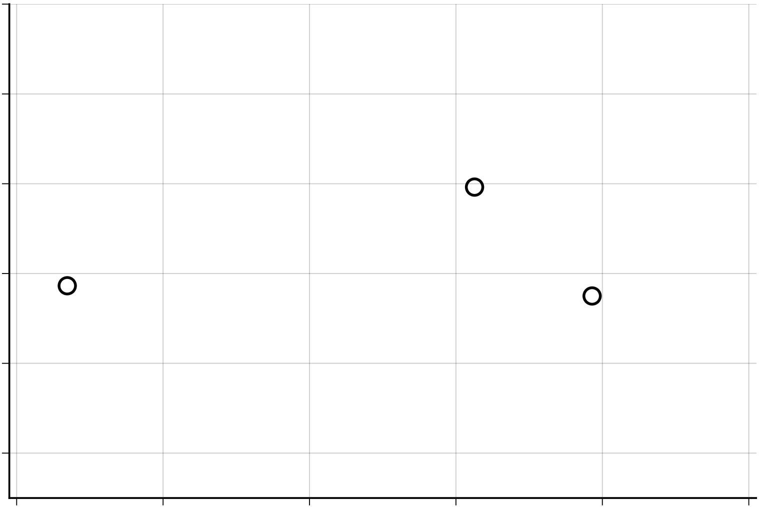
A tutorial on Gaussian Processes

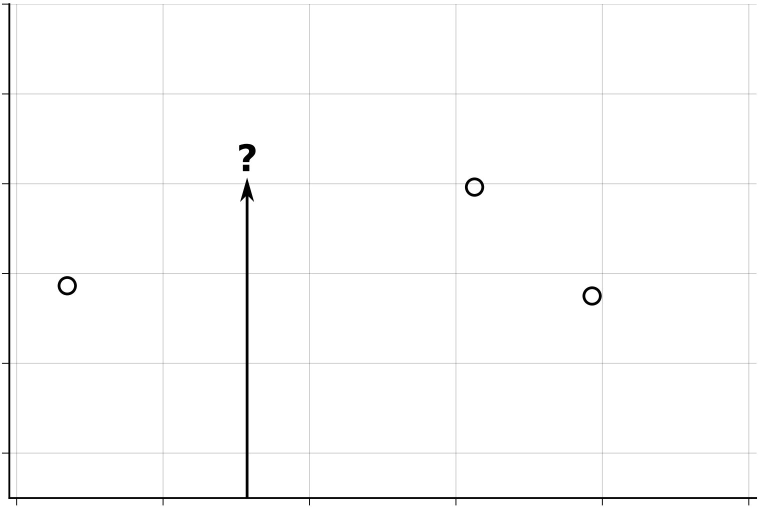
Daniel Hernández–Lobato
Computer Science Department
Universidad Autónoma de Madrid

http://dhnzl.org, daniel.hernandez@uam.es

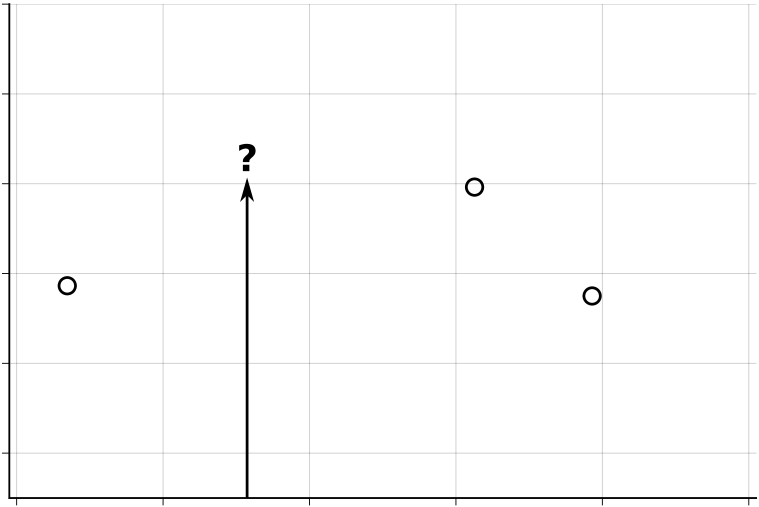
Motivation: Non-linear Regression



Motivation: Non-linear Regression

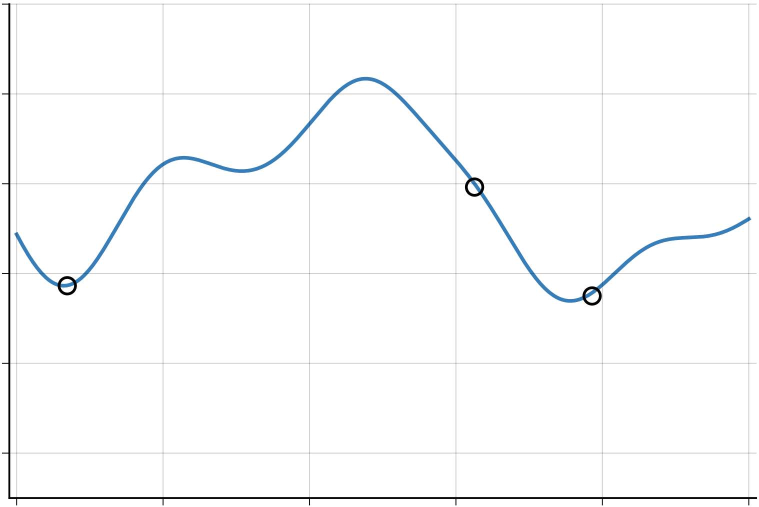


Motivation: Non-linear Regression



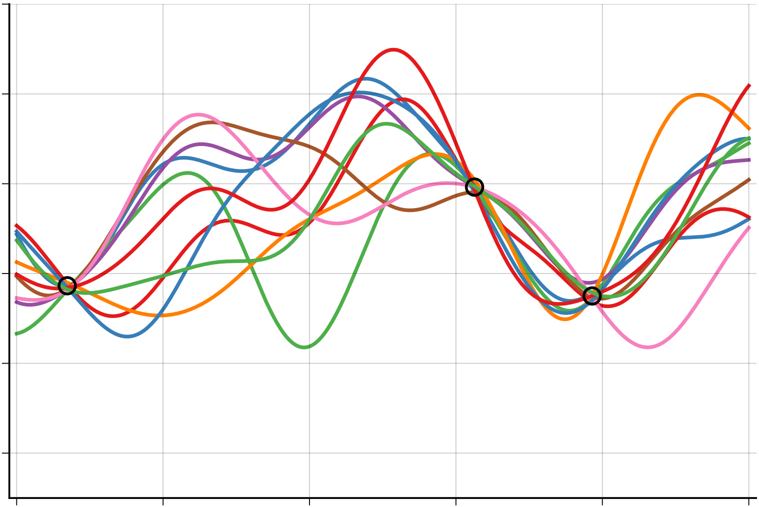
We have to specify a model that may depend on parameters W.

Motivation: Non-linear Regression



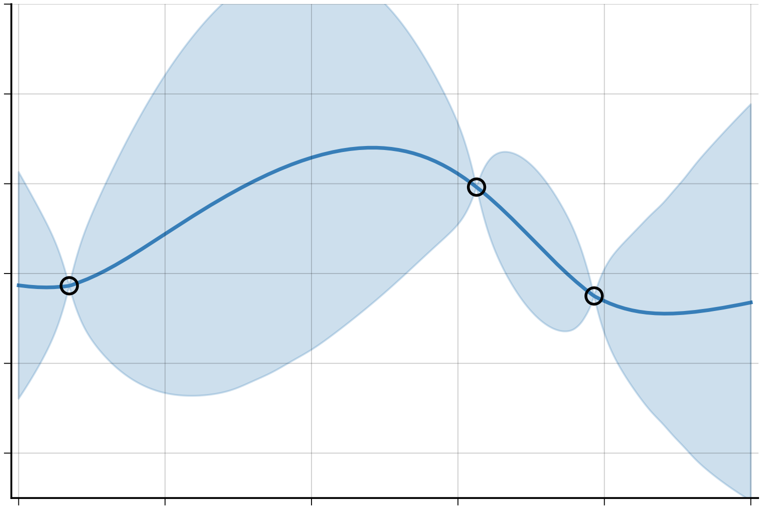
Given W the model will output a prediction.

Motivation: Non-linear Regression



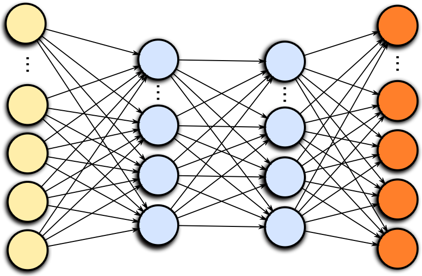
Many values for W can be compatible with the data!

Motivation: Non-linear Regression



We are interested in a predictive distribution!

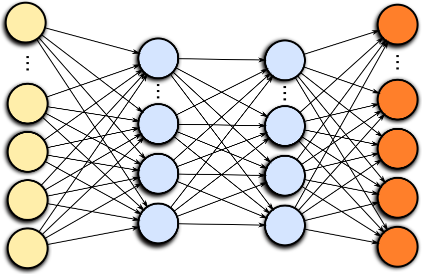
Computation of the Posterior Distribution



$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

Computation of the Posterior Distribution



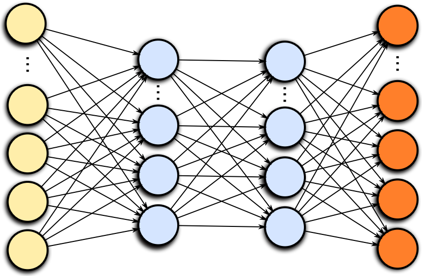
$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

The posterior distribution of **W** is:

$$p(\mathbf{W}|\mathbf{y}, \mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{W}, \mathbf{X})p(\mathbf{W})}{p(\mathbf{y}|\mathbf{X})}\,, \qquad p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{W}, \mathbf{X})p(\mathbf{W})d\mathbf{W}\,,$$

Computation of the Posterior Distribution



$$h_j(\mathbf{x}) = \tanh\left(\sum_{i=1}^{I} x_i w_{ji}\right)$$

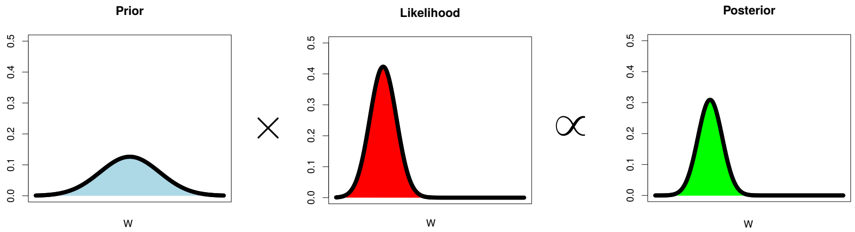
$$f(\mathbf{x}) = \sum_{j=1}^{H} v_j h_j(\mathbf{x})$$

The posterior distribution of $\mathbf{W}$ is:

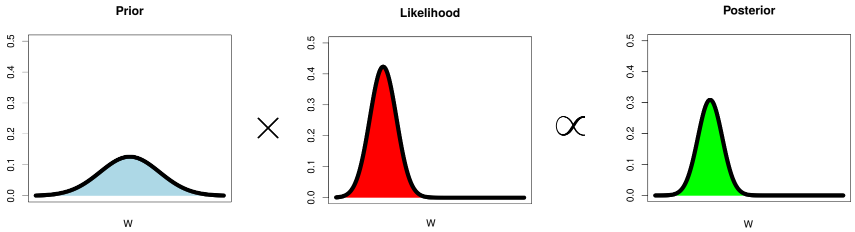
$$p(\mathbf{W}|\mathbf{y},\mathbf{X}) = \frac{p(\mathbf{y}|\mathbf{W},\mathbf{X})p(\mathbf{W})}{p(\mathbf{y}|\mathbf{X})}\,, \qquad p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{W},\mathbf{X})p(\mathbf{W})d\mathbf{W}\,,$$

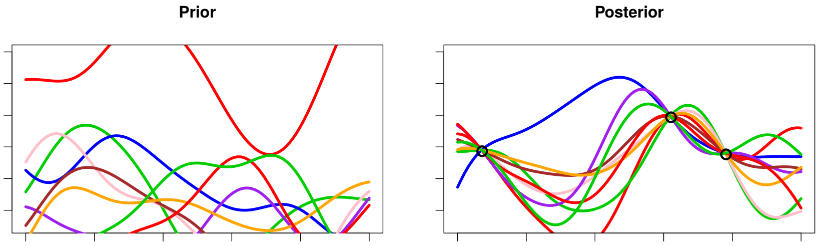
The posterior captures the values of W compatible with y and X.

Computation of the Posterior Distribution



Computation of the Posterior Distribution





Computation of the Predictive Distribution

The predictive distribution $y^\star$ is computed using the posterior:

$$p(y^\star|\mathbf{y}, \mathbf{X}) = \int p(y^\star|\mathbf{W}, \mathbf{x}^\star) p(\mathbf{W}|\mathbf{y}, \mathbf{X}) d\mathbf{W} \, .$$

Computation of the Predictive Distribution

The predictive distribution $y^\star$ is computed using the posterior:

$$p(y^\star|\mathbf{y},\mathbf{X}) = \int p(y^\star|\mathbf{W},\mathbf{x}^\star)p(\mathbf{W}|\mathbf{y},\mathbf{X})d\mathbf{W}\,.$$

Takes into account all potential values for W!

Computation of the Predictive Distribution

The predictive distribution $y^\star$ is computed using the posterior:

$$p(y^\star|\mathbf{y},\mathbf{X}) = \int p(y^\star|\mathbf{W},\mathbf{x}^\star)p(\mathbf{W}|\mathbf{y},\mathbf{X})d\mathbf{W}\,.$$

Takes into account all potential values for W!

Challenges:

Computation of the Predictive Distribution

The predictive distribution $y^\star$ is computed using the posterior:

$$p(y^\star|\mathbf{y},\mathbf{X}) = \int p(y^\star|\mathbf{W},\mathbf{x}^\star)p(\mathbf{W}|\mathbf{y},\mathbf{X})d\mathbf{W}\,.$$

Takes into account all potential values for W!

Challenges:

- $p(\mathbf{y}|\mathbf{X})$ cannot be computed!

Computation of the Predictive Distribution

The predictive distribution $y^\star$ is computed using the posterior:

$$p(y^\star|\mathbf{y}, \mathbf{X}) = \int p(y^\star|\mathbf{W}, \mathbf{x}^\star)p(\mathbf{W}|\mathbf{y}, \mathbf{X})d\mathbf{W}\,.$$

Takes into account all potential values for W!

Challenges:

- $p(\mathbf{y}|\mathbf{X})$ cannot be computed!
- The model should be non-parametric (the world is complex)!.

Computation of the Predictive Distribution

The predictive distribution $y^\star$ is computed using the posterior:

$$p(y^\star|\mathbf{y}, \mathbf{X}) = \int p(y^\star|\mathbf{W}, \mathbf{x}^\star)p(\mathbf{W}|\mathbf{y}, \mathbf{X})d\mathbf{W}\,.$$

Takes into account all potential values for W!

Challenges:

- $p(\mathbf{y}|\mathbf{X})$ cannot be computed!
- The model should be non-parametric (the world is complex)!.

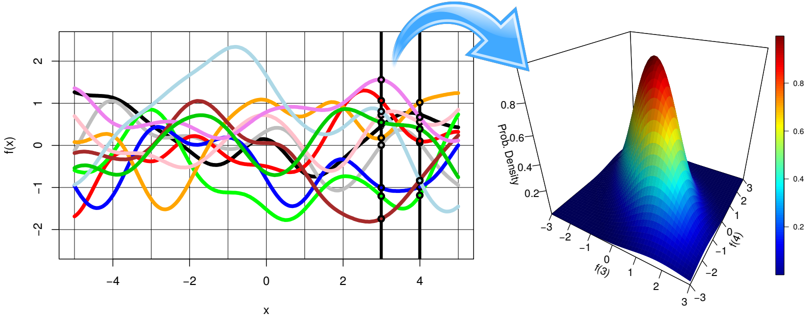
Solved by setting $p(\mathbf{W}) = \prod_{ij} \mathcal{N}(w_{ji}|0, \sigma^2 H^{-1})$ **and letting** $H \to \infty$**!**

Gaussian Processes

Distribution over functions $f(\cdot)$ so that for any finite $\{\mathbf{x}_i\}_{i=1}^N$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\mathsf{T}$ follows an N-dimensional Gaussian distribution.

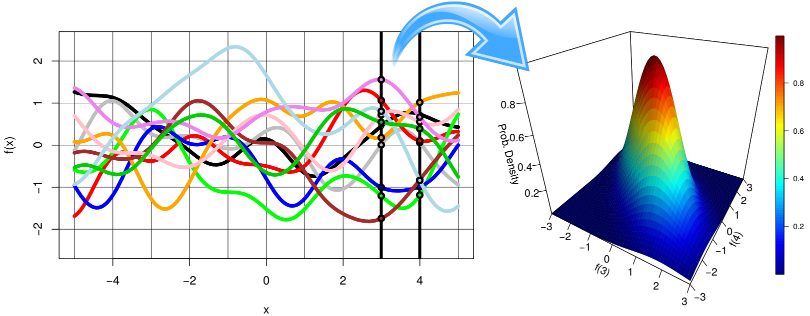
Gaussian Processes

Distribution over functions $f(\cdot)$ so that for any finite $\{\mathbf{x}_i\}_{i=1}^N$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\mathsf{T}$ follows an N-dimensional Gaussian distribution.



Gaussian Processes

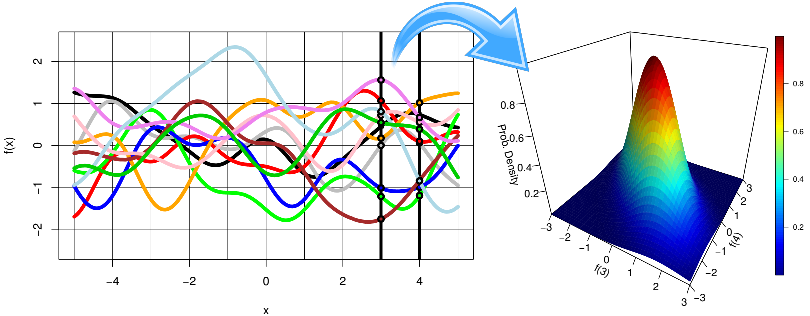
Distribution over functions $f(\cdot)$ so that for any finite $\{\mathbf{x}_i\}_{i=1}^N$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\mathsf{T}$ follows an N-dimensional Gaussian distribution.



When $H \to \infty$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\mathsf{T}$ follows an N-dimensional Gaussian where $\mathbb{E}[f(\mathbf{x}_i)f(\mathbf{x}_k)] = \sigma^2\mathbb{E}[h_j(\mathbf{x}_i)h_j(\mathbf{x}_k)]$ by the **central limit theorem**.

Gaussian Processes

Distribution over functions $f(\cdot)$ so that for any finite $\{\mathbf{x}_i\}_{i=1}^N$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\mathsf{T}$ follows an N-dimensional Gaussian distribution.

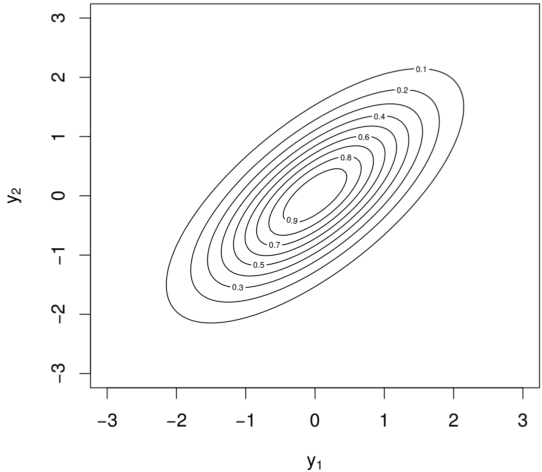


When $H \to \infty$, $(f(\mathbf{x}_1), \ldots, f(\mathbf{x}_N))^\mathsf{T}$ follows an N-dimensional Gaussian where $\mathbb{E}[f(\mathbf{x}_i)f(\mathbf{x}_k)] = \sigma^2 \mathbb{E}[h_j(\mathbf{x}_i)h_j(\mathbf{x}_k)]$ by the **central limit theorem**.

Due to Gaussian form, there are closed-form solutions for many useful questions about finite data.

Gaussian Distribution

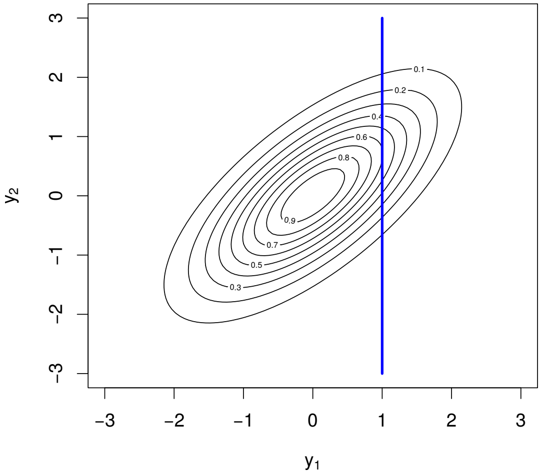
$$p(\mathbf{y}|\boldsymbol{\Sigma}) \propto \exp\left\{-0.5\mathbf{y}^\mathsf{T}\boldsymbol{\Sigma}^{-1}\mathbf{y}\right\} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1.0 & 0.7 \\ 0.7 & 1.0 \end{bmatrix}.$$



Gaussian Distribution

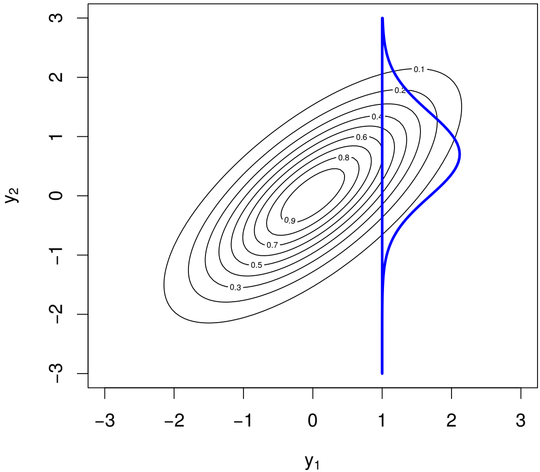
$p(\mathbf{y}|\boldsymbol{\Sigma}) \propto \exp\left\{-0.5\mathbf{y}^{\mathsf{T}}\boldsymbol{\Sigma}^{-1}\mathbf{y}\right\}$

$$\boldsymbol{\Sigma} = \begin{bmatrix} 1.0 & 0.7 \\ 0.7 & 1.0 \end{bmatrix}.$$



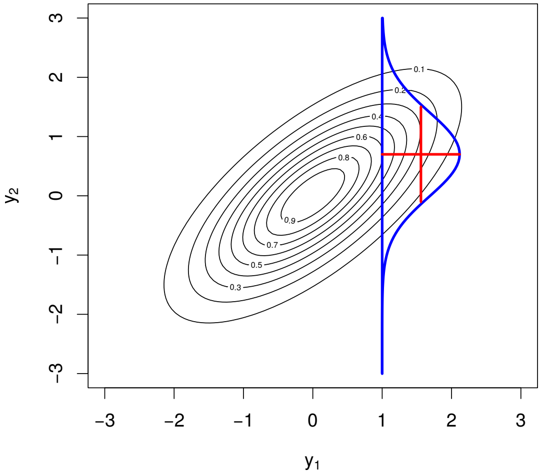
Gaussian Distribution

$$p(y_2|y_1, \boldsymbol{\Sigma}) \propto \exp\left\{-0.5(y_2 - \mu_\star)\Sigma_\star^{-1}(y_2 - \mu_\star)\right\} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1.0 & 0.7 \\ 0.7 & 1.0 \end{bmatrix}.$$



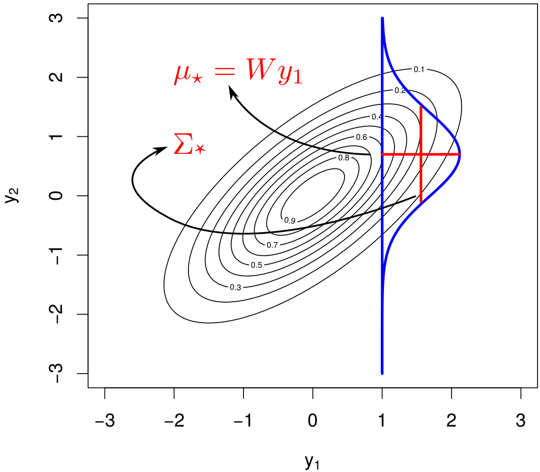
Gaussian Distribution

$$p(y_2|y_1, \boldsymbol{\Sigma}) \propto \exp\left\{-0.5(y_2 - \mu_\star)\Sigma_\star^{-1}(y_2 - \mu_\star)\right\} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1.0 & 0.7 \\ 0.7 & 1.0 \end{bmatrix}.$$

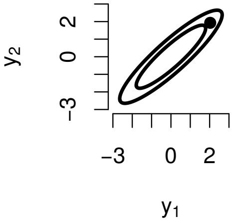


Gaussian Distribution

$$p(y_2|y_1, \boldsymbol{\Sigma}) \propto \exp\left\{-0.5(y_2 - \mu_\star)\Sigma_\star^{-1}(y_2 - \mu_\star)\right\} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1.0 & 0.7 \\ 0.7 & 1.0 \end{bmatrix}.$$

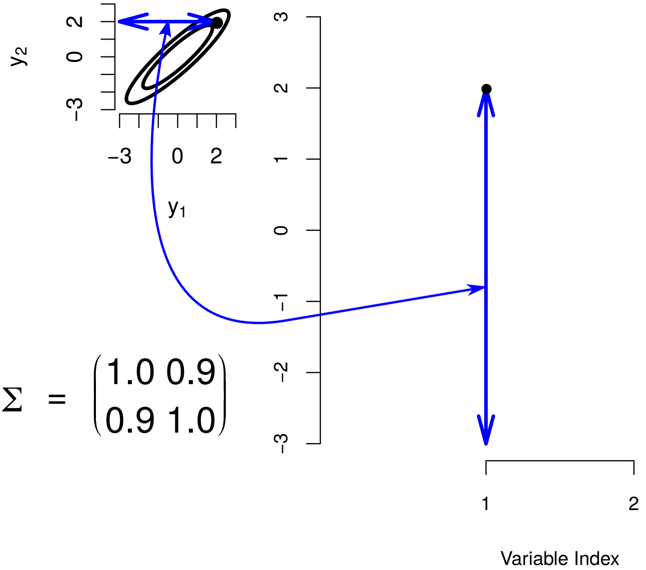


Two Dimensional Example



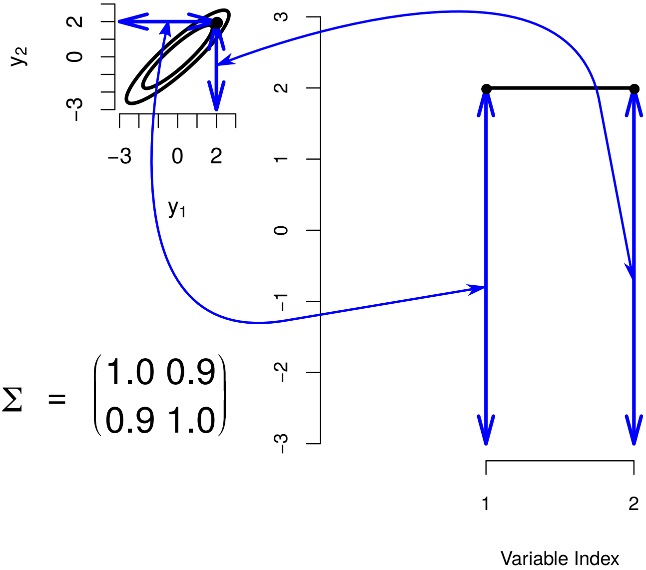
$$\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$

Two Dimensional Example



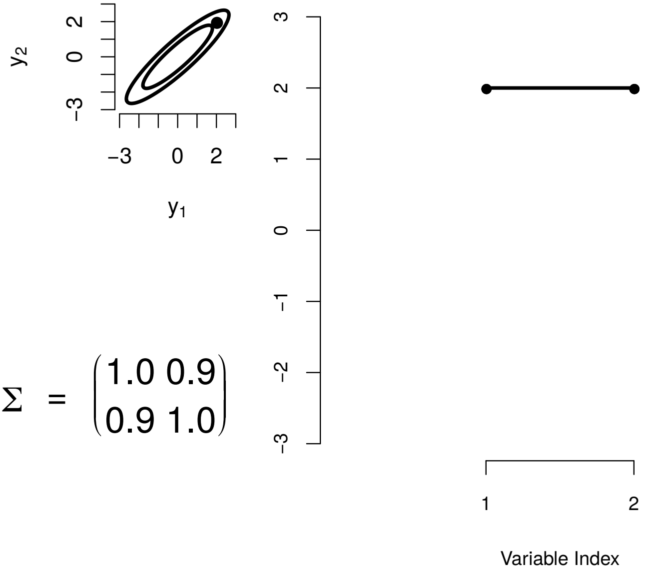
$$\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$

Two Dimensional Example



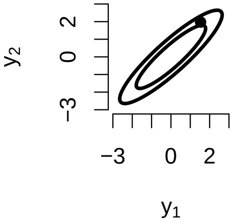
$$\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$

Two Dimensional Example

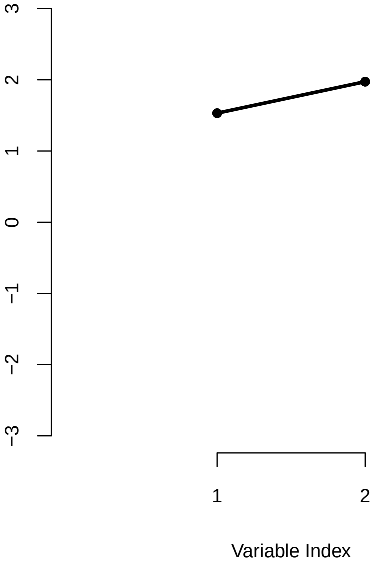


$$\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$

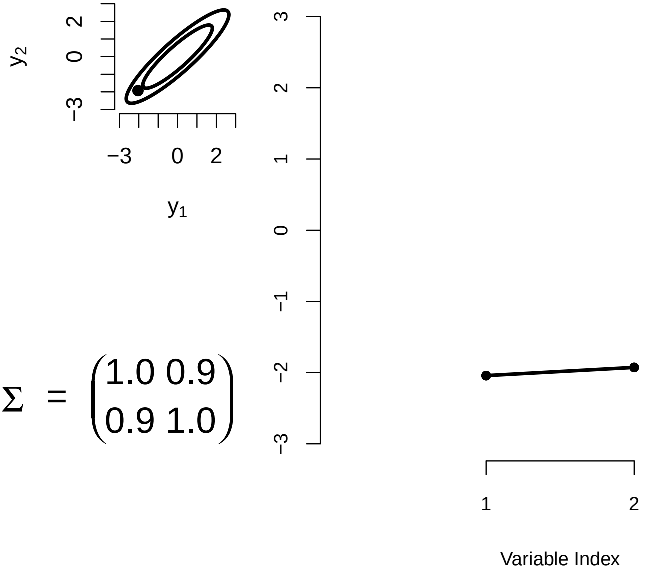
Two Dimensional Example



$$\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$

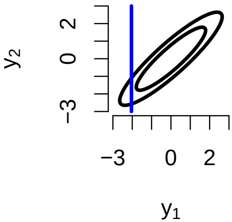


Two Dimensional Example

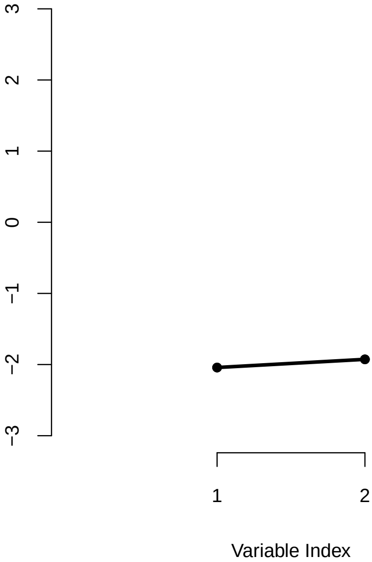


$$\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$

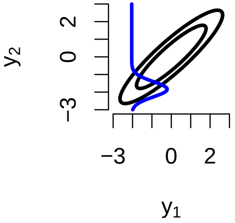
Two Dimensional Example



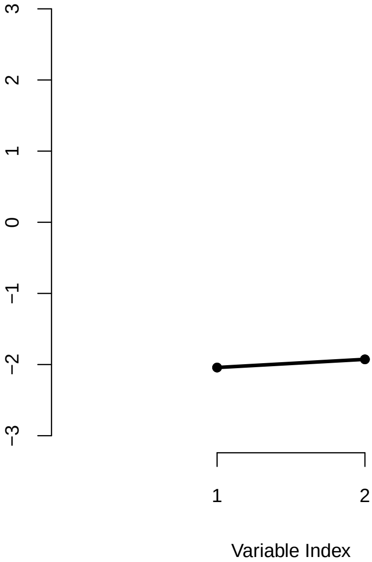
$$\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$



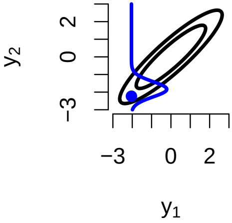
Two Dimensional Example



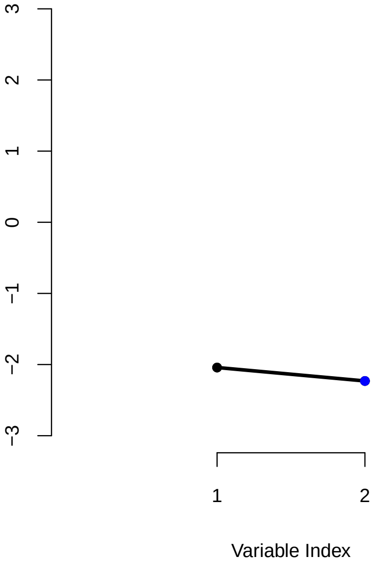
$$\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$



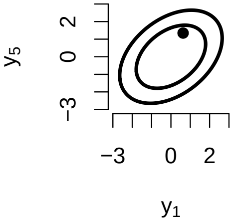
Two Dimensional Example



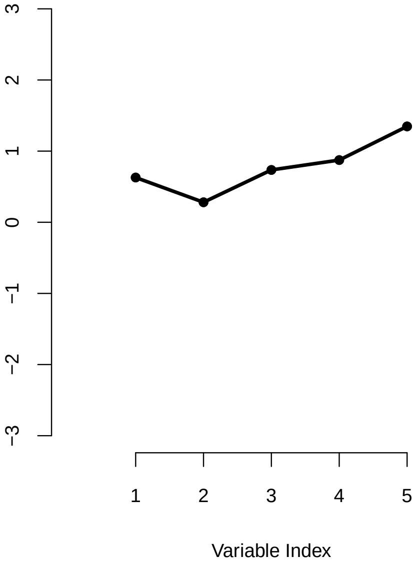
$$\Sigma = \begin{pmatrix} 1.0 & 0.9 \\ 0.9 & 1.0 \end{pmatrix}$$



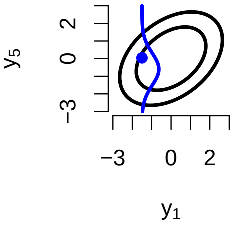
Five Dimensional Example



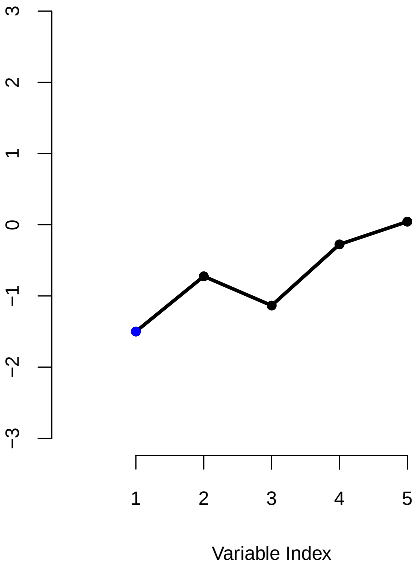
$$\Sigma = \begin{bmatrix} 1.0 & .9 & .8 & .6 & .4 \\ .9 & 1.0 & .9 & .8 & .6 \\ .8 & .9 & 1.0 & .9 & .8 \\ .6 & .8 & .9 & 1.0 & .9 \\ .4 & .6 & .8 & .9 & 1.0 \end{bmatrix}$$



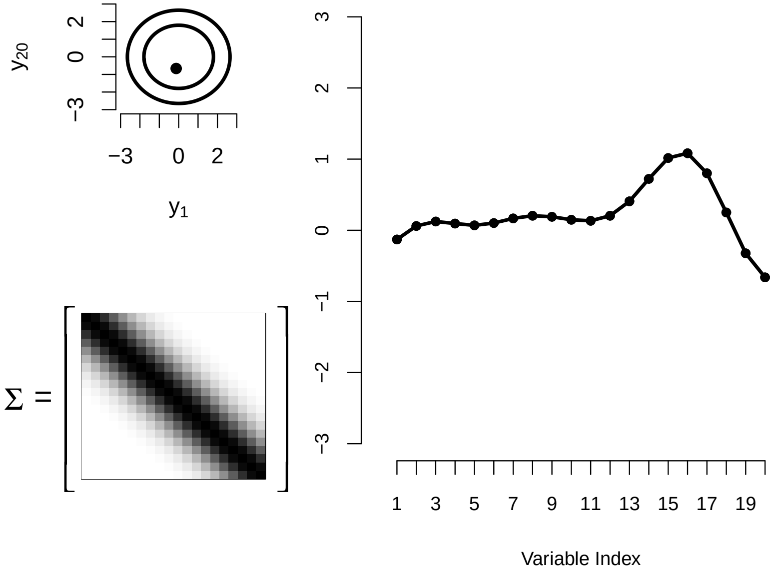
Five Dimensional Example



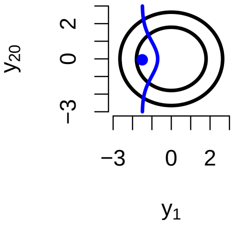
$$\Sigma = \begin{bmatrix} 1.0 & .9 & .8 & .6 & .4 \\ .9 & 1.0 & .9 & .8 & .6 \\ .8 & .9 & 1.0 & .9 & .8 \\ .6 & .8 & .9 & 1.0 & .9 \\ .4 & .6 & .8 & .9 & 1.0 \end{bmatrix}$$

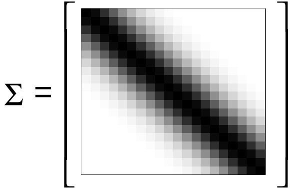


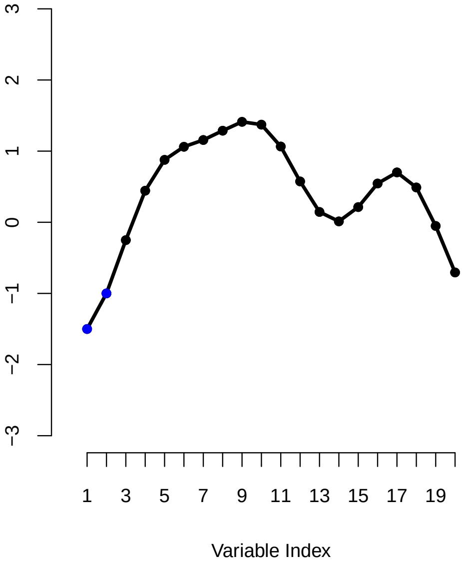
Twenty Dimensional Example



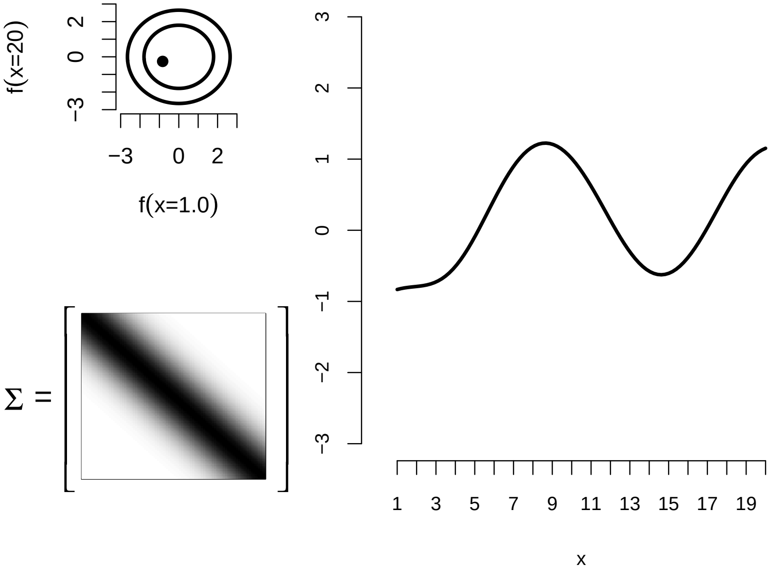
Twenty Dimensional Example



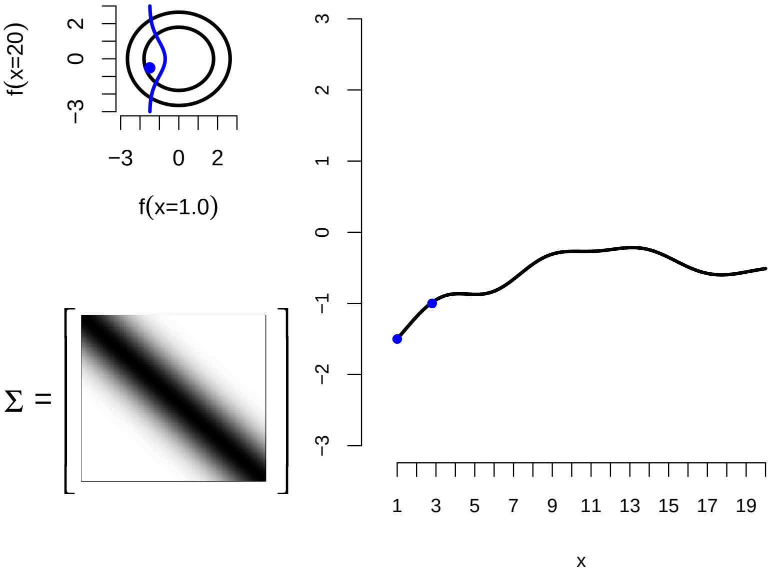




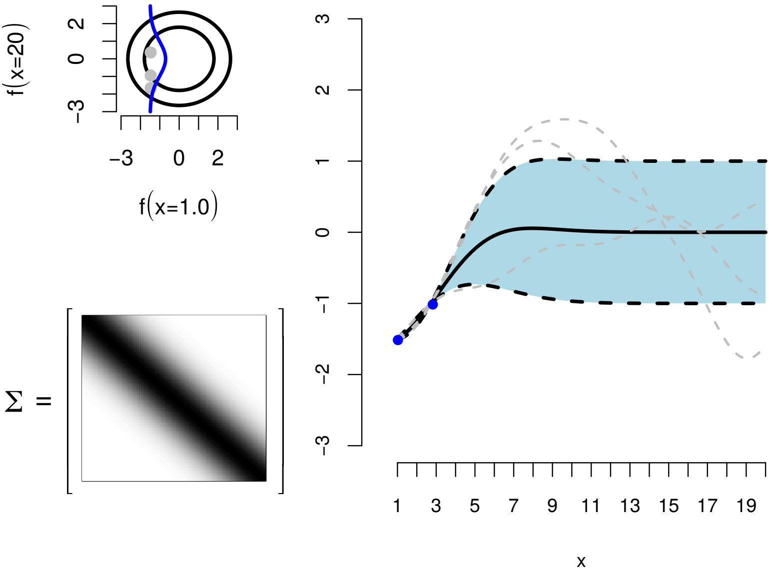
Infinite Dimensional Example



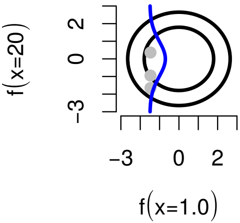
Infinite Dimensional Example

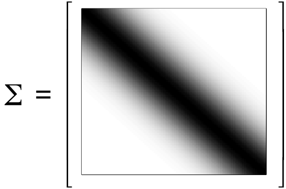


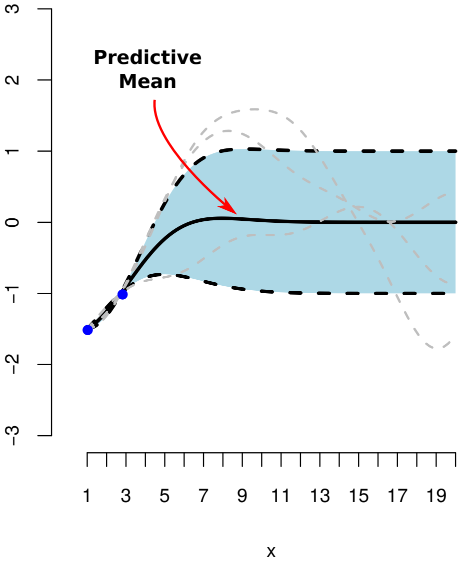
Predictive Distribution



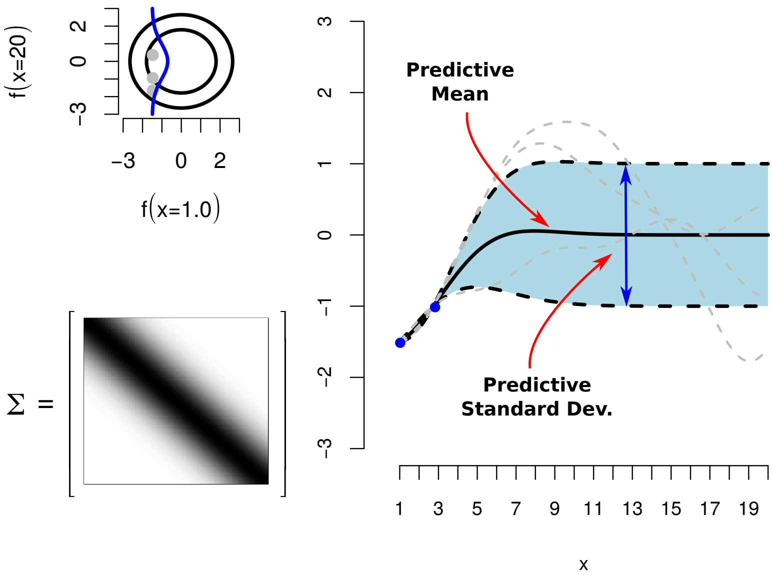
Predictive Distribution



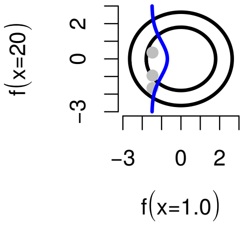


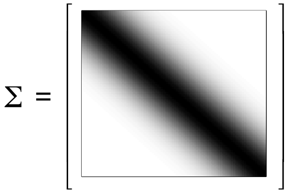


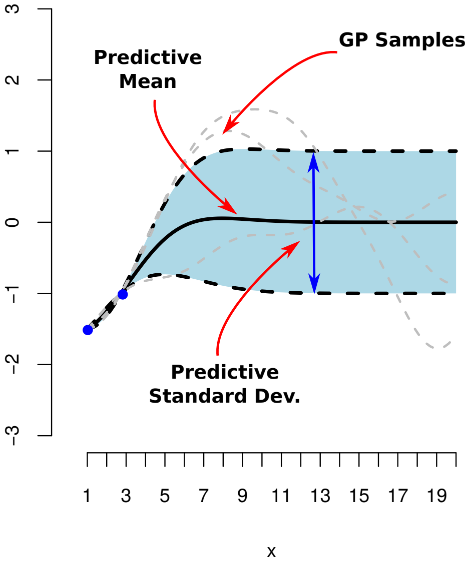
Predictive Distribution



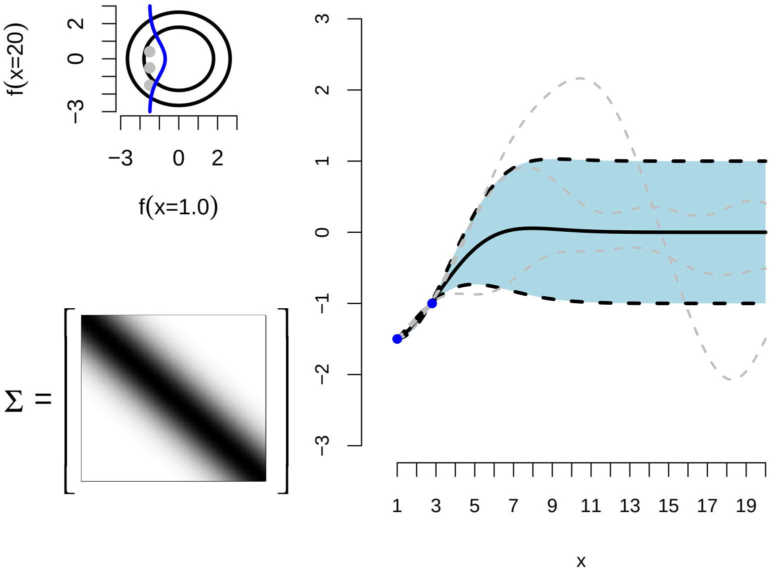
Predictive Distribution



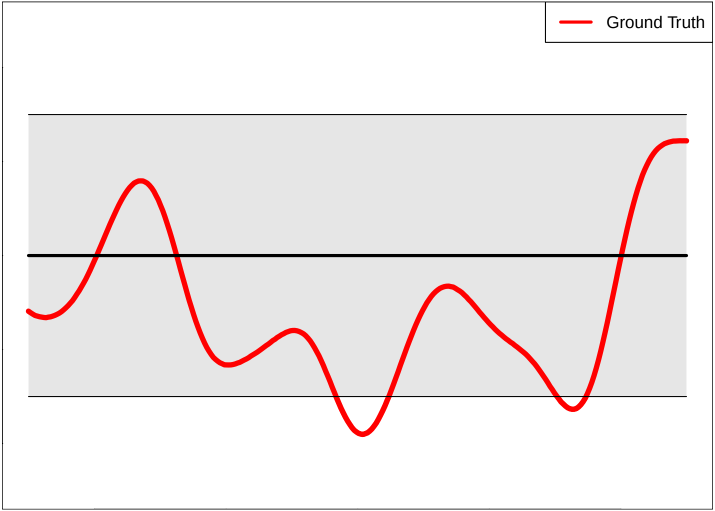




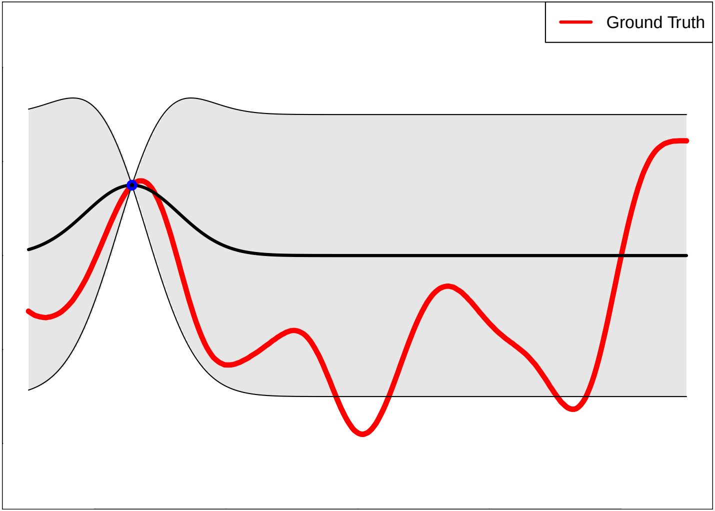
Predictive Distribution



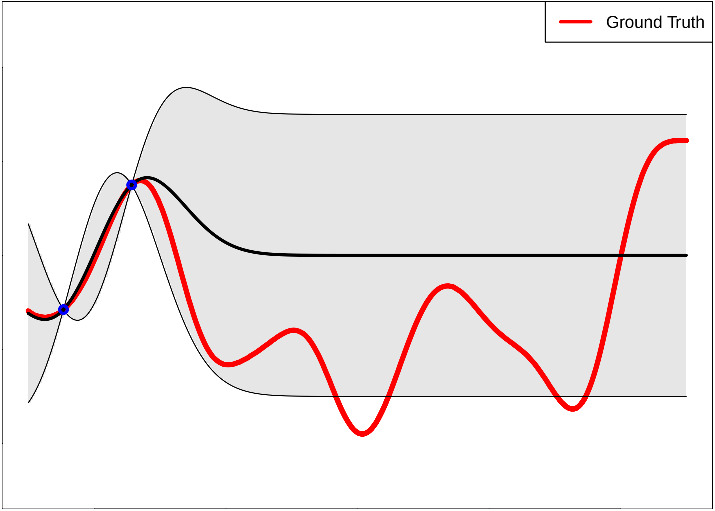
Predictive Distribution



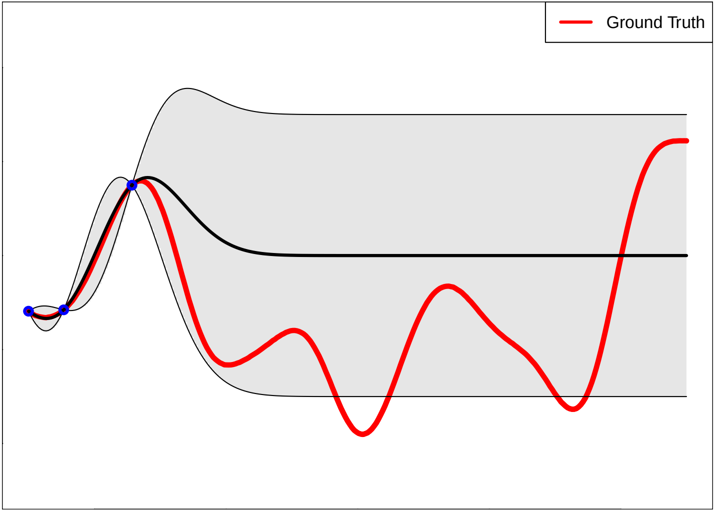
Predictive Distribution



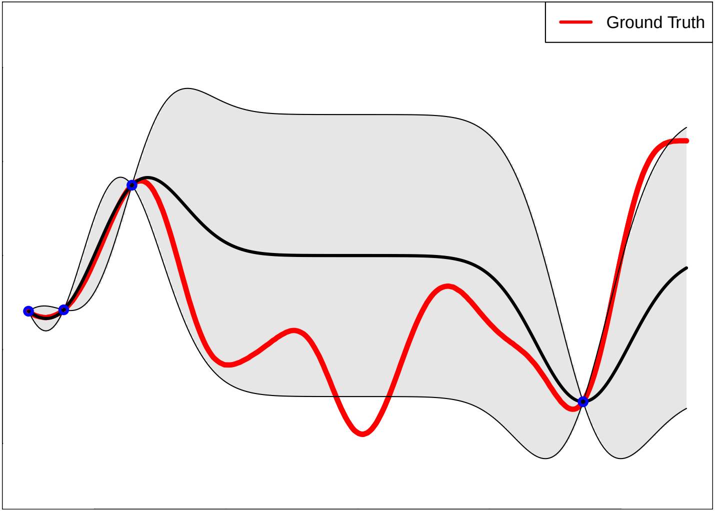
Predictive Distribution



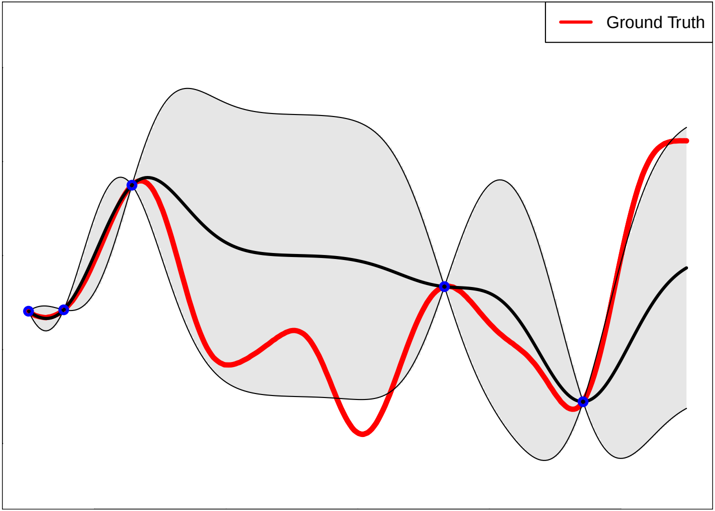
Predictive Distribution



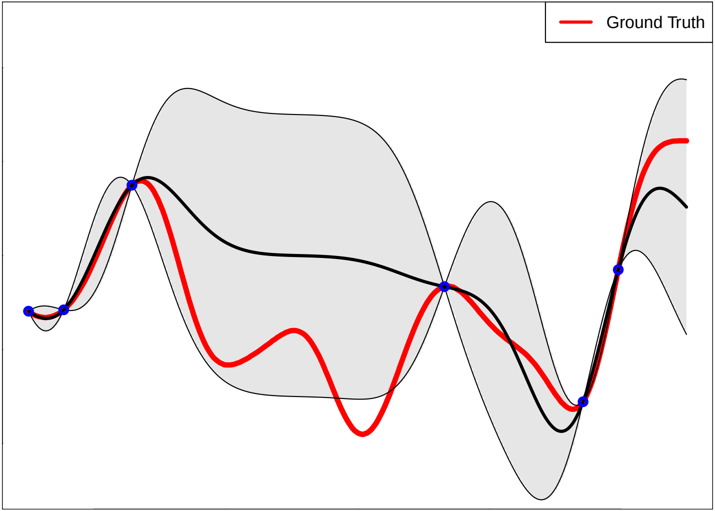
Predictive Distribution



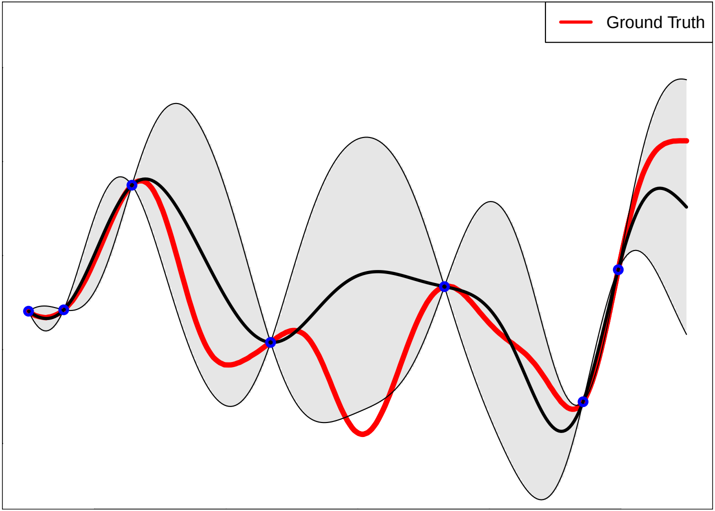
Predictive Distribution



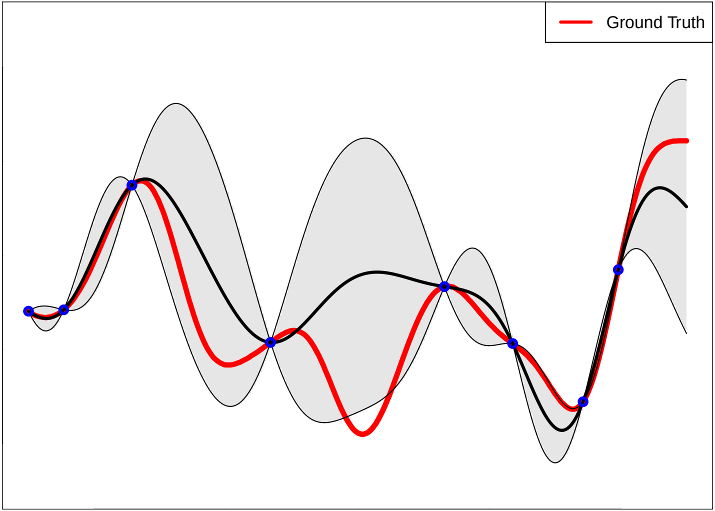
Predictive Distribution



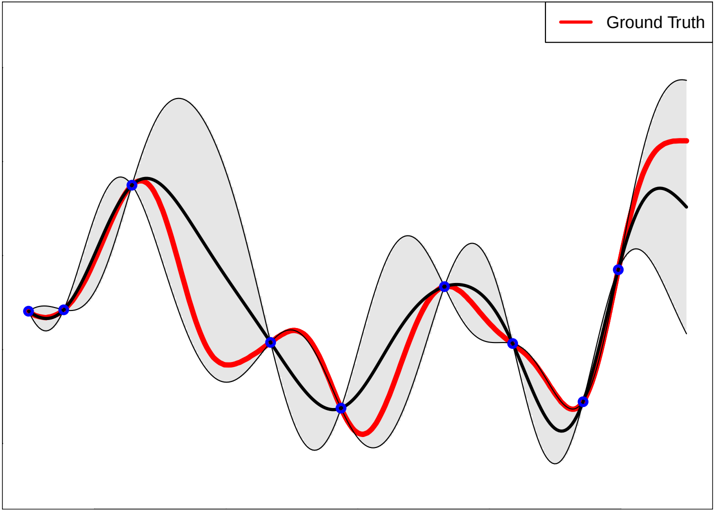
Predictive Distribution



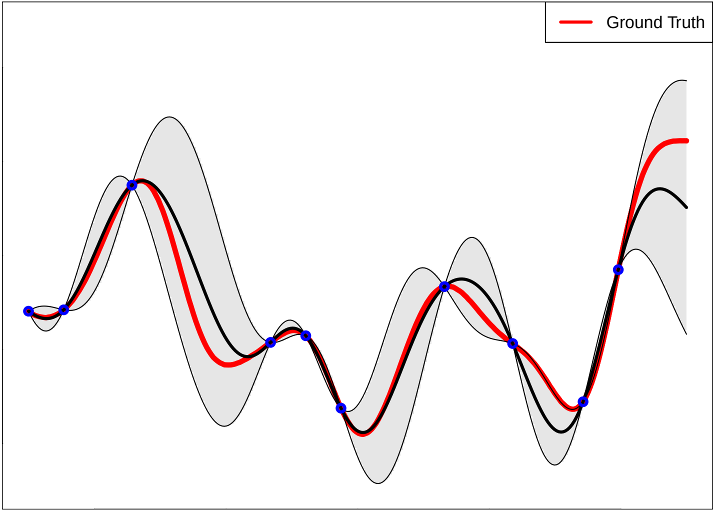
Predictive Distribution



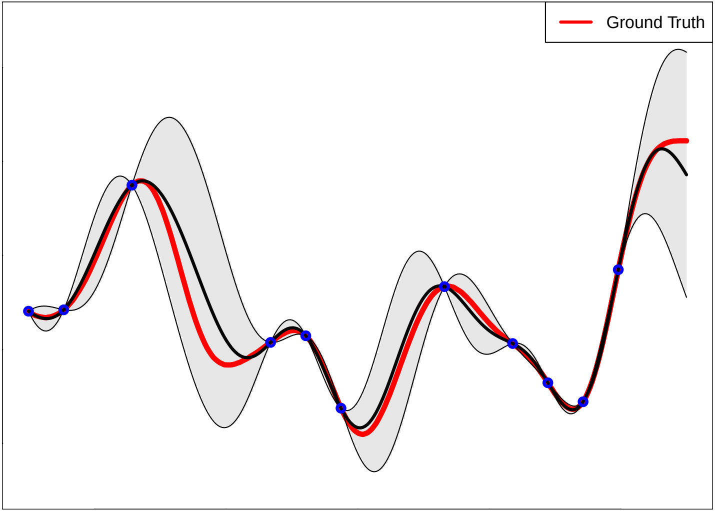
Predictive Distribution



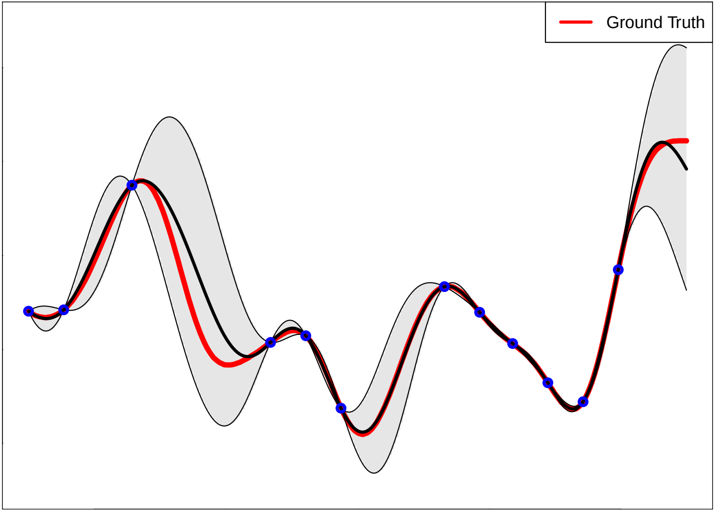
Predictive Distribution



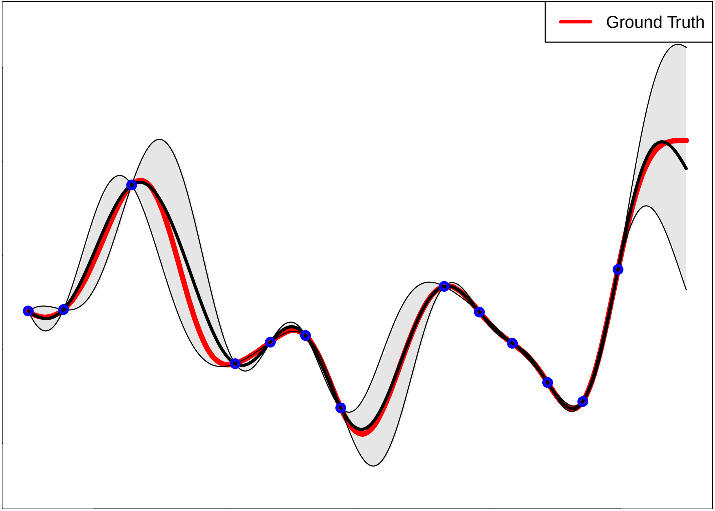
Predictive Distribution



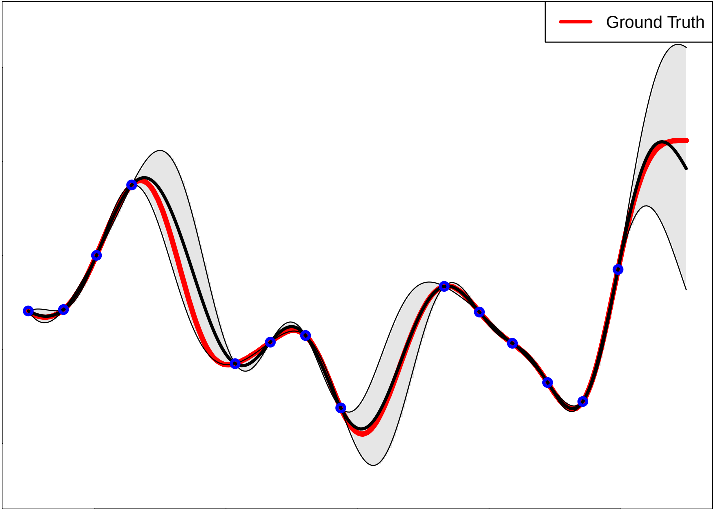
Predictive Distribution



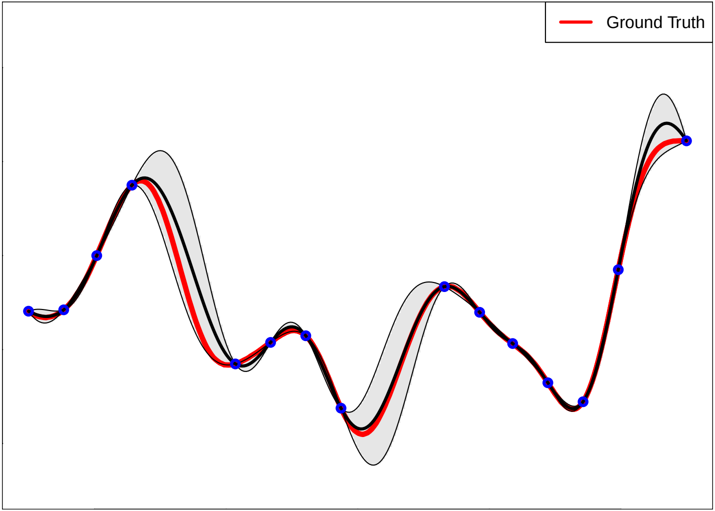
Predictive Distribution



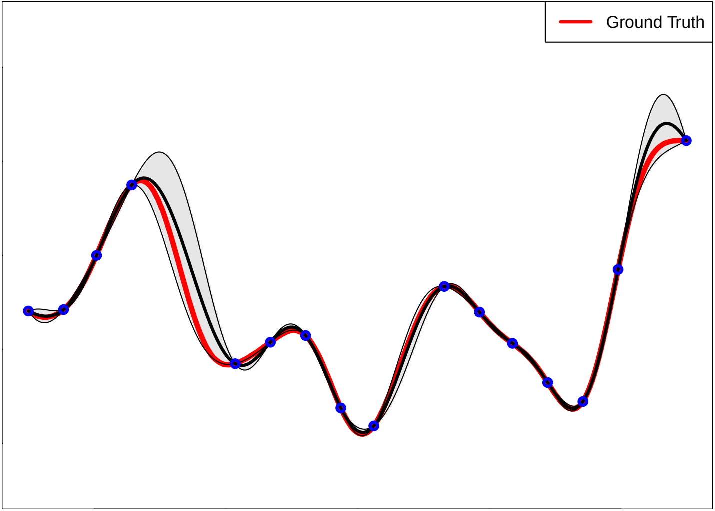
Predictive Distribution



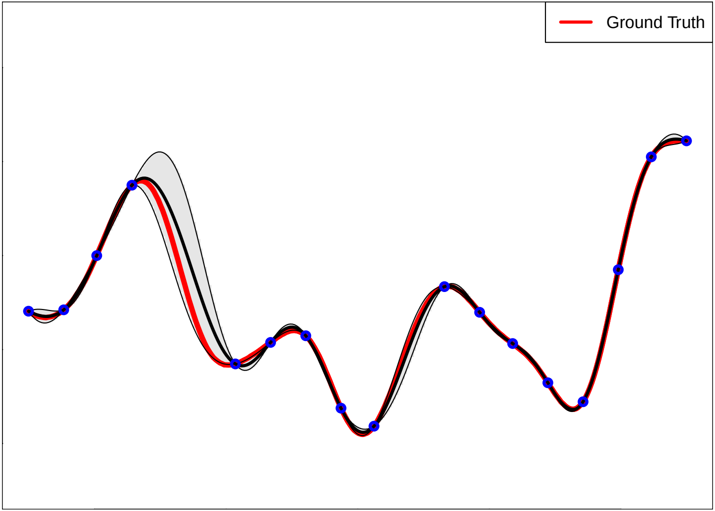
Predictive Distribution



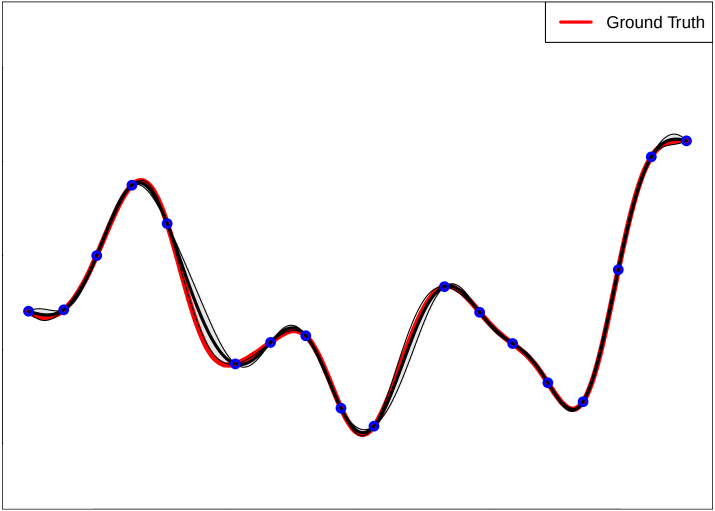
Predictive Distribution



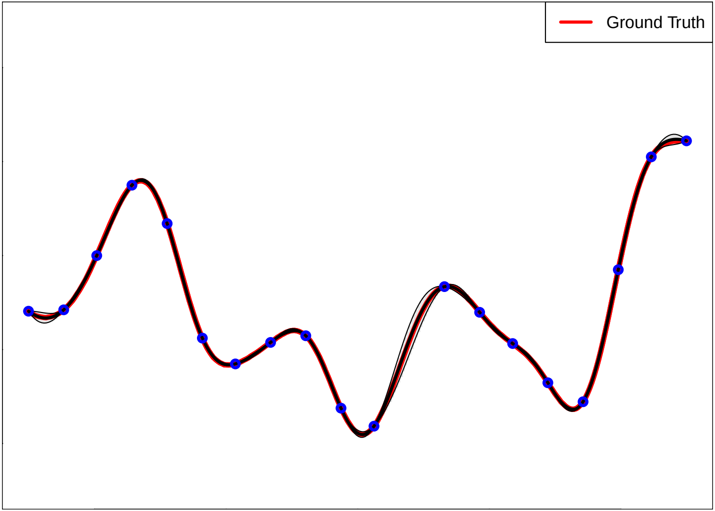
Predictive Distribution



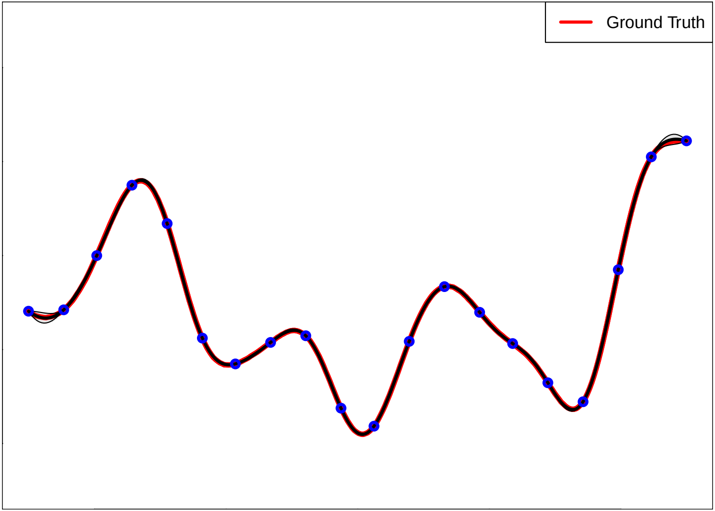
Predictive Distribution



Predictive Distribution



Predictive Distribution



Summary so Far...

- A GP is *like* a Gaussian distribution with an **infinitely long mean** vector and an $\infty \times \infty$ **covariance matrix**.

Summary so Far...

- A GP is *like* a Gaussian distribution with an **infinitely long mean** vector and an $\infty \times \infty$ **covariance matrix**.
- The covariance matrix often enforces that function values corresponding to near-by points take **similar values**.

Summary so Far...

- A GP is *like* a Gaussian distribution with an **infinitely long mean** vector and an $\infty \times \infty$ **covariance matrix**.
- The covariance matrix often enforces that function values corresponding to near-by points take **similar values**.
- Due to the Gaussian distribution of finite function values, there are many **closed form expressions** like the predictive distribution.

Summary so Far...

- A GP is *like* a Gaussian distribution with an **infinitely long mean** vector and an $\infty \times \infty$ **covariance matrix**.
- The covariance matrix often enforces that function values corresponding to near-by points take **similar values**.
- Due to the Gaussian distribution of finite function values, there are many **closed form expressions** like the predictive distribution.
- GPs are **non-parametric models** and become more expressive the more data we have.

Definition

A Gaussian process is a collection of random variables, any finite number of which have a Gaussian distribution.

Definition

A Gaussian process is a collection of random variables, any finite number of which have a Gaussian distribution.

A Gaussian distribution is fully specified by a mean vector, $\boldsymbol{\mu}$, and covariance matrix $\boldsymbol{\Sigma}$:

$$\mathbf{f} = (f_1, \ldots, f_N)^\mathsf{T} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \text{indices} \quad i = 1, \ldots, N\,.$$

Definition

A Gaussian process is a collection of random variables, any finite number of which have a Gaussian distribution.

A Gaussian distribution is fully specified by a mean vector, $\boldsymbol{\mu}$, and covariance matrix $\boldsymbol{\Sigma}$:

$$\mathbf{f} = (f_1, \ldots, f_N)^\mathsf{T} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad \text{indices} \quad i = 1, \ldots, N\,.$$

A Gaussian process is fully specified by a mean function $m(\mathbf{x})$ and covariance function $C(\mathbf{x}, \mathbf{x}')$:

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), C(\mathbf{x}, \mathbf{x}')), \quad \text{indices} \quad \mathbf{x}\,.$$

GP Prior Mean

The GP prior mean $m(\cdot)$ can be specified by any function!

$$\mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}).$$

GP Prior Mean

The GP prior mean $m(\cdot)$ can be specified by any function!

$$\mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}).$$

It determines the global tendency of the latent function before observing the data. Often, simply set to zero.

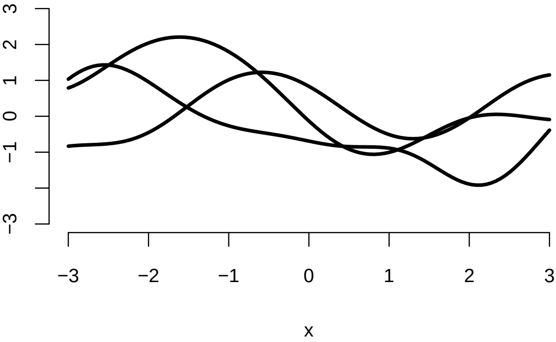
GP Prior Mean

The GP prior mean $m(\cdot)$ can be specified by any function!

$$\mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}).$$

It determines the global tendency of the latent function before observing the data. Often, simply set to zero.

m(x)=0



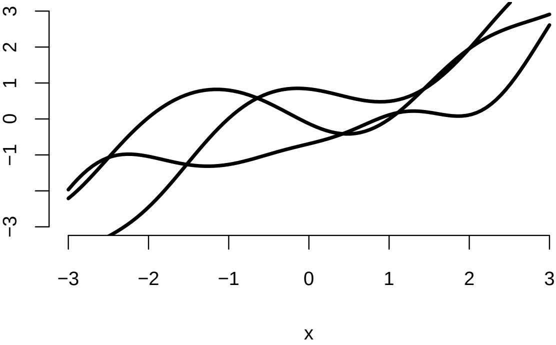
GP Prior Mean

The GP prior mean $m(\cdot)$ can be specified by any function!

$$\mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}).$$

It determines the global tendency of the latent function before observing the data. Often, simply set to zero.

m(x)=x



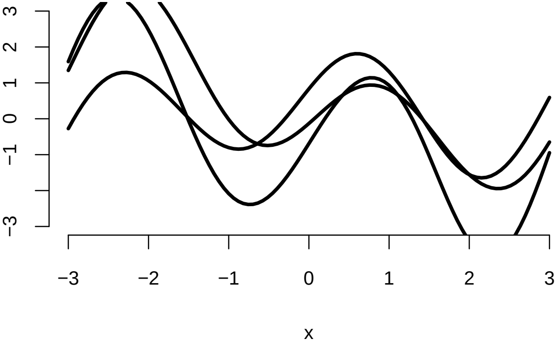
GP Prior Mean

The GP prior mean $m(\cdot)$ can be specified by any function!

$$\mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}).$$

It determines the global tendency of the latent function before observing the data. Often, simply set to zero.

m(x)=2sin(2x)



GP Prior Covariances

The covariance function sets prior covariances among function values!

$$\mathbb{E}\left[(f(\mathbf{x}_i) - m(\mathbf{x}_i))(f(\mathbf{x}_j) - m(\mathbf{x}_j))\right] = C(\mathbf{x}_i, \mathbf{x}_j).$$

GP Prior Covariances

The covariance function sets prior covariances among function values!

$$\mathbb{E}\left[(f(\mathbf{x}_i) - m(\mathbf{x}_i))(f(\mathbf{x}_j) - m(\mathbf{x}_j))\right] = C(\mathbf{x}_i, \mathbf{x}_j).$$

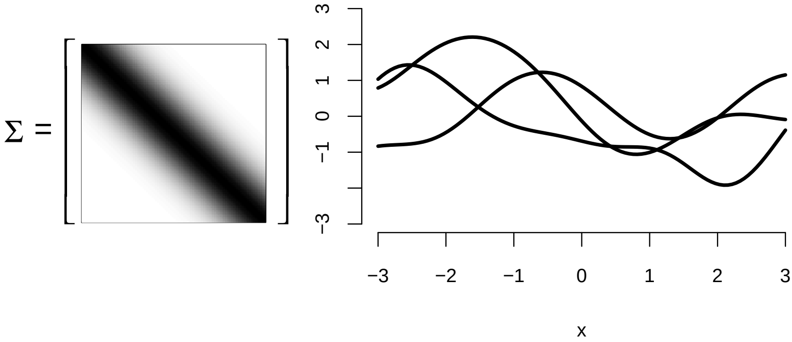
It determines the global properties of the latent function before observing the data.

GP Prior Covariances

The covariance function sets prior covariances among function values!

$$\mathbb{E}\left[(f(\mathbf{x}_i) - m(\mathbf{x}_i))(f(\mathbf{x}_j) - m(\mathbf{x}_j))\right] = C(\mathbf{x}_i, \mathbf{x}_j).$$

It determines the global properties of the latent function before observing the data.

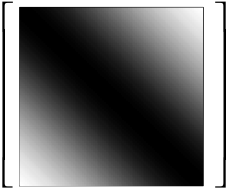


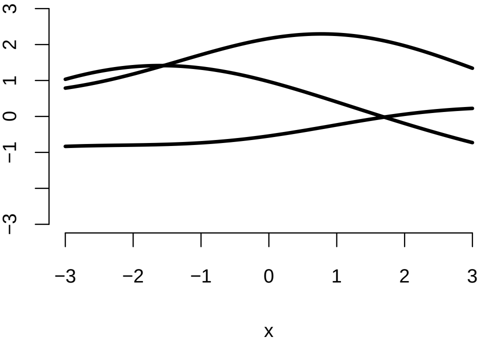
GP Prior Covariances

The covariance function sets prior covariances among function values!

$$\mathbb{E}\left[(f(\mathbf{x}_i) - m(\mathbf{x}_i))(f(\mathbf{x}_j) - m(\mathbf{x}_j))\right] = C(\mathbf{x}_i, \mathbf{x}_j).$$

It determines the global properties of the latent function before observing the data.

$\Sigma =$ 

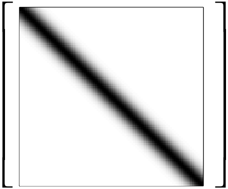


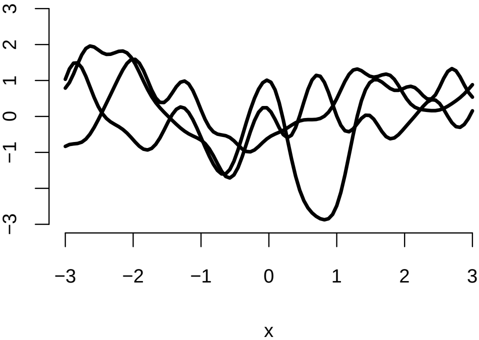
GP Prior Covariances

The covariance function sets prior covariances among function values!

$$\mathbb{E}\left[(f(\mathbf{x}_i) - m(\mathbf{x}_i))(f(\mathbf{x}_j) - m(\mathbf{x}_j))\right] = C(\mathbf{x}_i, \mathbf{x}_j).$$

It determines the global properties of the latent function before observing the data.

$\Sigma =$ 



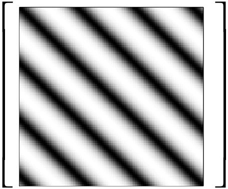
GP Prior Covariances

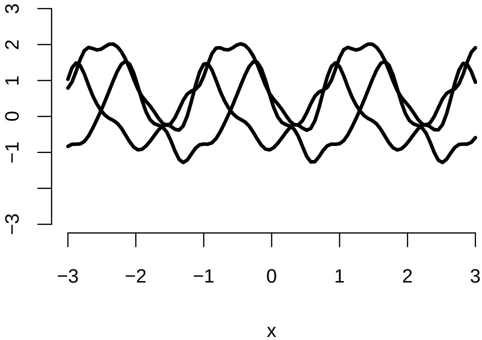
The covariance function sets prior covariances among function values!

$$\mathbb{E}\left[(f(\mathbf{x}_i) - m(\mathbf{x}_i))(f(\mathbf{x}_j) - m(\mathbf{x}_j))\right] = C(\mathbf{x}_i, \mathbf{x}_j).$$

It determines the global properties of the latent function before observing the data.

$\Sigma =$





Marginalization

If the GP mean has infinite length and the GP covariance matrix is $\infty \times \infty$, how do we represent a GP on a computer?

Marginalization

If the GP mean has infinite length and the GP covariance matrix is $\infty \times \infty$, how do we represent a GP on a computer?

We can use the marginalization property of distributions:

$$p(\mathbf{y}_1) = \int p(\mathbf{y}_1, \mathbf{y}_2) d\mathbf{y}_2 \,,$$

$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{B} \end{bmatrix} \right) \,,$$

Marginalization

If the GP mean has infinite length and the GP covariance matrix is $\infty \times \infty$, how do we represent a GP on a computer?

We can use the marginalization property of distributions:

$$
\begin{aligned}
p(\mathbf{y}_1) &= \int p(\mathbf{y}_1, \mathbf{y}_2) d\mathbf{y}_2 \,, \\
p(\mathbf{y}_1, \mathbf{y}_2) &= \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\mathsf{T} & \mathbf{B} \end{bmatrix} \right) \,, \\
p(\mathbf{y}_1) &= \mathcal{N}(\mathbf{y}_1 | \mathbf{a}, \mathbf{A}) \,,
\end{aligned}
$$

Marginalization

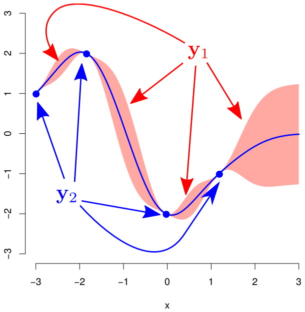
If the GP mean has infinite length and the GP covariance matrix is $\infty \times \infty$, how do we represent a GP on a computer?

We can use the marginalization property of distributions:

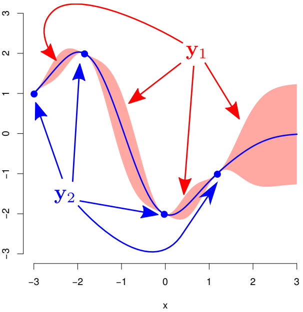
$$
\begin{aligned}
p(\mathbf{y}_1) &= \int p(\mathbf{y}_1, \mathbf{y}_2) d\mathbf{y}_2 \,, \\
p(\mathbf{y}_1, \mathbf{y}_2) &= \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\mathsf{T} & \mathbf{B} \end{bmatrix} \right) \,, \\
p(\mathbf{y}_1) &= \mathcal{N}(\mathbf{y}_1 | \mathbf{a}, \mathbf{A}) \,,
\end{aligned}
$$

We only need to work with finite sets of random variables!

Computing the Predictive Distribution

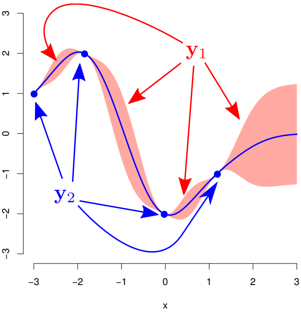


Computing the Predictive Distribution



$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\mathsf{T} & \mathbf{B} \end{bmatrix} \right),$$

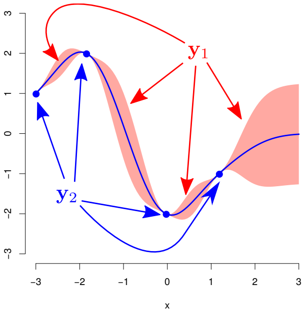
Computing the Predictive Distribution



$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{B} \end{bmatrix}\right),$$

$$p(\mathbf{y}_1 | \mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)},$$

Computing the Predictive Distribution

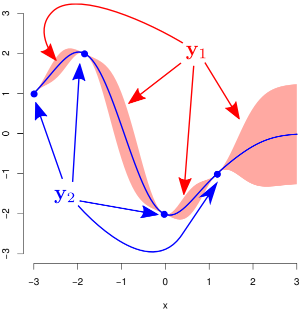


$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{B} \end{bmatrix}\right),$$

$$p(\mathbf{y}_1 | \mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)},$$

$$p(\mathbf{y}_1|\mathbf{y}_2) = \mathcal{N}\left(\mathbf{y}_1 \middle| \mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y}_2 - \mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^\top\right)$$

Computing the Predictive Distribution



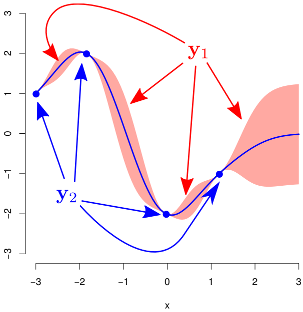
$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{B} \end{bmatrix}\right),$$

$$p(\mathbf{y}_1 | \mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)},$$

$$p(\mathbf{y}_1 | \mathbf{y}_2) = \mathcal{N}\left(\mathbf{y}_1 \middle| \mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y}_2 - \mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^\top\right)$$

- The predictive mean is linear in $\mathbf{y}_2$.

Computing the Predictive Distribution



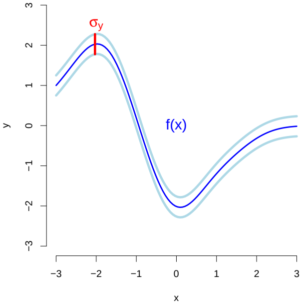
$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{B} \end{bmatrix}\right),$$

$$p(\mathbf{y}_1|\mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)},$$

$$p(\mathbf{y}_1|\mathbf{y}_2) = \mathcal{N}\left(\mathbf{y}_1 \middle| \mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y}_2 - \mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^\top\right)$$

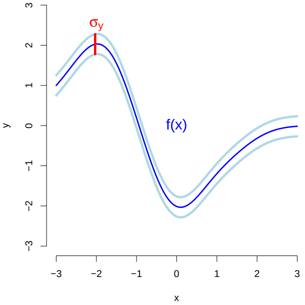
- The predictive mean is linear in $\mathbf{y}_2$.
- The predictive covariance is **more confident** than the prior!.

Considering Additive Noise



$$y(\mathbf{x}) = f(\mathbf{x}) + \epsilon\sigma_y\,,$$
$$p(\epsilon) = \mathcal{N}(\epsilon|0,1)\,.$$

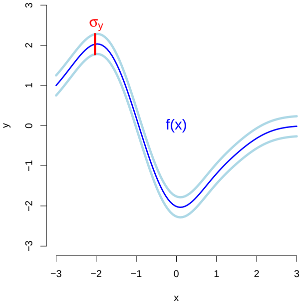
Considering Additive Noise



$$y(\mathbf{x}) = f(\mathbf{x}) + \epsilon\sigma_y \,,$$
$$p(\epsilon) = \mathcal{N}(\epsilon|0,1)\,.$$

Since $f(\mathbf{x})$ follows a GP and ϵ is Gaussian $y(\mathbf{x})$ is another GP!

Considering Additive Noise

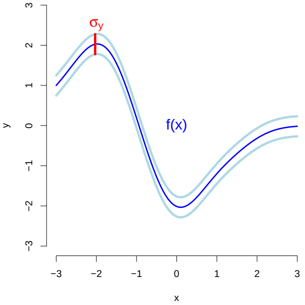


$$y(\mathbf{x}) = f(\mathbf{x}) + \epsilon\sigma_y\,,$$
$$p(\epsilon) = \mathcal{N}(\epsilon|0,1)\,.$$

Since $f(\mathrm{x})$ follows a GP and ϵ is Gaussian $y(\mathrm{x})$ is another GP!

$$y(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), C(\mathbf{x},\mathbf{x}') + \mathbb{I}(\mathbf{x}=\mathbf{x}')\sigma_y^2)$$

Considering Additive Noise



$$y(\mathbf{x}) = f(\mathbf{x}) + \epsilon\sigma_y ,$$
$$p(\epsilon) = \mathcal{N}(\epsilon|0, 1) .$$

Since $f(\mathbf{x})$ follows a GP and ϵ is Gaussian $y(\mathbf{x})$ is another GP!

$$y(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), C(\mathbf{x}, \mathbf{x}') + \mathbb{I}(\mathbf{x} = \mathbf{x}')\sigma_y^2)$$

The predictive distribution is:

$$p(\mathbf{y}_1|\mathbf{y}_2) = \mathcal{N}\left(\mathbf{y}_1 \middle| \mathbf{a} + \mathbf{C}(\mathbf{B} + \mathbf{I}\sigma_y^2)^{-1}(\mathbf{y}_2 - \mathbf{b}), \mathbf{A} - \mathbf{C}(\mathbf{B} + \mathbf{I}\sigma_y^2)^{-1}\mathbf{C}^\mathsf{T}\right)$$

An Example of a Covariance Function

Squared Exponential: $$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left\{ -\frac{1}{2} \sum_{j=1}^{d} \left(\frac{x_j - x'_j}{l_j} \right)^2 \right\}$$

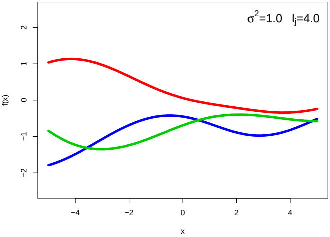
An Example of a Covariance Function

Squared Exponential: $$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left\{ -\frac{1}{2} \sum_{j=1}^{d} \left(\frac{x_j - x_j'}{l_j} \right)^2 \right\}$$

- Vertical scale
- Horizontal scale

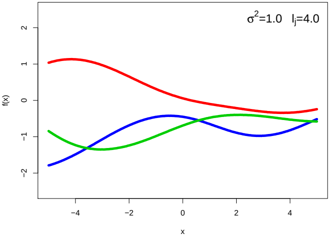
An Example of a Covariance Function

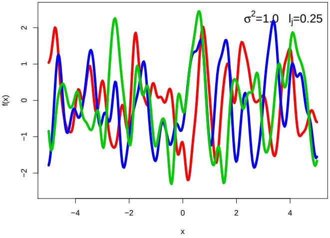
Squared Exponential: $$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left\{ -\frac{1}{2} \sum_{j=1}^{d} \left(\frac{x_j - x_j'}{l_j} \right)^2 \right\}$$



An Example of a Covariance Function

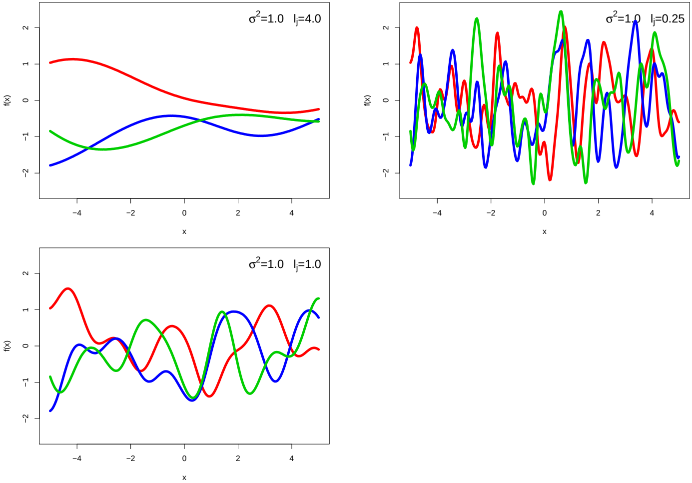
Squared Exponential: $$C(\mathbf{x},\mathbf{x}') = \boxed{\sigma^2} \exp\left\{ -\frac{1}{2}\sum_{j=1}^{d}\left(\frac{x_j - x'_j}{\boxed{l_j}}\right)^2 \right\}$$





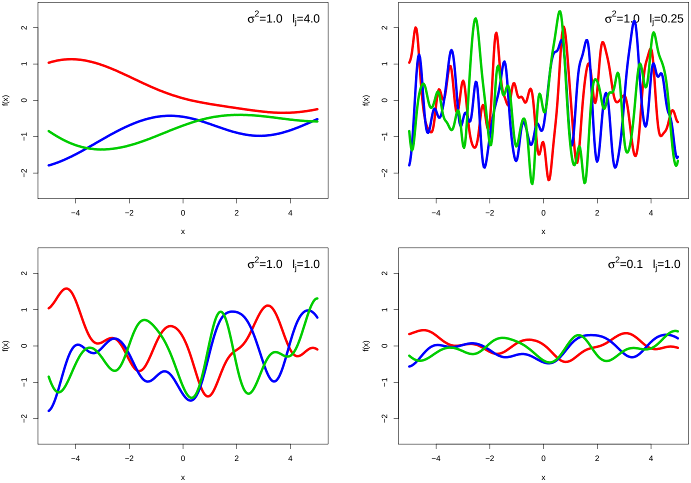
An Example of a Covariance Function

Squared Exponential: $$C(\mathbf{x}, \mathbf{x}') = \boxed{\sigma^2} \exp\left\{ -\frac{1}{2} \sum_{j=1}^{d} \left(\frac{x_j - x_j'}{\boxed{l_j}} \right)^2 \right\}$$



An Example of a Covariance Function

Squared Exponential: $$C(\mathbf{x}, \mathbf{x}') = \boxed{\sigma^2} \exp\left\{ -\frac{1}{2} \sum_{j=1}^{d} \left(\frac{x_j - x_j'}{\boxed{l_j}} \right)^2 \right\}$$



How do we choose the hyper-parameters?

Intuition: find parameters θ that are compatible with the observed data.

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

How do we choose the hyper-parameters?

Intuition: find parameters θ that are compatible with the observed data.

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

what we know after seeing the data (*posterior*)	$\propto$	what the data tell us (likelihood)	$\times$	what we know before seeing the data (prior)

How do we choose the hyper-parameters?

Intuition: find parameters θ that are compatible with the observed data.

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

what we know after seeing the data (*posterior*)	$\propto$	what the data tell us (likelihood)	$\times$	what we know before seeing the data (prior)

$$\begin{aligned} p(\mathbf{y}|\theta) &\equiv \text{how well does } \theta \text{ explain the observed data} \\ &= \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \boldsymbol{\Sigma} + \mathbf{I}\sigma_y^2\right) \end{aligned}$$

How do we choose the hyper-parameters?

Intuition: find parameters θ that are compatible with the observed data.

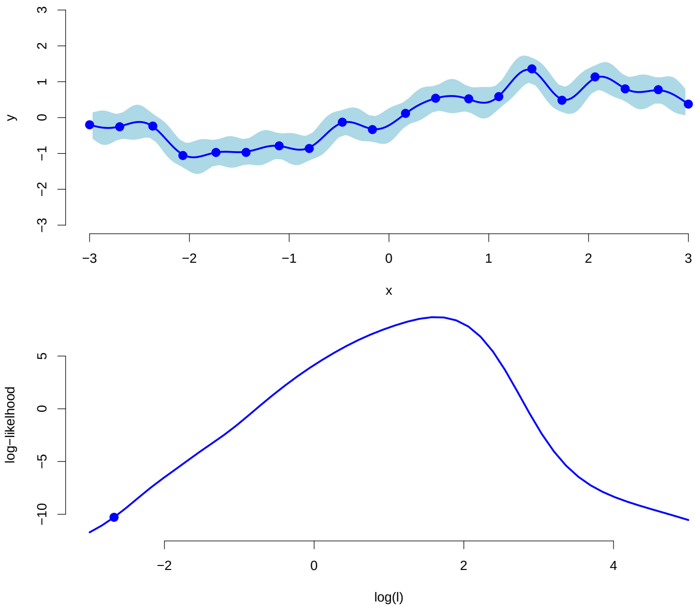
$$p(\theta|\mathbf{y}) = \frac{p(\mathbf{y}|\theta)p(\theta)}{p(\mathbf{y})}$$

what we know after seeing the data (*posterior*)	$\propto$	what the data tell us (likelihood)	$\times$	what we know before seeing the data (prior)

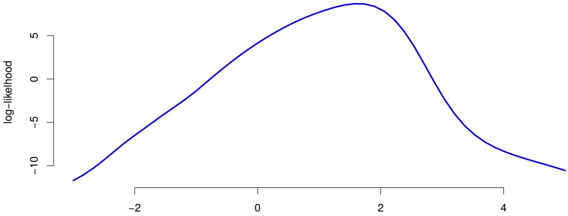
$$\begin{aligned} p(\mathbf{y}|\theta) &\equiv \text{how well does } \theta \text{ explain the observed data} \\ &= \mathcal{N}\left(\mathbf{y}|\mathbf{0}, \mathbf{\Sigma} + \mathbf{I}\sigma_y^2\right) \end{aligned}$$

Often, with a reasonable amount of data, maximizing $p(\mathbf{y}|\theta)$ w.r.t. θ gives good results as it favors the right model!

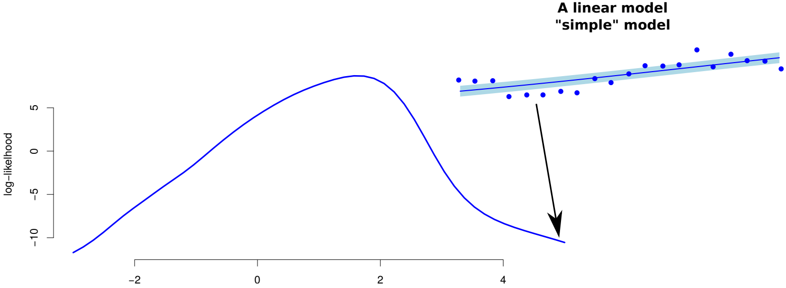
How do we choose the hyper-parameters?



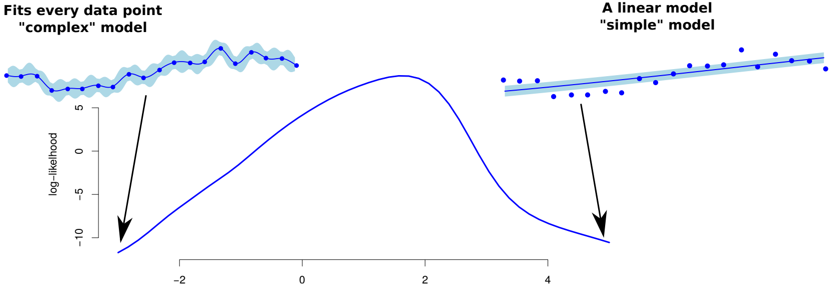
Why maximizing the likelihood is robust?



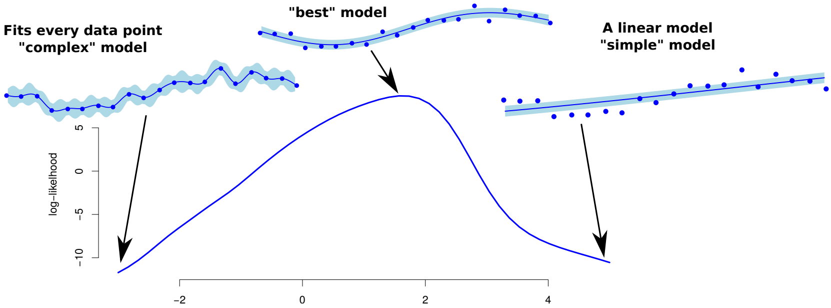
Why maximizing the likelihood is robust?



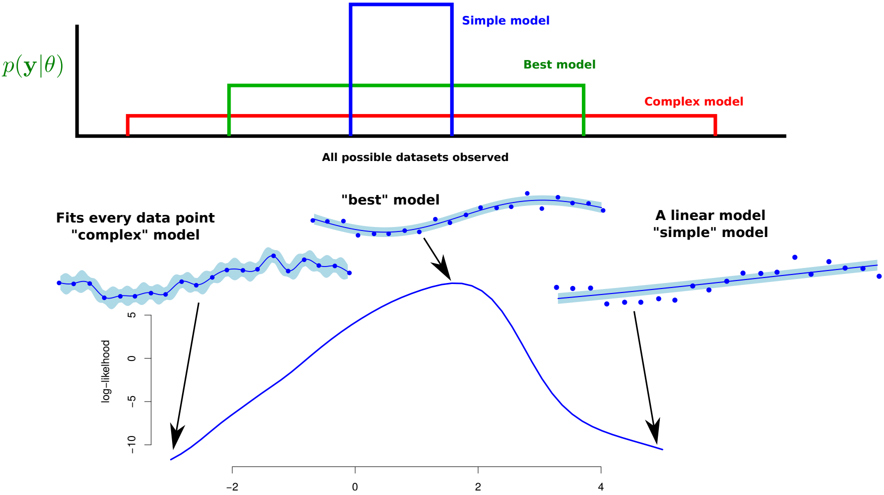
Why maximizing the likelihood is robust?



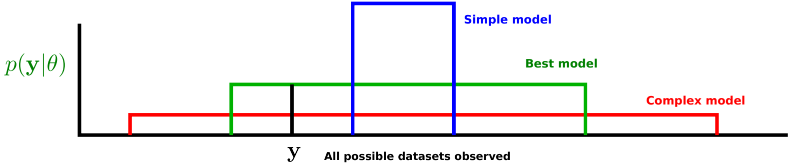
Why maximizing the likelihood is robust?

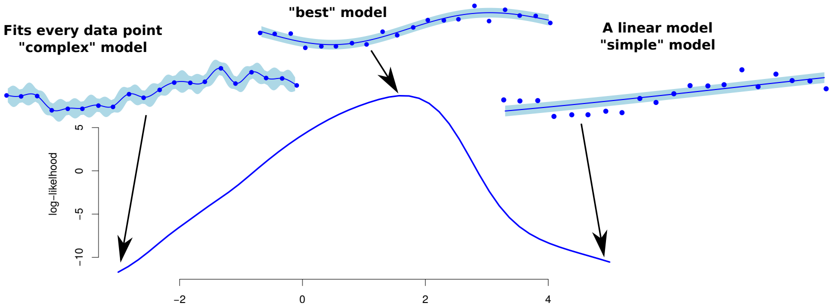


Why maximizing the likelihood is robust?

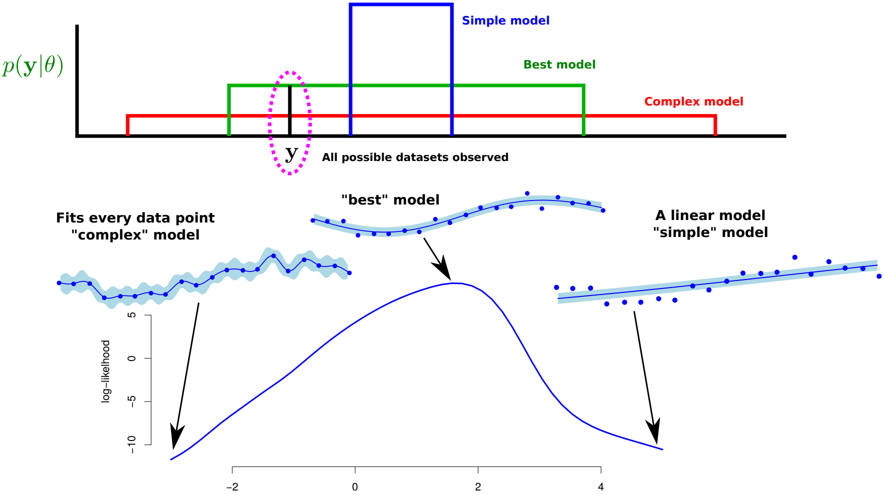


Why maximizing the likelihood is robust?





Why maximizing the likelihood is robust?

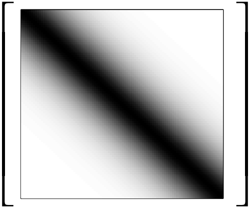


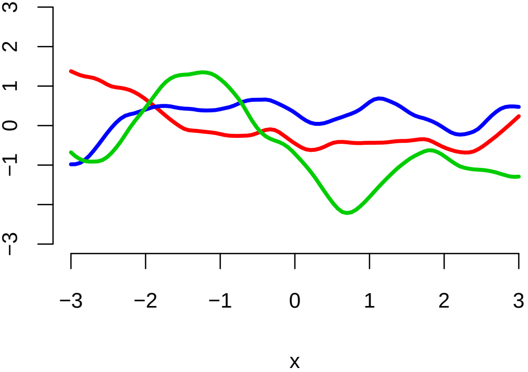
Covariance Functions: Matérn

$$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2^{1-\nu)}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu r}}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu} r}{l} \right)$$

Covariance Functions: Matérn

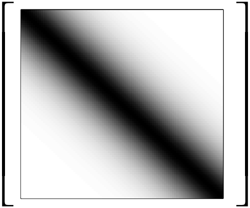
$$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2^{1-\nu)}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu r}}{l} \right)^{\nu} K_{\nu} \left(\frac{\sqrt{2\nu}r}{l} \right)$$

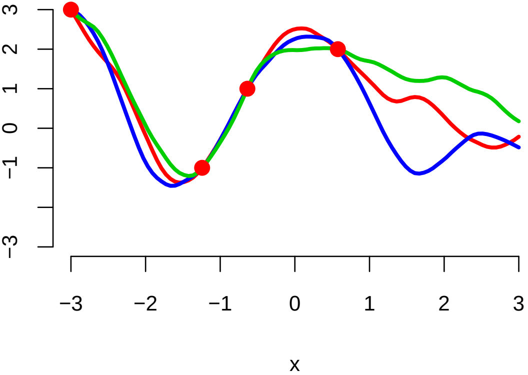
$\Sigma =$ 



Covariance Functions: Matérn

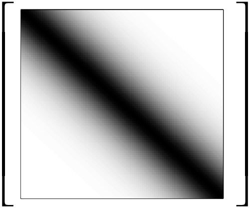
$$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2^{1-\nu)}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu r}}{l} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu r}}{l} \right)$$

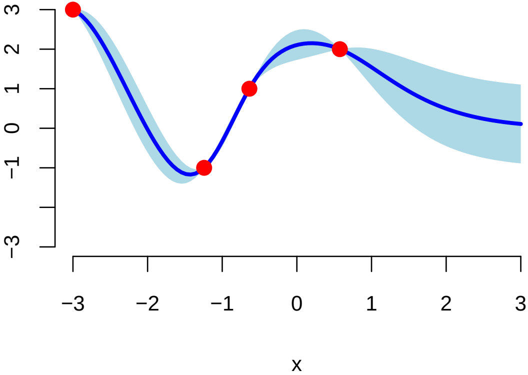
$\Sigma =$ 



Covariance Functions: Matérn

$$C(\mathbf{x},\mathbf{x}') = \sigma^2 \frac{2^{1-\nu)}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu r}}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu r}}{l}\right)$$

$\Sigma =$ 

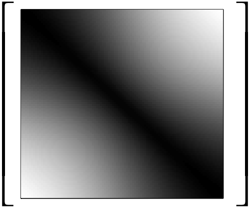


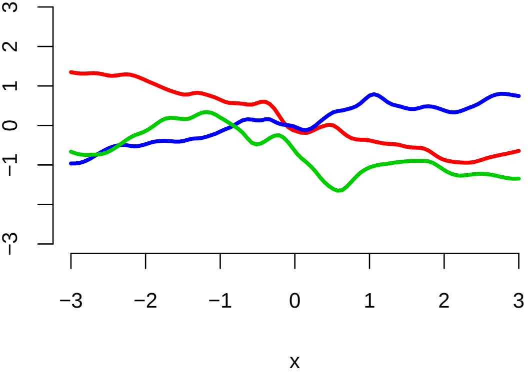
Covariance Functions: Neural Network

$$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2}{\pi} \sin^{-1} \left(\frac{\mathbf{x}^\mathsf{T} \mathbf{\Sigma} \mathbf{x}'}{\sqrt{(1 + 2\mathbf{x}^\mathsf{T} \mathbf{\Sigma} \mathbf{x}')(1 + 2\mathbf{x}^\mathsf{T} \mathbf{\Sigma} \mathbf{x}')}} \right)$$

Covariance Functions: Neural Network

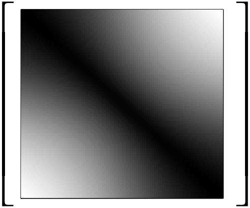
$$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2}{\pi} \sin^{-1} \left(\frac{\mathbf{x}^\top \boldsymbol{\Sigma} \mathbf{x}'}{\sqrt{(1 + 2\mathbf{x}^\top \boldsymbol{\Sigma} \mathbf{x}')(1 + 2\mathbf{x}^\top \boldsymbol{\Sigma} \mathbf{x}')}} \right)$$

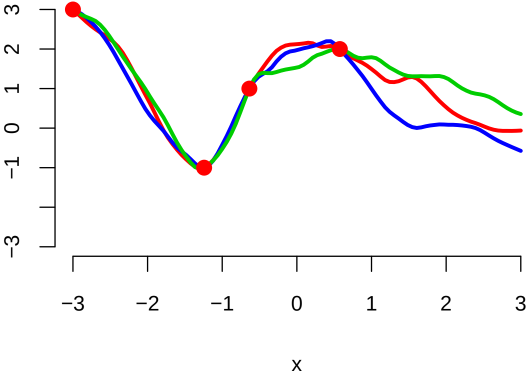
$\Sigma =$ 



Covariance Functions: Neural Network

$$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2}{\pi} \sin^{-1} \left(\frac{\mathbf{x}^{\mathsf{T}} \boldsymbol{\Sigma} \mathbf{x}'}{\sqrt{(1 + 2\mathbf{x}^{\mathsf{T}} \boldsymbol{\Sigma} \mathbf{x}')(1 + 2\mathbf{x}^{\mathsf{T}} \boldsymbol{\Sigma} \mathbf{x}')}} \right)$$

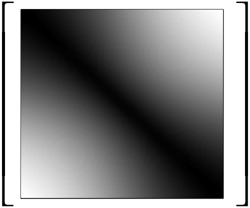
$\boldsymbol{\Sigma} =$ 

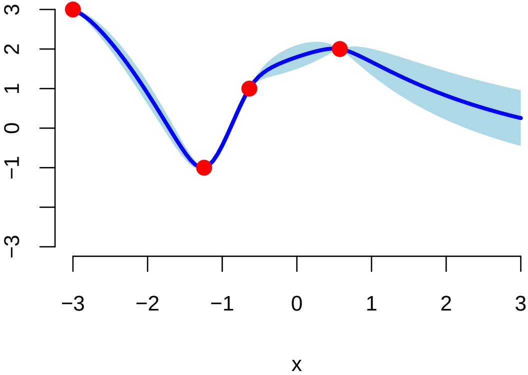


Covariance Functions: Neural Network

$$C(\mathbf{x}, \mathbf{x}') = \sigma^2 \frac{2}{\pi} \sin^{-1} \left(\frac{\mathbf{x}^\mathsf{T} \boldsymbol{\Sigma} \mathbf{x}'}{\sqrt{(1 + 2\mathbf{x}^\mathsf{T} \boldsymbol{\Sigma} \mathbf{x}')(1 + 2\mathbf{x}^\mathsf{T} \boldsymbol{\Sigma} \mathbf{x}')}} \right)$$

$\boldsymbol{\Sigma} =$





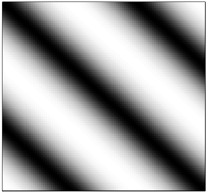
Covariance Functions: Periodic

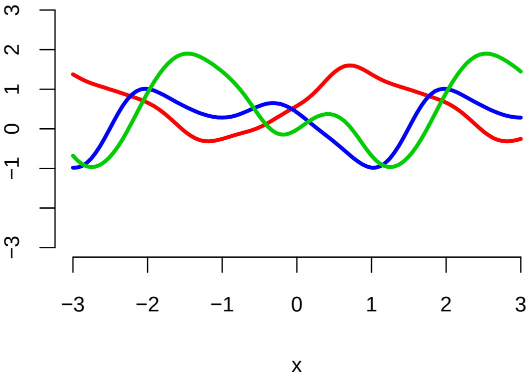
$$C(\mathbf{x}, \mathbf{x}') = \exp\left\{ -\frac{2\sin^2\left(\frac{|\mathbf{x}-\mathbf{x}'|}{2}\right)}{l^2} \right\}$$

Covariance Functions: Periodic

$$C(\mathbf{x}, \mathbf{x}') = \exp\left\{ -\frac{2\sin^2\left(\frac{|\mathbf{x}-\mathbf{x}'|}{2}\right)}{l^2} \right\}$$

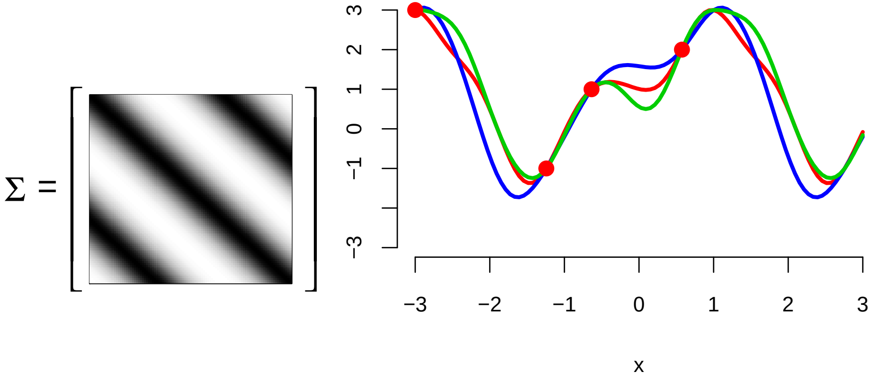
$$\Sigma = \left[\quad\right]$$





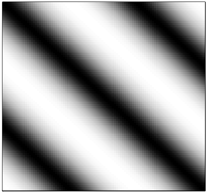
Covariance Functions: Periodic

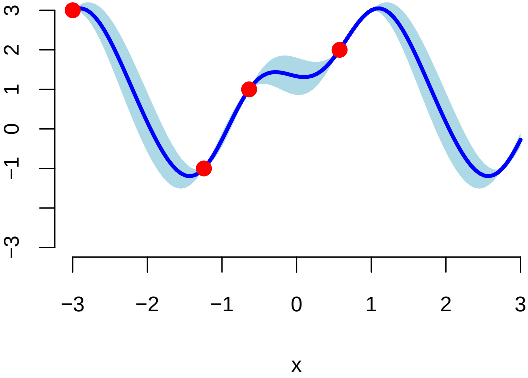
$$C(\mathbf{x}, \mathbf{x}') = \exp\left\{ -\frac{2\sin^2\left(\frac{|\mathbf{x}-\mathbf{x}'|}{2}\right)}{l^2} \right\}$$



Covariance Functions: Periodic

$$C(\mathbf{x}, \mathbf{x}') = \exp\left\{ -\frac{2\sin^2\left(\frac{|\mathbf{x}-\mathbf{x}'|}{2}\right)}{l^2} \right\}$$

$\Sigma =$ 

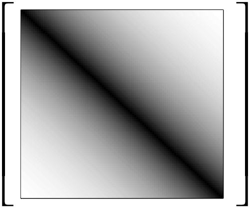


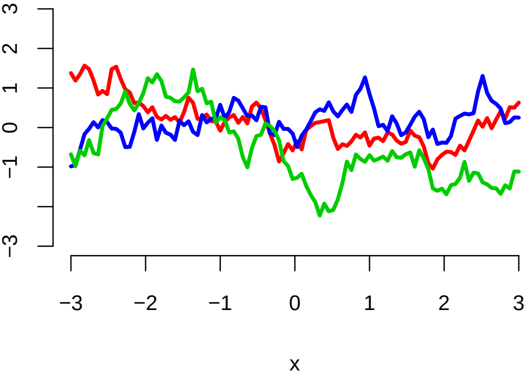
Covariance Functions: Ornstein-Uhlenbeck

$$C(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{|\mathbf{x} - \mathbf{x}'|}{2l^2}\right\}$$

Covariance Functions: Ornstein-Uhlenbeck

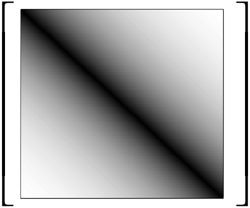
$$C(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{|\mathbf{x} - \mathbf{x}'|}{2l^2}\right\}$$

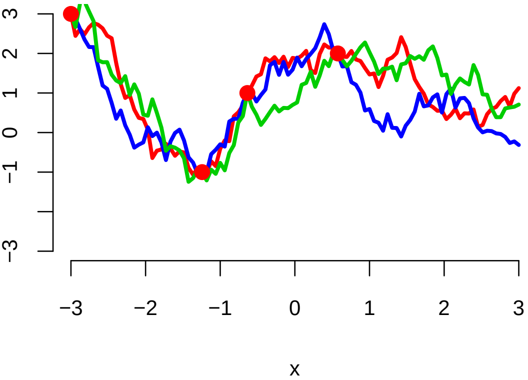
$\Sigma =$ 



Covariance Functions: Ornstein-Uhlenbeck

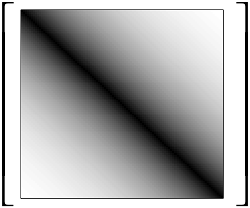
$$C(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{|\mathbf{x} - \mathbf{x}'|}{2l^2}\right\}$$

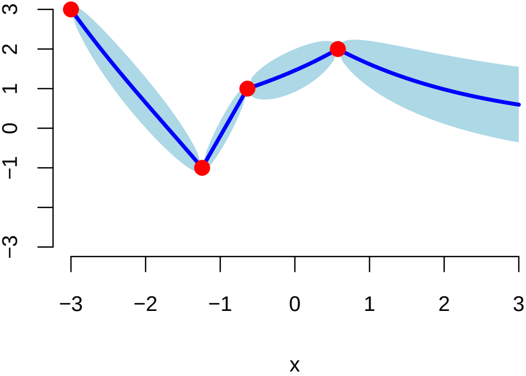
$\Sigma =$ 



Covariance Functions: Ornstein-Uhlenbeck

$$C(\mathbf{x}, \mathbf{x}') = \exp\left\{-\frac{|\mathbf{x} - \mathbf{x}'|}{2l^2}\right\}$$

$\Sigma =$ 



Summary about Covariance Functions

- Covariance functions include strong assumptions about $f(\mathbf{x})$.

Summary about Covariance Functions

- Covariance functions include strong assumptions about $f(\mathbf{x})$.
- Often the sq. exponential or Matérn work fine for regression.

Summary about Covariance Functions

- Covariance functions include strong assumptions about $f(\mathbf{x})$.
- Often the sq. exponential or Matérn work fine for regression.
- Covariance functions parameters allow to interpret the data.

Summary about Covariance Functions

- Covariance functions include strong assumptions about $f(\mathbf{x})$.
- Often the sq. exponential or Matérn work fine for regression.
- Covariance functions parameters allow to interpret the data.
- Covariance functions can be combined (sum $+$ and product $\times$).

Summary about Covariance Functions

- Covariance functions include strong assumptions about $f(\mathbf{x})$.
- Often the sq. exponential or Matérn work fine for regression.
- Covariance functions parameters allow to interpret the data.
- Covariance functions can be combined (sum $+$ and product $\times$).
- The likelihood $p(\mathbf{y})$ can *discriminate* among them (use with care).

Computational Cost of Gaussian Processes

The memory cost is in $\mathcal{O}(N^2)$ since we have to compute Σ.

Computational Cost of Gaussian Processes

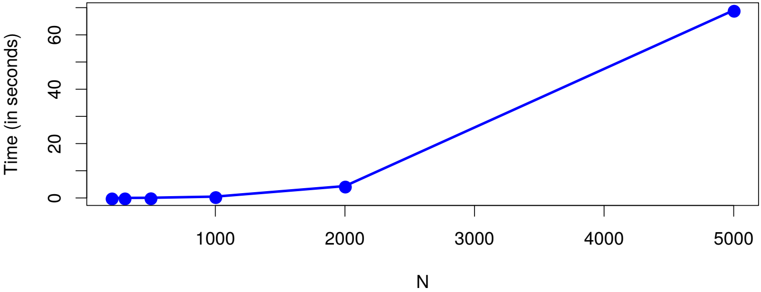
The memory cost is in $\mathcal{O}(N^2)$ since we have to compute $\mathbf{\Sigma}$.

The computational cost is in $\mathcal{O}(N^3)$ since we have to invert $\mathbf{\Sigma}$.

Computational Cost of Gaussian Processes

The memory cost is in $\mathcal{O}(N^2)$ since we have to compute Σ.

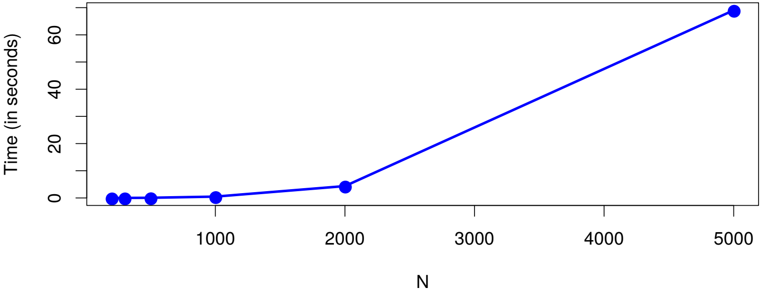
The computational cost is in $\mathcal{O}(N^3)$ since we have to invert Σ.



Computational Cost of Gaussian Processes

The memory cost is in $\mathcal{O}(N^2)$ since we have to compute Σ.

The computational cost is in $\mathcal{O}(N^3)$ since we have to invert Σ.



We can handle just a few thousand data instances at most!

Improving the Cost of Gaussian Processes

GPs are non-parametric models whose flexibility grows with N!

Improving the Cost of Gaussian Processes

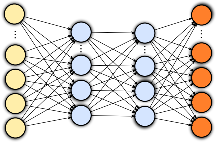
GPs are non-parametric models whose flexibility grows with N!

Idea: go back to the parametric model, but in such a way that we can still make inference easily!

Improving the Cost of Gaussian Processes

GPs are non-parametric models whose flexibility grows with N!

Idea: go back to the parametric model, but in such a way that we can still make inference easily!

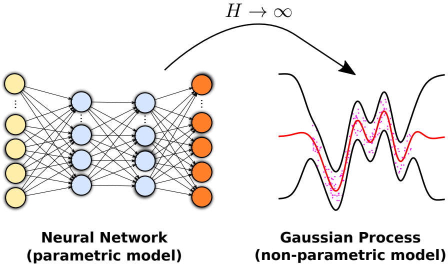


Neural Network
(parametric model)

Improving the Cost of Gaussian Processes

GPs are non-parametric models whose flexibility grows with N!

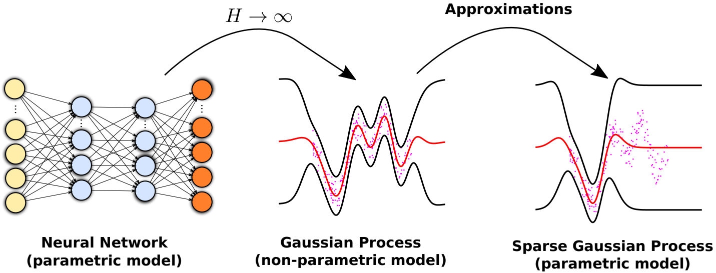
Idea: go back to the parametric model, but in such a way that we can still make inference easily!



Improving the Cost of Gaussian Processes

GPs are non-parametric models whose flexibility grows with N!

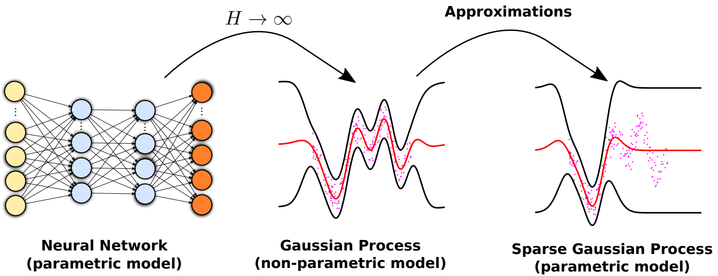
Idea: go back to the parametric model, but in such a way that we can still make inference easily!



Improving the Cost of Gaussian Processes

GPs are non-parametric models whose flexibility grows with N!

Idea: go back to the parametric model, but in such a way that we can still make inference easily!

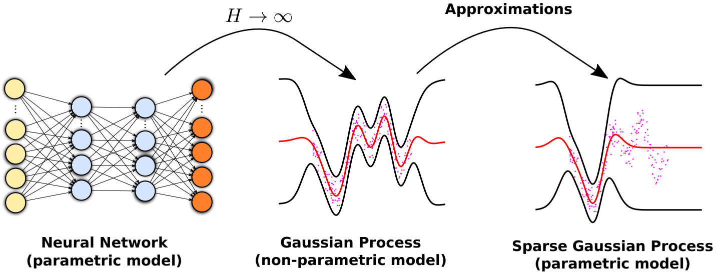


Approximations based on inducing points:

Improving the Cost of Gaussian Processes

GPs are non-parametric models whose flexibility grows with N!

Idea: go back to the parametric model, but in such a way that we can still make inference easily!



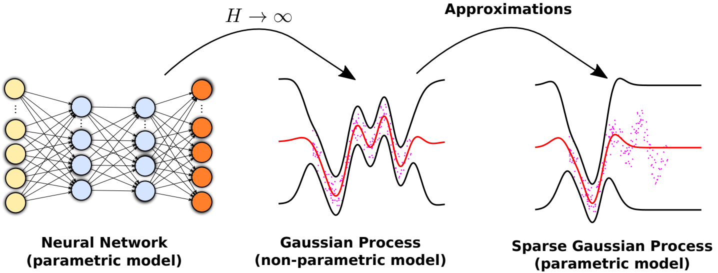
Approximations based on inducing points:

- **FITC**: changes the GP model to remove some dependencies!

Improving the Cost of Gaussian Processes

GPs are non-parametric models whose flexibility grows with N!

Idea: go back to the parametric model, but in such a way that we can still make inference easily!



Approximations based on inducing points:

- **FITC**: changes the GP model to remove some dependencies!
- **VFE**: does approximate inference with a simplified distribution q.

Full Independent Training Conditional (FITC)

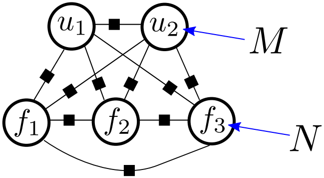
1. Extend model with $M \ll N$ inducing points and outputs at $\overline{\mathbf{X}}$.

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\left[\begin{array}{c} \mathbf{f} \\ \mathbf{u} \end{array}\right] \middle| \left[\begin{array}{c} \mathbf{0} \\ \mathbf{0} \end{array}\right], \left[\begin{array}{cc} \mathbf{K_{ff}} & \mathbf{K_{fu}} \\ \mathbf{K_{uf}} & \mathbf{K_{uu}} \end{array}\right]\right)$$

Full Independent Training Conditional (FITC)

1. Extend model with $M \ll N$ inducing points and outputs at $\overline{\mathbf{X}}$.

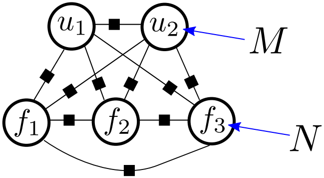
$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\left[\begin{array}{c} \mathbf{f} \\ \mathbf{u} \end{array}\right] \middle| \left[\begin{array}{c} \mathbf{0} \\ \mathbf{0} \end{array}\right], \left[\begin{array}{cc} \mathbf{K}_{\mathbf{ff}} & \mathbf{K}_{\mathbf{fu}} \\ \mathbf{K}_{\mathbf{uf}} & \mathbf{K}_{\mathbf{uu}} \end{array}\right]\right)$$



Full Independent Training Conditional (FITC)

1. Extend model with $M \ll N$ inducing points and outputs at $\overline{\mathbf{X}}$.

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K_{ff}} & \mathbf{K_{fu}} \\ \mathbf{K_{uf}} & \mathbf{K_{uu}} \end{bmatrix} \right)$$



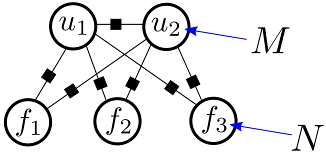
2. Introduce conditional independences:

$$p(\mathbf{f}|\mathbf{u}) = \prod_{i=1}^{N} p(f_i|\mathbf{u})$$

Full Independent Training Conditional (FITC)

1. Extend model with $M \ll N$ inducing points and outputs at $\overline{\mathbf{X}}$.

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\left[\begin{array}{c} \mathbf{f} \\ \mathbf{u} \end{array} \right] \middle| \left[\begin{array}{c} \mathbf{0} \\ \mathbf{0} \end{array} \right], \left[\begin{array}{cc} \mathbf{K_{ff}} & \mathbf{K_{fu}} \\ \mathbf{K_{uf}} & \mathbf{K_{uu}} \end{array} \right] \right)$$

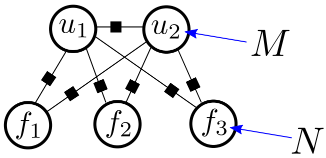


2. Introduce conditional independences:

$$p(\mathbf{f}|\mathbf{u}) = \prod_{i=1}^{N} p(f_i|\mathbf{u})$$

Full Independent Training Conditional (FITC)

1. Extend model with $M \ll N$ inducing points and outputs at $\bar{\mathbf{X}}$.



$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{K_{ff}} & \mathbf{K_{fu}} \\ \mathbf{K_{uf}} & \mathbf{K_{uu}} \end{bmatrix} \right)$$

2. Introduce conditional independences:

$$p(\mathbf{f}|\mathbf{u}) = \prod_{i=1}^{N} p(f_i|\mathbf{u})$$

3. Marginalize $\mathbf{u}$ to obtain an approximate GP prior for $\mathbf{f}$.

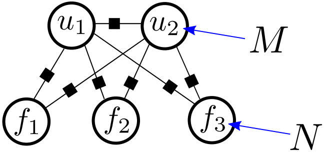
$$p(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u})p(\mathbf{u})d\mathbf{u} = \prod_{i=1}^{N} p(f_i|\mathbf{u})p(\mathbf{u})d\mathbf{u} = \mathcal{N}(\mathbf{f}|0, \tilde{\mathbf{K}}_{\mathbf{ff}})$$

where $\tilde{\mathbf{K}}_{\mathbf{ff}} = \mathbf{D} + \mathbf{Q_{ff}}$ with $\mathbf{D}$ diagonal and $\mathbf{Q_{ff}} = \mathbf{K_{fu}}\mathbf{K_{uu}}^{-1}\mathbf{K_{uf}}$ of rank M.

Full Independent Training Conditional (FITC)

5. We make the prediction of $f^\star$ at $\mathbf{x}^\star$ by considering the approximate GP prior:

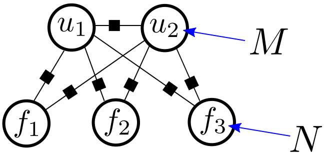
$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \tilde{\mathbf{K}}_{\mathbf{ff}} & \mathbf{Q}_{\mathbf{ff}^\star} \\ \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} & \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} \end{bmatrix} \right)$$

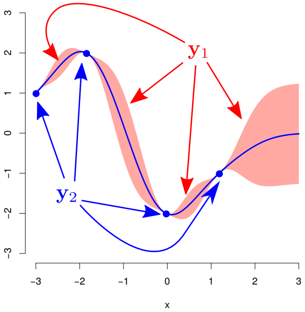


Full Independent Training Conditional (FITC)

5. We make the prediction of $f^\star$ at $\mathbf{x}^\star$ by considering the approximate GP prior:

$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \tilde{\mathbf{K}}_{\mathbf{ff}} & \mathbf{Q}_{\mathbf{ff}^\star} \\ \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} & \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} \end{bmatrix} \right)$$



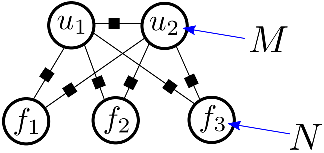


$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\mathsf{T} & \mathbf{B} \end{bmatrix} \right),$$

$$p(\mathbf{y}_1 | \mathbf{y}_2) = \frac{p(\mathbf{y}_1, \mathbf{y}_2)}{p(\mathbf{y}_2)},$$

Full Independent Training Conditional (FITC)

5. We make the prediction of $f^\star$ at $\mathbf{x}^\star$ by considering the approximate GP prior:

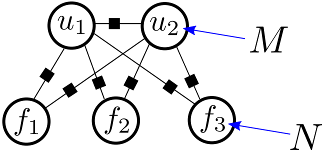


$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \tilde{\mathbf{K}}_{\mathbf{ff}} & \mathbf{Q}_{\mathbf{ff}^\star} \\ \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} & \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} \end{bmatrix} \right)$$

$$p(\mathbf{f}^\star | \mathbf{f}) = \mathcal{N}\left(\mathbf{f}^\star \middle| \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \tilde{\mathbf{K}}_{\mathbf{ff}}^{-1} \mathbf{f}, \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} - \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}^{\mathsf{T}} \tilde{\mathbf{K}}_{\mathbf{ff}}^{-1} \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}\right)$$

Full Independent Training Conditional (FITC)

5. We make the prediction of $f^\star$ at $\mathbf{x}^\star$ by considering the approximate GP prior:



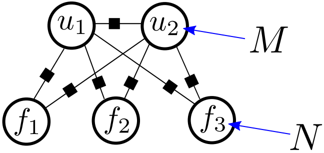
$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \tilde{\mathbf{K}}_{\mathbf{ff}} & \mathbf{Q}_{\mathbf{ff}^\star} \\ \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} & \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} \end{bmatrix} \right)$$

$$p(\mathbf{f}^\star | \mathbf{f}) = \mathcal{N}\left(\mathbf{f}^\star \middle| \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \tilde{\mathbf{K}}_{\mathbf{ff}}^{-1} \mathbf{f}, \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} - \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}^\top \tilde{\mathbf{K}}_{\mathbf{ff}}^{-1} \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \right)$$

Due to the structure in $\tilde{\mathbf{K}}_{\mathbf{ff}}$ all computations have cost in $\mathcal{O}(NM^2)$.

Full Independent Training Conditional (FITC)

5. We make the prediction of $f^\star$ at $\mathbf{x}^\star$ by considering the approximate GP prior:



$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \tilde{\mathbf{K}}_{\mathbf{ff}} & \mathbf{Q}_{\mathbf{ff}^\star} \\ \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} & \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} \end{bmatrix} \right)$$

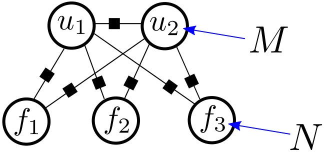
$$p(\mathbf{f}^\star|\mathbf{f}) = \mathcal{N}\left(\mathbf{f}^\star \middle| \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \tilde{\mathbf{K}}_{\mathbf{ff}}^{-1} \mathbf{f}, \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} - \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}^\top \tilde{\mathbf{K}}_{\mathbf{ff}}^{-1} \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \right)$$

Due to the structure in $\tilde{\mathbf{K}}_{\mathbf{ff}}$ all computations have cost in $\mathcal{O}(NM^2)$.

6. How do we find the location of the inducing points $\overline{\mathbf{X}}$?

Full Independent Training Conditional (FITC)

5. We make the prediction of $f^\star$ at $\mathbf{x}^\star$ by considering the approximate GP prior:



$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \tilde{\mathbf{K}}_{\mathbf{ff}} & \mathbf{Q}_{\mathbf{ff}^\star} \\ \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} & \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} \end{bmatrix} \right)$$

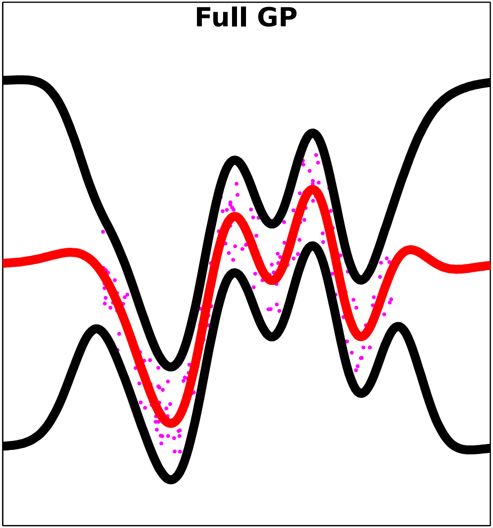
$$p(\mathbf{f}^\star|\mathbf{f}) = \mathcal{N}\left(\mathbf{f}^\star | \, \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \tilde{\mathbf{K}}_{\mathbf{ff}}^{-1} \mathbf{f}, \mathbf{K}_{\mathbf{f}^\star\mathbf{f}^\star} - \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}^\mathsf{T} \tilde{\mathbf{K}}_{\mathbf{ff}}^{-1} \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}\right)$$

Due to the structure in $\tilde{\mathbf{K}}_{\mathbf{ff}}$ all computations have cost in $\mathcal{O}(NM^2)$.

6. How do we find the location of the inducing points $\overline{\mathbf{X}}$?

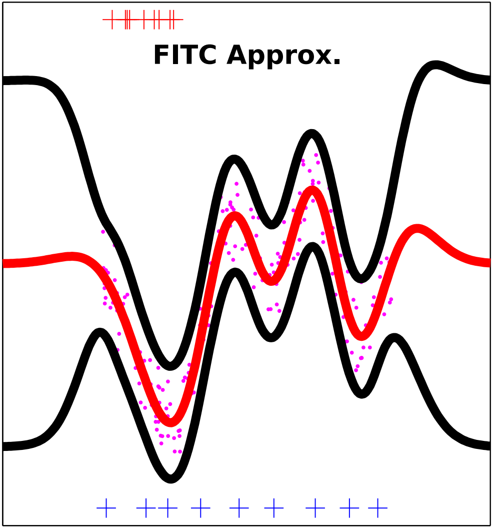
Simply treat them as prior parameters and maximize the approximate likelihood $p(\mathbf{f}|\mathbf{0}, \tilde{\mathbf{K}}_{\mathbf{ff}})$!

Full Independent Training Conditional (FITC)



(Snelson & Gahramani, 2006)

Full Independent Training Conditional (FITC)

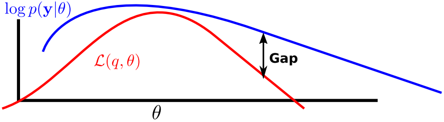


(Snelson & Gahramani, 2006)

Variational Free Energy (VFE)

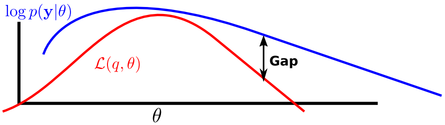
Lower bound the log-likelihood:

$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) d\mathbf{f} d\mathbf{u}$$



Variational Free Energy (VFE)

Lower bound the log-likelihood:

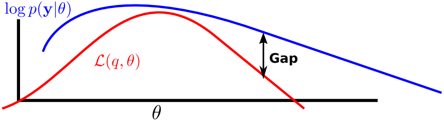


$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) d\mathbf{f} d\mathbf{u}$$

$$= \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) \frac{q(\mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u}$$

Variational Free Energy (VFE)

Lower bound the log-likelihood:



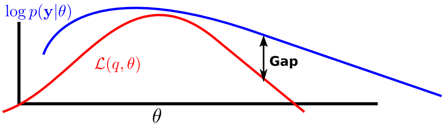
$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) d\mathbf{f} d\mathbf{u}$$

$$= \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) \frac{q(\mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} \geq \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} \equiv \mathcal{L}(q, \theta)$$

Variational Free Energy (VFE)

Lower bound the log-likelihood:

$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) d\mathbf{f} d\mathbf{u}$$

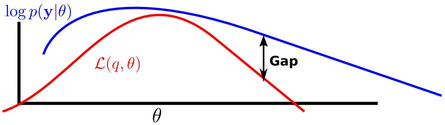


$$= \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) \frac{q(\mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} \geq \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} \equiv \mathcal{L}(q, \theta)$$

$$\mathcal{L}(q, \theta) = \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} = \log p(\mathbf{y}|\theta) - \mathbf{KL}[q(\mathbf{f}, \mathbf{u})|p(\mathbf{f}, \mathbf{u}|\mathbf{y})]$$

Variational Free Energy (VFE)

Lower bound the log-likelihood:



$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) d\mathbf{f} d\mathbf{u}$$

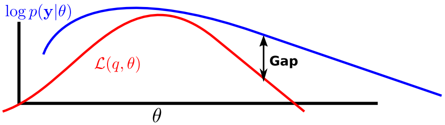
$$= \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) \frac{q(\mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} \geq \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} \equiv \mathcal{L}(q, \theta)$$

$$\mathcal{L}(q, \theta) = \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} = \log p(\mathbf{y}|\theta) - \mathbf{KL}[q(\mathbf{f}, \mathbf{u}) | p(\mathbf{f}, \mathbf{u}|\mathbf{y})]$$

KL $\equiv$ Kullback-Leibler divergence

Variational Free Energy (VFE)

Lower bound the log-likelihood:



$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y},\mathbf{f},\mathbf{u}|\theta) d\mathbf{f} d\mathbf{u}$$

$$= \log \int p(\mathbf{y},\mathbf{f},\mathbf{u}|\theta) \frac{q(\mathbf{f},\mathbf{u})}{q(\mathbf{f},\mathbf{u})} d\mathbf{f} d\mathbf{u} \geq \int q(\mathbf{f},\mathbf{u}) \log \frac{p(\mathbf{y},\mathbf{f},\mathbf{u}|\theta)}{q(\mathbf{f},\mathbf{u})} d\mathbf{f} d\mathbf{u} \equiv \mathcal{L}(q,\theta)$$

$$\mathcal{L}(q,\theta) = \int q(\mathbf{f},\mathbf{u}) \log \frac{p(\mathbf{y},\mathbf{f},\mathbf{u}|\theta)}{q(\mathbf{f},\mathbf{u})} d\mathbf{f} d\mathbf{u} = \log p(\mathbf{y}|\theta) - \mathbf{KL}[q(\mathbf{f},\mathbf{u})|p(\mathbf{f},\mathbf{u}|\mathbf{y})]$$

KL $\equiv$ Kullback-Leibler divergence

By maximizing $\mathcal{L}(q,\theta)$ w.r.t q we are enforcing that $q(\mathbf{f},\mathbf{u})$ looks similar to $p(\mathbf{f},\mathbf{u}|\mathbf{y})$ in terms of the KL!

Variational Free Energy (VFE)

Consider the following approximate distribution:

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u}) \; q(\mathbf{u}) = p(\mathbf{f}|\mathbf{u}) \; \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S})$$

Variational Free Energy (VFE)

Consider the following approximate distribution:

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})\, q(\mathbf{u}) = p(\mathbf{f}|\mathbf{u})\, \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S})$$

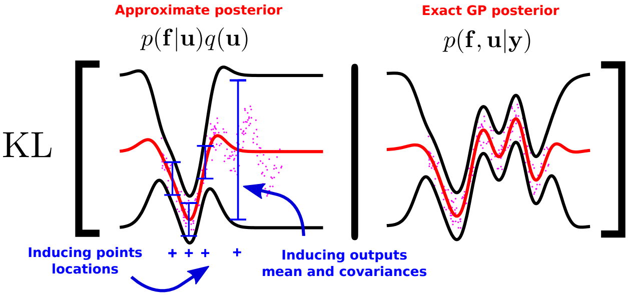
- Fixed
- Tunable

Variational Free Energy (VFE)

Consider the following approximate distribution:

$$q(\mathbf{f}, \mathbf{u}) = \boxed{p(\mathbf{f}|\mathbf{u})}\ \boxed{q(\mathbf{u})} = \boxed{p(\mathbf{f}|\mathbf{u})}\ \boxed{\mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S})}$$

- Fixed
- Tunable

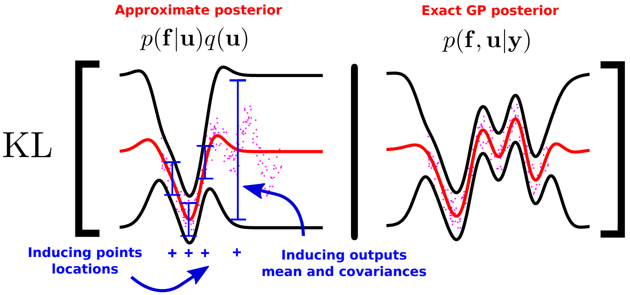


Variational Free Energy (VFE)

Consider the following approximate distribution:

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u}) \; q(\mathbf{u}) = p(\mathbf{f}|\mathbf{u}) \; \mathcal{N}(\mathbf{u}|\mathbf{m}, \mathbf{S})$$

- Fixed
- Tunable



The inducing points are now parameters of the approx. dist. q!

Variational Free Energy (VFE)

Plugging $q(\mathbf{f}, \mathbf{u})$ into the lower bound we have:

Variational Free Energy (VFE)

Plugging $q(\mathbf{f}, \mathbf{u})$ into the lower bound we have:

$$
\begin{aligned}
\mathcal{L}(q, \theta) &= \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} \\
&= \int p(\mathbf{f}|\mathbf{u}) q(\mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f}, \theta) p(\mathbf{f}|\mathbf{u}) p(\mathbf{u})}{p(\mathbf{f}|\mathbf{u}) q(\mathbf{u})} d\mathbf{f} d\mathbf{u}
\end{aligned}
$$

Variational Free Energy (VFE)

Plugging $q(\mathbf{f}, \mathbf{u})$ into the lower bound we have:

$$\begin{aligned} \mathcal{L}(q, \theta) &= \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u} | \theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} \\ &= \int p(\mathbf{f} | \mathbf{u}) q(\mathbf{u}) \log \frac{p(\mathbf{y} | \mathbf{f}, \theta) \cancel{p(\mathbf{f} | \mathbf{u})} p(\mathbf{u})}{\cancel{p(\mathbf{f} | \mathbf{u})} q(\mathbf{u})} d\mathbf{f} d\mathbf{u} \end{aligned}$$

Variational Free Energy (VFE)

Plugging $q(\mathbf{f}, \mathbf{u})$ into the lower bound we have:

$$\mathcal{L}(q, \theta) = \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u}$$

$$= \int p(\mathbf{f}|\mathbf{u}) q(\mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f}, \theta) \cancel{p(\mathbf{f}|\mathbf{u})} p(\mathbf{u})}{\cancel{p(\mathbf{f}|\mathbf{u})} q(\mathbf{u})} d\mathbf{f} d\mathbf{u}$$

$$\mathcal{L}(q, \theta) = \boxed{\mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f}, \theta)]} - \boxed{\mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})]}$$

- Mean squared prediction error
- KL between Gaussians

Variational Free Energy (VFE)

Plugging $q(\mathbf{f}, \mathbf{u})$ into the lower bound we have:

$$\mathcal{L}(q, \theta) = \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u}$$

$$= \int p(\mathbf{f}|\mathbf{u}) q(\mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f}, \theta) \cancel{p(\mathbf{f}|\mathbf{u})} p(\mathbf{u})}{\cancel{p(\mathbf{f}|\mathbf{u})} q(\mathbf{u})} d\mathbf{f} d\mathbf{u}$$

$$\mathcal{L}(q, \theta) = \boxed{\mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f}, \theta)]} - \boxed{\mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})]}$$

- Mean squared prediction error
- KL between Gaussians

- No change in the model is made and the cost is in $\mathcal{O}(M^2 N)$!

Variational Free Energy (VFE)

Plugging $q(\mathbf{f}, \mathbf{u})$ into the lower bound we have:

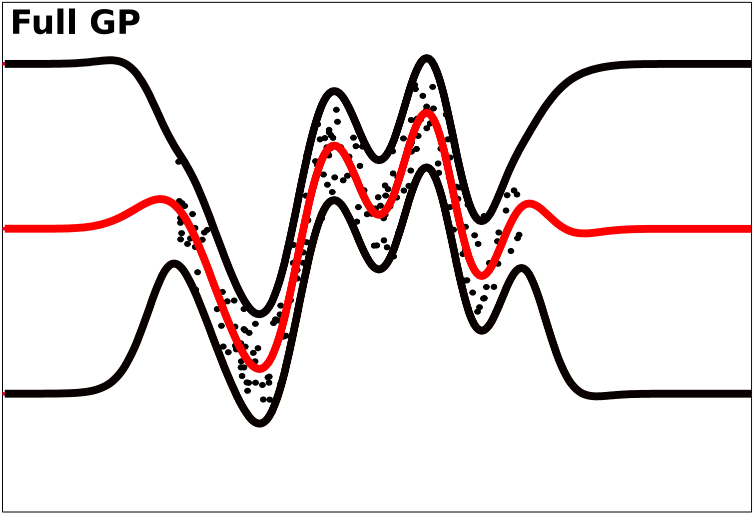
$$\begin{aligned}\mathcal{L}(q, \theta) &= \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} \\ &= \int p(\mathbf{f}|\mathbf{u}) q(\mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f}, \theta) \cancel{p(\mathbf{f}|\mathbf{u})} p(\mathbf{u})}{\cancel{p(\mathbf{f}|\mathbf{u})} q(\mathbf{u})} d\mathbf{f} d\mathbf{u}\end{aligned}$$

$$\mathcal{L}(q, \theta) = \boxed{\mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f}, \theta)]} - \boxed{\mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})]}$$

- Mean squared prediction error
- KL between Gaussians

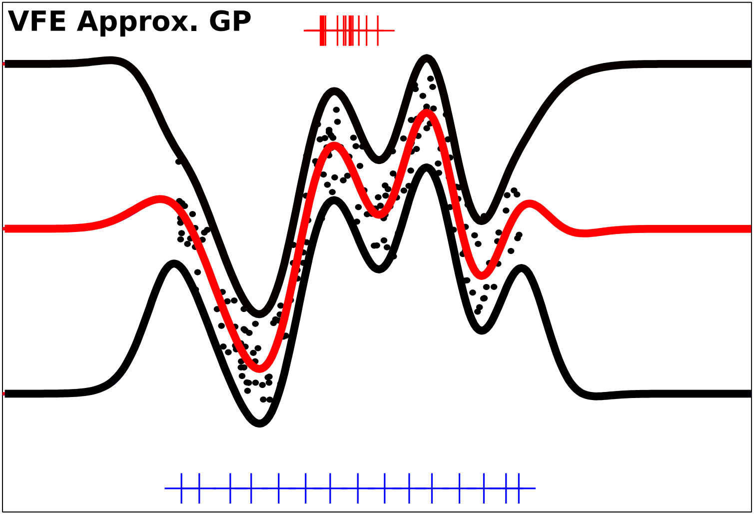
- No change in the model is made and the cost is in $\mathcal{O}(M^2 N)$!
- Predictions are made using $p(\mathbf{f}^{\star}|\mathbf{u}) q(\mathbf{u})$ marginalizing out $\mathbf{u}$.

Variational Free Energy (VFE)



(Titsias, 2009)

Variational Free Energy (VFE)



(Titsias, 2009)

FITC vs. VFE

Two approaches:

FITC vs. VFE

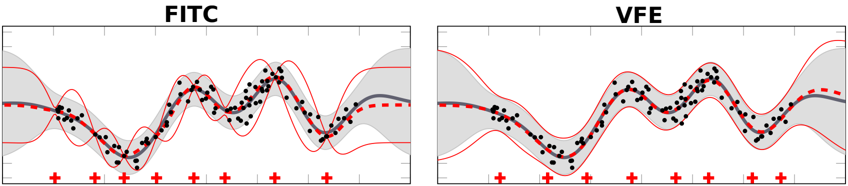
Two approaches:

- FITC: optimize the marginal likelihood of an approximate GP model.
- VFE: maximize fidelity to the original exact GP.

FITC vs. VFE

Two approaches:

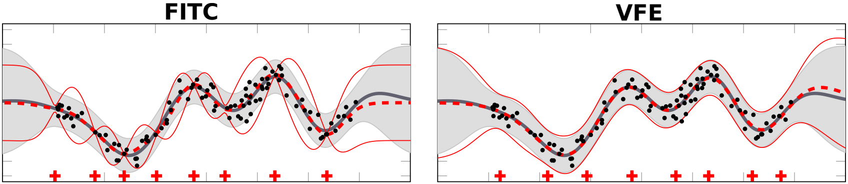
- FITC: optimize the marginal likelihood of an approximate GP model.
- VFE: maximize fidelity to the original exact GP.



FITC vs. VFE

Two approaches:

- FITC: optimize the marginal likelihood of an approximate GP model.
- VFE: maximize fidelity to the original exact GP.



- FITC: less local optima and easier to optimize, also less accurate.
- VFE: more accurate, more local optima, more difficult to optimize.

(Bui et al., 2017) (Bauer et al., 2016)

GPs for Big Data

Can we further improve the computational cost in $\mathcal{O}(NM^2)$?

GPs for Big Data

Can we further improve the computational cost in $\mathcal{O}(NM^2)$?

Minibatch training in NN allows to scale to massive datasets!

GPs for Big Data

Can we further improve the computational cost in $\mathcal{O}(NM^2)$?

Minibatch training in NN allows to scale to massive datasets!

Straight forward to do that in the VFE approach:

$$\begin{aligned}
\mathcal{L}(q,\theta) &= \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f},\theta)] - \mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})] \\
&= \sum_{i=1}^{N} \mathbb{E}_{q(f_i)}[\log p(y_i|f_i,\theta)] - \mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})] \\
&\approx \frac{B}{N} \sum_{i\in\mathcal{B}} \mathbb{E}_{q(f_i)}[\log p(y_i|f_i,\theta)] - \mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})]
\end{aligned}$$

GPs for Big Data

Can we further improve the computational cost in $\mathcal{O}(NM^2)$?

Minibatch training in NN allows to scale to massive datasets!

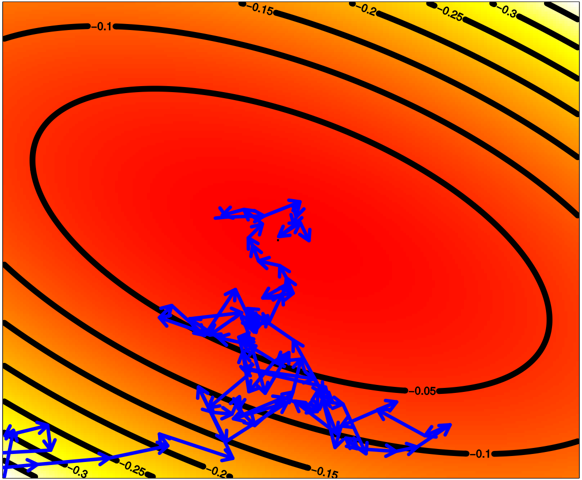
Straight forward to do that in the VFE approach:

$$
\begin{aligned}
\mathcal{L}(q,\theta) &= \mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f},\theta)] - \mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})] \\
&= \sum_{i=1}^{N} \mathbb{E}_{q(f_i)}[\log p(y_i|f_i,\theta)] - \mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})] \\
&\approx \frac{B}{N} \sum_{i\in\mathcal{B}} \mathbb{E}_{q(f_i)}[\log p(y_i|f_i,\theta)] - \mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})]
\end{aligned}
$$

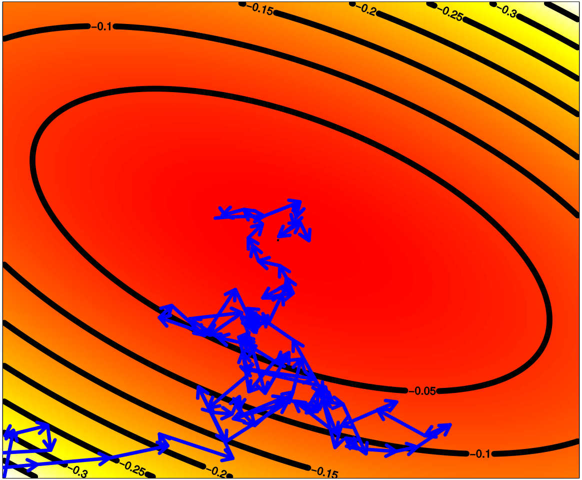
The training cost goes down to $\mathcal{O}(M^3)$ which allows to address datasets with millions of instances!

(Hensman et al., 2013)

GPs for Big Data

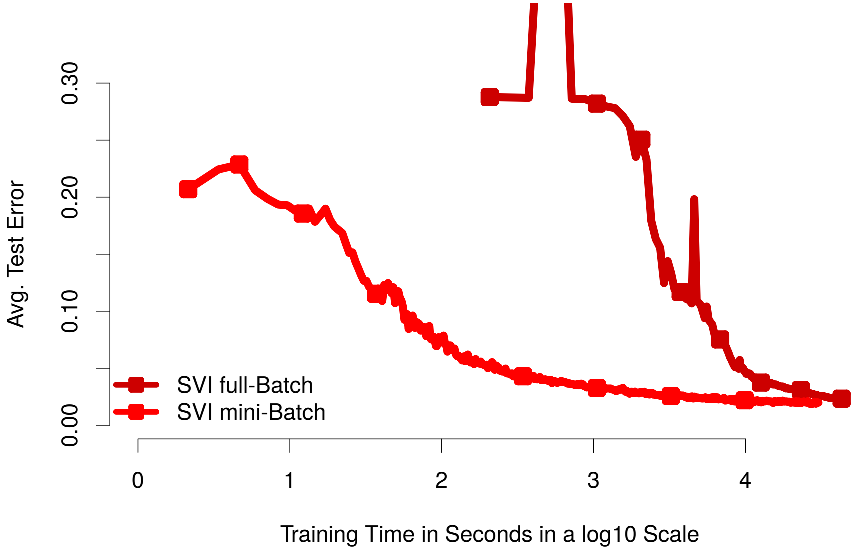


GPs for Big Data



To converge to a local neighborhood of the optimum stochastic methods require an estimate of the gradient which can be very cheap!

GPs for Big Data



(Hernández-Lobato, 2015)

Summary so Far about GPs

Summary so Far about GPs

Advantages of GPs:

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Disadvantages of GPs:

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Disadvantages of GPs:

- Strong assumptions made about $f(\mathbf{x})$!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Disadvantages of GPs:

- Strong assumptions made about $f(\mathbf{x})$!
- The predictive distribution is always Gaussian!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

Disadvantages of GPs:

- Strong assumptions made about $f(\mathbf{x})$!
- The predictive distribution is always Gaussian!
- Do not learn specific features to represent the observed data!

Summary so Far about GPs

Advantages of GPs:

- Non-parametric models!
- Exact Bayesian inference is tractable!
- They scale to very large datasets!
- Easy to introduce prior knowledge!

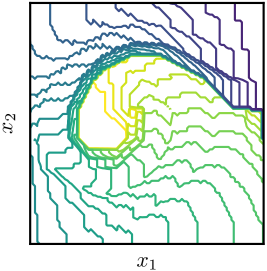
Disadvantages of GPs:

- Strong assumptions made about $f(\mathbf{x})$!
- The predictive distribution is always Gaussian!
- Do not learn specific features to represent the observed data!

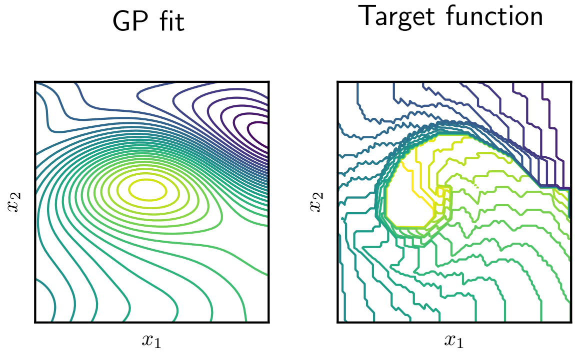
Deep GPs constitute a nice alternative to address these issues!

Motivation for Deep Gaussian Processes

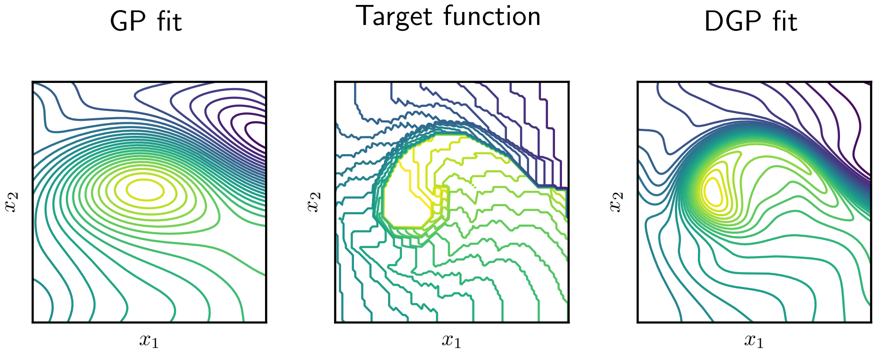
Target function



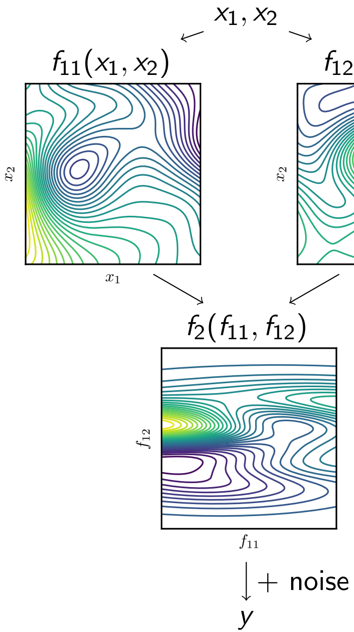
Motivation for Deep Gaussian Processes



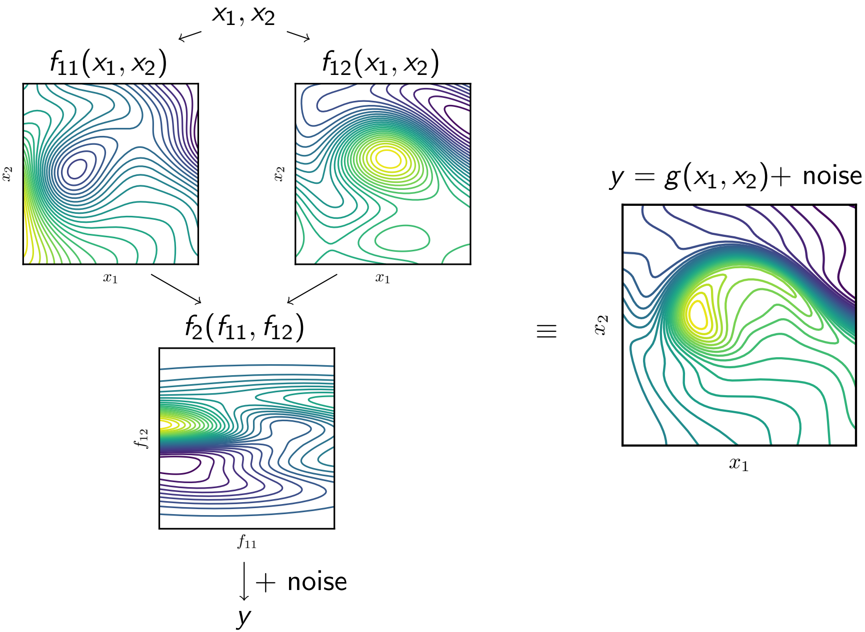
Motivation for Deep Gaussian Processes



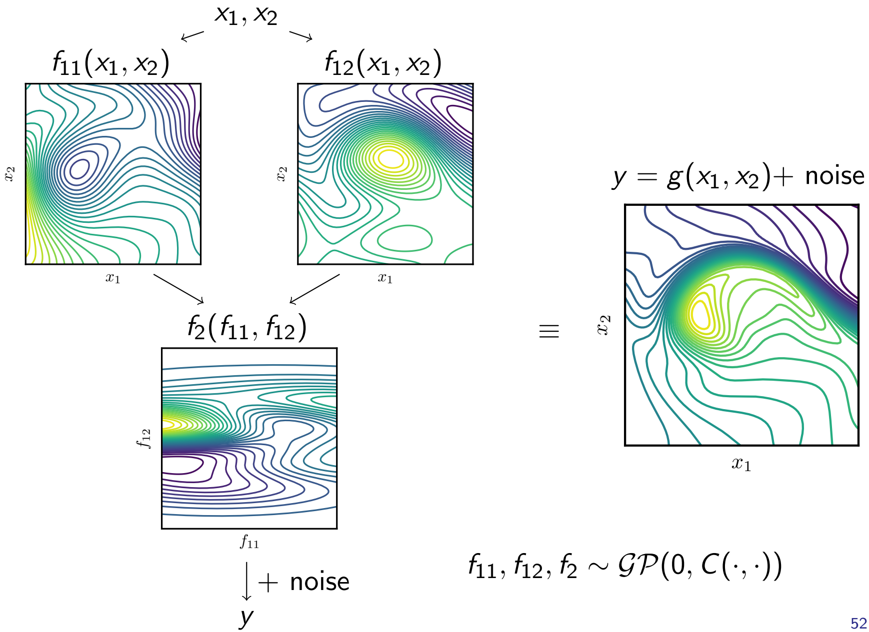
How do deep GPs work?



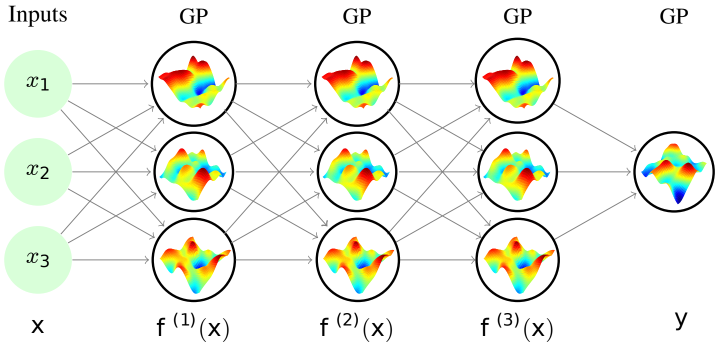
How do deep GPs work?



How do deep GPs work?



Deep GPs as Deep Neural Networks

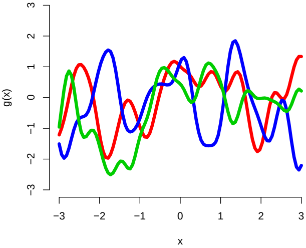


Deep GPs: Composition of Functions

$$y = f(g(\mathbf{x}))\,, \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$

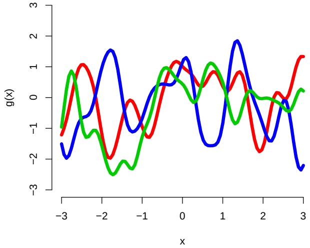
Deep GPs: Composition of Functions

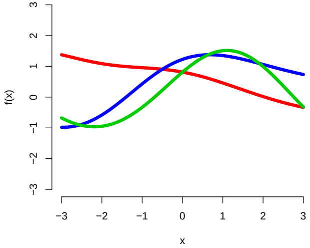
$$y = f(g(\mathbf{x}))\,, \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$



Deep GPs: Composition of Functions

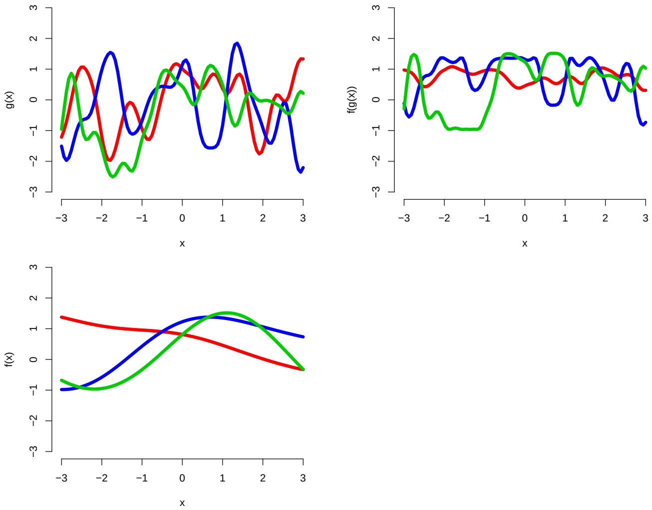
$$y = f(g(\mathbf{x}))\,, \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$





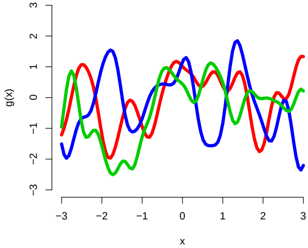
Deep GPs: Composition of Functions

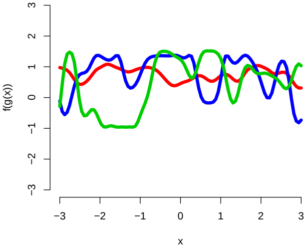
$$y = f(g(\mathbf{x}))\,, \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$

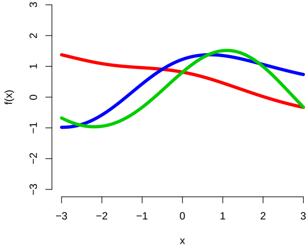


Deep GPs: Composition of Functions

$$y = f(g(\mathbf{x})), \quad f(\mathbf{x}) \sim \mathcal{GP}(0, C_f(\mathbf{x}, \mathbf{x}')) \quad g(\mathbf{x}) \sim \mathcal{GP}(0, C_g(\mathbf{x}, \mathbf{x}'))$$

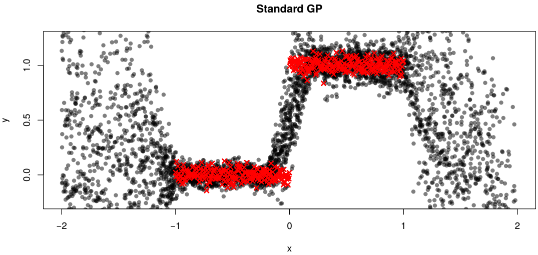




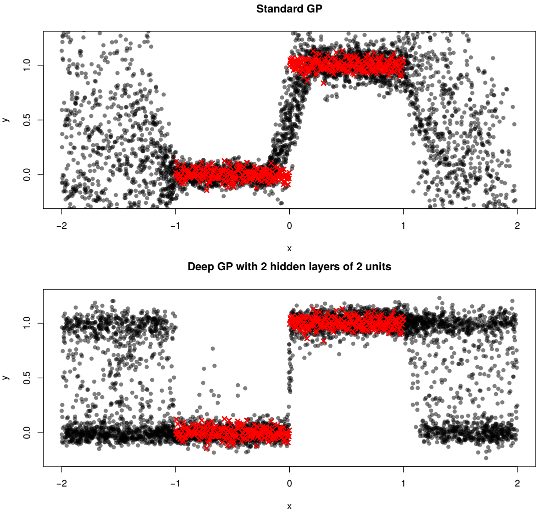


Deep GPs perform automatic covariance function design!

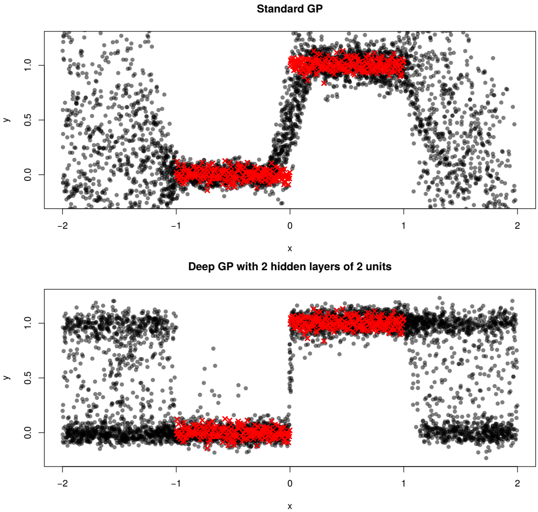
Deep GP Predictive Distribution



Deep GP Predictive Distribution



Deep GP Predictive Distribution



In a deep GP the predictive distribution needs not be Gaussian!

Why deep GPs?

Advantages:

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.

Drawbacks:

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.

Drawbacks:

- Require complicated approximate inference methods.

Why deep GPs?

Advantages:

- Useful input warping: automatic, non-parametric kernel design.
- Repair damage done by sparse approximations to GPs.
- More accurate predictions and better uncertainty estimates.

Drawbacks:

- Require complicated approximate inference methods.
- High computational cost.

Bayesian inference

Posterior over latent functions (typically at the observed data $\mathbf{X}$):

$$p(\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^3 | \mathbf{Y}) = \frac{p(\mathbf{f}^1)p(\mathbf{f}^2)p(\mathbf{f}^3) \; p(\mathbf{Y}|\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^3, \mathbf{X})}{p(\mathbf{Y})}$$

- GP priors
- Likelihood function
- Marginal likelihood

But the posterior $p(\mathbf{f}^1, \mathbf{f}^2, \mathbf{f}^3 | \mathbf{Y})$ is **intractable**.

Inducing Points Representation

Latent variables: from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, with $M \ll N$.

Distribution on f given by GP with inducing inputs $\bar{\mathbf{X}}$ and outputs $\mathbf{u}$.

Inducing Points Representation

Latent variables: from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, with $M \ll N$.

Distribution on f given by GP with inducing inputs $\bar{\mathbf{X}}$ and outputs $\mathbf{u}$.

If $\mathbf{u}$ is known, then $p(f(\mathbf{x}^\star)|\mathbf{u}) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

$$m^\star = \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}\,,$$
$$v^\star = k_{f^\star,f^\star} - \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{k}_{\mathbf{u},f^\star}\,.$$

Inducing Points Representation

Latent variables: from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, with $M \ll N$.

Distribution on f given by GP with inducing inputs $\bar{\mathbf{X}}$ and outputs $\mathbf{u}$.

If $\mathbf{u}$ is known, then $p(f(\mathbf{x}^\star)|\mathbf{u}) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

$$m^\star = \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}\,,$$
$$v^\star = k_{f^\star,f^\star} - \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{k}_{\mathbf{u},f^\star}\,.$$

If $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m},\mathbf{S})$, then $p(f(\mathbf{x}^\star)) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

$$m^\star = \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{m}\,,$$
$$v^\star = k_{f^\star,f^\star} - \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{k}_{\mathbf{u},f^\star} + \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{S}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{k}_{\mathbf{u},f^\star}$$

Inducing Points Representation

Latent variables: from $\mathcal{O}(N)$ to $\mathcal{O}(M)$, with $M \ll N$.

Distribution on f given by GP with inducing inputs $\bar{\mathbf{X}}$ and outputs $\mathbf{u}$.

If $\mathbf{u}$ is known, then $p(f(\mathbf{x}^\star)|\mathbf{u}) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

$$m^\star = \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{u}\,,$$
$$v^\star = k_{f^\star,f^\star} - \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{k}_{\mathbf{u},f^\star}\,.$$

If $p(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mathbf{m},\mathbf{S})$, then $p(f(\mathbf{x}^\star)) = \mathcal{N}(f(\mathbf{x}^\star)|m^\star, v^\star)$, where

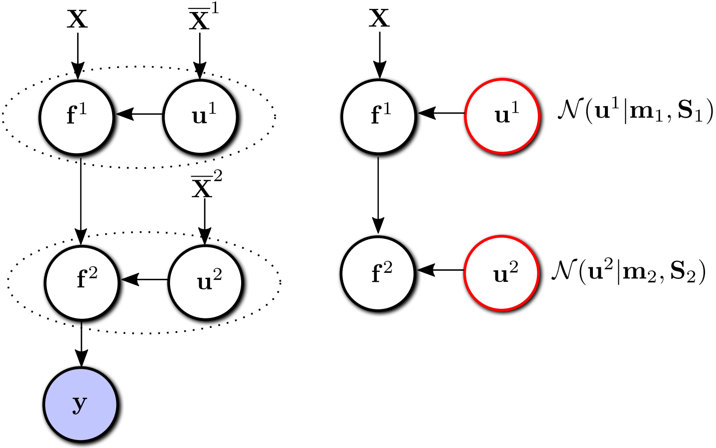
$$m^\star = \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{m}\,,$$
$$v^\star = k_{f^\star,f^\star} - \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{k}_{\mathbf{u},f^\star} + \mathbf{k}_{f^\star,\mathbf{u}}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{S}\mathbf{K}_{\mathbf{u},\mathbf{u}}^{-1}\mathbf{k}_{\mathbf{u},f^\star}$$

Given u or a Gaussian for u, f is fully specified!

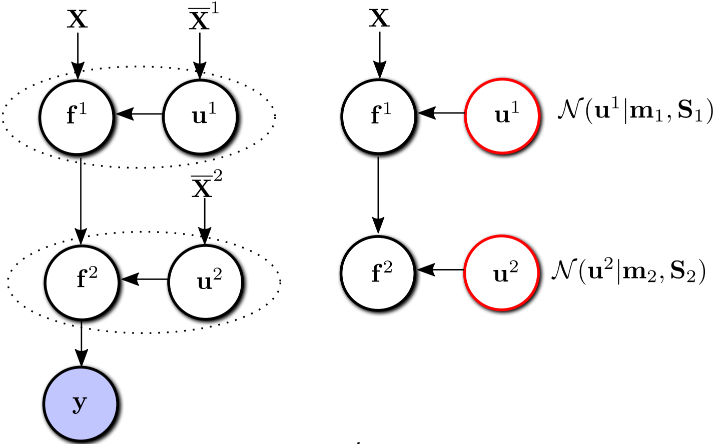
Deep GPs Joint Distribution

$$p(\mathbf{y}, \{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^{L}) = \overbrace{\prod_{i=1}^{N} p(y_i | f_i^L)}^{\text{Likelihood}} \times \underbrace{\prod_{l=1}^{L} p(\mathbf{f}^l | \mathbf{u}^l, \overline{\mathbf{X}}^l) p(\mathbf{u}^l | \overline{\mathbf{X}}^l)}_{\text{Deep GP prior}}$$

Graphical Model and Posterior Approximation



Graphical Model and Posterior Approximation



$$q(\{\mathbf{f}^l, \mathbf{u}^l\}_{l=1}^L) = \prod_{l=1}^{L} p(\mathbf{f}^l|\mathbf{u}^l) \; q(\mathbf{u}^l)$$

- Fixed
- Tunable

Variational Inference for Deep GPs

Based on minimizing KL$(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^{L}) | p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^{L} | \mathbf{y}))$

(Salimbeni, 2017)

Variational Inference for Deep GPs

Based on minimizing $\mathbf{KL}(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L)|p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L|\mathbf{y}))$

Equivalent to maximizing the lower bound on $\log p(\mathbf{y})$:

$$\mathcal{L} = \mathbb{E}_q\left[\log \frac{\prod_{i=1}^N p(y_i|f_i^L)\prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)}\,p(\mathbf{u}^l)}{\prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)}\,q(\mathbf{u}^l)}\right].$$

$$= \sum_{i=1}^N \mathbb{E}_q[\log p(y_i|f_i^L)] - \sum_{l=1}^L \mathrm{KL}(q(\mathbf{u}^l)|p(\mathbf{u}^l)).$$

(Salimbeni, 2017)

Variational Inference for Deep GPs

Based on minimizing $\mathbf{KL}(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L)|p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L|\mathbf{y}))$

Equivalent to maximizing the lower bound on $\log p(\mathbf{y})$:

$$\mathcal{L} = \mathbb{E}_q \left[\log \frac{\prod_{i=1}^N p(y_i|f_i^L) \prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)} p(\mathbf{u}^l)}{\prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)} q(\mathbf{u}^l)}\right] .$$

$$= \sum_{i=1}^N \mathbb{E}_q[\log p(y_i|f_i^L)] - \sum_{l=1}^L \mathrm{KL}(q(\mathbf{u}^l)|p(\mathbf{u}^l)) .$$

- Suitable for stochastic optimization.

(Salimbeni, 2017)

Variational Inference for Deep GPs

Based on minimizing KL$(q(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L)|p(\{\mathbf{u}^l, \mathbf{f}^l\}_{l=1}^L|\mathbf{y}))$

Equivalent to maximizing the lower bound on $\log p(\mathbf{y})$:

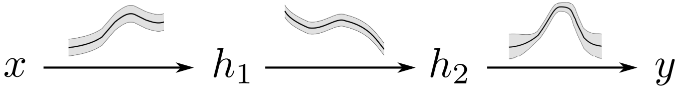
$$\mathcal{L} = \mathbb{E}_q\left[\log \frac{\prod_{i=1}^N p(y_i|f_i^L) \prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)} p(\mathbf{u}^l)}{\prod_{l=1}^L \cancel{p(\mathbf{f}^l|\mathbf{u}^l)} q(\mathbf{u}^l)}\right].$$

$$= \sum_{i=1}^N \mathbb{E}_q[\log p(y_i|f_i^L)] - \sum_{l=1}^L \mathrm{KL}(q(\mathbf{u}^l)|p(\mathbf{u}^l)).$$

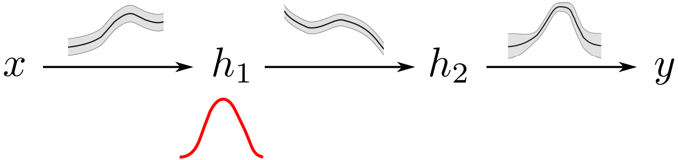
- Suitable for stochastic optimization.
- The expectations can be approximated by Monte Carlo.

(Salimbeni, 2017)

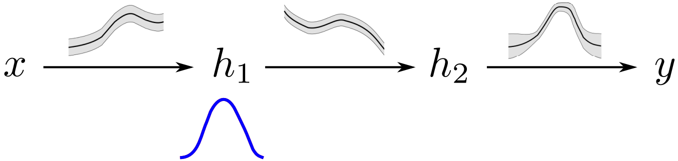
Monte Carlo Approximation



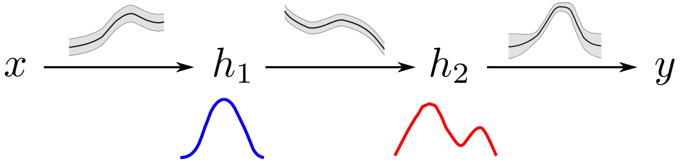
Monte Carlo Approximation



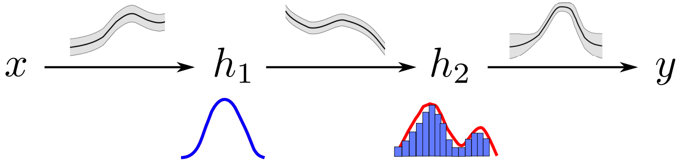
Monte Carlo Approximation



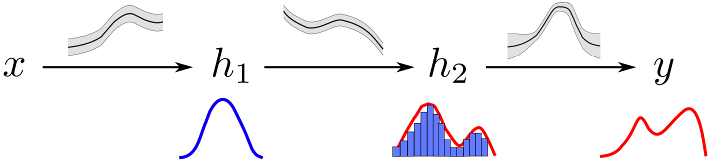
Monte Carlo Approximation



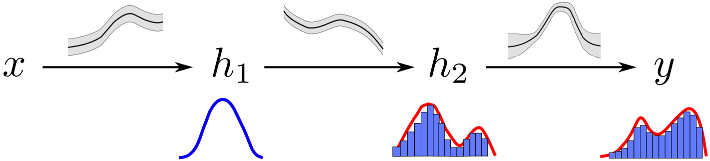
Monte Carlo Approximation



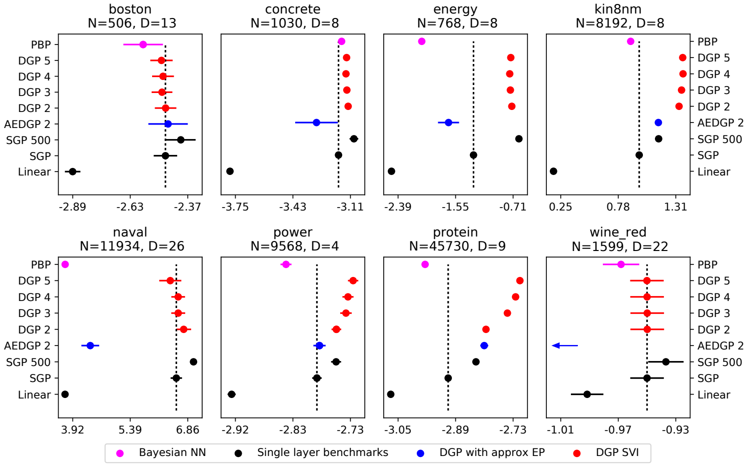
Monte Carlo Approximation



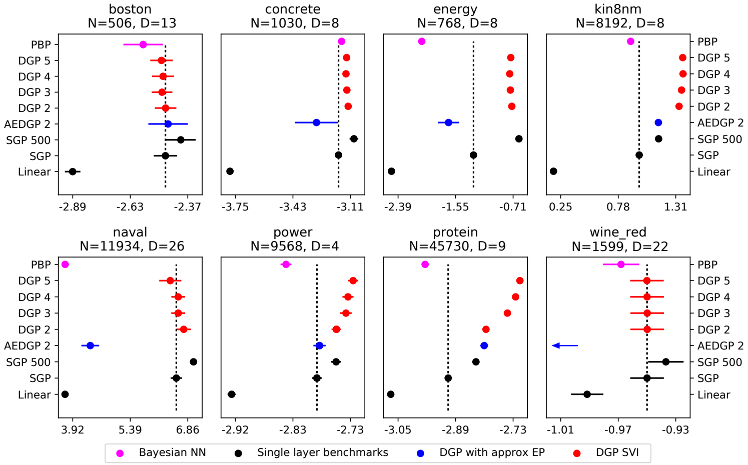
Monte Carlo Approximation



DGPs Experimental Results



DGPs Experimental Results



DGPs perform similar or better than the sparse GP and adding more layers does not seem to overfit!

(Salimbeni, 2017)

Software for GPs and Deep GPs

There are several packages providing implementations of GPs:

Software for GPs and Deep GPs

There are several packages providing implementations of GPs:

- **GPy**: Gaussian Processes in Python. Easy-to-use and extend. Supports multi-output GPs, different noise models and different approximate inference methods.

Software for GPs and Deep GPs

There are several packages providing implementations of GPs:

- **GPy**: Gaussian Processes in Python. Easy-to-use and extend. Supports multi-output GPs, different noise models and different approximate inference methods.
- **GPML**: Gaussian Processes in Matlab. No longer maintained. Implements the models and methods from the book "Gaussian Process for Machine learning".

Software for GPs and Deep GPs

There are several packages providing implementations of GPs:

- **GPy**: Gaussian Processes in Python. Easy-to-use and extend. Supports multi-output GPs, different noise models and different approximate inference methods.
- **GPML**: Gaussian Processes in Matlab. No longer maintained. Implements the models and methods from the book "Gaussian Process for Machine learning".
- **GPflow**: Gaussian processes in python using Tensorflow. Supports GPU acceleration. Focuses on variational inference and MCMC for approximate inference.

Software for GPs and Deep GPs

There are several packages providing implementations of GPs:

- **GPy**: Gaussian Processes in Python. Easy-to-use and extend. Supports multi-output GPs, different noise models and different approximate inference methods.
- **GPML**: Gaussian Processes in Matlab. No longer maintained. Implements the models and methods from the book "Gaussian Process for Machine learning".
- **GPflow**: Gaussian processes in python using Tensorflow. Supports GPU acceleration. Focuses on variational inference and MCMC for approximate inference.
- **GPyTorch**: Gaussian processes in python using PyTorch. Supports GPU acceleration. Also supports deep GPs.

Software for GPs and Deep GPs

There are several packages providing implementations of GPs:

- **GPy**: Gaussian Processes in Python. Easy-to-use and extend. Supports multi-output GPs, different noise models and different approximate inference methods.
- **GPML**: Gaussian Processes in Matlab. No longer maintained. Implements the models and methods from the book "Gaussian Process for Machine learning".
- **GPflow**: Gaussian processes in python using Tensorflow. Supports GPU acceleration. Focuses on variational inference and MCMC for approximate inference.
- **GPyTorch**: Gaussian processes in python using PyTorch. Supports GPU acceleration. Also supports deep GPs.

Deep GPs: uses doubly stochastic variational inference and GPflow.

Ongoing Research Directions

There is several research going on on GP:

Ongoing Research Directions

There is several research going on on GP:

1. **Scalable GPs**: More efficient and accurate methods to approximate the full GP. Need not be based on inducing points.

Ongoing Research Directions

There is several research going on on GP:

1. **Scalable GPs**: More efficient and accurate methods to approximate the full GP. Need not be based on inducing points.
2. **Flexible Approximations**: Most times parametric distributions are used for approximate inference. Non-Gaussian distributions such as those given by implicit models can have an advantage.

Ongoing Research Directions

There is several research going on on GP:

1. **Scalable GPs**: More efficient and accurate methods to approximate the full GP. Need not be based on inducing points.
2. **Flexible Approximations**: Most times parametric distributions are used for approximate inference. Non-Gaussian distributions such as those given by implicit models can have an advantage.
3. **Bayesian Neural Networks**: Instead of taking the limit $H \to \infty$ perform approximate inference in the Neural Network model.

Ongoing Research Directions

There is several research going on on GP:

1. **Scalable GPs**: More efficient and accurate methods to approximate the full GP. Need not be based on inducing points.
2. **Flexible Approximations**: Most times parametric distributions are used for approximate inference. Non-Gaussian distributions such as those given by implicit models can have an advantage.
3. **Bayesian Neural Networks**: Instead of taking the limit $H \to \infty$ perform approximate inference in the Neural Network model.
4. **Implicit Processes**: Process that is easy to sample from. Generalization of GPs being potentially more flexible. Approximate inference in functional space with advantages over Bayesian NN.

Ongoing Research Directions

There is several research going on on GP:

1. **Scalable GPs**: More efficient and accurate methods to approximate the full GP. Need not be based on inducing points.
2. **Flexible Approximations**: Most times parametric distributions are used for approximate inference. Non-Gaussian distributions such as those given by implicit models can have an advantage.
3. **Bayesian Neural Networks**: Instead of taking the limit $H \to \infty$ perform approximate inference in the Neural Network model.
4. **Implicit Processes**: Process that is easy to sample from. Generalization of GPs being potentially more flexible. Approximate inference in functional space with advantages over Bayesian NN.
5. **Convolutional GPs**: introduce prior knowledge about the latent function similar to that of convolutional neural networks.

Conclusions

Gaussian Processes:

Conclusions

Gaussian Processes:

- **Powerful non-parametric models** that can be used to describe latent functions.

Conclusions

Gaussian Processes:

- **Powerful non-parametric models** that can be used to describe latent functions.
- Provide **a closed-form expression** for the predictive distribution which takes into account prediction uncertainty.

Conclusions

Gaussian Processes:

- **Powerful non-parametric models** that can be used to describe latent functions.
- Provide **a closed-form expression** for the predictive distribution which takes into account prediction uncertainty.
- Scale to **very large datasets** and allow to introduce prior knowledge about the latent function.

Conclusions

Gaussian Processes:

- **Powerful non-parametric models** that can be used to describe latent functions.
- Provide **a closed-form expression** for the predictive distribution which takes into account prediction uncertainty.
- Scale to **very large datasets** and allow to introduce prior knowledge about the latent function.

Deep Gaussian Processes:

Conclusions

Gaussian Processes:

- **Powerful non-parametric models** that can be used to describe latent functions.
- Provide **a closed-form expression** for the predictive distribution which takes into account prediction uncertainty.
- Scale to **very large datasets** and allow to introduce prior knowledge about the latent function.

Deep Gaussian Processes:

- More flexible models that **address some of the GP limitations**.

Conclusions

Gaussian Processes:

- **Powerful non-parametric models** that can be used to describe latent functions.
- Provide **a closed-form expression** for the predictive distribution which takes into account prediction uncertainty.
- Scale to **very large datasets** and allow to introduce prior knowledge about the latent function.

Deep Gaussian Processes:

- More flexible models that **address some of the GP limitations**.

Thank you for your attention!

References

- M. Bauer, M. van der Wilk, C. E. Rasmussen. Understanding probabilistic sparse Gaussian process approximations. In Advances in neural information processing systems (pp. 1533-1541), 2016.
- Bui, T., Hernández-Lobato, D., Hernández-Lobato, J., Li, Y., Turner, R. (2016, June). Deep Gaussian processes for regression using approximate expectation propagation. In International conference on machine learning (pp. 1472-1481).
- Bui, T.D., Yan, J., Turner, R.E.: A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation. Journal of Machine Learning Research 18, 104:1104:72 (2017).
- Csató, L., Opper, M. (2002). Sparse on-line Gaussian processes. Neural computation, 14(3), 641-668.
- Damianou, A., Lawrence, N.: Deep Gaussian processes. In: International Conference on Artificial Intelligence and Statistics. pp. 207215 (2013).
- Havasi, M., Hernández-Lobato, J.M., Murillo-Fuentes, J.J.: Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo. In: Advances in Neural Information Processing Systems, pp. 75177527 (2018).
- Hensman, J., Fusi, N., Lawrence, N.D.: Gaussian processes for big data. In: Uncertainty in Artificial Intellegence. pp. 282290 (2013)
- Hernández-Lobato, D., Hernández-Lobato, J. M. (2016, May). Scalable Gaussian process classification via expectation propagation. In Artificial Intelligence and Statistics (pp. 168-176). PMLR.
- Matthews, A. G. D. G., Hensman, J., Turner, R., Ghahramani, Z. (2016, May). On sparse variational methods and the Kullback-Leibler divergence between stochastic processes. In Artificial Intelligence and Statistics (pp. 231-239).
- Quiñonero-Candela, J., Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. The Journal of Machine Learning Research, 6, 1939-1959.
- Salimbeni, H., Deisenroth, M.: Doubly stochastic variational inference for deep Gaussian processes. In: Advances in Neural Information Processing Systems, pp. 45884599 (2017).
- Snelson, E., Ghahramani, .: Sparse Gaussian processes using pseudo-inputs. In: Advances in Neural Information Processing Systems, pp. 12571264 (2006).
- Titsias, M.: Variational learning of inducing variables in sparse Gaussian processes. In: International Conference on Artificial Intelligence and Statistics. pp. 567574 (2009).
- M. Van der Wilk, C. E. Rasmussen, and J. Hensman. Convolutional Gaussian processes. In Advances in Neural Information Processing Systems, pages 28492858, 2017.
- Williams, C. K., Rasmussen, C. E. (2006). Gaussian processes for machine learning (Vol. 2, No. 3, p. 4). Cambridge, MA: MIT press.