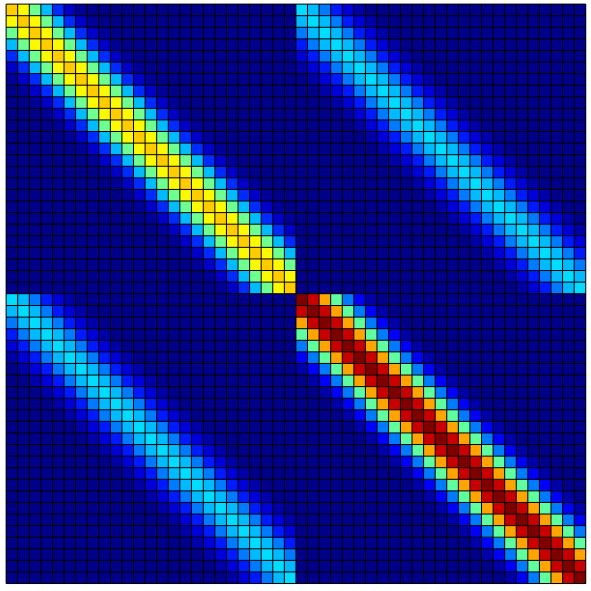
$$\mathbf{B} = \begin{bmatrix} 1 & 5 \\ 5 & 25 \end{bmatrix}$$



Intrinsic Coregionalization Model

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

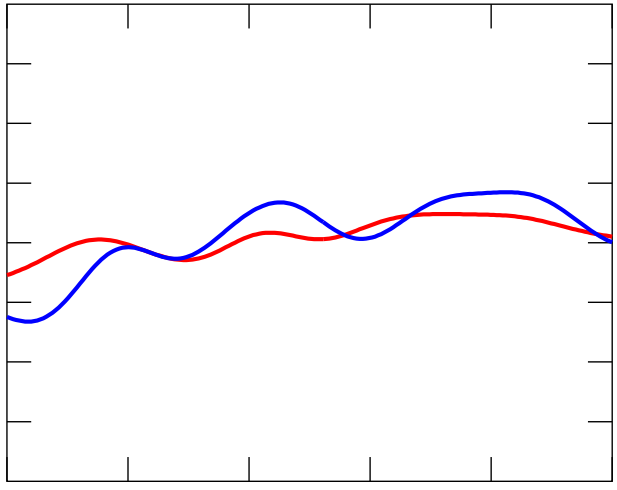
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



Intrinsic Coregionalization Model

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

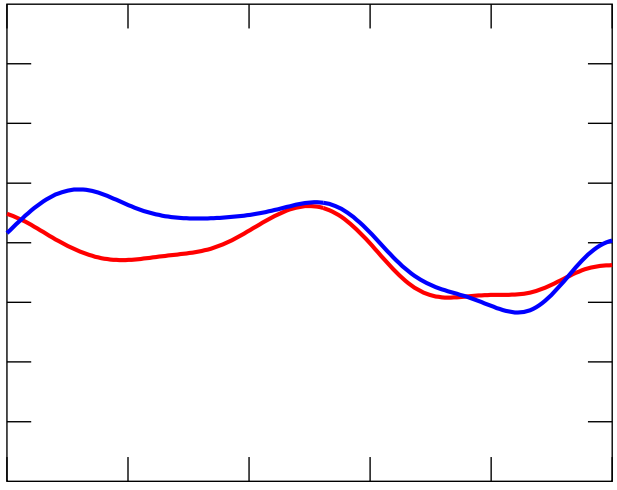
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



Intrinsic Coregionalization Model

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

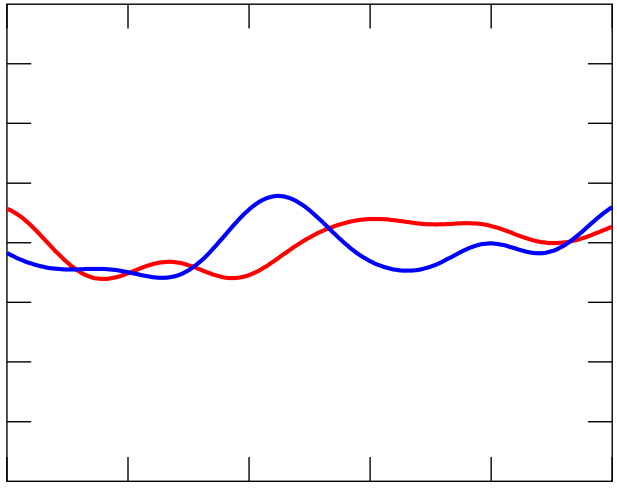
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



Intrinsic Coregionalization Model

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

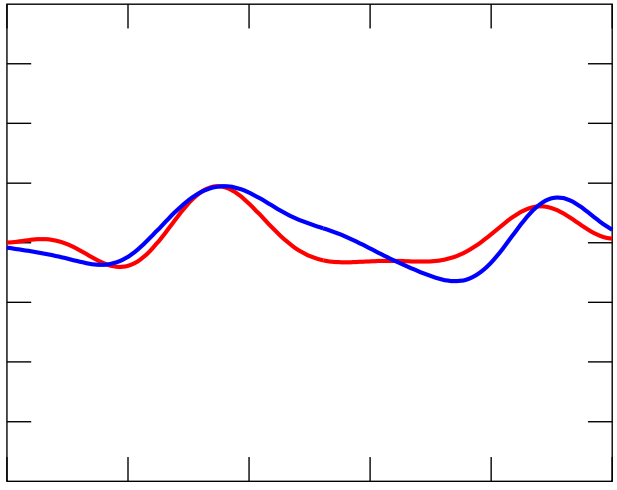
$$\mathbf{B} = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1.5 \end{bmatrix}$$



Intrinsic Coregionalization Model

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

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



LMC Samples

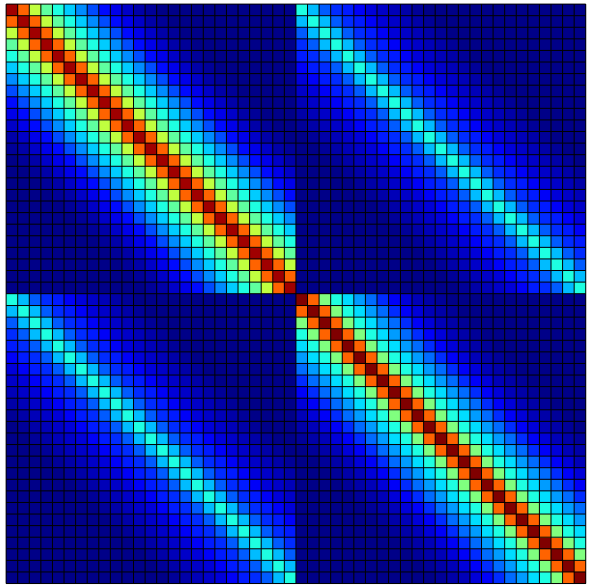
$$\mathbf{K}(\mathbf{X},\mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X},\mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X},\mathbf{X})$$

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

$$\ell_1 = 1$$

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

$$\ell_2 = 0.2$$



LMC Samples

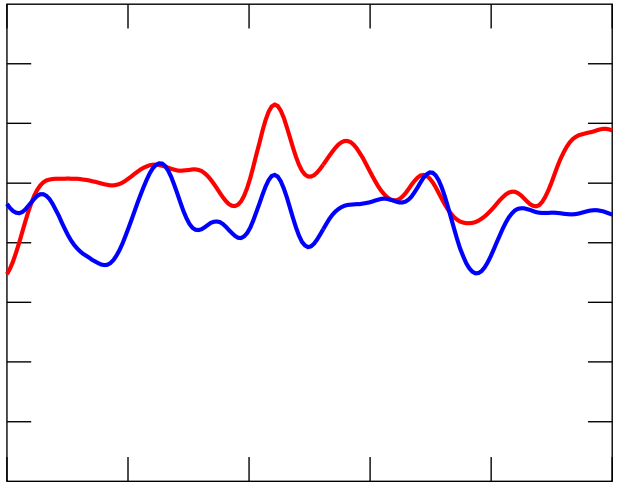
$$\mathbf{K}(\mathbf{X},\mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X},\mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X},\mathbf{X})$$

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

$$\ell_1 = 1$$

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

$$\ell_2 = 0.2$$



LMC Samples

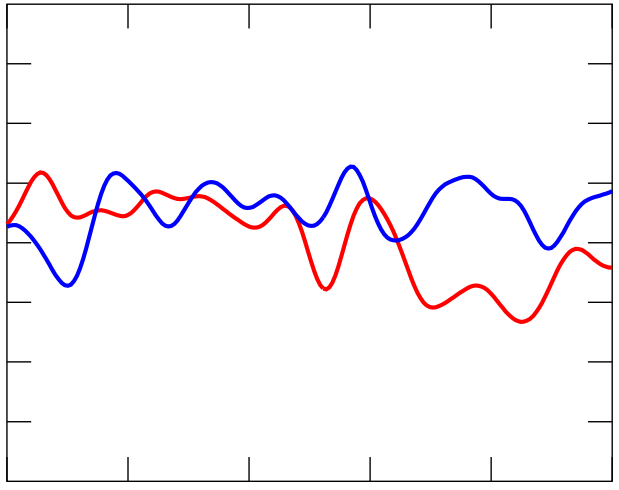
$$\mathbf{K}(\mathbf{X},\mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X},\mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X},\mathbf{X})$$

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

$$\ell_1 = 1$$

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

$$\ell_2 = 0.2$$



LMC Samples

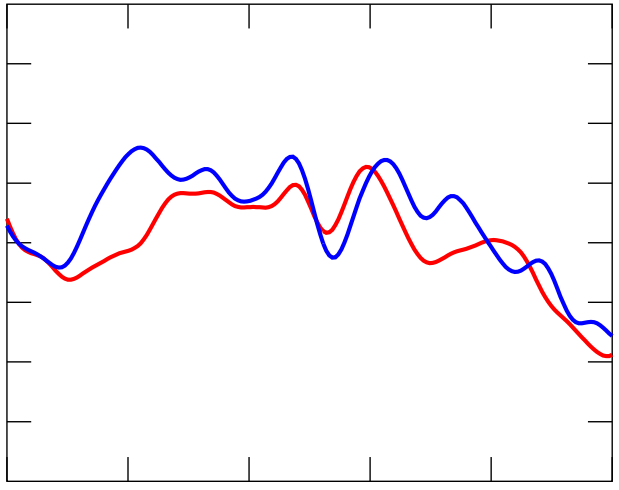
$$\mathbf{K}(\mathbf{X},\mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X},\mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X},\mathbf{X})$$

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

$$\ell_1 = 1$$

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

$$\ell_2 = 0.2$$



LMC Samples

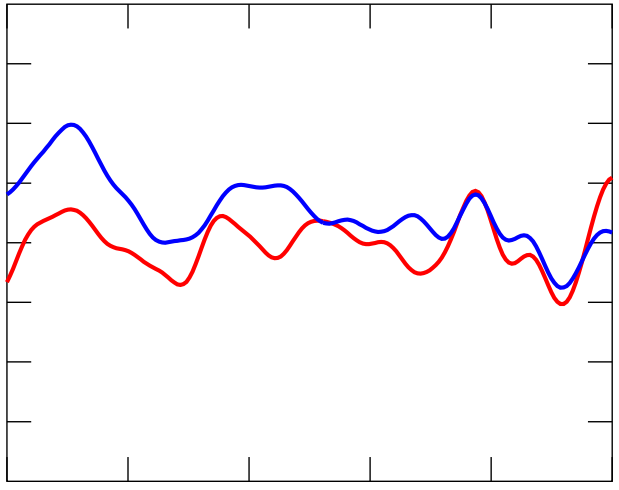
$$\mathbf{K}(\mathbf{X},\mathbf{X}) = \mathbf{B}_1 \otimes k_1(\mathbf{X},\mathbf{X}) + \mathbf{B}_2 \otimes k_2(\mathbf{X},\mathbf{X})$$

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

$$\ell_1 = 1$$

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

$$\ell_2 = 0.2$$



LMC in Machine Learning and Statistics

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

Semiparametric Latent Factor Model

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

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

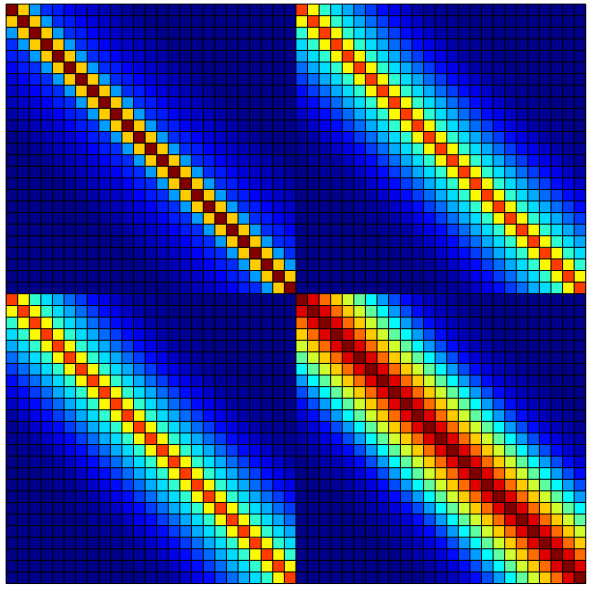
- Like the Kalman filter, but each latent function has a *different* covariance.
- Authors suggest using an exponentiated quadratic characteristic length-scale for each input dimension.

Semiparametric Latent Factor Model Samples

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

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

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

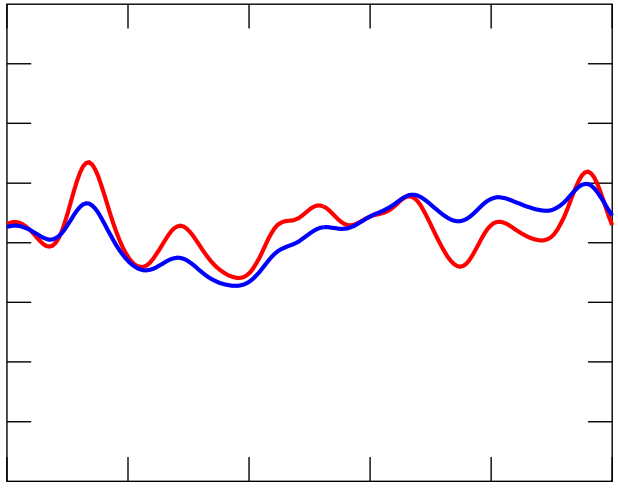


Semiparametric Latent Factor Model Samples

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

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

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

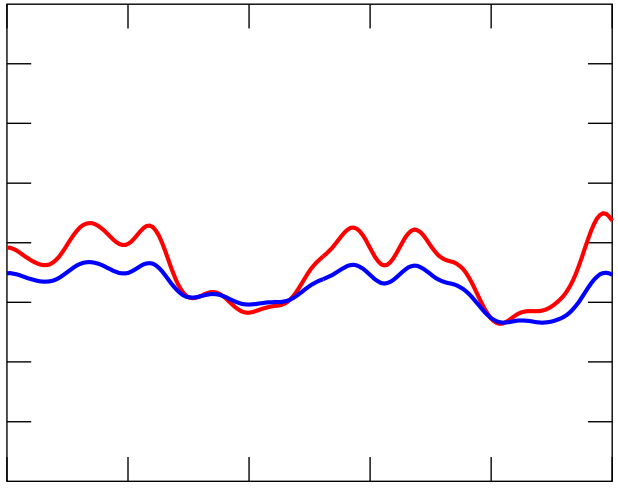


Semiparametric Latent Factor Model Samples

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

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

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

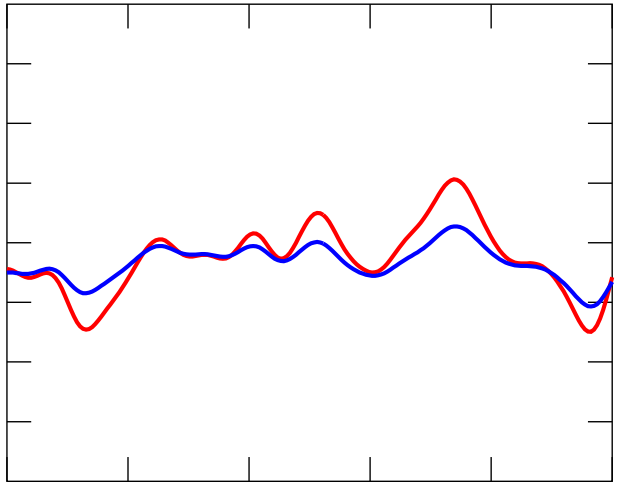


Semiparametric Latent Factor Model Samples

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

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

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

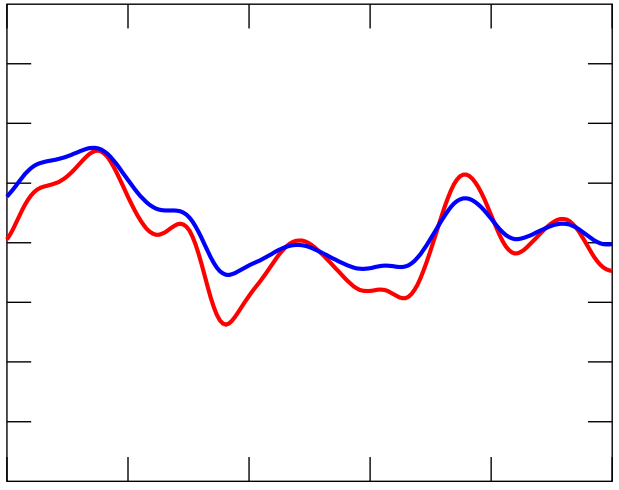


Semiparametric Latent Factor Model Samples

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

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

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



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

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

Multitask Classification

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

Computer Emulation

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

Approximations

Neil D. Lawrence

GPRS
25th–27th February 2015

Outline

Outline

Approximations in GPs

- Two main challenges:
  - Computational complexity and storage of exact inference $O(n^3)$ and $O(n^2)$ respectively.
  - Non Gaussian likelihoods making requisite integrals intractable.
- In this section we address these challenges.

Bayes Rule and Gaussian Processes

- So far we have focussed on *joint* Gaussians and exploited their properties.

$$p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \mathbf{K} + \sigma^2\mathbf{I}\right)$$

This is derived from

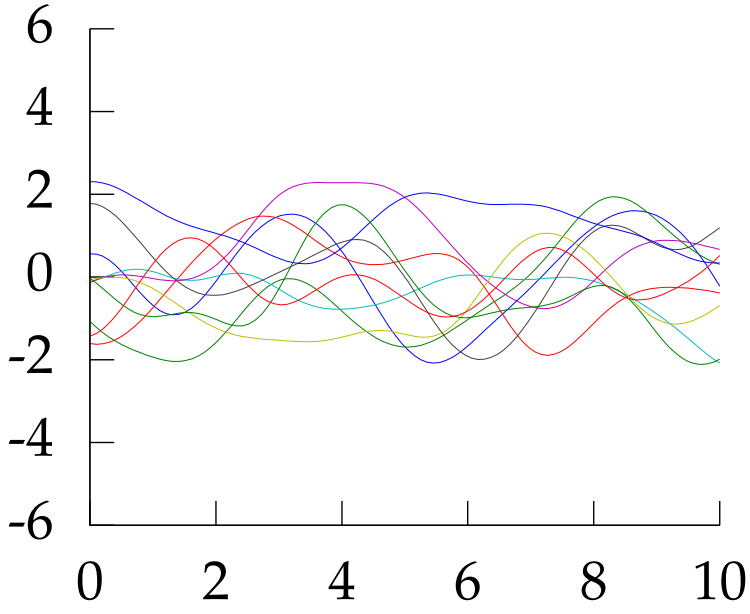
$$y(\mathbf{x}_i) = f(\mathbf{x}_i) + \epsilon_i$$

where

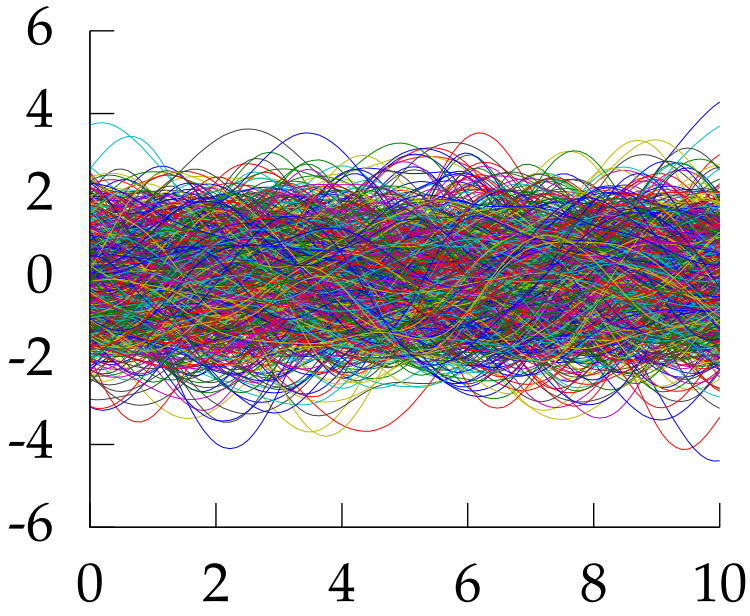
$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}) \quad \text{and} \quad \epsilon \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$$

- Let's remind ourselves of principles of probabilistic inference.

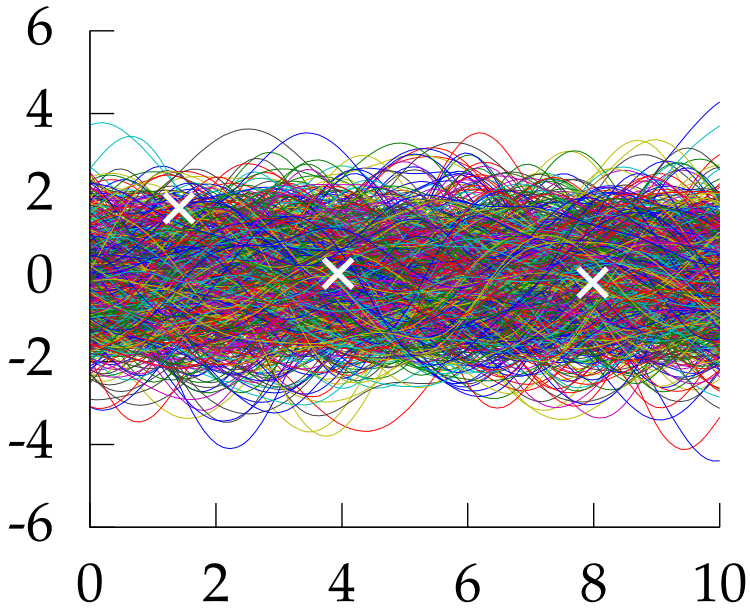
Gaussian Processes: Extremely Short Overview



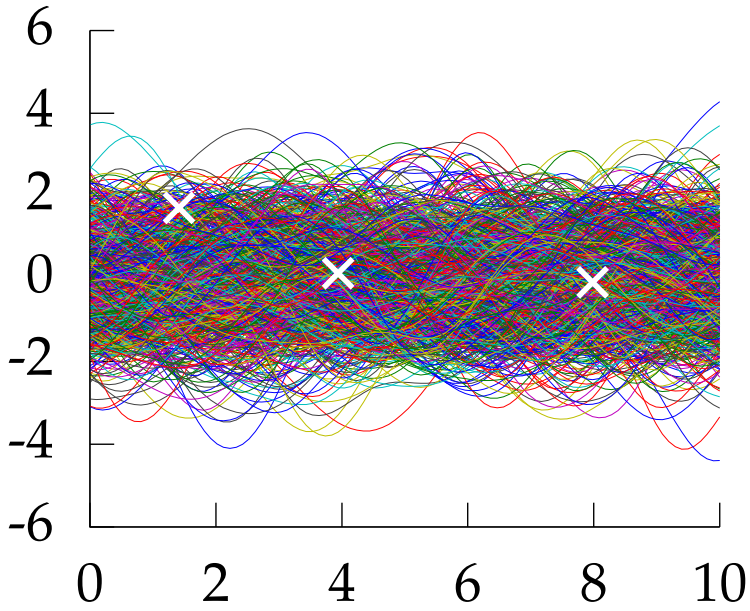
Gaussian Processes: Extremely Short Overview

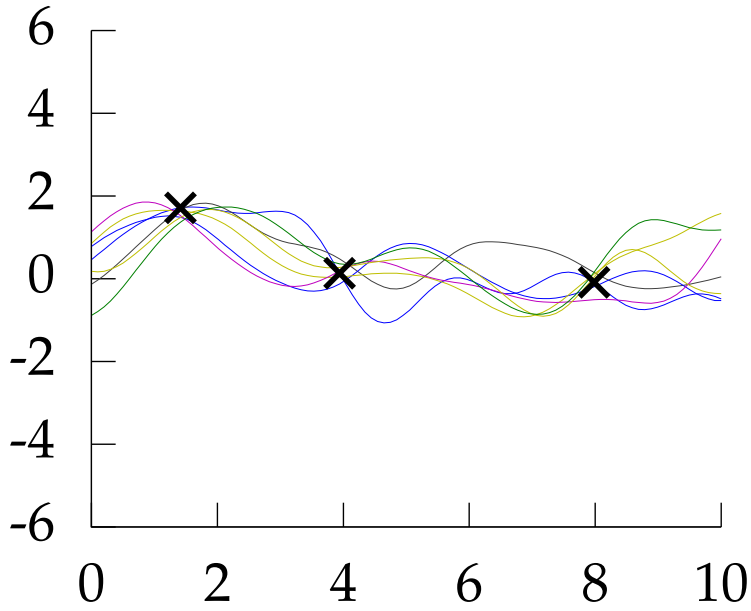


Gaussian Processes: Extremely Short Overview



Gaussian Processes: Extremely Short Overview





Classical Bayesian Inference

- The way we can perform inference in Gaussian systems is special (properties of multivarate Gaussians).
- Classically we need to declare a prior, $p(\mathbf{f})$.
- Combine it with a likelihood, $p(\mathbf{y}|\mathbf{f})$,

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f})}{p(\mathbf{y})}$$

- The easy bit is the multiplication on top. Normally the tough bit is

$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f})\mathrm{d}\mathbf{f}$$

it just happens to be trivial for the joint Gaussian case ...

Bayesian Inference, i.i.d. Likelihood

- Or for i.i.d. likelihood,

$$p(\mathbf{f}|\mathbf{y}) = \frac{\prod_{i=1}^{n} p(y_i|f_i)p(\mathbf{f})}{p(\mathbf{y})}$$

- If

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

  inference is trivial because

$$y_i = f_i + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right).$$

- In approximate GPs we will return to the more general formulation.

Variational Compression

(Lawrence, 2007; Titsias, 2009)

- Complexity of standard GP:
  - $O(n^3)$ in computation.
  - $O(n^2)$ in storage.

Variational Compression

(Lawrence, 2007; Titsias, 2009)

- Complexity of standard GP:
  - $O(n^3)$ in computation.
  - $O(n^2)$ in storage.
- Via low rank representations of covariance:
  - $O(nm^2)$ in computation.
  - $O(nm)$ in storage.
- Where m is user chosen number of *inducing* variables. They give the rank of the resulting covariance.

Variational Compression

(Lawrence, 2007; Titsias, 2009)

- Complexity of standard GP:
  - $O(n^3)$ in computation.
  - $O(n^2)$ in storage.
- Via low rank representations of covariance:
  - $O(nm^2)$ in computation.
  - $O(nm)$ in storage.
- Where m is user chosen number of *inducing* variables. They give the rank of the resulting covariance.

Variational Compression

- Inducing variables are a compression of the real observations.
- They can live in space of $\mathbf{f}$ or a space that is related through a linear operator (Álvarez et al., 2010) — could be gradient or convolution.
- There are inducing variables associated with each set of hidden variables, $\mathbf{x}^i$.

Variational Compression II

- **Importantly** conditioning on inducing variables renders the likelihood independent across the data.
  - It turns out that this allows us to variationally handle uncertainty on the kernel (including the inputs to the kernel).
  - It also allows standard scaling approaches: stochastic variational inference Hensman et al. (2013), parallelization Gal et al. (2014) and work by Zhenwen Dai on GPUs to be applied: an *engineering* challenge?

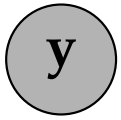
Inducing Variable Approximations

- Date back to (Williams and Seeger, 2001; Smola and Bartlett, 2001; Csató and Opper, 2002; Seeger et al., 2003; Snelson and Ghahramani, 2006). See Quiñonero Candela and Rasmussen (2005) for a review.
- We follow variational perspective of (Titsias, 2009).
- This is an augmented variable method, followed by a collapsed variational approximation (King and Lawrence, 2006; Hensman et al., 2012).

Augmented Variable Model: Not Wrong but Useful?

Augment standard model with a set of m new inducing variables, $\mathbf{u}$.

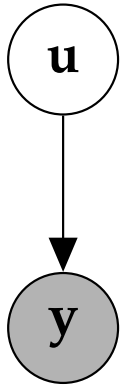
$$p(\mathbf{y}) = \int p(\mathbf{y}, \mathbf{u}) \mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Augment standard model with a set of m new inducing variables, $\mathbf{u}$.

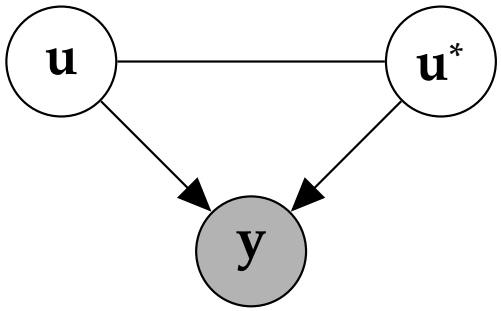
$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Important: Ensure inducing variables are *also* Kolmogorov consistent (we have m^* other inducing variables we are not *yet* using.)

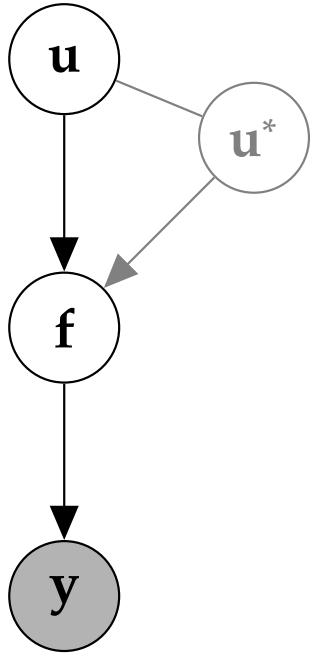
$$p(\mathbf{u}) = \int p(\mathbf{u}, \mathbf{u}^*) \mathrm{d}\mathbf{u}^*$$



Augmented Variable Model: Not Wrong but Useful?

Assume that relationship is through **f** (represents 'fundamentals'—push Kolmogorov consistency up to here).

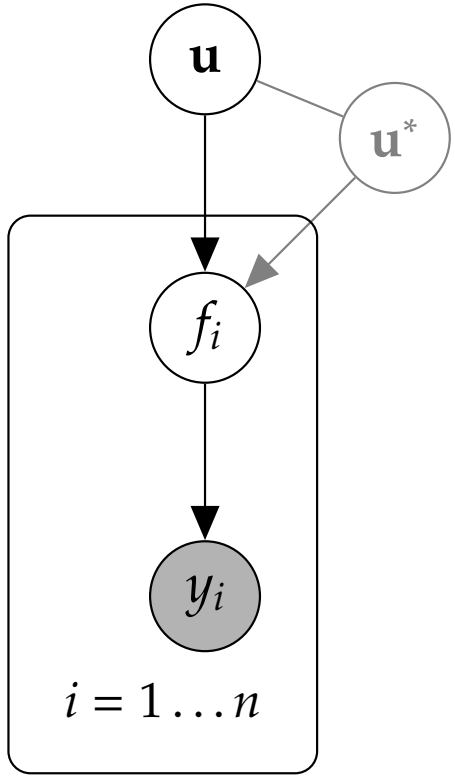
$$p(\mathbf{y}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Convenient to assume factorization (*doesn't* invalidate model—think delta function as worst case).

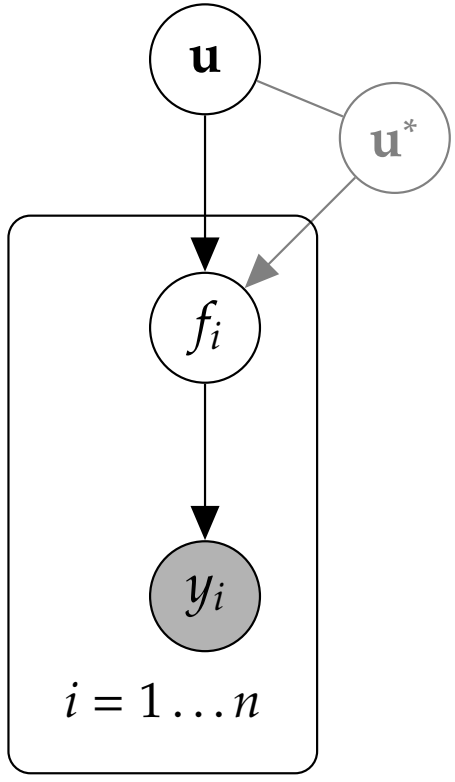
$$p(\mathbf{y}) = \int \prod_{i=1}^{n} p(y_i|f_i)p(\mathbf{f}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{f}\mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Focus on integral over $\mathbf{f}$.

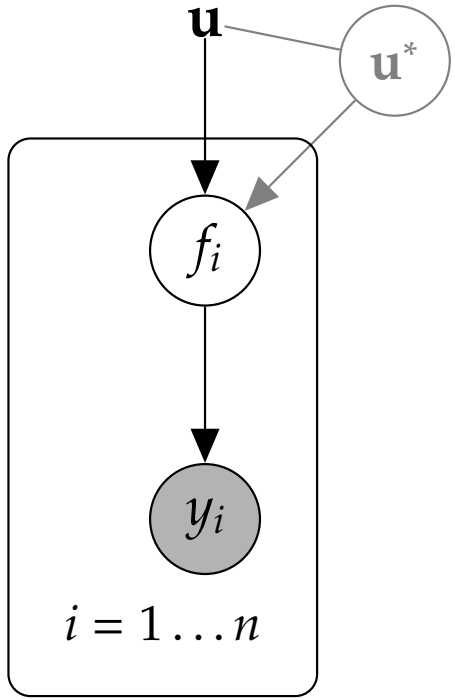
$$p(\mathbf{y}) = \int \int \prod_{i=1}^{n} p(y_i|f_i)p(\mathbf{f}|\mathbf{u})\mathrm{d}\mathbf{f}p(\mathbf{u})\mathrm{d}\mathbf{u}$$



Augmented Variable Model: Not Wrong but Useful?

Focus on integral over $\mathbf{f}$.

$$p(\mathbf{y}|\mathbf{u}) = \int \prod_{i=1}^{n} p(y_i|f_i)p(\mathbf{f}|\mathbf{u})\mathrm{d}\mathbf{f}$$



Variational Bound on $p(\mathbf{y}|\mathbf{u})$

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{u}) &= \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})\mathrm{d}\mathbf{f} \\ &= \int q(\mathbf{f}) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})}{q(\mathbf{f})}\mathrm{d}\mathbf{f} + \mathrm{KL}\left(q(\mathbf{f}) \,||\, p(\mathbf{f}|\mathbf{y}, \mathbf{u})\right)\end{aligned}$$

Variational Bound on $p(\mathbf{y}|\mathbf{u})$

$$\log p(\mathbf{y}|\mathbf{u}) = \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})\mathrm{d}\mathbf{f}$$
$$= \int q(\mathbf{f}) \log \frac{p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\mathbf{u})}{q(\mathbf{f})}\mathrm{d}\mathbf{f} + \mathrm{KL}\left(q(\mathbf{f}) \,||\, p(\mathbf{f}|\mathbf{y}, \mathbf{u})\right)$$

(Titsias, 2009)

- Example, set $q(\mathbf{f}) = p(\mathbf{f}|\mathbf{u})$,

$$\log p(\mathbf{y}|\mathbf{u}) \geq \log \int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f})\mathrm{d}\mathbf{f}.$$

$$p(\mathbf{y}|\mathbf{u}) \geq \exp \int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f})\mathrm{d}\mathbf{f}.$$

Optimal Compression in Inducing Variables

- Maximizing lower bound minimizes the KL divergence (information gain):

$$\mathrm{KL}\left(p(\mathbf{f}|\mathbf{u}) \,||\, p(\mathbf{f}|\mathbf{y},\mathbf{u})\right) = \int p(\mathbf{f}|\mathbf{u}) \log \frac{p(\mathbf{f}|\mathbf{u})}{p(\mathbf{f}|\mathbf{y},\mathbf{u})} \mathrm{d}\mathbf{u}$$

- This is minimized when the information stored about $\mathbf{y}$ is stored already in $\mathbf{u}$.
- The bound seeks an *optimal compression* from the *information gain* perspective.
- If $\mathbf{u} = \mathbf{f}$ bound is exact ($\mathbf{f}$ d-separates $\mathbf{y}$ from $\mathbf{u}$).

Choice of Inducing Variables

- Optimizing the bound directly not always practical.
- Free to choose whatever heuristics for the inducing variables.
- Can quantify which heuristics perform better through checking lower bound.

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \exp \int p(\mathbf{f}|\mathbf{u}) \log \prod_{i=1}^{n} p(y_i|f_i) \mathrm{d}\mathbf{f}.$$

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \exp \int p(\mathbf{f}|\mathbf{u}) \log \prod_{i=1}^{n} p(y_i|f_i) \mathrm{d}\mathbf{f}.$$

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \exp \int p(\mathbf{f}|\mathbf{u}) \sum_{i=1}^{n} \log p(y_i|f_i) \mathrm{d}\mathbf{f}.$$

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \exp \int p(\mathbf{f}|\mathbf{u}) \sum_{i=1}^{n} \log p(y_i|f_i) \mathrm{d}\mathbf{f}.$$

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \exp \sum_{i=1}^{n} \int p(f_i|\mathbf{u}) \log p(y_i|f_i) \mathrm{d}\mathbf{f}.$$

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \exp \sum_{i=1}^{n} \int p(f_i|\mathbf{u}) \log p(y_i|f_i) \mathrm{d}\mathbf{f}.$$

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \prod_{i=1}^{n} \exp \int p(f_i|\mathbf{u}) \log p(y_i|f_i) \mathrm{d}\mathbf{f}.$$

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \prod_{i=1}^{n} \exp \int p(f_i|\mathbf{u}) \log p(y_i|f_i) \mathrm{d}\mathbf{f}.$$

- Then the bound factorizes.

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \prod_{i=1}^{n} \exp \left\langle \log p(y_i|f_i) \right\rangle_{p(f_i|\mathbf{u})}$$

- Then the bound factorizes.

Factorizing Likelihoods

- If the likelihood, $p(\mathbf{y}|\mathbf{f})$, factorizes

$$p(\mathbf{y}|\mathbf{u}) \geq \prod_{i=1}^{n} \exp \left\langle \log p(y_i|f_i) \right\rangle_{p(f_i|\mathbf{u})}$$

- Then the bound factorizes.
- Now need a choice of distributions for $\mathbf{f}$ and $\mathbf{y}|\mathbf{f}$ ...

Gaussian $p(y_i|f_i)$

For Gaussian likelihoods:

$$\left\langle \log p(y_i|f_i) \right\rangle_{p(f_i|\mathbf{u})} = -\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left(y_i - \langle f_i \rangle\right)^2 - \frac{1}{2\sigma^2}\left(\left\langle f_i^2 \right\rangle - \langle f_i \rangle^2\right)$$

Gaussian $p(y_i|f_i)$

For Gaussian likelihoods:

$$\left\langle \log p(y_i|f_i) \right\rangle_{p(f_i|\mathbf{u})} = -\frac{1}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left(y_i - \langle f_i \rangle\right)^2 - \frac{1}{2\sigma^2}\left(\left\langle f_i^2 \right\rangle - \langle f_i \rangle^2\right)$$

Implying:

$$p(y_i|\mathbf{u}) \geq \exp\left\langle \log c_i \right\rangle \mathcal{N}\left(y_i | \left\langle f_i \right\rangle, \sigma^2\right)$$

Gaussian Process Over $\mathbf{f}$ and $\mathbf{u}$

Define:

$$q_{i,i} = \text{var}_{p(f_i|\mathbf{u})}\left(f_i\right) = \left\langle f_i^2 \right\rangle_{p(f_i|\mathbf{u})} - \left\langle f_i \right\rangle^2_{p(f_i|\mathbf{u})}$$

We can write:

$$c_i = \exp\left(-\frac{q_{i,i}}{2\sigma^2}\right)$$

If joint distribution of $p(\mathbf{f}, \mathbf{u})$ is Gaussian then:

$$q_{i,i} = k_{i,i} - \mathbf{k}_{i,\mathbf{u}}^\top \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{k}_{i,\mathbf{u}}$$

c_i is not a function of $\mathbf{u}$ but *is* a function of $\mathbf{X}_\mathbf{u}$.

Lower Bound on Likelihood

Substitute variational bound into marginal likelihood:

$$p(\mathbf{y}) \geq \prod_{i=1}^{n} c_i \int \mathcal{N}\left(\mathbf{y} | \langle \mathbf{f} \rangle, \sigma^2 \mathbf{I}\right) p(\mathbf{u}) \mathrm{d}\mathbf{u}$$

Note that:

$$\langle \mathbf{f} \rangle_{p(\mathbf{f}|\mathbf{u})} = \mathbf{K}_{\mathbf{f},\mathbf{u}} \mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1} \mathbf{u}$$

is *linearly* dependent on $\mathbf{u}$.

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \int \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}, \sigma^2\right)\mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)\mathrm{d}\mathbf{u}$$

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Maximize log of the bound to find covariance function parameters,

$$L \geq \sum_{i=1}^{n} \log c_i + \log \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f},}\right)$$

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Maximize log of the bound to find covariance function parameters,

$$L \geq \sum_{i=1}^{n} \log c_i + \log \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f},}\right)$$

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Maximize log of the bound to find covariance function parameters,

$$L \approx \log \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f},}\right)$$

- If the bound is normalized, the c_i terms are removed.

Deterministic Training Conditional

Making the marginalization of $\mathbf{u}$ straightforward. In the Gaussian case:

$$p(\mathbf{u}) = \mathcal{N}\left(\mathbf{u}|\mathbf{0}, \mathbf{K}_{\mathbf{u},\mathbf{u}}\right)$$

$$\int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})\mathrm{d}\mathbf{u} \geq \prod_{i=1}^{n} c_i \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

Maximize log of the bound to find covariance function parameters,

- If the bound is normalized, the c_i terms are removed.
- This results in the projected process approximation (Rasmussen and Williams, 2006) or DTC (Quiñonero Candela and Rasmussen, 2005). Proposed by (Smola and Bartlett, 2001; Seeger et al., 2003; Csató and Opper, 2002; Csató, 2002).

Fully Independent Training Conditional

Define c_i' to be

$$c_i' = c_i \exp\left(\frac{\mathbf{y}_i^2 q_{i,i}}{2}\right) = \exp\left(\frac{q_{i,i}(\mathbf{y}_i^2 - \sigma^{-2})}{2}\right)$$

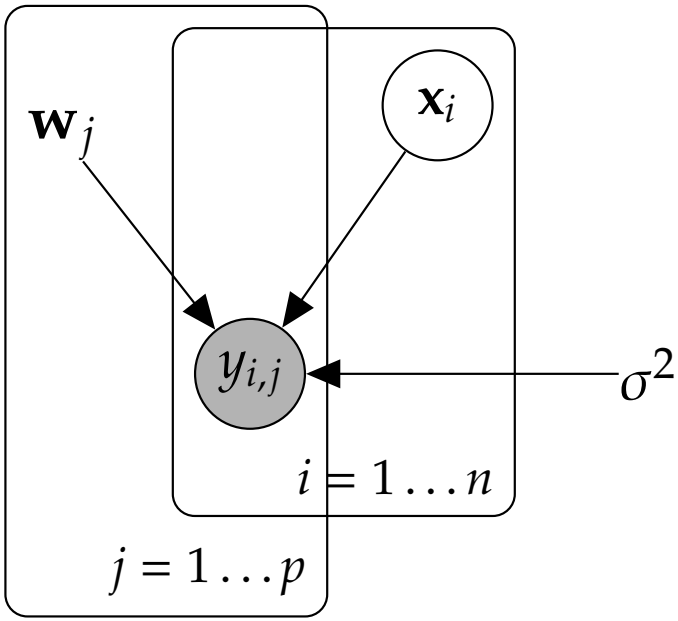
Then rewrite the bound:

$$\sum_{i=1}^{n} \log c_i' + \log \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \sigma^2\mathbf{I} + \operatorname{diag}\left(\mathbf{Q}_{\mathbf{f},\mathbf{f}}\right) + \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}\right)$$

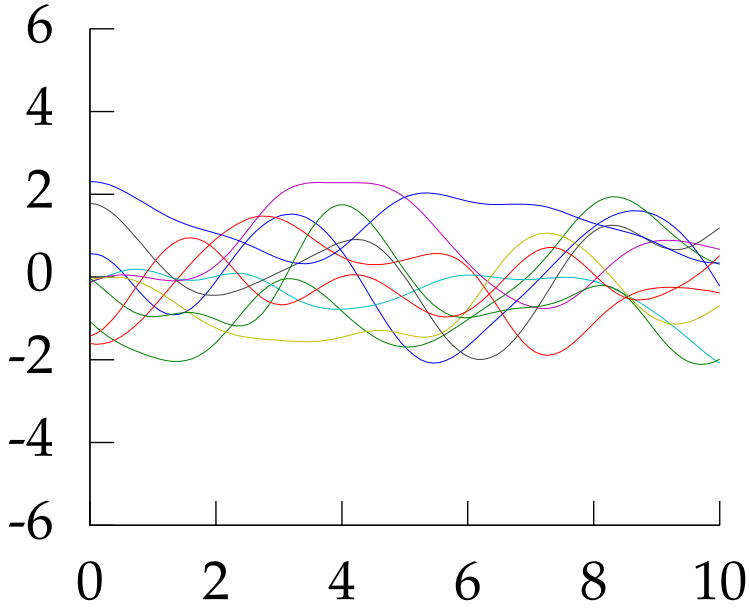
where

$$\mathbf{Q}_{\mathbf{f},\mathbf{f}} = \operatorname{cov}\left(\mathbf{f}\mathbf{f}^\top\right)_{p(\mathbf{f}|\mathbf{u})} = \mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{K}_{\mathbf{f},\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{K}_{\mathbf{u},\mathbf{f}}$$

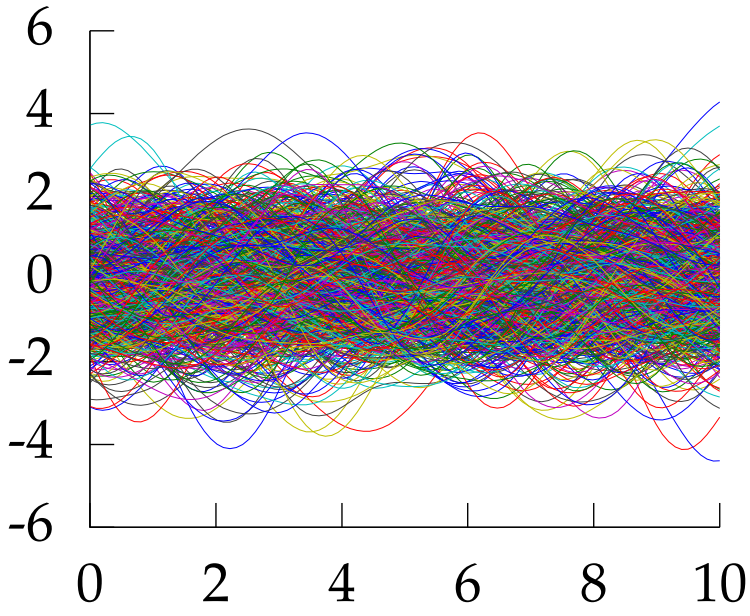
In FITC the $\log c_i'$ terms could be negative or positive.


$\mathbf{w}_j$
$\mathbf{x}_i$
$y_{i,j}$
σ^2
$i = 1 \ldots n$
$j = 1 \ldots p$

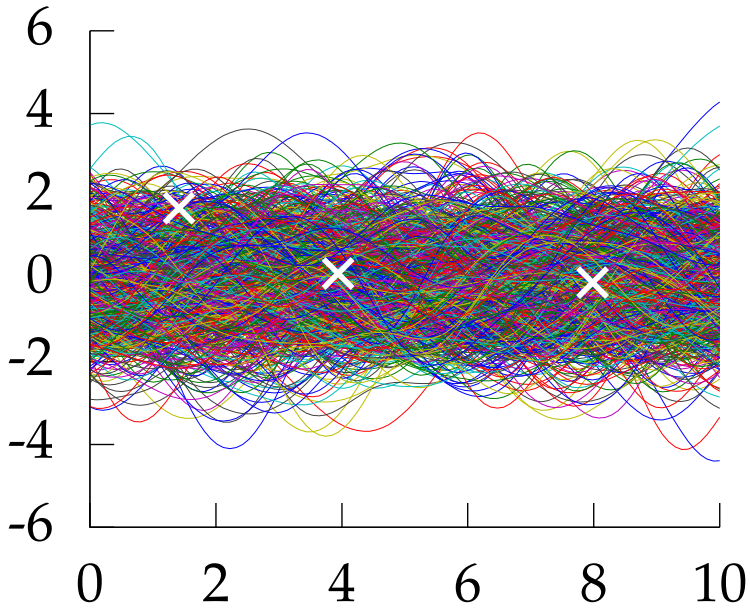
Gaussian Processes: Extremely Short Overview



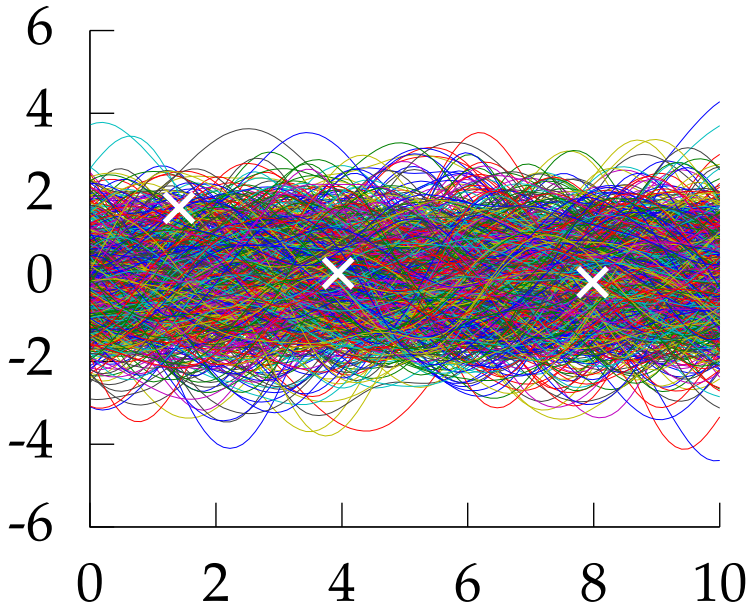
Gaussian Processes: Extremely Short Overview

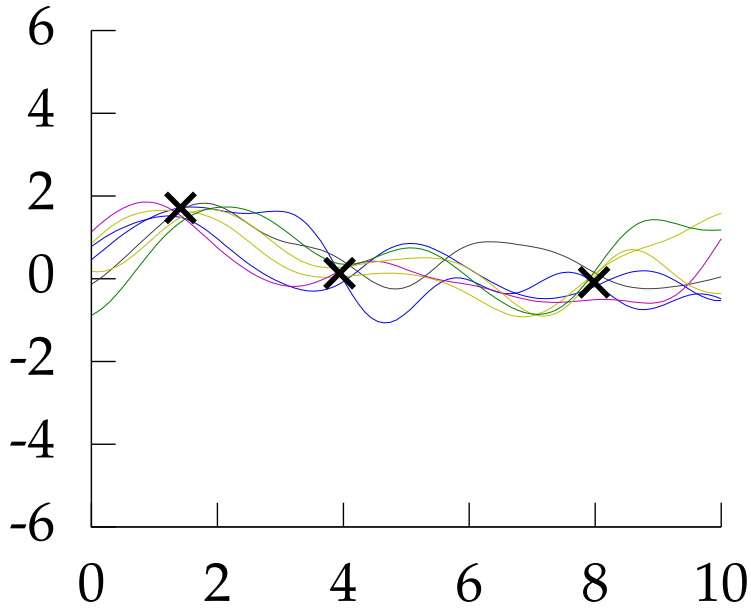


Gaussian Processes: Extremely Short Overview



Gaussian Processes: Extremely Short Overview





GP Regression

Analytical tractability of the posterior distribution is assured:

- Gaussian prior:

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K_{ff}})$$

- Gaussian likelihood:

$$\prod_{i=1}^{n} p(y_i|f_i) = \mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma_i^2\mathbf{I}\right)$$

- Gaussian posterior:

$$p(\mathbf{f}|\mathbf{y}) \propto \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K_{ff}})\,\mathcal{N}\left(\mathbf{y}|\mathbf{f}, \sigma_i^2\mathbf{I}\right)$$

Bernoulli Distribution

- A mathematical switch allows us to write a probability table as a function.

$$P(Y = 1) = \pi$$

$$P(Y = 0) = (1 - \pi)$$

- Write as a function

$$P(Y = y) = \pi^y (1 - \pi)^{1-y}$$

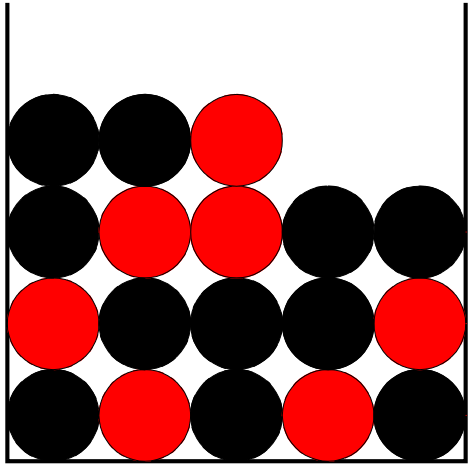
- Can think of this construction as a "mathematical switch". Known as the Bernoulli distribution.
- Widely used in classification algorithms: π parameter is made to be dependent on "inputs".

Binomial Distribution

- Generalization of Bernoulli to multiple trials.
- Jakob Bernoulli: black and red balls in an urn. Proportion of red is π.
- Sample with replacement. Binomial gives the distribution of number of reds, y, from S extractions

$$P(y|\pi, S) = \frac{S!}{y!(S-y)!}\pi^y(1-\pi)^{(S-y)}$$

- Mean is given by $S\pi$ and variance $S\pi(1-\pi)$.



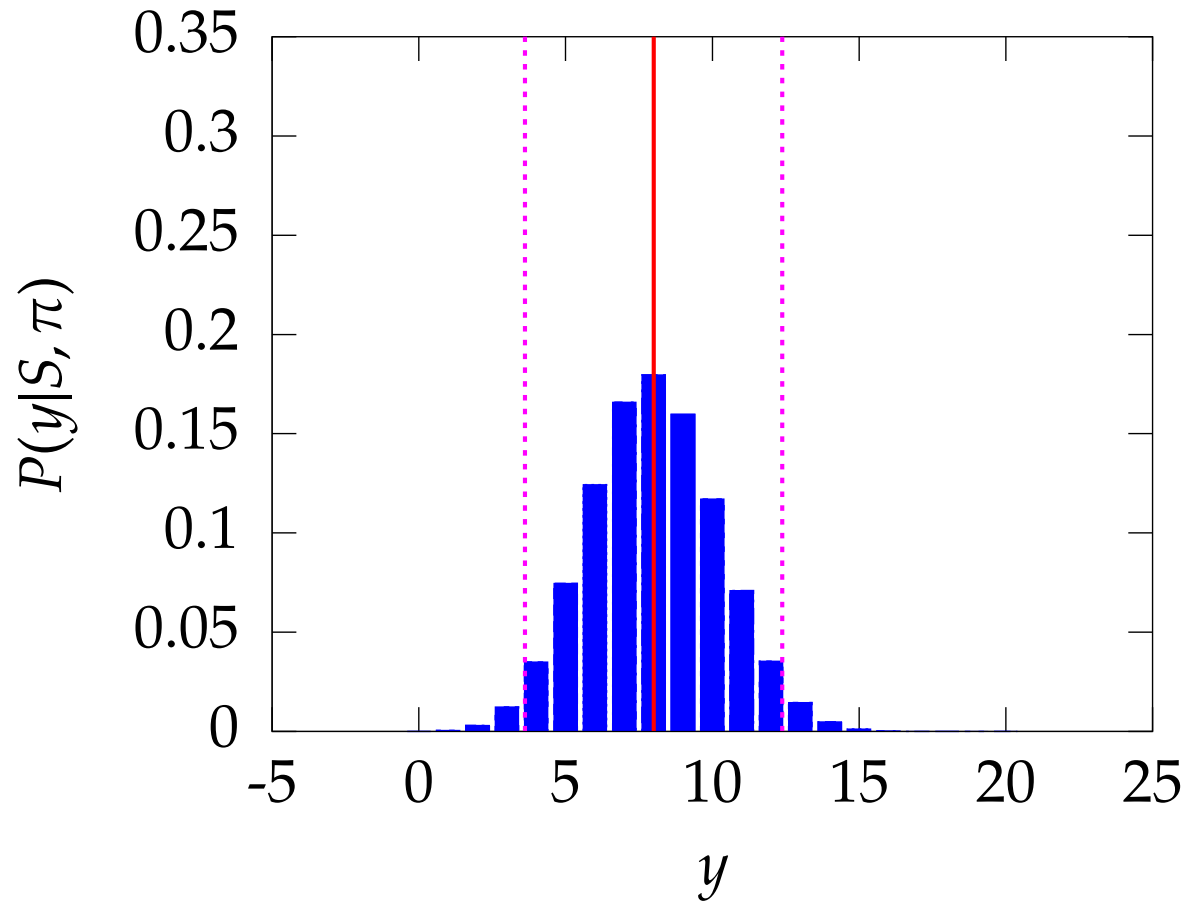


Figure : The binomial distribution for $\pi = 0.4$ and $S = 20$. Mean is shown as red line, 2 standard deviations are magenta.

The Gamma Density

- Density over positive real values.

$$p(y|a,b) = \frac{b^a}{\Gamma(a)} y^{a-1} \exp(-by)$$
$$= \mathcal{G}\left(y|\mu,\sigma^2\right)$$

- Mean is $\frac{a}{b}$ and variance is $\frac{a}{b^2}$.
- Also available in multivariate as the Wishart (positive definite matrices).

Gamma PDF I

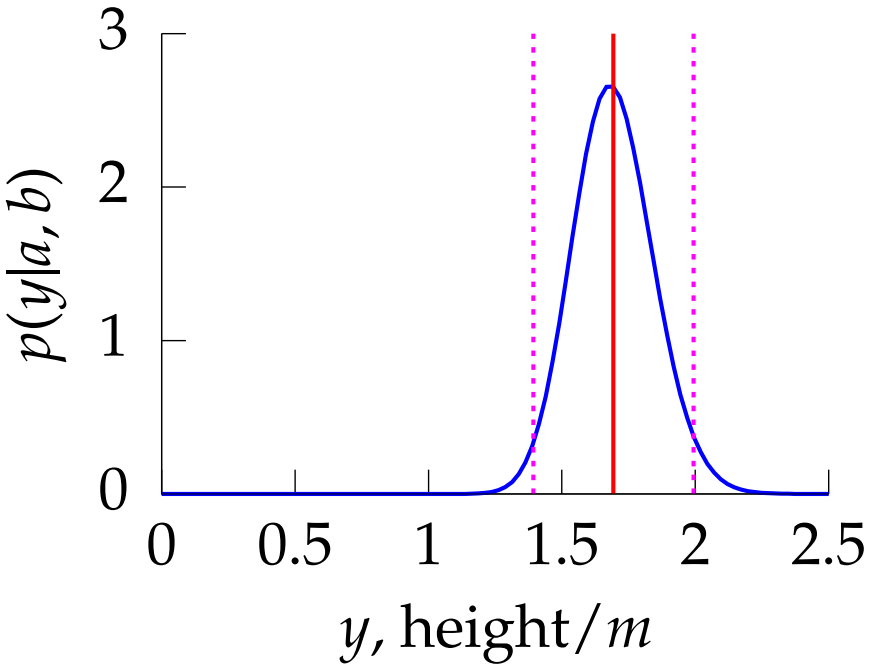


Figure : The Gamma PDF with $a = 127$ and $b = 75$. Here it represents the heights of a population of students and constrains them positive.

Gamma PDF I

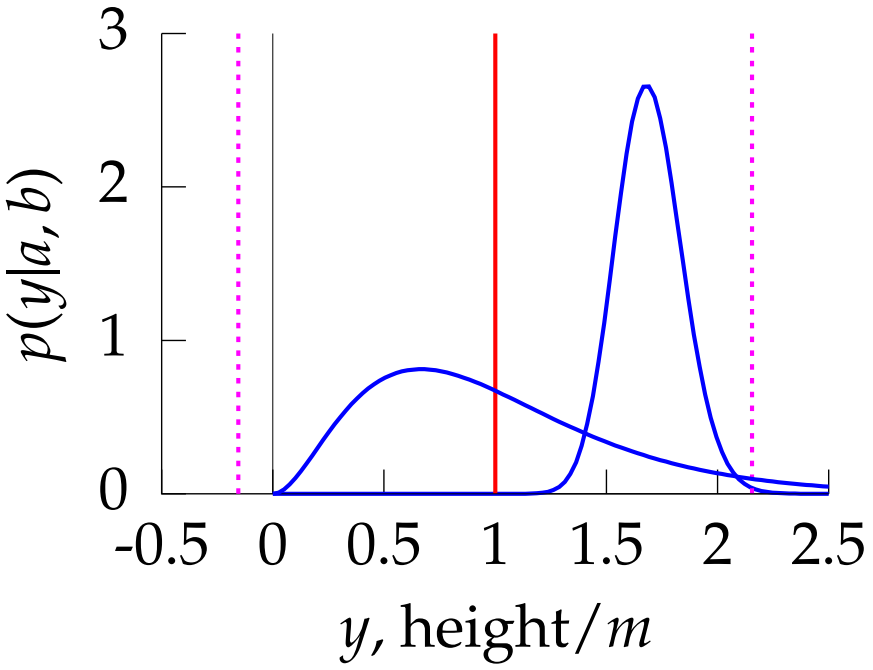


Figure : The Gamma PDF with $a = 127$ and $b = 75$ alongside a Gamma PDF with $a = 3$ and $b = 3$.

Categorical Distribution

Multiple outcomes, example: die roll.

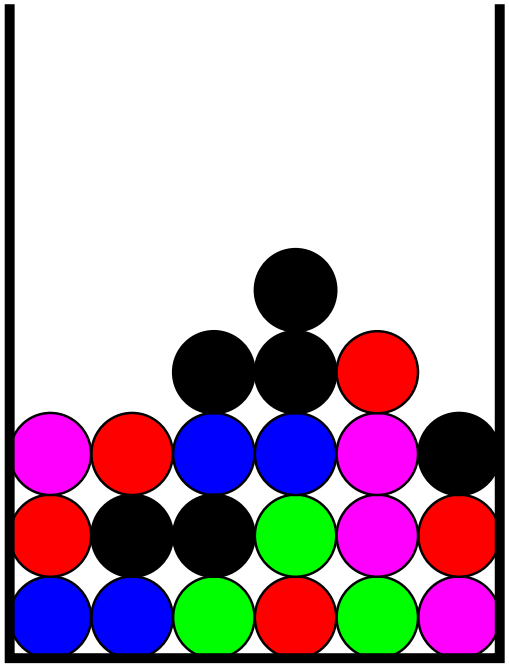
die role	probability	$\mathbf{y}$
1	π_1	[1 0 0 0 0 0]
2	π_2	[0 1 0 0 0 0]
3	π_3	[0 0 1 0 0 0]
4	π_4	[0 0 0 1 0 0]
5	π_5	[0 0 0 0 1 0]
6	π_6	[0 0 0 0 0 1]

$$P(\mathbf{y}) = \prod_{i=1}^{k} \pi_i^{y_i}$$

Multinomial Distribution

- Generalization of categorical to multiple trials.
- Generalization of binomial to multiple outcomes. Proportion of each colour ball is now π_i.
- Sample with replacement. Multinomial gives the distribution of number of each of k different balls, y, from S extractions

$$P(y|\pi, S) = \frac{S!}{\prod_{i=1}^{k} y_i!} \prod_{i=1}^{k} \pi_i^{y_i}$$



- Mean for each colour is given by $S\pi_i$ and variance $S\pi_i(1 - \pi_i)$.

Distributions as Functions

- Probability distribution with a simple table can be limiting.
- The Poisson Distribution — a distribution a a function
- First published by **Siméon Denis Poisson** (1781-1840) in 1837.
- Defined over the space of all non-negative integers.
- This set is countably infinite: impossible to summarise in a table!
- The Poisson distribution is therefore defined as

$$P(y|\mu) = \frac{\mu^y}{y!} \exp(-\mu). \tag{2}$$

where y is any integer from 0 to ∞, and μ is a parameter of the distribution.

A Poisson with $\mu = 2$

- To work out the probability of y in a Poisson $\mu = 2$ we can start filling a table.
- The values in a table are computed from (2)

y	0	1	2	...
$P(y)$	0.135	0.271	0.271	...

Table : Some values for the Poisson distribution with $\mu = 2$.

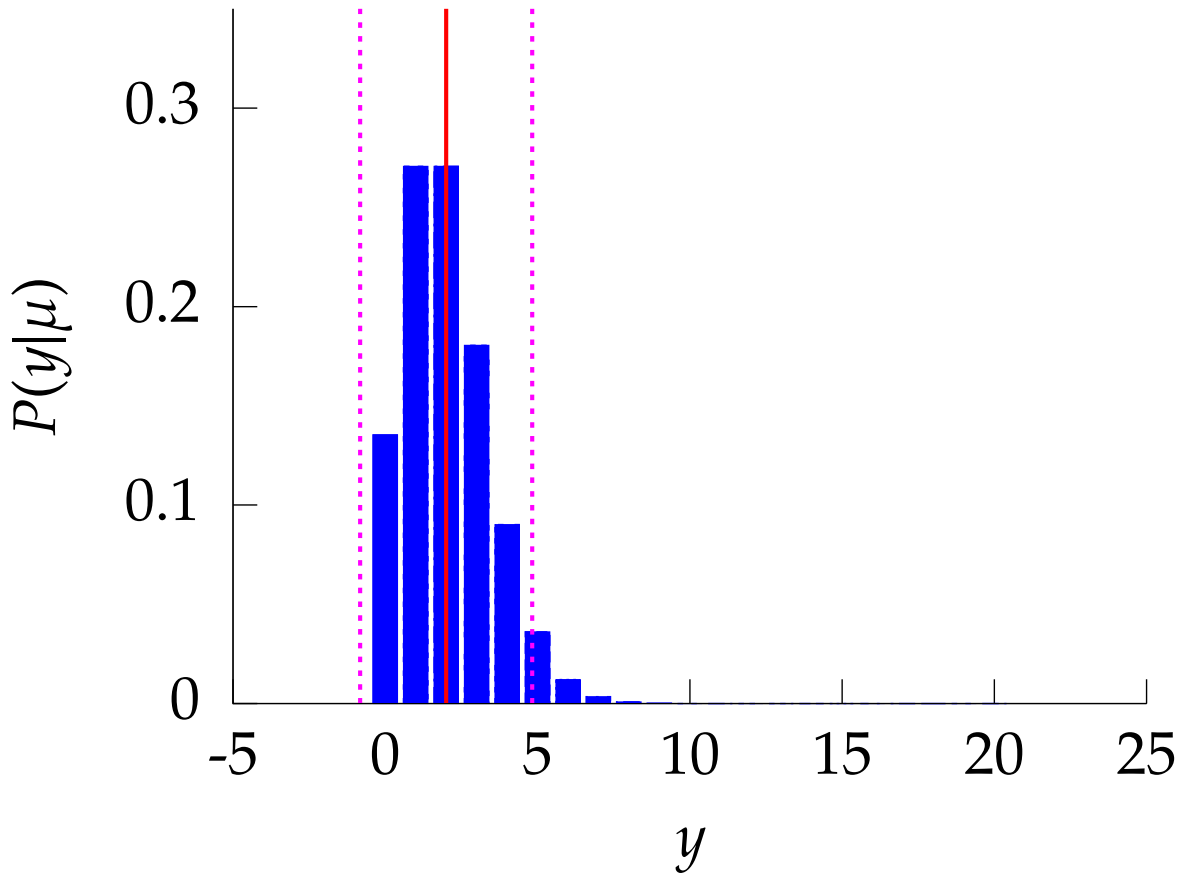


Figure : The Poisson distribution for $\mu = 2$. Mean is given by μ (red line), standard deviation is given by $\sqrt{\mu}$ (magenta lines show 2 standard deviations).

Gaussian Noise

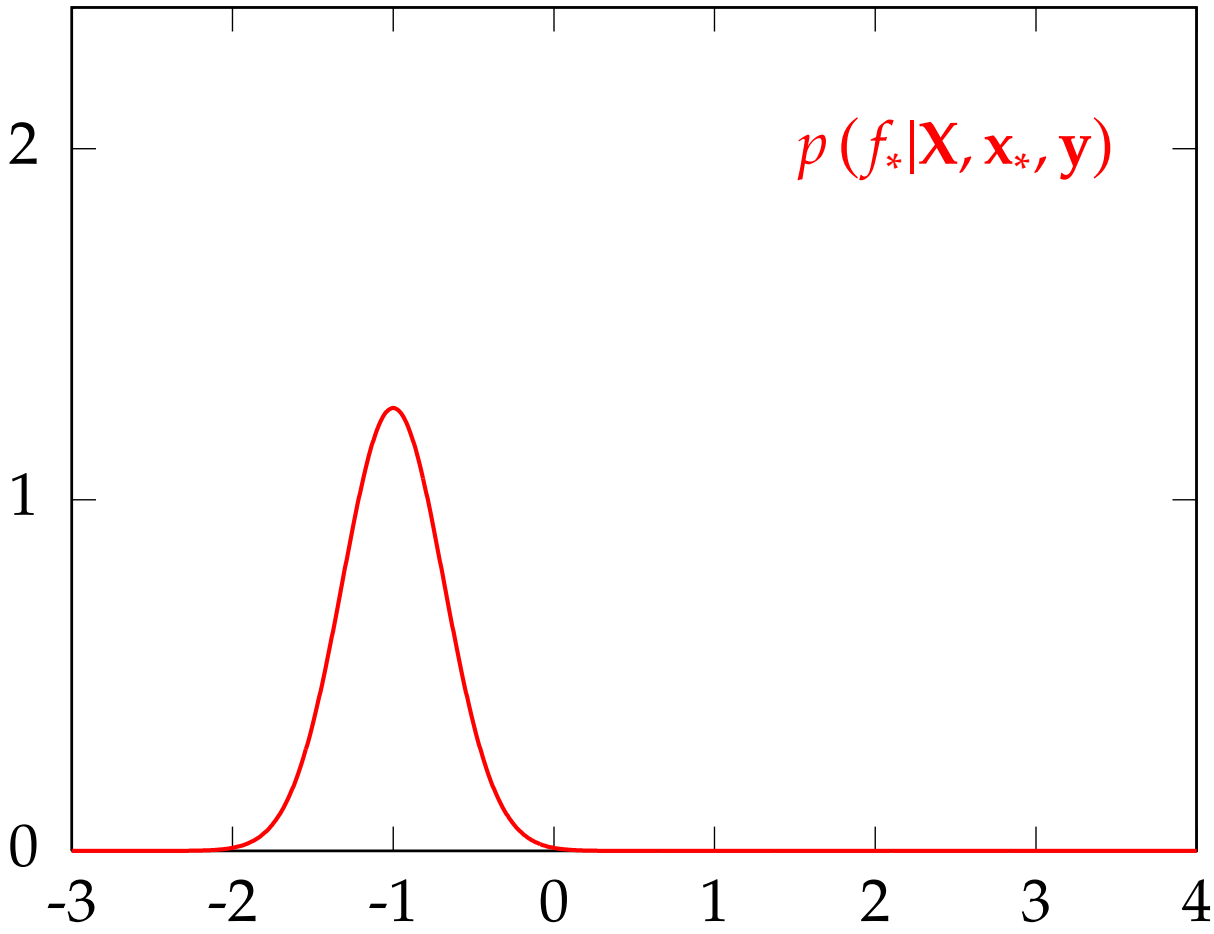


Figure : Inclusion of a data point with Gaussian noise.

Gaussian Noise

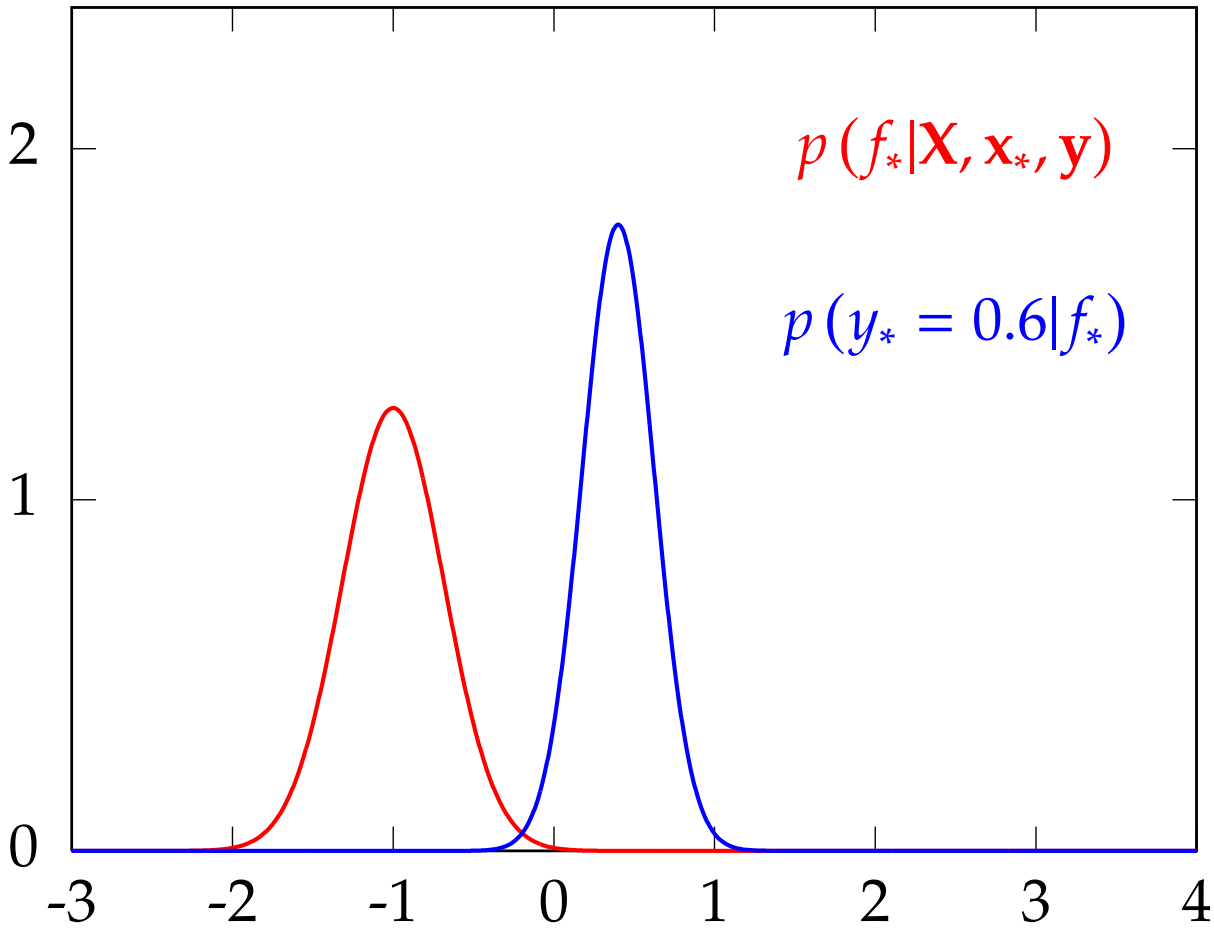


Figure : Inclusion of a data point with Gaussian noise.

Gaussian Noise

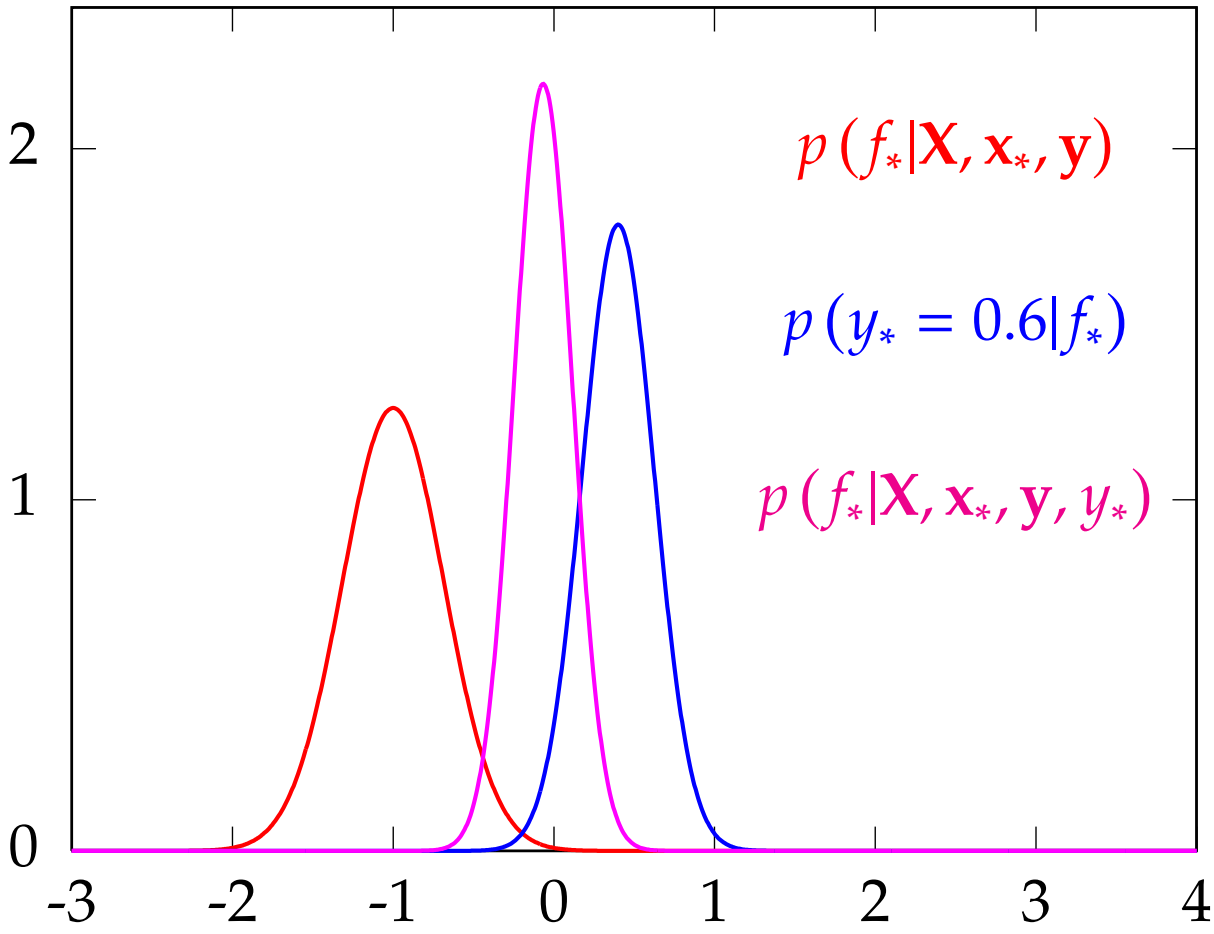


Figure : Inclusion of a data point with Gaussian noise.

Classification Noise Model

Probit Noise Model

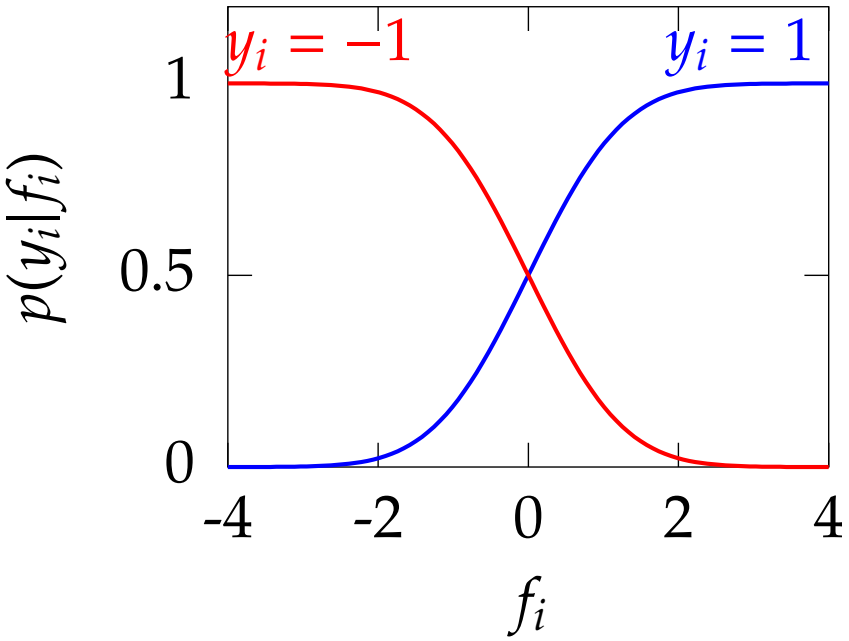


Figure : The probit model (classification). The plot shows $p(y_i|f_i)$ for different values of y_i. For $y_i = 1$ we have

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

Ordinal Noise Model

Ordered Categories

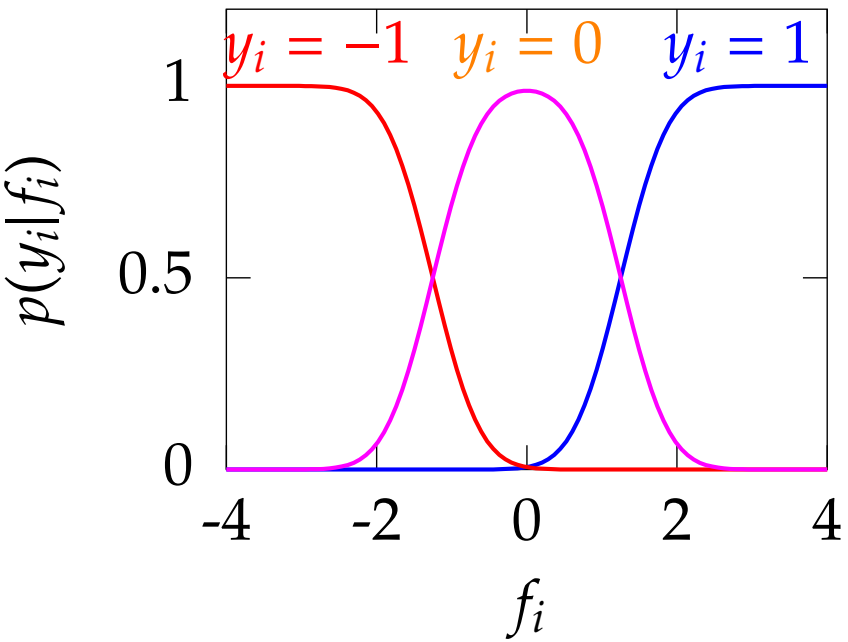


Figure : The ordered categorical noise model (ordinal regression). The plot shows $p(y_i|f_i)$ for different values of y_i. Here we have assumed three categories.

Null Category Noise Model

Classification with a Missing Category

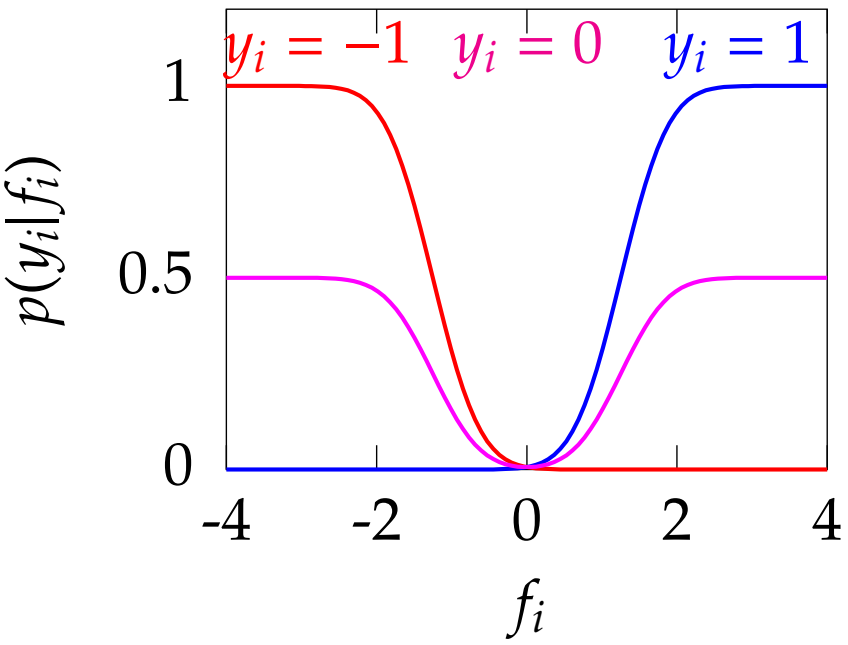


Figure : The null category noise model (semi-supervised learning). The plot shows $p(y_i|f_i)$ for different values of y_i. Here we have assumed three categories.

Non-linear Response Functions

- Non Gaussian likelihood:

$$p(y_i|f_i) = \Phi(f_i)$$

- Exact computation of the posterior is no longer possible analytically.

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{f}) \prod_{i=1}^{n} p(y_i|f_i)}{\int p(\mathbf{f}) \prod_{i=1}^{n} p(y_i|f_i) \mathrm{d}\mathbf{f}}$$

Link Functions

- Take the output of our function, $f(\cdot)$ use as:
  - Success probability in binomial distribution.
  - Rate function in Poisson likelihood.
  - shape parameter of Gamma distribution.
- Problem: $f(\cdot)$ defined over real line.
- Needs to be squashed down to 0-1 or constrained positive.

Link Functions

- Log link function, model the log rate.

$$\log \lambda(\mathbf{x}) = f(\mathbf{x})$$

- Logit link function, model the log odds.

$$\frac{\log \pi(\mathbf{x})}{\log(1 - \pi(\mathbf{x}))} = f(\mathbf{x})$$

Generative Model

- From a generative perspective we often naturally think of the inverse link:

$$\lambda(\mathbf{x}) = \exp(f(\mathbf{x}))$$

$$\pi(\mathbf{x}) = \frac{1}{1 + \exp(-f(\mathbf{x}))}$$

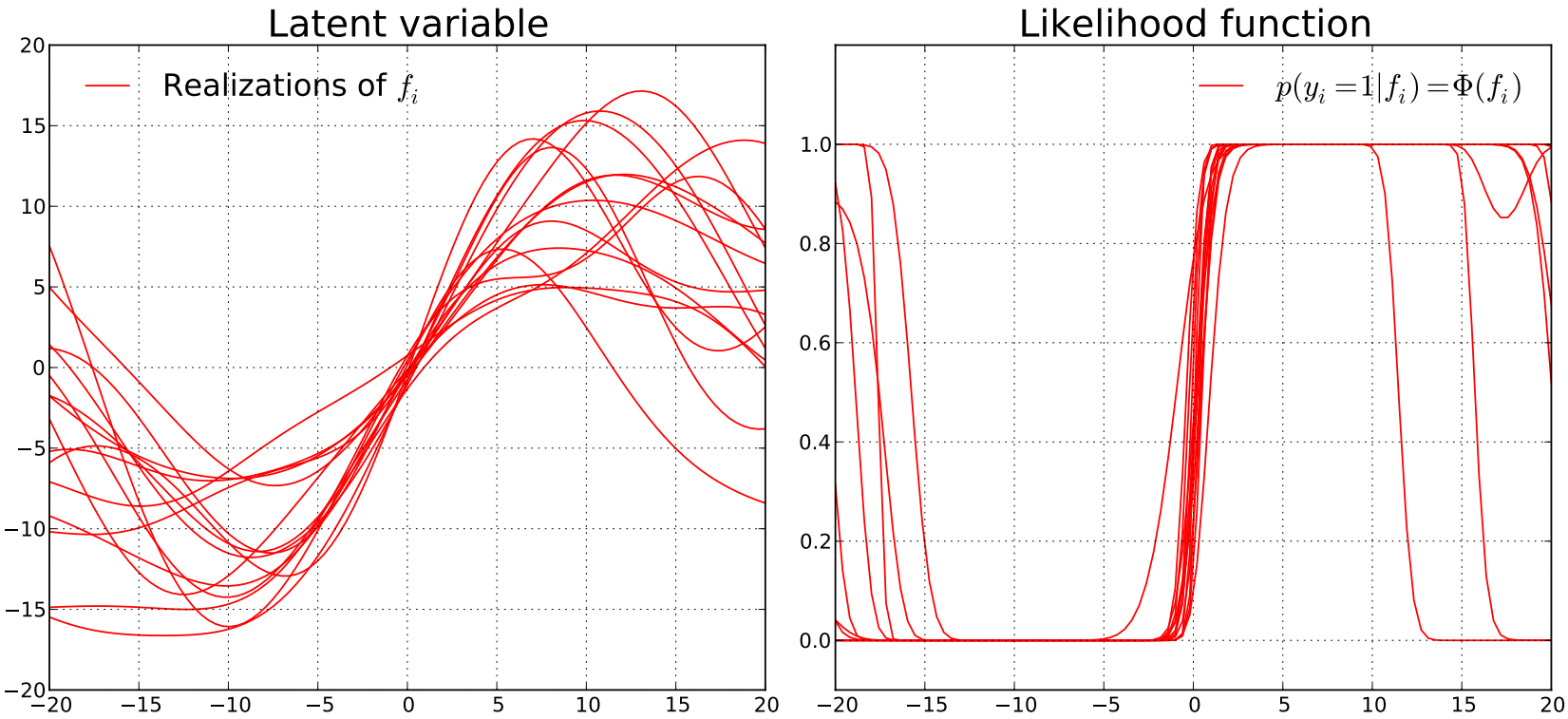
- Can make some assumptions of the link function clearer. For example log additive link function:

$$\log \lambda(\mathbf{x}) = f_1(\mathbf{x}) + f_2(\mathbf{x})$$

is a product of functions:

$$\lambda(\mathbf{x}) = \exp(f_1(\mathbf{x}))\exp(f_2(\mathbf{x}))$$

Example: Logit/Probit Link Function



Laplace Approximation

- Second order Taylor expansion at mode of log likelihood.
- First suggested by Laplace for his English dice example.
- How Laplace independently (of de Moivre) reinvented the Gaussian density.

Laplace Approximation

$$\log p(\mathbf{f}|\mathbf{y}) = \log p(\mathbf{y}|\mathbf{f}) + \log p(\mathbf{f}) + \text{const}$$

$$\log p(\mathbf{f}|\mathbf{y}) = \log p(\mathbf{y}|\mathbf{f}) - \frac{1}{2}\mathbf{f}^\top \mathbf{K}_{\mathbf{ff}}^{-1}\mathbf{f}$$

- Find MAP estimate $\hat{\mathbf{f}}$. This is mean of Gaussian approximation.
- Find Hessian of this system.
- Covariance of approximation is $-\mathbf{H}^{-1}$.

$$\mathbf{H} = \left(\frac{\mathrm{d}^2 \log p(\mathbf{y}|\mathbf{f})}{\mathrm{d}f_i \mathrm{d}f_j}\right)_{ij} - \mathbf{K}_{\mathbf{ff}}^{-1}$$

Expectation Propagation: General Case

- Exact (intractable) posterior:

$$p(\mathbf{f}|\mathbf{y}) = \frac{p(\mathbf{f}) \prod_{i=1}^{n} p(y_i|f_i)}{\int p(\mathbf{f}) \prod_{i=1}^{n} p(y_i|f_i) \mathrm{d}\mathbf{f}}$$

- EP posterior approximation:

$$q(\mathbf{f}|\mathbf{y}) = \frac{\prod_{i=1}^{K} t_i(f_i)}{Z_{EP}}$$

Expectation Propagation: Gaussian Approximation

Consider the special case:

$$p(y_i|f_i) \approx t_i(f_i) = Z_i \mathcal{N}\left(\tilde{\mu}_i|f_i, \tilde{\sigma}_i^2\right)$$

Here Z_i is a scaling factor so t_i is unnormalized.

If

$$p(\mathbf{f}) \sim \mathcal{N}\left(\mathbf{0}, \mathbf{K}_{\mathbf{f},\mathbf{f}}\right).$$

No approximation needed.

EP Posterior Approximation

$$q(\mathbf{f}|\mathbf{y}) = \frac{\prod_{i=1}^{n} t(f_i)p(\mathbf{f})}{Z_{EP}} = \mathcal{N}(\mathbf{f}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

Site functions provide "fake Gaussian observations" with target value $\hat{\mu}_i$ and observation variance $\hat{\sigma}_i^2$.

$$Z_{EP} = \prod_{i=1}^{n} Z_i \int \prod_{i=1}^{n} \mathcal{N}\left(\hat{\mu}_i | f_i, \hat{\sigma}_i^2\right) p(\mathbf{f}) \mathrm{d}\mathbf{f}$$

EP Posterior Approximation

$$q(\mathbf{f}|\mathbf{y}) = \frac{\prod_{i=1}^{n} Z_i \mathcal{N}\left(\hat{\mu}_i | f_i, \hat{\sigma}_i^2\right) p(\mathbf{f})}{Z_{EP}} = \mathcal{N}\left(\mathbf{f}|\boldsymbol{\mu}, \boldsymbol{\Sigma}\right)$$

Site functions provide "fake Gaussian observations" with target value $\hat{\mu}_i$ and observation variance $\hat{\sigma}_i^2$.

$$Z_{EP} = \prod_{i=1}^{n} Z_i \int \prod_{i=1}^{n} \mathcal{N}\left(\hat{\mu}_i | f_i, \hat{\sigma}_i^2\right) p(\mathbf{f}) \mathrm{d}\mathbf{f}$$

Site approximations

- Given initial site approximations: $t_j(f_j)$ for $j \neq i$.
- Need to set

$$t_i(f_i) \approx p(y_i|f_i)$$

$$p(y_i|f_i)p(\mathbf{f})\prod_{j\neq i} t_j(f_j) \approx p(\mathbf{f})\prod_{j=1}^{n} t_j(f_j)$$

$$p(y_i|f_i)\int p(\mathbf{f})\prod_{j\neq i} t_j(f_j)\mathrm{d}f_{j\neq i} \approx \int p(\mathbf{f})\prod_{j=1}^{n} t_j(f_j)\mathrm{d}f_{j\neq i}$$

$$p(y_i|f_i)q_{\setminus i}(f_i) \approx \mathcal{N}\left(f_i|\hat{\mu}_i, \hat{\sigma}_i^2\right)\hat{Z}_i$$

Cavity Distribution

$$q_{\setminus i}(f_i) = \frac{\prod_{j \neq i} t(f_j) p(\mathbf{f})}{\int \prod_{j \neq i} t(f_j) p(\mathbf{f})} \mathrm{d}\mathbf{f}$$

Tilted Distribution

$$\hat{p}_i(f_i|y_i) = \frac{p(y_i|f_i)q_{\setminus i}(f_i)}{\hat{Z}}$$

where

$$\hat{Z}_i = \int p(y_i|f_i)q_{\setminus i}(f_i)\mathrm{d}f_i$$

Minimization of the KL divergence

$$\hat{\mu}_i, \hat{\sigma}_i = \text{argmin}_{\hat{\mu}_i, \hat{\sigma}_i} \text{KL}\left(\frac{p(y_i|f_i) q_{\setminus i}(f_i)}{\hat{Z}} \| \mathcal{N}\left(f_i | \hat{\mu}_i, \hat{\sigma}_i^2\right) \right)$$

This is the KL between *tilted distribution* and *marginal of approximation*.
Since the approximation is Gaussian, KL is minimal when:

- $\hat{\mu}_i = \langle f_i \rangle_{p(y_i|f_i) q_{\setminus i}(f_i)}$
- $\hat{\sigma}_i^2 = \langle f_i \rangle^2_{p(y_i|f_i) q_{\setminus i}(f_i)} - \tilde{\mu}_i^2$

Scale of Site Approximation

- Since the approximation is un-normalized, we set scale as follows:

$$\hat{Z}_i = \int p(y_i|f_i) q_{\setminus i}(f_i) \mathrm{d}f_i$$

Classification Noise Model

Probit Noise Model

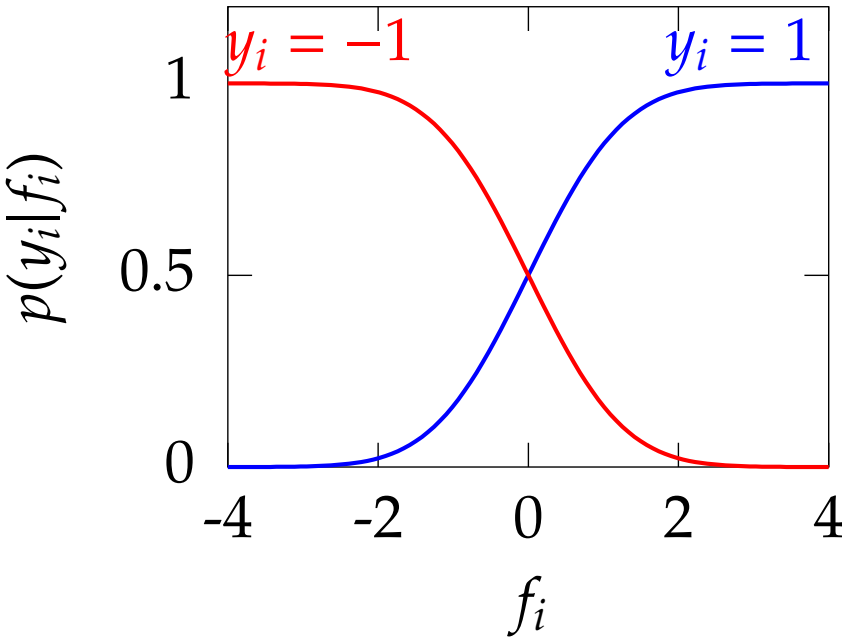


Figure : The probit model (classification). The plot shows $p(y_i|f_i)$ for different values of y_i. For $y_i = 1$ we have

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

Classification

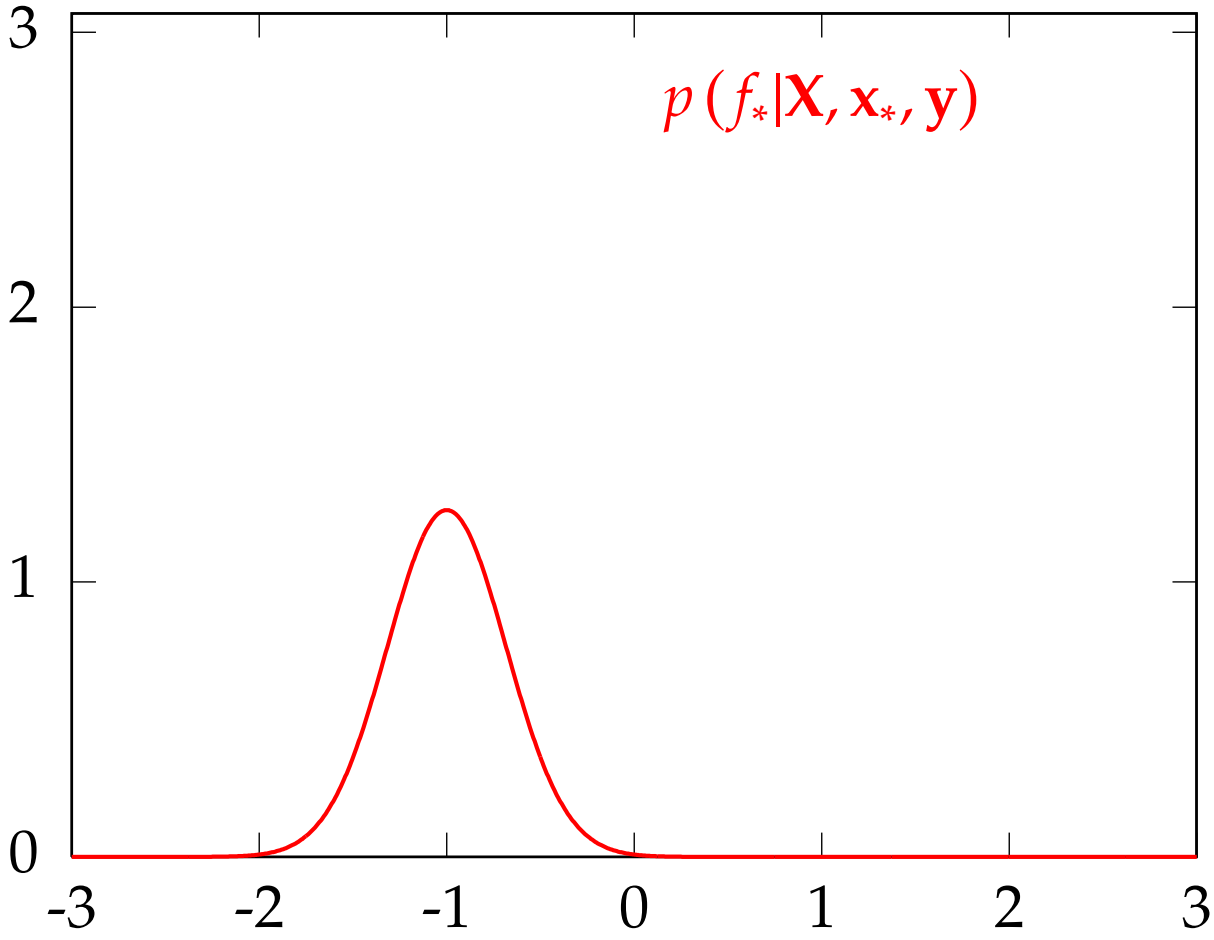


Figure : An EP style update with a classification noise model.

Classification

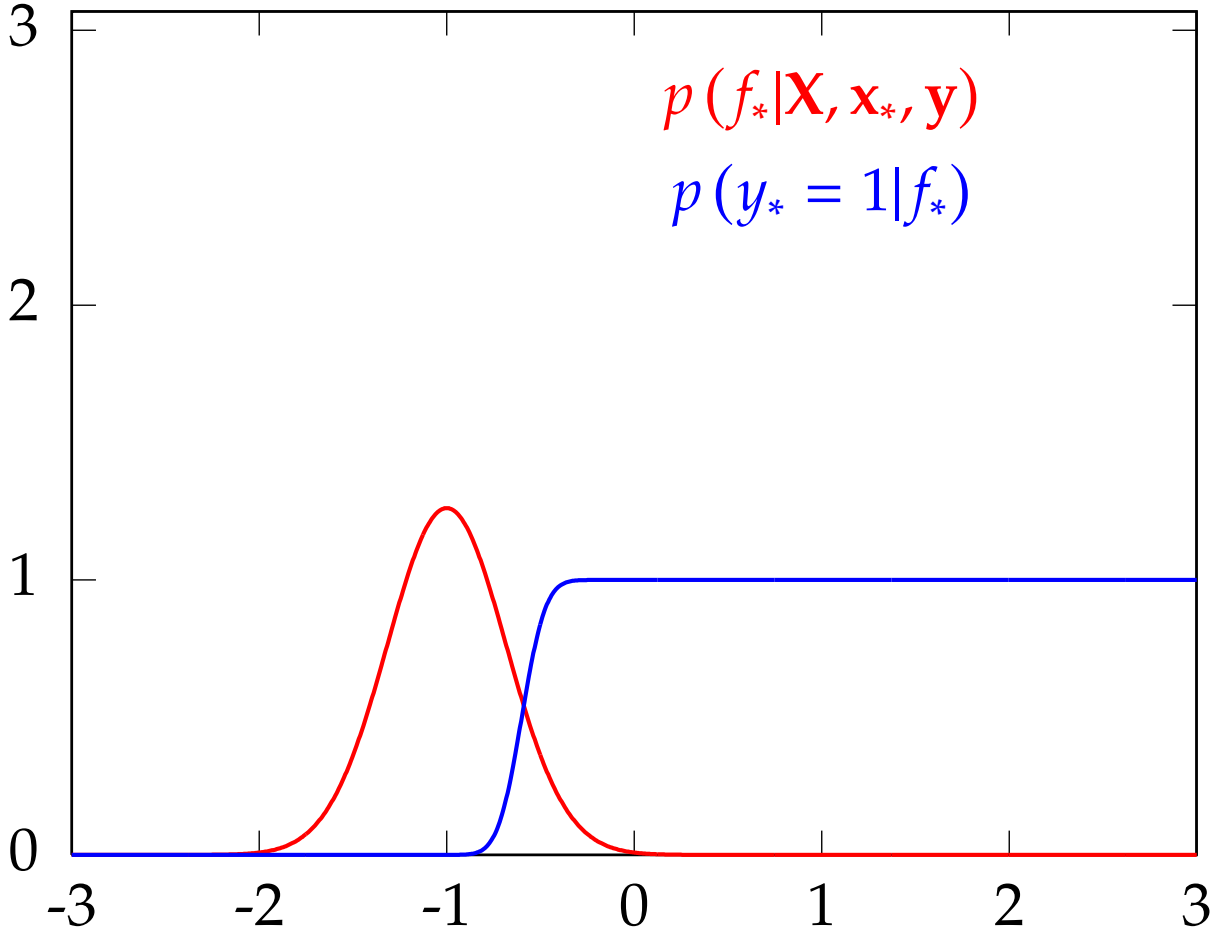


Figure : An EP style update with a classification noise model.

Classification

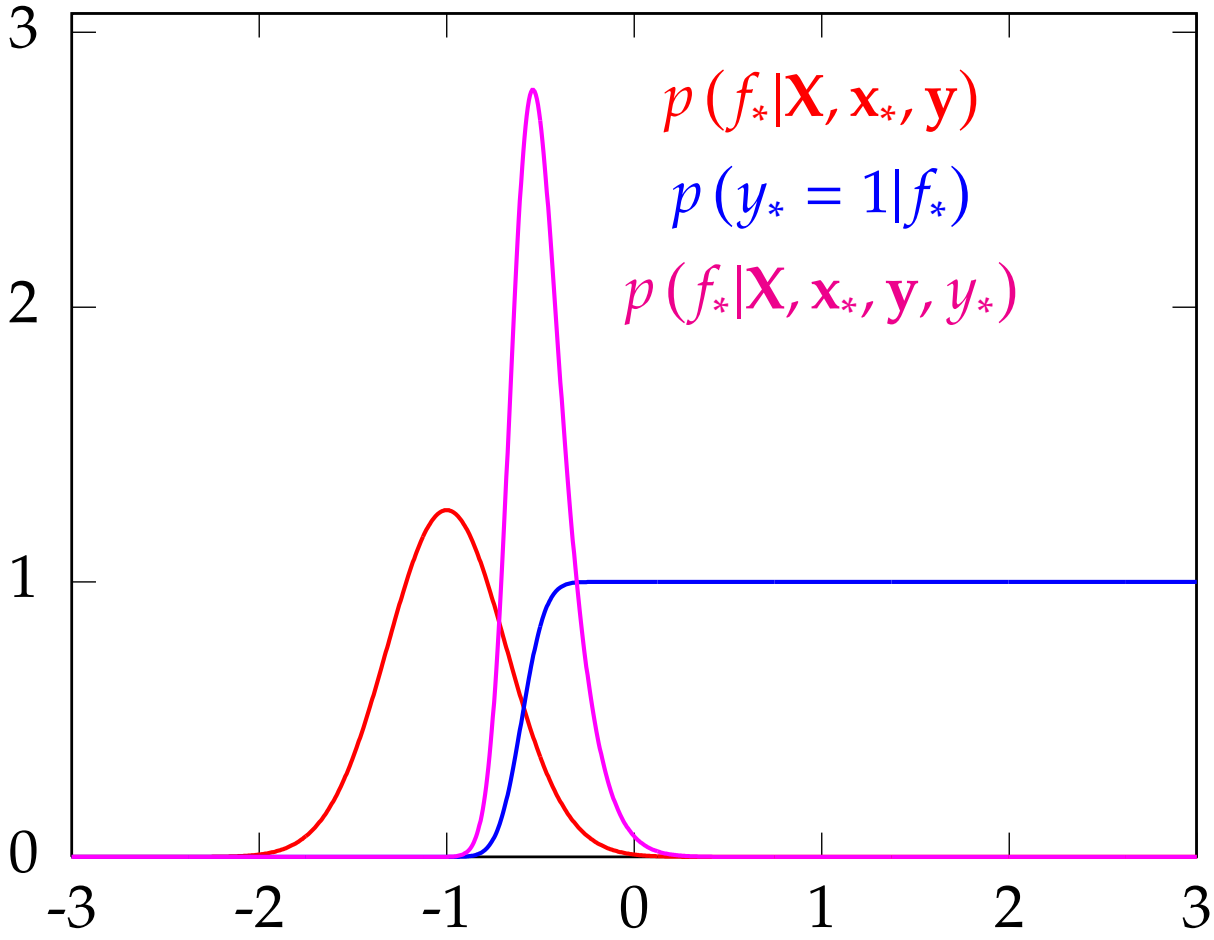


Figure : An EP style update with a classification noise model.

Classification

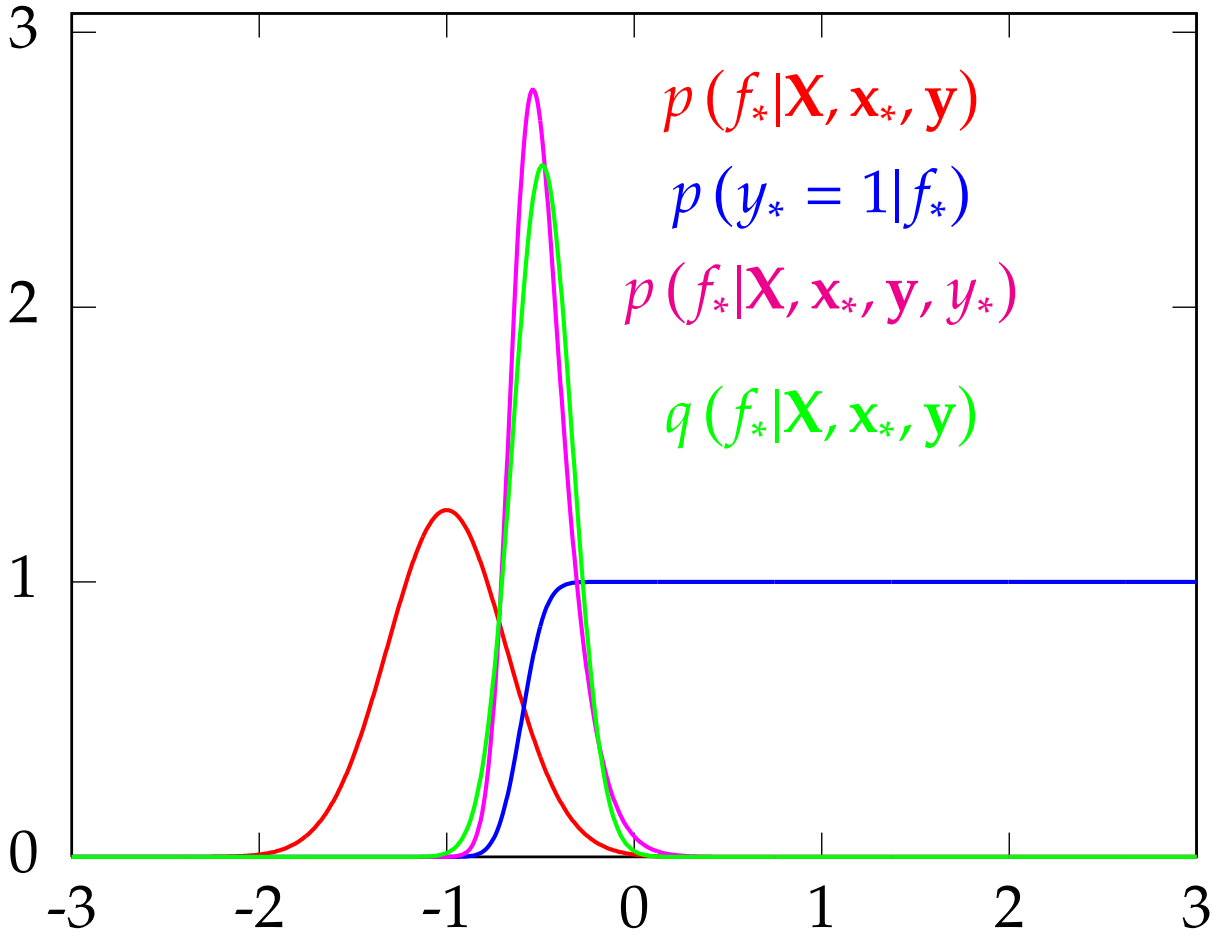


Figure : An EP style update with a classification noise model.

Ordinal Noise Model

Ordered Categories

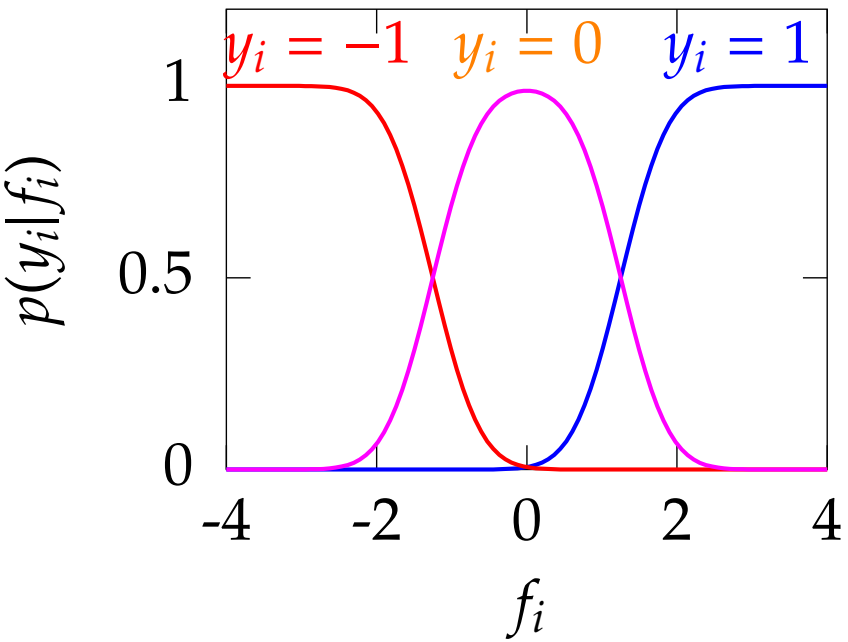


Figure : The ordered categorical noise model (ordinal regression). The plot shows $p(y_i|f_i)$ for different values of y_i. Here we have assumed three categories.

Ordinal Regression

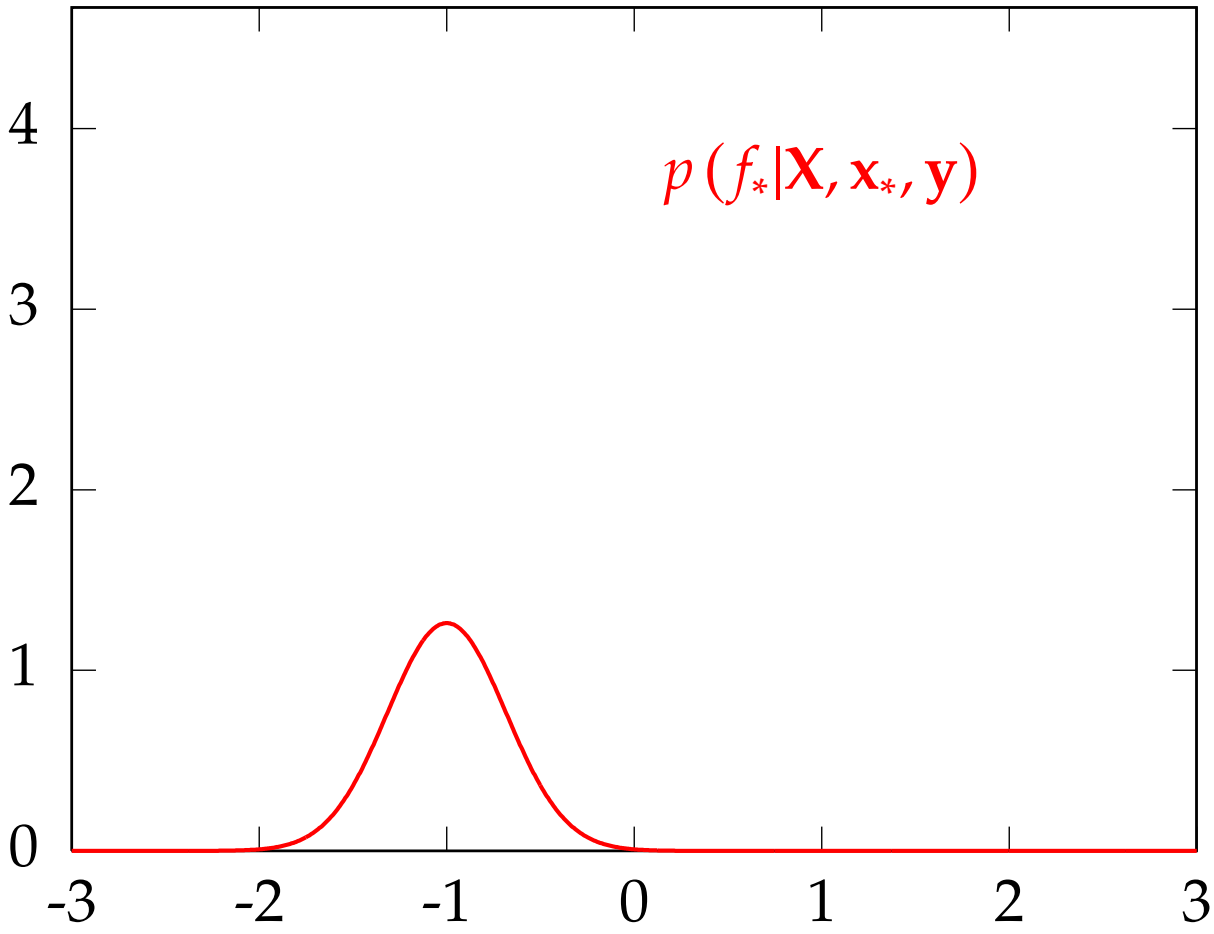


Figure : An EP style update with an ordered category noise model.

Ordinal Regression

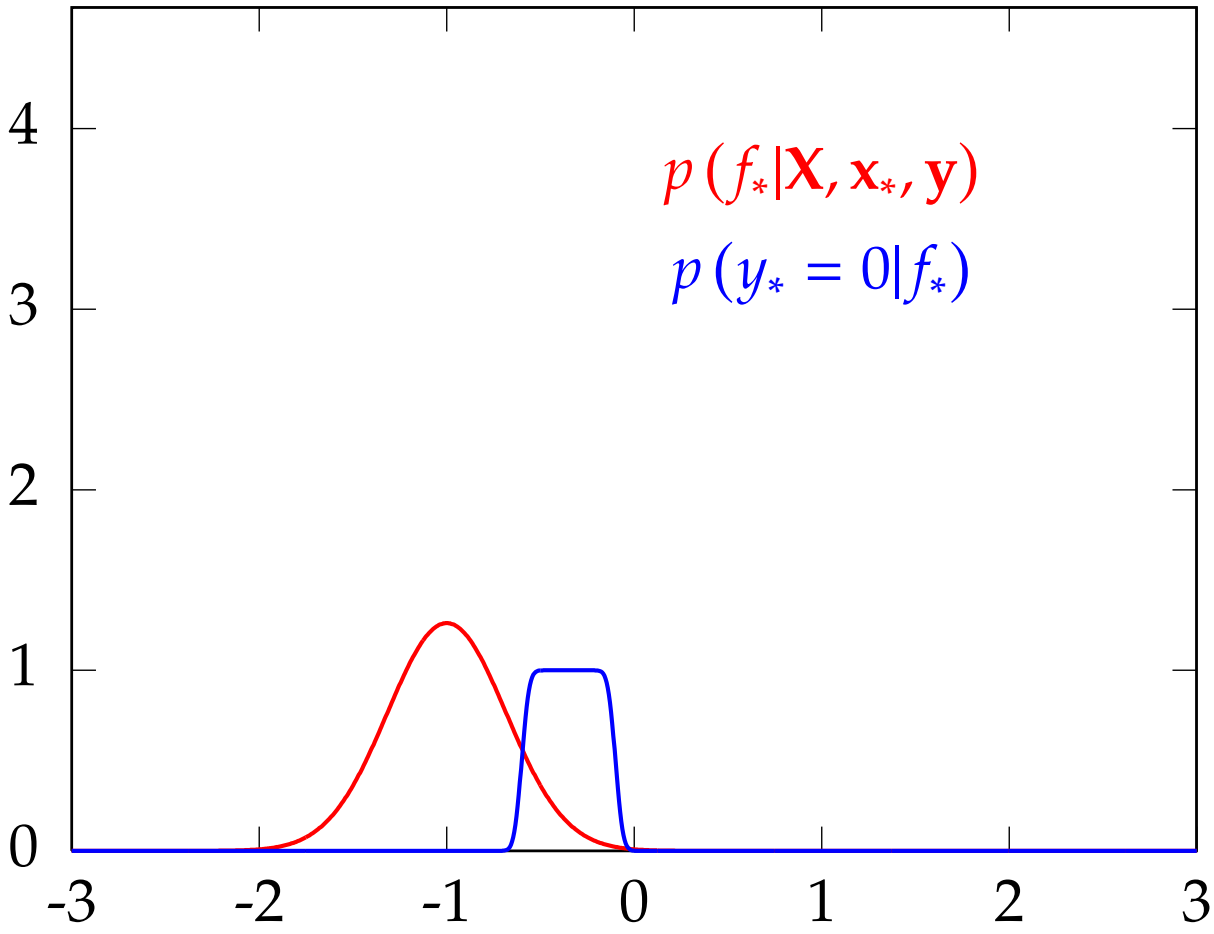


Figure : An EP style update with an ordered category noise model.

Ordinal Regression

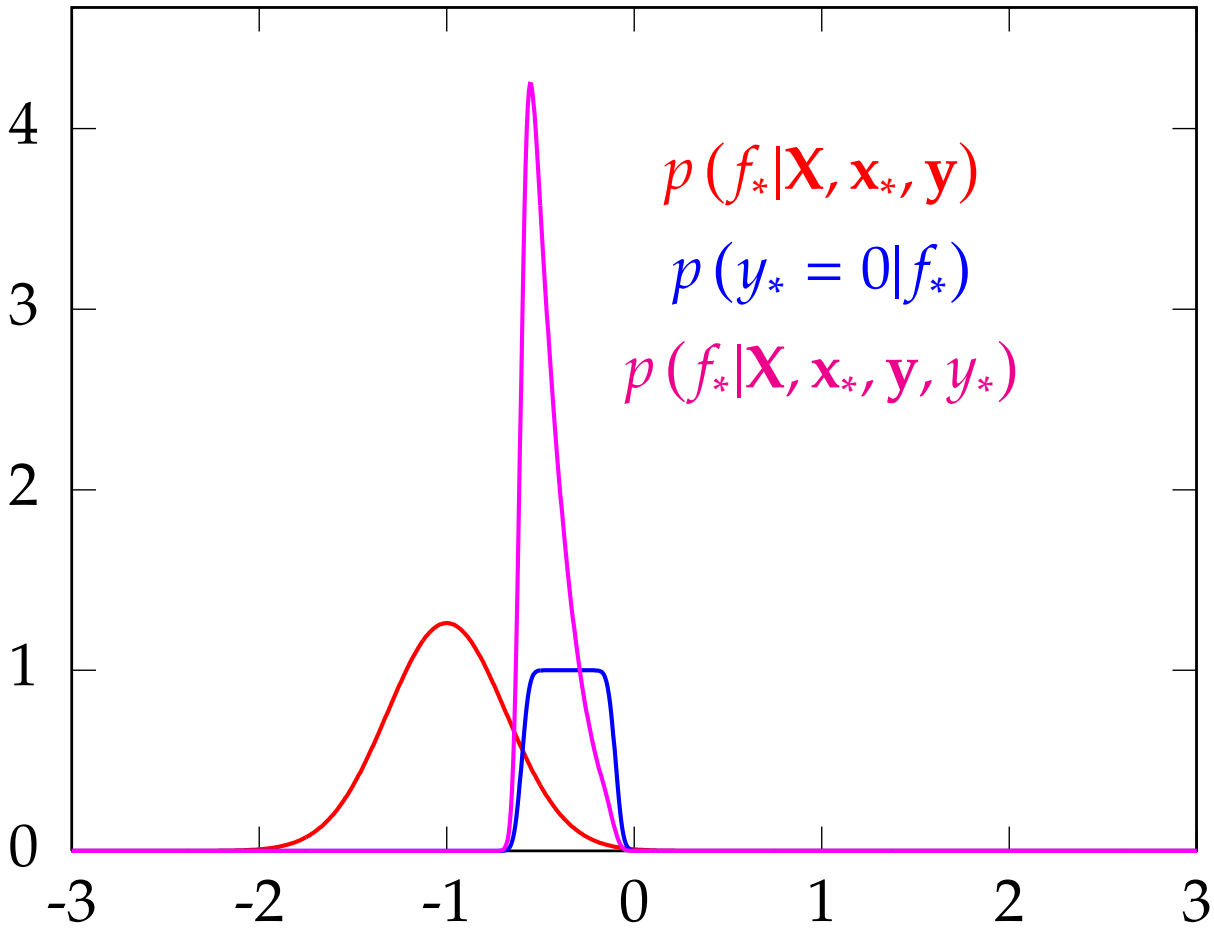


Figure : An EP style update with an ordered category noise model.

Ordinal Regression

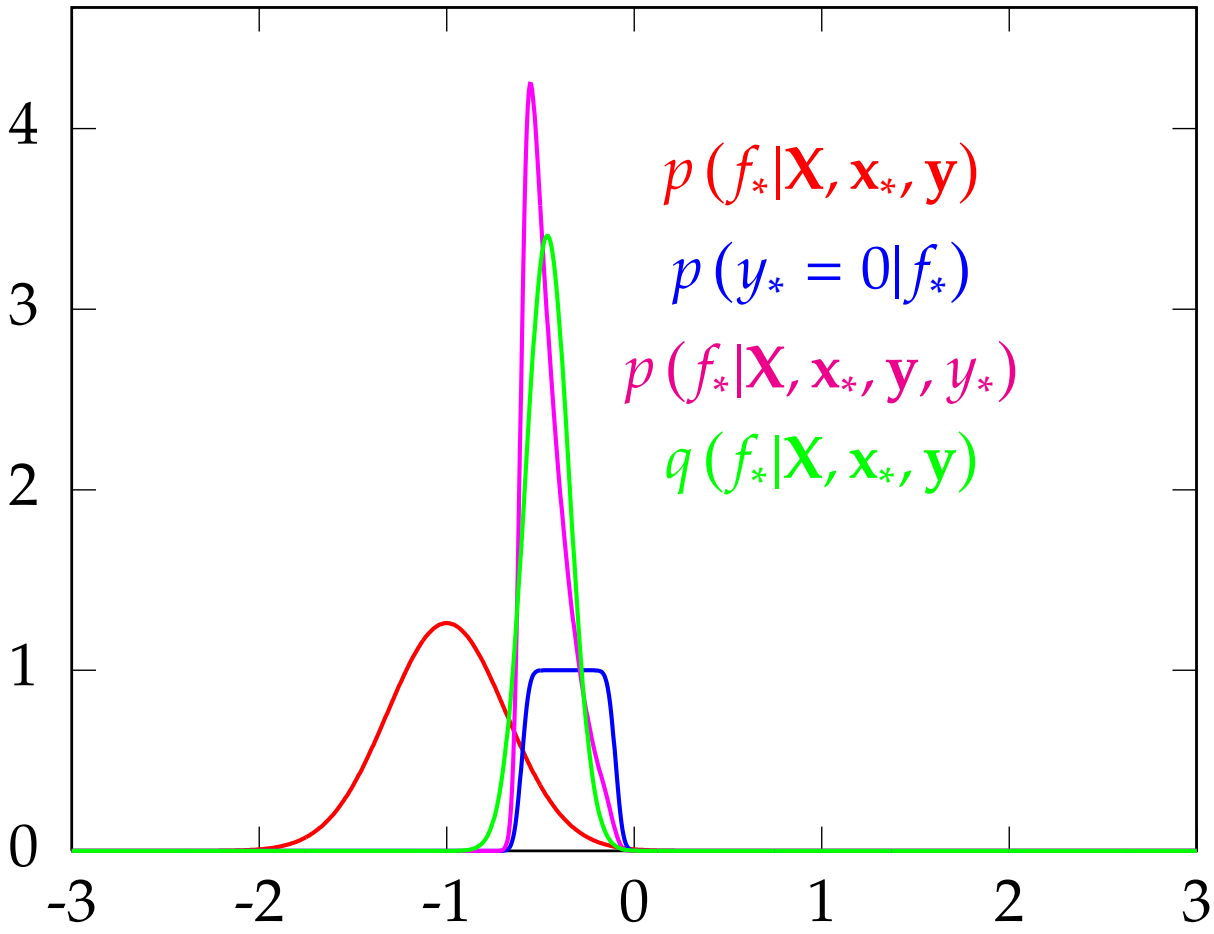


Figure : An EP style update with an ordered category noise model.

Null Category Noise Model

Classification with a Missing Category

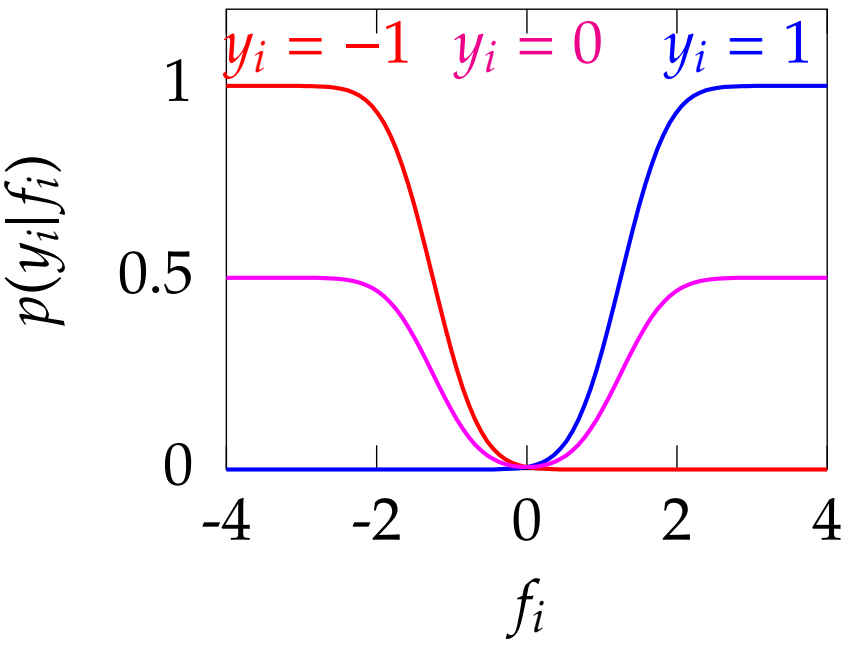


Figure : The null category noise model (semi-supervised learning). The plot shows $p(y_i|f_i)$ for different values of y_i. Here we have assumed three categories.

Semi-supervised Learning

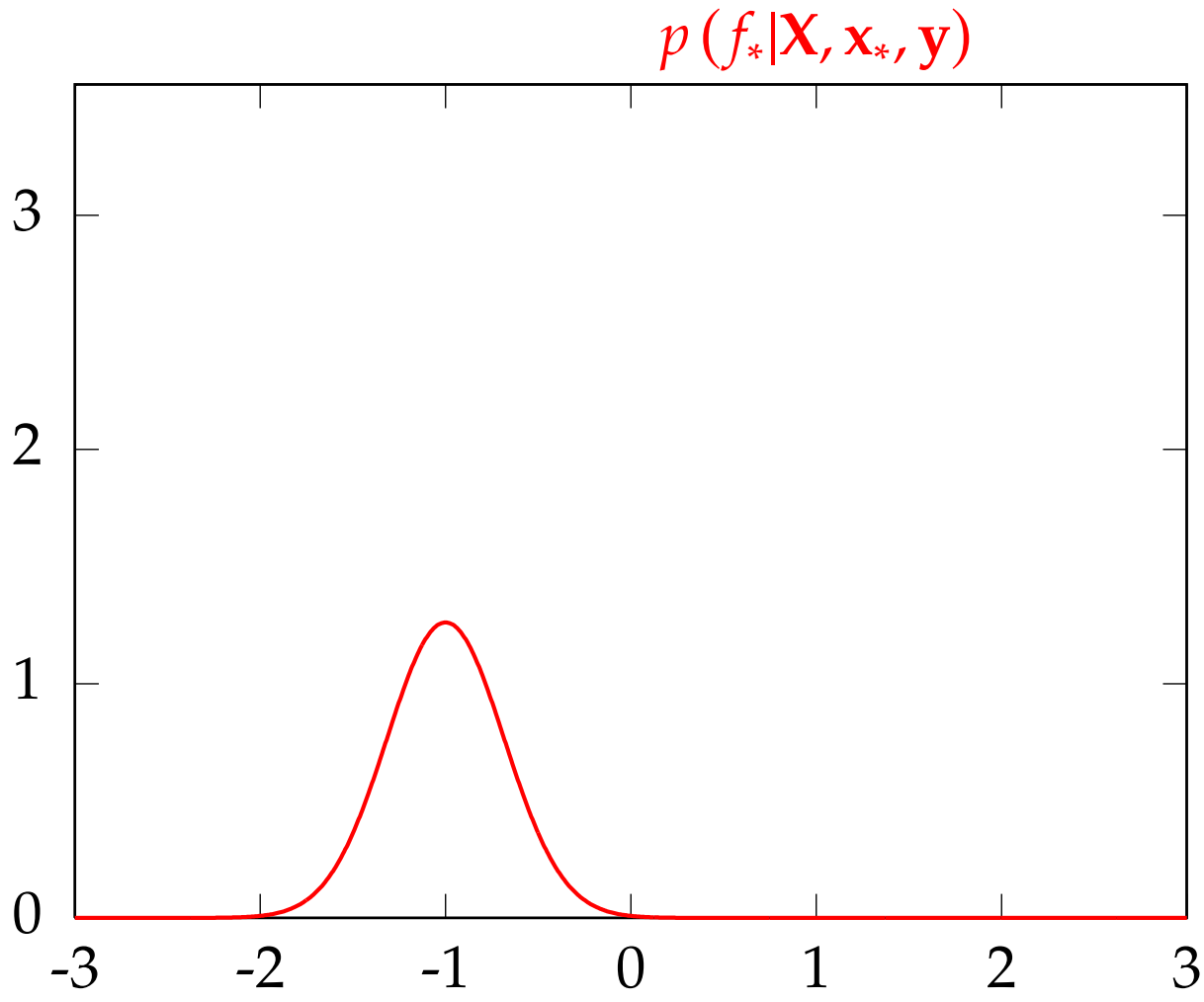


Figure : An EP style update with an null category noise model.

Semi-supervised Learning

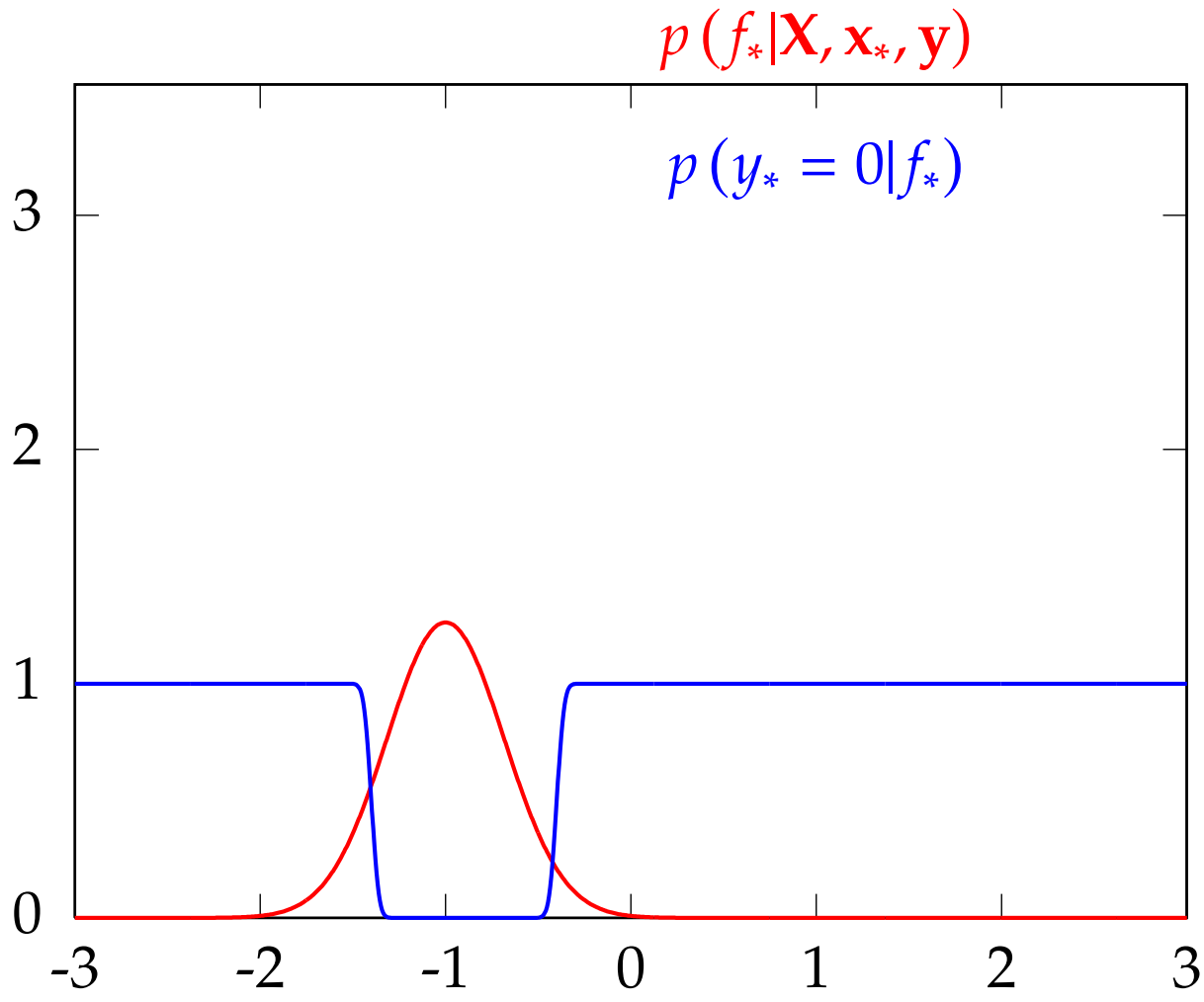


Figure : An EP style update with an null category noise model.

Semi-supervised Learning

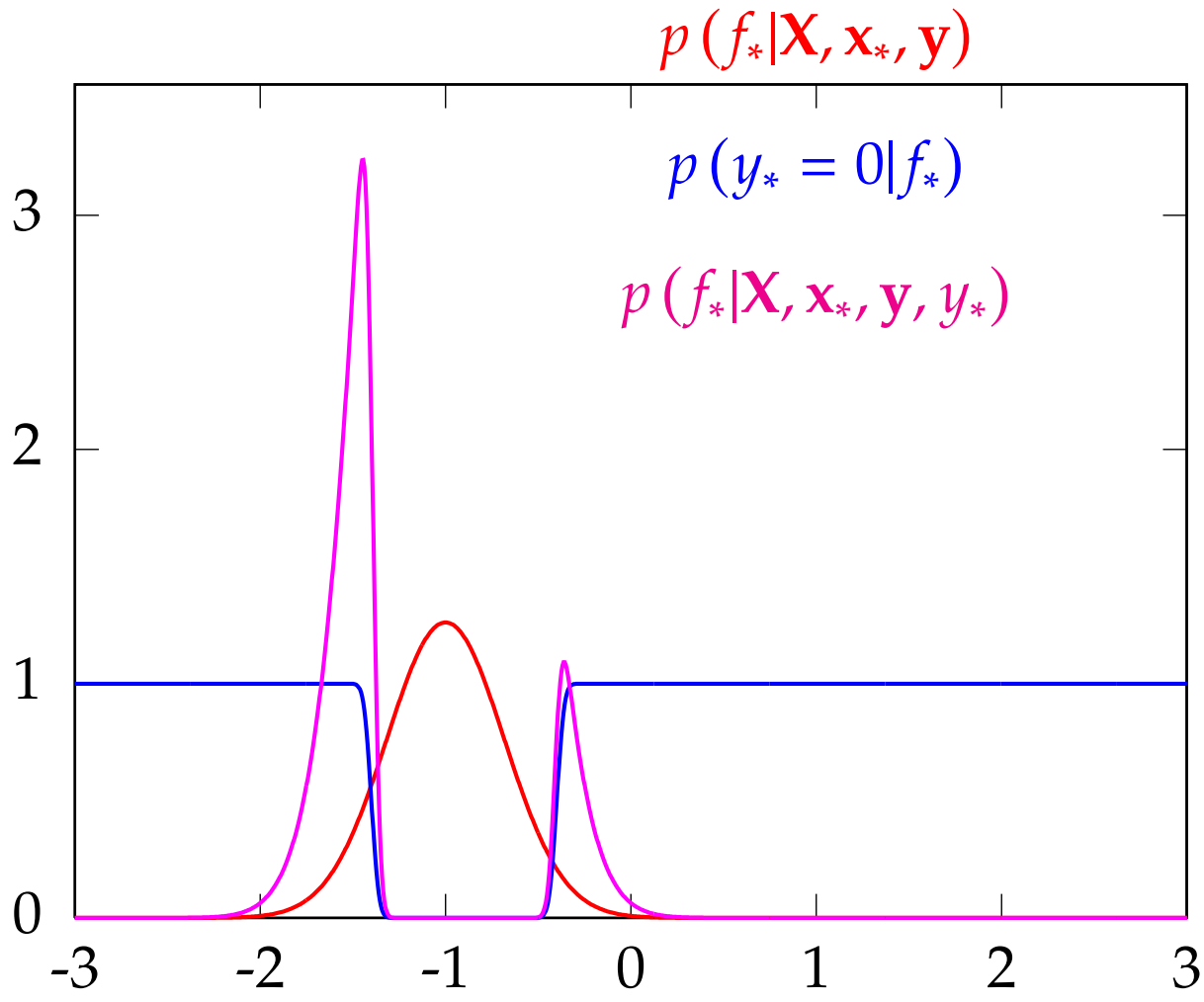


Figure : An EP style update with an null category noise model.

Semi-supervised Learning

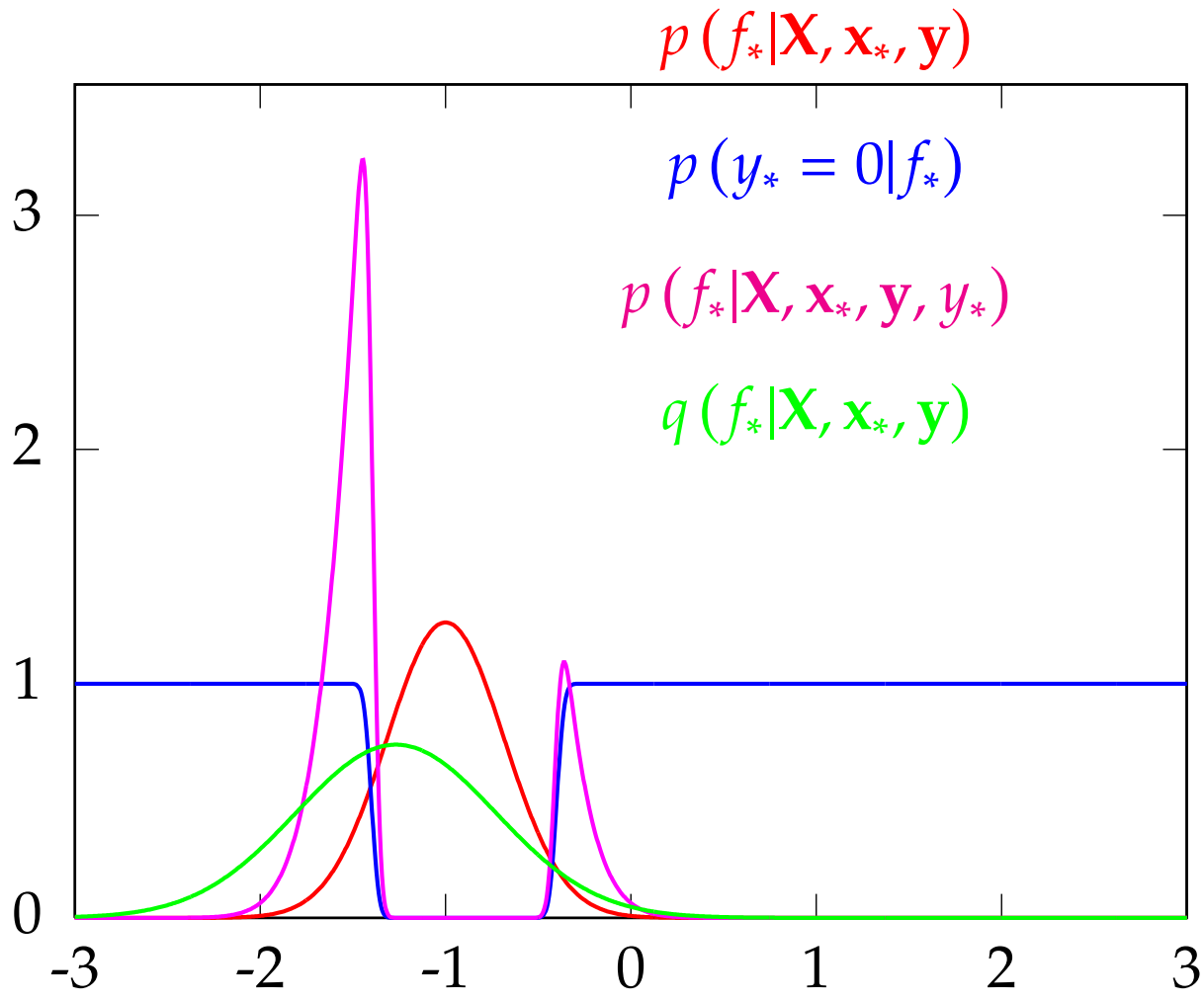


Figure : An EP style update with an null category noise model.

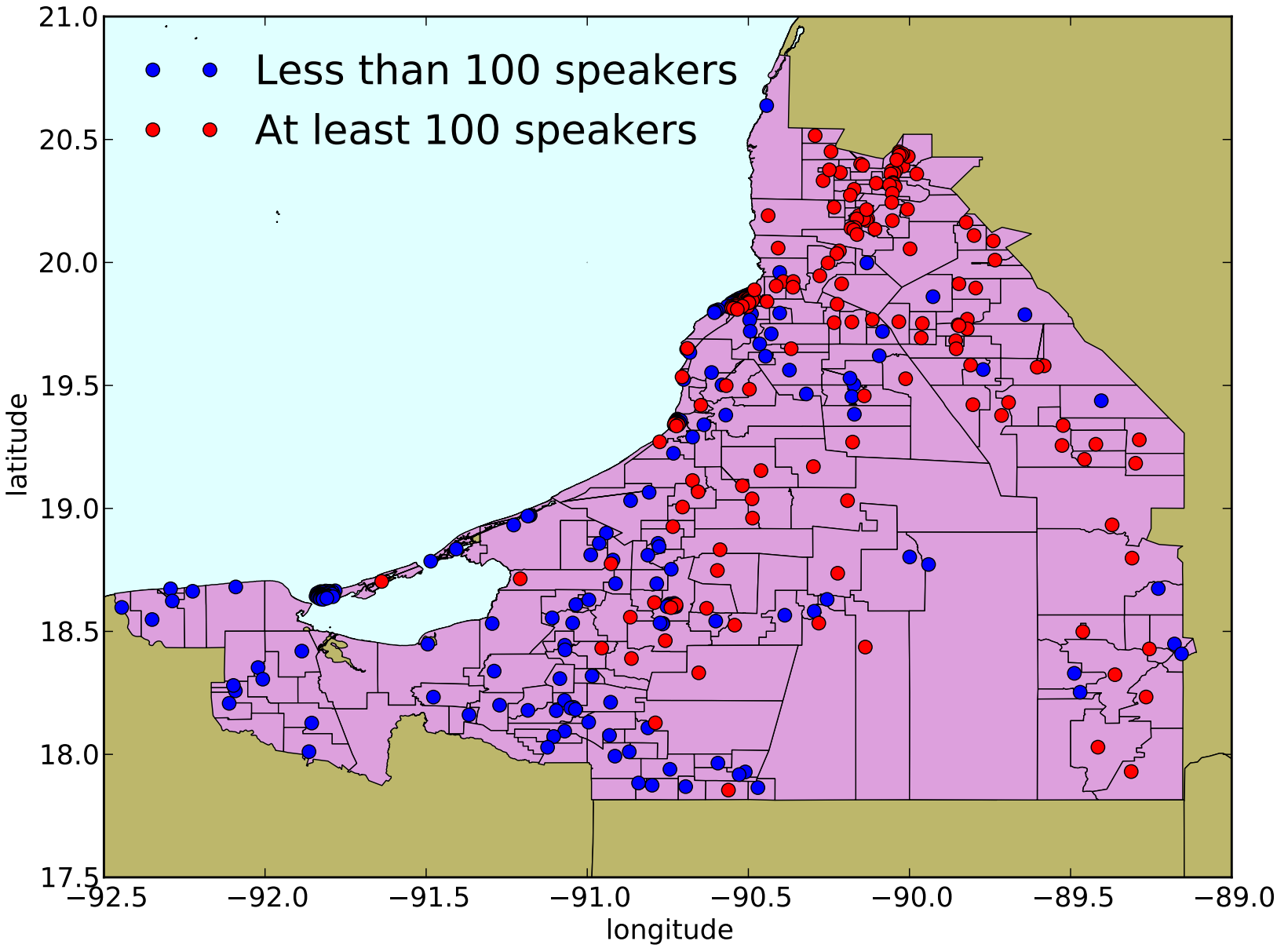
Predictions

- Predictive distribution of $q(f_*|\mathbf{y})$ is also Gaussian:

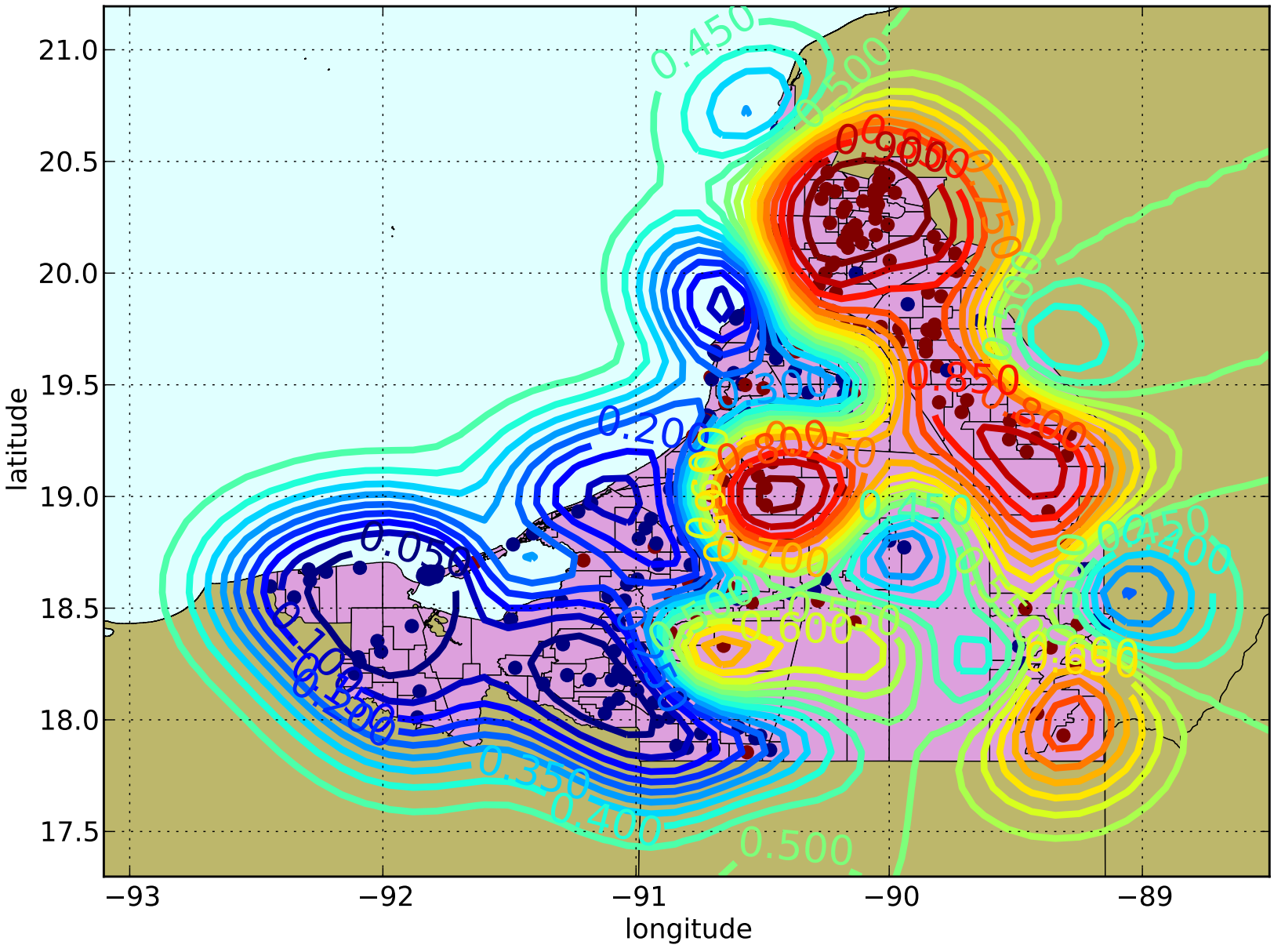
$$\langle f_* \rangle_{q(f_*|\mathbf{y})} = \mathbf{k}_*^\top \left(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \boldsymbol{\Sigma} t \right)^{-1} \tilde{\boldsymbol{\mu}}$$

$$\text{var}\left(f_*\right) = k_{*,*} - \mathbf{k}_*^\top \left(\mathbf{K}_{\mathbf{f},\mathbf{f}} + \boldsymbol{\Sigma}_t \right)^{-1} \mathbf{k}_*$$

Example: People who speak an indigenous language



Example: People who speak an indigenous language



Computational Complexity

- Major problem for Gaussian processes is the high computational complexity.
- $O(n^3)$ computation and $O(n^2)$ storage. For multioutput case $O(n^3p^3)$ computation and $O(n^2p^2)$ storage.
- Motivates sparse and low rank approximations.

The Informative Vector Machine

Reduce Complexity

- Including n data points through EP 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 m most informative points (Lawrence et al., 2003; Seeger, 2004; Lawrence et al., 2005).
- Computational complexity: $O\left(n^3\right)$ to $O\left(m^2 n\right)$.
- 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 jth inclusion of ith data point:

$$\begin{aligned}\Delta H_{j,i} &= -\frac{1}{2}\log\left|\Sigma_{j,i}\right| + \frac{1}{2}\log\left|\Sigma_{j-1}\right| \\ &= -\frac{1}{2}\log\left|\mathbf{I} - \Sigma_{j-1}\mathrm{diag}\left(\boldsymbol{\nu}_j\right)\right| \\ &= -\frac{1}{2}\log\left(1 - \nu_{j,i}\varsigma_{j-1,i}\right). \end{aligned} \tag{3}$$

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 β, 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

Classification

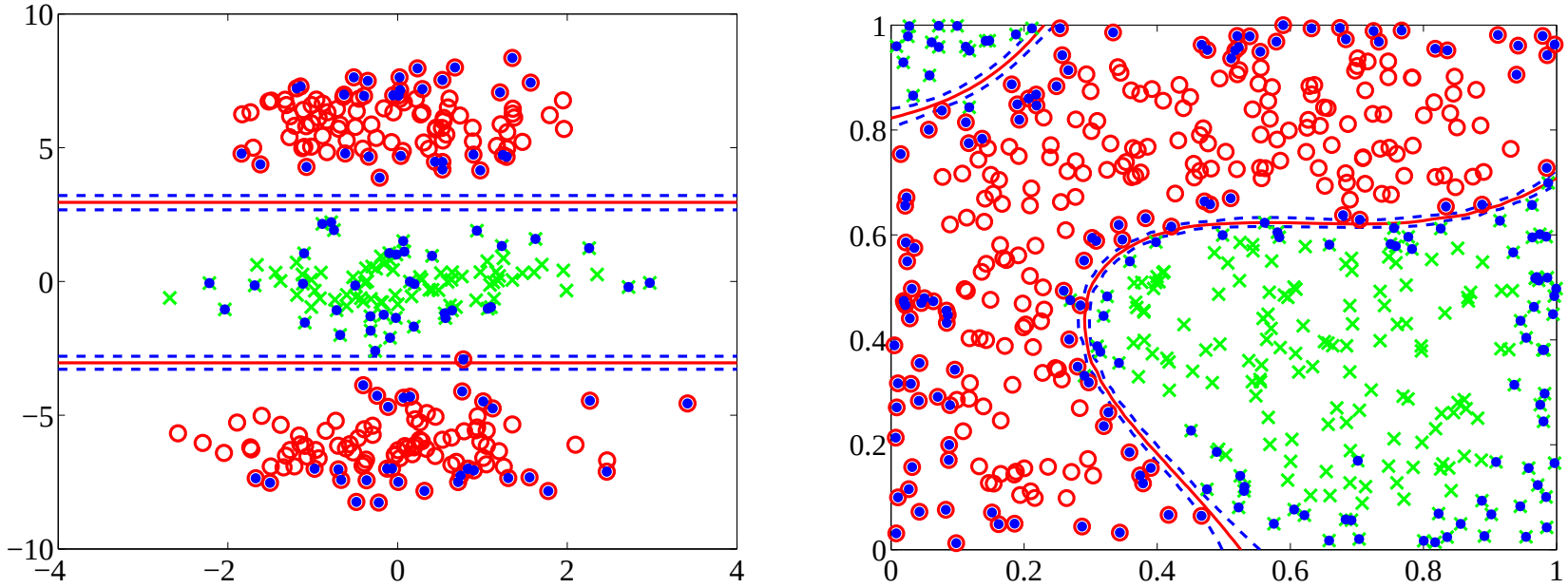


Figure : *Contours*: Red solid line at $p(y|\mathbf{x}) = 0.5$, blue dashed lines at $p(y|\mathbf{x}) = 0.25$ and $p(y|\mathbf{x}) = 0.75$. Active points are blue dots. *Left*: data sampled from from a mixture of Gaussians. *Right*: Data uniformly sampled on the 2–dimensional unit square. Class labels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

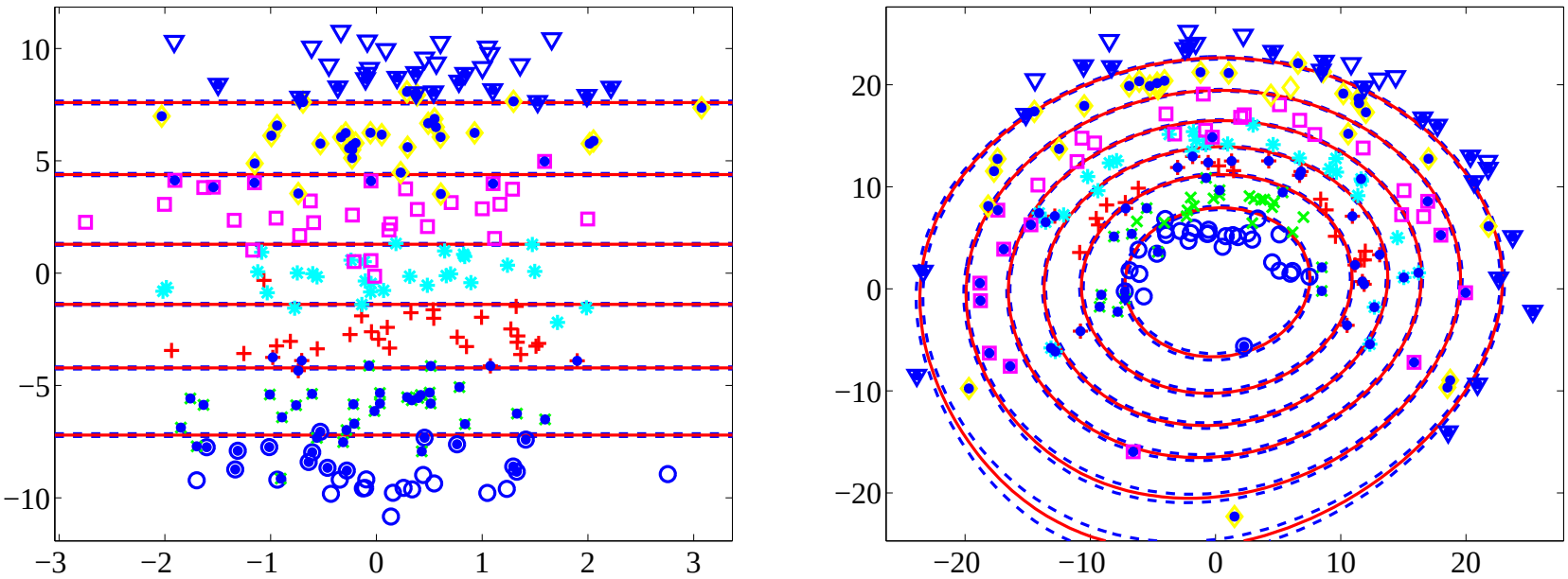


Figure : .*Left*: a linear solution is found. *Right*: this categories in this example were sampled in polar co-ordinates.

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 use 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 $m = 500$.

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 (Lawrence et al., 2005)

- 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 $m = 1,000$ 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.

Posterior variance update

- Complexity is dominated by the computation of the posterior covariance:

$$\boldsymbol{\Sigma} = \left(\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1} + \boldsymbol{\Sigma}_t^{-1}\right)^{-1}$$

Sparse EP

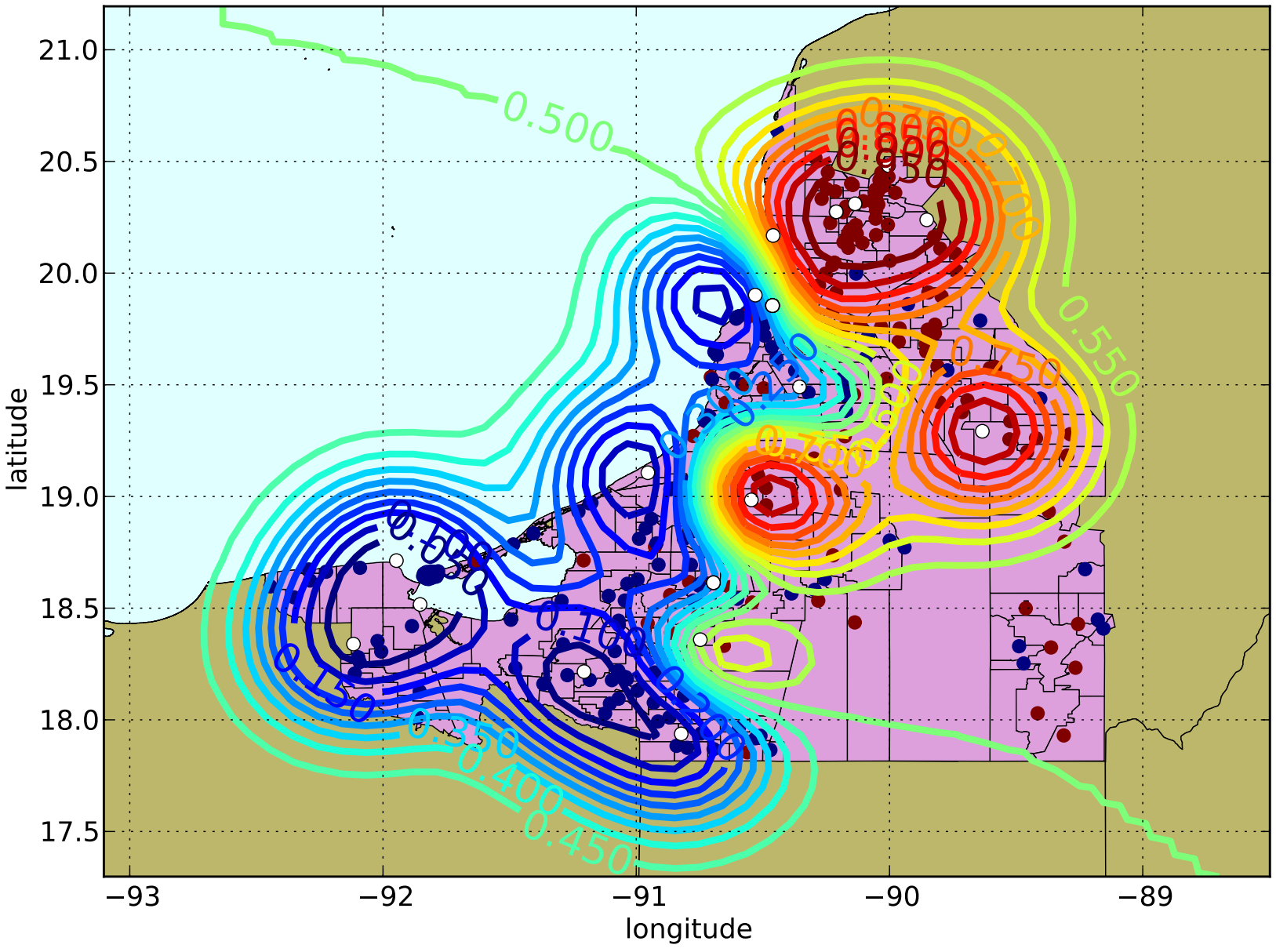
- $q(\mathbf{f}|\mathbf{y})$ is computed as before, but an sparse approximation is used instead of the exact covariance $\mathbf{K_{f,f}}$.
- FITC approximation: $O(nm^2)$

$$\mathbf{K_{f,f}} \approx \mathbf{K_{f,u}}\mathbf{K_{u,u}^{-1}}\mathbf{K_{u,f}} + \text{diag}\left(\mathbf{K_{f,f}} - \mathbf{Q_{f,f}}\right)$$

- DTC approximation: $O(nm^2)$

$$\mathbf{K_{f,f}} \approx \mathbf{K_{f,u}}\mathbf{K_{u,u}^{-1}}\mathbf{K_{u,f}}$$

EP-FITC (generalized FITC)



EP-DTC

Compatible with sparse variational approach:

$$\mathcal{L} = \log \mathcal{N}\left(\boldsymbol{\mu}_t | \mathbf{0}, \mathbf{Q}_{\mathbf{f},\mathbf{f}} + \boldsymbol{\Sigma}_t\right) - \frac{1}{2}\mathrm{tr}\left((\mathbf{K}_{\mathbf{f},\mathbf{f}} - \mathbf{Q}_{\mathbf{f},\mathbf{f}})\boldsymbol{\Sigma}_{t_i}\right) - Z_{EP}$$

Sparse variational + EP-DTC