Part II: Sparse Gaussian Processes

Daniel Hernández–Lobato
Computer Science Department
Universidad Autónoma de Madrid

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

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

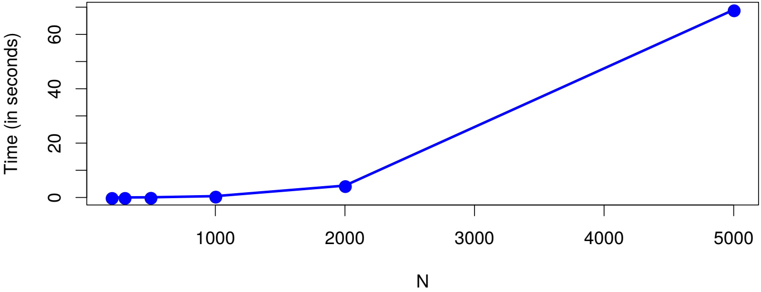
The memory cost is in $\mathcal{O}(N^2)$ since we have to compute $\boldsymbol{\Sigma}$.

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

Computational Cost of Gaussian Processes

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

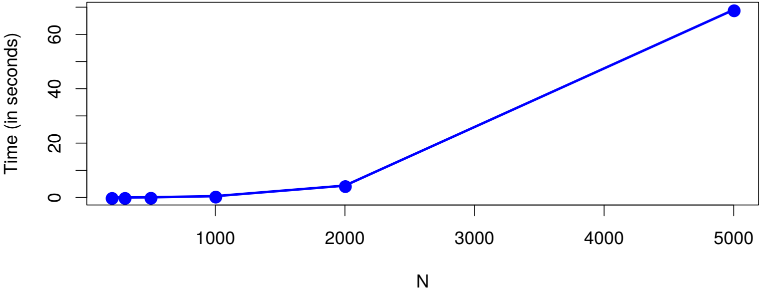
The computational cost is in $\mathcal{O}(N^3)$ since we have to invert Σ.



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!

Run the first cells of the notebook to measure GP fitting time and complete task 1!

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!

GPs are the limiting case ($H \to \infty$) of Bayesian Neural Networks!

Improving the Cost of Gaussian Processes

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

GPs are the limiting case ($H \to \infty$) of Bayesian Neural Networks!

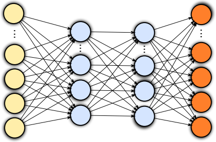
Idea: go back to the parametric model, but in such a way that we can still make inference easily!

Improving the Cost of Gaussian Processes

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

GPs are the limiting case ($H \to \infty$) of Bayesian Neural Networks!

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



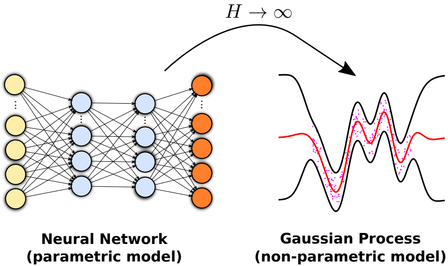
Neural Network (parametric model)

Improving the Cost of Gaussian Processes

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

GPs are the limiting case ($H \to \infty$) of Bayesian Neural Networks!

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

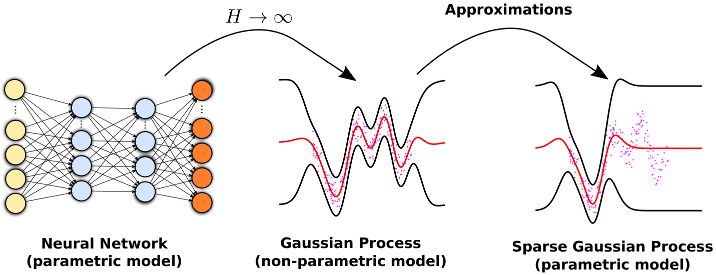


Improving the Cost of Gaussian Processes

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

GPs are the limiting case ($H \to \infty$) of Bayesian Neural Networks!

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

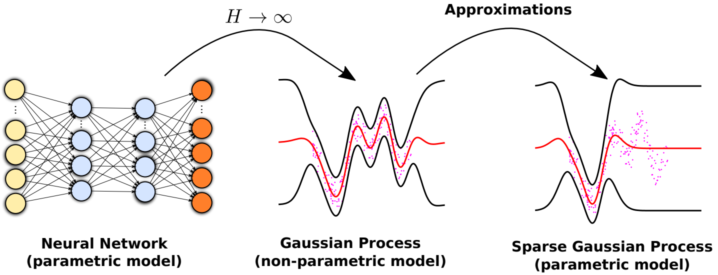


Improving the Cost of Gaussian Processes

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

GPs are the limiting case ($H \to \infty$) of Bayesian Neural Networks!

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



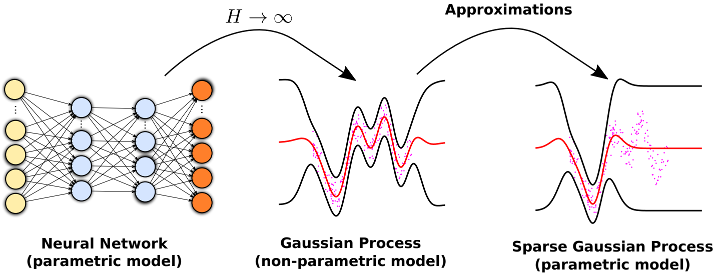
- **Nyström, Random Features and FITC**: approximate GP prior!

Improving the Cost of Gaussian Processes

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

GPs are the limiting case ($H \to \infty$) of Bayesian Neural Networks!

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



- **Nyström, Random Features and FITC**: approximate GP prior!
- **VFE**: does approximate inference with a simplified distribution q.

The Nyström Method

Motivation: The posterior mean and covariances require $(\mathbf{I}\sigma^2 + \mathbf{\Sigma})^{-1}$.

The Nyström Method

Motivation: The posterior mean and covariances require $(\mathbf{I}\sigma^2 + \mathbf{\Sigma})^{-1}$.

Can we approximate the inverse of $\mathbf{I}\sigma^2 + \mathbf{\Sigma}$ with a cheaper cost?

The Nyström Method

Motivation: The posterior mean and covariances require $(\mathbf{I}\sigma^2 + \boldsymbol{\Sigma})^{-1}$.

Can we approximate the inverse of $\mathbf{I}\sigma^2 + \boldsymbol{\Sigma}$ with a cheaper cost?

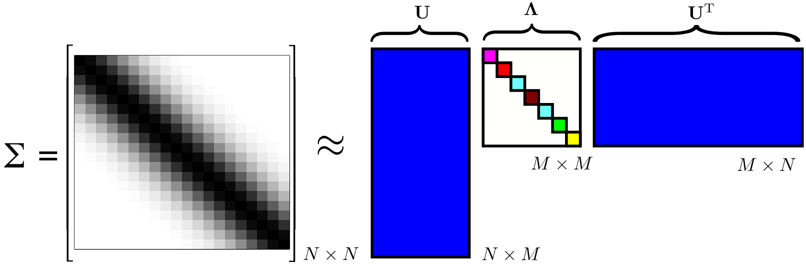
A low rank *m* approximation of $\boldsymbol{\Sigma}$ does the job:

The Nyström Method

Motivation: The posterior mean and covariances require $(\mathbf{I}\sigma^2 + \mathbf{\Sigma})^{-1}$.

Can we approximate the inverse of $\mathbf{I}\sigma^2 + \mathbf{\Sigma}$ with a cheaper cost?

A low rank *m* approximation of $\mathbf{\Sigma}$ does the job:

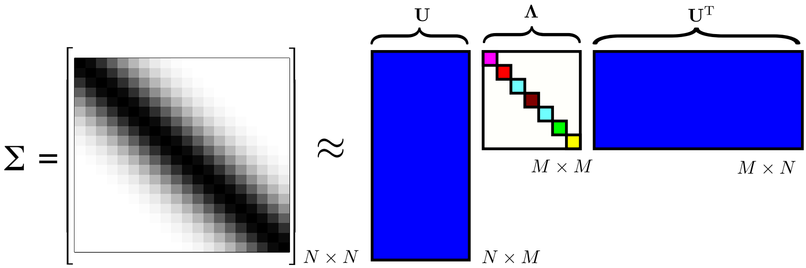


The Nyström Method

Motivation: The posterior mean and covariances require $(\mathbf{I}\sigma^2 + \boldsymbol{\Sigma})^{-1}$.

Can we approximate the inverse of $\mathbf{I}\sigma^2 + \boldsymbol{\Sigma}$ with a cheaper cost?

A low rank m approximation of $\boldsymbol{\Sigma}$ does the job:



The Woodbury formula gives $(\mathbf{I}\sigma^2 + \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\mathsf{T})^{-1}$ with cost $\mathcal{O}(M^2 N)$!

Woodbury Formula

$$(\mathbf{A} + \mathbf{PCQ})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{P}\left(\mathbf{C}^{-1} + \mathbf{QA}^{-1}\mathbf{P}\right)^{-1}\mathbf{QA}^{-1}$$

Woodbury Formula

$$(\mathbf{A} + \mathbf{PCQ})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{P}\left(\mathbf{C}^{-1} + \mathbf{QA}^{-1}\mathbf{P}\right)^{-1}\mathbf{QA}^{-1}$$

Let us now use $\mathbf{A} = \mathbf{I}\sigma^2$, $\mathbf{P} = \mathbf{U}$, $\mathbf{Q} = \mathbf{U}^\top$ and $\mathbf{C} = \mathbf{\Lambda}$.

Woodbury Formula

$$(\mathbf{A} + \mathbf{PCQ})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{P}\left(\mathbf{C}^{-1} + \mathbf{QA}^{-1}\mathbf{P}\right)^{-1}\mathbf{QA}^{-1}$$

Let us now use $\mathbf{A} = \mathbf{I}\sigma^2$, $\mathbf{P} = \mathbf{U}$, $\mathbf{Q} = \mathbf{U}^\mathsf{T}$ and $\mathbf{C} = \mathbf{\Lambda}$.

Note that A and C are diagonal with sizes $N \times N$ and $M \times M$!

Woodbury Formula

$$(\mathbf{A} + \mathbf{PCQ})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{P}\left(\mathbf{C}^{-1} + \mathbf{QA}^{-1}\mathbf{P}\right)^{-1}\mathbf{QA}^{-1}$$

Let us now use $\mathbf{A} = \mathbf{I}\sigma^2$, $\mathbf{P} = \mathbf{U}$, $\mathbf{Q} = \mathbf{U}^\top$ and $\mathbf{C} = \mathbf{\Lambda}$.

Note that A and C are diagonal with sizes $N \times N$ and $M \times M$!

$\mathbf{C}^{-1} + \mathbf{QA}^{-1}\mathbf{P} = \mathbf{\Lambda}^{-1} + \mathbf{U}^\top\mathbf{U}\sigma^{-2}$ has size $M \times M$!

Woodbury Formula

$$(\mathbf{A} + \mathbf{PCQ})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{P}\left(\mathbf{C}^{-1} + \mathbf{QA}^{-1}\mathbf{P}\right)^{-1}\mathbf{QA}^{-1}$$

Let us now use $\mathbf{A} = \mathbf{I}\sigma^2$, $\mathbf{P} = \mathbf{U}$, $\mathbf{Q} = \mathbf{U}^\mathsf{T}$ and $\mathbf{C} = \mathbf{\Lambda}$.

Note that A and C are diagonal with sizes $N \times N$ and $M \times M$!

$\mathbf{C}^{-1} + \mathbf{QA}^{-1}\mathbf{P} = \mathbf{\Lambda}^{-1} + \mathbf{U}^\mathsf{T}\mathbf{U}\sigma^{-2}$ has size $M \times M$!

$$(\mathbf{I}\sigma^2 + \mathbf{U\Lambda U}^\mathsf{T})^{-1} = \mathbf{I}\sigma^{-2} - \sigma^{-2}\mathbf{U}\left(\mathbf{\Lambda}^{-1} + \mathbf{U}^\mathsf{T}\sigma^{-2}\mathbf{U}\right)^{-1}\mathbf{U}^\mathsf{T}\sigma^{-2}$$

Woodbury Formula

$$(\mathbf{A} + \mathbf{PCQ})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{P}\left(\mathbf{C}^{-1} + \mathbf{QA}^{-1}\mathbf{P}\right)^{-1}\mathbf{QA}^{-1}$$

Let us now use $\mathbf{A} = \mathbf{I}\sigma^2$, $\mathbf{P} = \mathbf{U}$, $\mathbf{Q} = \mathbf{U}^\mathsf{T}$ and $\mathbf{C} = \mathbf{\Lambda}$.

Note that A and C are diagonal with sizes $N \times N$ and $M \times M$!

$\mathbf{C}^{-1} + \mathbf{QA}^{-1}\mathbf{P} = \mathbf{\Lambda}^{-1} + \mathbf{U}^\mathsf{T}\mathbf{U}\sigma^{-2}$ has size $M \times M$!

$$(\mathbf{I}\sigma^2 + \mathbf{U\Lambda U}^\mathsf{T})^{-1} = \mathbf{I}\sigma^{-2} - \sigma^{-2}\mathbf{U}\left(\mathbf{\Lambda}^{-1} + \mathbf{U}^\mathsf{T}\sigma^{-2}\mathbf{U}\right)^{-1}\mathbf{U}^\mathsf{T}\sigma^{-2}$$

Computing the whole expression has cost $\mathcal{O}(NM^2)$!

Eigenfunction Analysis of Covariance Functions

GPs are equivalent to a Bayesian linear model on an extended input space given by the eigenfunctions of the covariance function.

Eigenfunction Analysis of Covariance Functions

GPs are equivalent to a Bayesian linear model on an extended input space given by the eigenfunctions of the covariance function.

Extended input space: A function $\phi(\cdot)$ that obeys

$$\int C(\mathbf{x}, \mathbf{x}')\phi(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \lambda\phi(\mathbf{x}')\,,$$

is an eigenfunction of $C(\cdot, \cdot)$ with eigenvalue λ, w.r.t., $p(\mathbf{x})$.

Eigenfunction Analysis of Covariance Functions

GPs are equivalent to a Bayesian linear model on an extended input space given by the eigenfunctions of the covariance function.

Extended input space: A function $\phi(\cdot)$ that obeys

$$\int C(\mathbf{x}, \mathbf{x}')\phi(\mathbf{x})p(\mathbf{x})d\mathbf{x} = \lambda\phi(\mathbf{x}')\,,$$

is an eigenfunction of $C(\cdot, \cdot)$ with eigenvalue λ, w.r.t., $p(\mathbf{x})$.

Mercer's theorem:

$$C(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x})\phi_i(\mathbf{x}')\,.$$

An Analytic Example

Consider:

$$p(x) = \mathcal{N}(x|0, \sigma^2)\,, \qquad C(x, x') = \exp\left\{-\frac{1}{2\ell^2}(x - x')^2\right\}\,.$$

An Analytic Example

Consider:

$$p(x) = \mathcal{N}(x|0, \sigma^2)\,, \qquad C(x, x') = \exp\left\{-\frac{1}{2\ell^2}(x - x')^2\right\}\,.$$

Then,

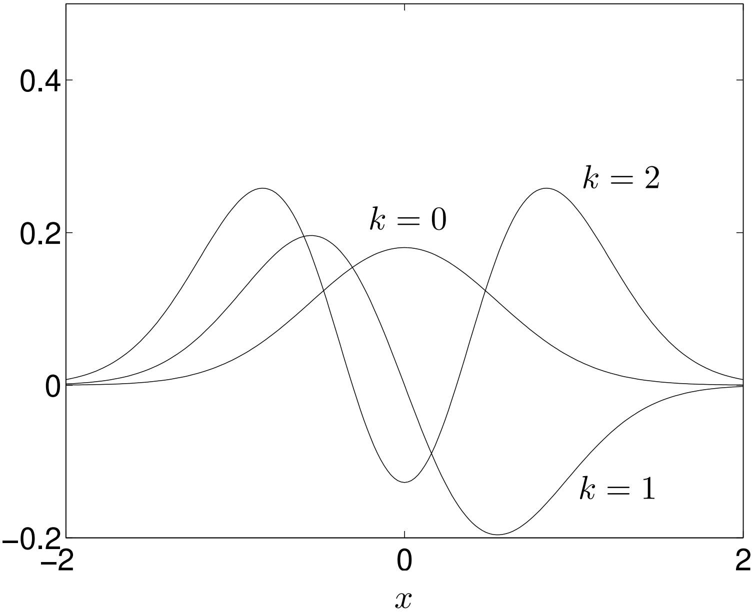
$$\lambda_k = \sqrt{\frac{2a}{A}} B^k\,, \qquad \phi_k(x) = \exp\left\{-(c - a)x^2\right\} H_k(\sqrt{2c}x)\,,$$

for $k = 0, 1, 2, \ldots$, with

$$a^{-1} = 4\sigma^2\,, \quad b^{-1} = 2\ell^2\,, \quad c = \sqrt{a^2 + 2ab}\,, \quad A = a + b + c\,, \quad B = b/a\,,$$

and $H_k(\cdot)$, the k-th order Hermite polynomial.

Hermite Polynomials

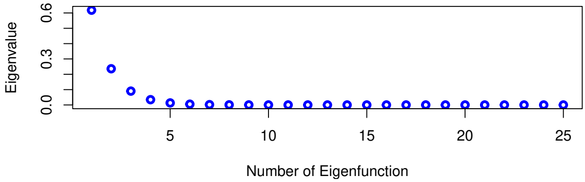


Covariance Function Approximation

Considering only the first eigenfunctions and eigenvalues is expected to give a good approximation of the covariance function!

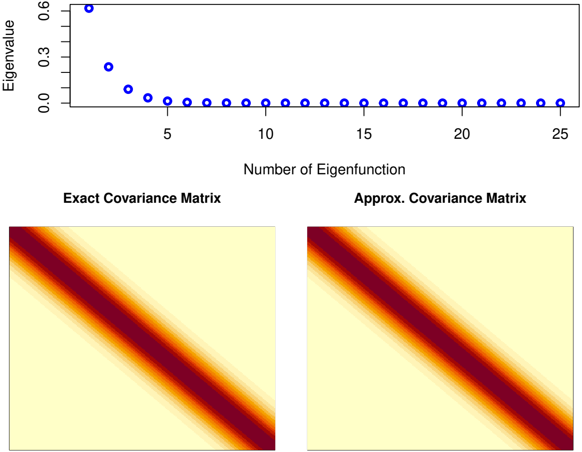
Covariance Function Approximation

Considering only the first eigenfunctions and eigenvalues is expected to give a good approximation of the covariance function!



Covariance Function Approximation

Considering only the first eigenfunctions and eigenvalues is expected to give a good approximation of the covariance function!



Nyström Approximation of Eigenfunctions

Let $p(\mathbf{x})$ be the distribution of the observed data.

Nyström Approximation of Eigenfunctions

Let $p(\mathbf{x})$ be the distribution of the observed data.

Consider the Monte Carlo estimator:

$$\lambda_i \phi_i(\mathbf{x}') = \int C(\mathbf{x}, \mathbf{x}') \phi_i(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \approx \frac{1}{N} \sum_{n=1}^{N} C(\mathbf{x}_n, \mathbf{x}') \phi_i(\mathbf{x}_n) \,.$$

Nyström Approximation of Eigenfunctions

Let $p(\mathbf{x})$ be the distribution of the observed data.

Consider the Monte Carlo estimator:

$$\lambda_i \phi_i(\mathbf{x}') = \int C(\mathbf{x}, \mathbf{x}') \phi_i(\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \approx \frac{1}{N} \sum_{n=1}^{N} C(\mathbf{x}_n, \mathbf{x}') \phi_i(\mathbf{x}_n) \,.$$

For $\mathbf{x}'$ in the training set, this motivates the following eigenvalue problem:

$$\lambda_i^{\mathrm{mat}} \mathbf{u}_i = \mathbf{\Sigma} \mathbf{u}_i \,,$$

with $\Sigma_{i,j} = C(\mathbf{x}_i, \mathbf{x}_j)$. Then, we approximate $\phi_i(\mathbf{x}_j) \approx \sqrt{N}(\mathbf{u}_i)_j = \tilde{\phi}_i(\mathbf{x}_j)$, and $\lambda_i \approx \lambda_i^{\mathrm{mat}}/N = \tilde{\lambda}_i$, which guarantees that $\mathbf{\Sigma} = \tilde{\mathbf{\Phi}} \tilde{\mathbf{\Lambda}} \tilde{\mathbf{\Phi}}^{\mathsf{T}}$, with $j = 1, \ldots, N$.

Nyström Approximation of Eigenfunctions

Let $p(\mathbf{x})$ be the distribution of the observed data.

Consider the Monte Carlo estimator:

$$\lambda_i \phi_i(\mathbf{x}') = \int C(\mathbf{x}, \mathbf{x}')\phi_i(\mathbf{x})p(\mathbf{x})d\mathbf{x} \approx \frac{1}{N}\sum_{n=1}^{N} C(\mathbf{x}_n, \mathbf{x}')\phi_i(\mathbf{x}_n)\,.$$

For $\mathbf{x}'$ in the training set, this motivates the following eigenvalue problem:

$$\lambda_i^{\text{mat}}\mathbf{u}_i = \mathbf{\Sigma}\mathbf{u}_i\,,$$

with $\Sigma_{i,j} = C(\mathbf{x}_i, \mathbf{x}_j)$. Then, we approximate $\phi_i(\mathbf{x}_j) \approx \sqrt{N}(\mathbf{u}_i)_j = \tilde{\phi}_i(\mathbf{x}_j)$, and $\lambda_i \approx \lambda_i^{\text{mat}}/N = \tilde{\lambda}_i$, which guarantees that $\mathbf{\Sigma} = \tilde{\mathbf{\Phi}}\tilde{\mathbf{\Lambda}}\tilde{\mathbf{\Phi}}^{\text{T}}$, with $j = 1, \ldots, N$.

For an arbitrary $\mathbf{x}'$ not in the training set, then:

$$\tilde{\phi}_i(\mathbf{x}') = \frac{1}{N\lambda_i}\sum_{n=1}^{N} C(\mathbf{x}', \mathbf{x}_n)\phi_i(\mathbf{x}_n) \approx \frac{\sqrt{N}}{\lambda_i^{\text{mat}}}\sum_{n=1}^{N} C(\mathbf{x}', \mathbf{x}_n)(\mathbf{u}_i)_n = \frac{\sqrt{N}}{\lambda_i^{\text{mat}}}\mathbf{\Sigma}(\mathbf{x}')^{\text{T}}\mathbf{u}_i\,.$$

Putting All Together

We choose a random subset of size $M < N$ of the training data, to approximate the eigenfunctions and eigenvalues!

Putting All Together

We choose a random subset of size $M < N$ of the training data, to approximate the eigenfunctions and eigenvalues!

Using Mercer's theorem and the previous approximation, we approximate the covariance function as:

$$C(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{x}') \approx \sum_{i=1}^{M} \tilde{\lambda}_i \tilde{\phi}_i(\mathbf{x}) \tilde{\phi}_i(\mathbf{x}') \, .$$

Putting All Together

We choose a random subset of size $M < N$ of the training data, to approximate the eigenfunctions and eigenvalues!

Using Mercer's theorem and the previous approximation, we approximate the covariance function as:

$$C(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{x}') \approx \sum_{i=1}^{M} \tilde{\lambda}_i \tilde{\phi}_i(\mathbf{x}) \tilde{\phi}_i(\mathbf{x}') \,.$$

which results in a rank M approximation of the covariance matrix $\mathbf{\Sigma}$:

$$\mathbf{\Sigma} \approx \tilde{\mathbf{\Sigma}} = \tilde{\mathbf{\Phi}} \tilde{\mathbf{\Lambda}} \tilde{\mathbf{\Phi}}^{\mathsf{T}} = \mathbf{\Sigma}_{N,M} \mathbf{\Sigma}_{M,M}^{-1} \mathbf{\Sigma}_{M,N} \,.$$

Putting All Together

We choose a random subset of size $M < N$ of the training data, to approximate the eigenfunctions and eigenvalues!

Using Mercer's theorem and the previous approximation, we approximate the covariance function as:

$$C(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{\infty} \lambda_i \phi_i(\mathbf{x}) \phi_i(\mathbf{x}') \approx \sum_{i=1}^{M} \tilde{\lambda}_i \tilde{\phi}_i(\mathbf{x}) \tilde{\phi}_i(\mathbf{x}') \,.$$

which results in a rank M approximation of the covariance matrix $\mathbf{\Sigma}$:

$$\mathbf{\Sigma} \approx \tilde{\mathbf{\Sigma}} = \tilde{\mathbf{\Phi}} \tilde{\mathbf{\Lambda}} \tilde{\mathbf{\Phi}}^{\mathsf{T}} = \mathbf{\Sigma}_{N,M} \mathbf{\Sigma}_{M,M}^{-1} \mathbf{\Sigma}_{M,N} \,.$$

The inverse of $\mathbf{I}\sigma^2 + \tilde{\mathbf{\Sigma}}$ can be efficiently computed using the Woodbury formula with cost $\mathcal{O}(NM^2)$!

Predictive Distribution

We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

Predictive Distribution

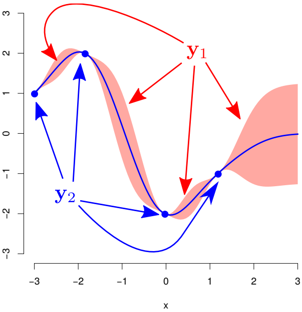
We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

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

Predictive Distribution

We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

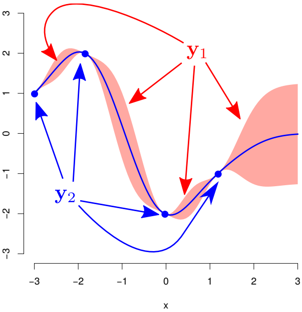
$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \tilde{\boldsymbol{\Sigma}} & \boldsymbol{\Sigma}_{\mathbf{f}\mathbf{f}^\star} \\ \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{f}}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



Predictive Distribution

We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

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

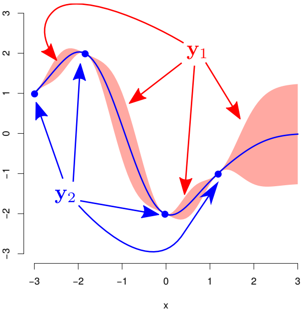


$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{B} \end{bmatrix} \right),$$

Predictive Distribution

We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

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



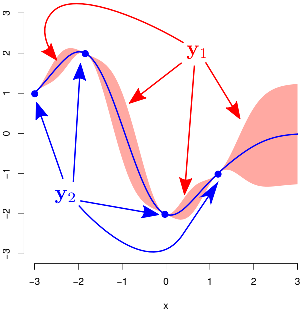
$$p(\mathbf{y}_1,\mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}_1\\ \mathbf{y}_2\end{bmatrix},\begin{bmatrix}\mathbf{a}\\ \mathbf{b}\end{bmatrix},\begin{bmatrix}\mathbf{A} & \mathbf{C}\\ \mathbf{C}^\top & \mathbf{B}\end{bmatrix}\right),$$

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

Predictive Distribution

We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

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

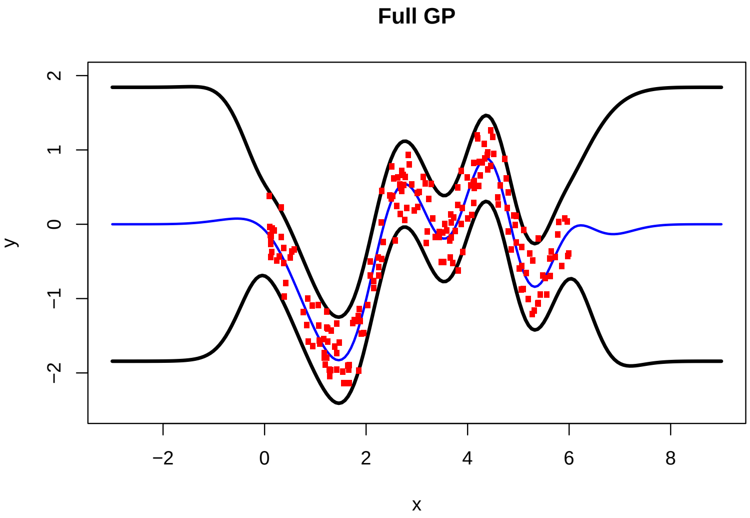


$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\top & \mathbf{B} \end{bmatrix}\right),$$

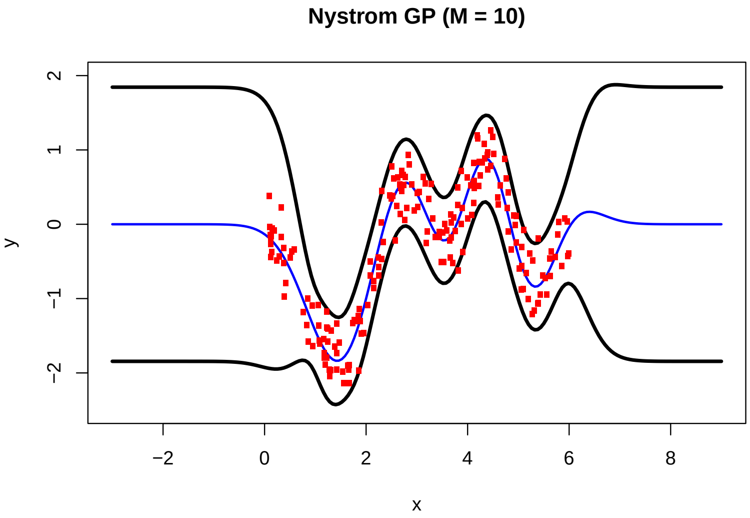
$$p(\mathbf{y}_1 | \mathbf{y}_2) = \mathcal{N}\left(\mathbf{y}_1 \middle| \mathbf{a} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y}_2 - \mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^\top\right)$$

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

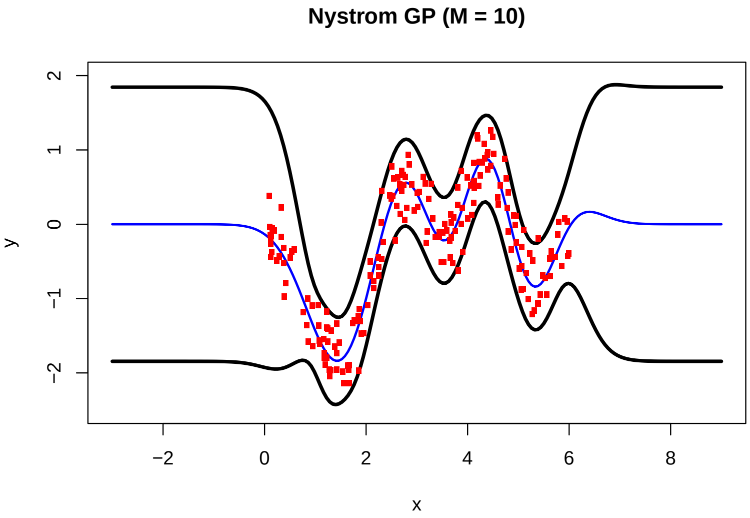
Nyström Approximation: Illustrative Example



Nyström Approximation: Illustrative Example



Nyström Approximation: Illustrative Example



The approximation is similar to the full GP in some regions!

Summary of Nyström Approximation

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.

Summary of Nyström Approximation

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- If $M = N$ the method is exact since $\tilde{\mathbf{\Sigma}} = \mathbf{\Sigma}$.

Summary of Nyström Approximation

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- If $M = N$ the method is exact since $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}$.
- For small M it can give bad results according to empirical evidence.

Summary of Nyström Approximation

- Reduces the cost to $\mathcal{O}(M^2N)$ with $M \ll N$.
- If $M = N$ the method is exact since $\tilde{\mathbf{\Sigma}} = \mathbf{\Sigma}$.
- For small M it can give bad results according to empirical evidence.
- It can perform well if $\mathbf{\Sigma}$ is dominated by a few eigenvalues.

Summary of Nyström Approximation

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- If $M = N$ the method is exact since $\tilde{\mathbf{\Sigma}} = \mathbf{\Sigma}$.
- For small M it can give bad results according to empirical evidence.
- It can perform well if $\mathbf{\Sigma}$ is dominated by a few eigenvalues.
- As the M points are chosen at random it may give different results.

Summary of Nyström Approximation

- Reduces the cost to $\mathcal{O}(M^2N)$ with $M \ll N$.
- If $M = N$ the method is exact since $\tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}$.
- For small M it can give bad results according to empirical evidence.
- It can perform well if $\boldsymbol{\Sigma}$ is dominated by a few eigenvalues.
- As the M points are chosen at random it may give different results.
- Since the approximation is done only over the covariance matrix of the training data, negative predictive variances are possible, but rare.

Run the cells of the notebook to fit a sparse GP using the Nyström approximation and complete task 2!

Run the cells of the notebook compare the exact and approximate predictive distribution and complete task 3!

Random Features Approximations

They can be used to approximate any stationary covariance function (only depends on the distance between points).

Random Features Approximations

They can be used to approximate any stationary covariance function (only depends on the distance between points).

Bochner's theorem:

A covariance function $C(\mathbf{x}, \mathbf{x}') = C(\mathbf{x} - \mathbf{x}')$ on $\mathbb{R}^D$ is positive definite if and only if $C(\mathbf{x} - \mathbf{x}')$ is the Fourier transform of a distribution $s(\mathbf{w})$.

Random Features Approximations

They can be used to approximate any stationary covariance function (only depends on the distance between points).

Bochner's theorem:

A covariance function $C(\mathbf{x}, \mathbf{x}') = C(\mathbf{x} - \mathbf{x}')$ on $\mathbb{R}^D$ is positive definite if and only if $C(\mathbf{x} - \mathbf{x}')$ is the Fourier transform of a distribution $s(\mathbf{w})$.

$$C(\mathbf{x}, \mathbf{x}') = \int \exp\{-i\mathbf{w}^\mathsf{T}(\mathbf{x} - \mathbf{x}')\} s(\mathbf{w}) d\mathbf{w} \,,$$

$$s(\mathbf{w}) = \frac{1}{(2\pi)^D} \int \exp\{i\mathbf{w}^\mathsf{T}\boldsymbol{\tau}\} C(\boldsymbol{\tau}, \mathbf{0}) d\boldsymbol{\tau} \,.$$

Random Features Approximations

They can be used to approximate any stationary covariance function (only depends on the distance between points).

Bochner's theorem:

A covariance function $C(\mathbf{x}, \mathbf{x}') = C(\mathbf{x} - \mathbf{x}')$ on $\mathbb{R}^D$ is positive definite if and only if $C(\mathbf{x} - \mathbf{x}')$ is the Fourier transform of a distribution $s(\mathbf{w})$.

$$C(\mathbf{x}, \mathbf{x}') = \int \exp\{-i\mathbf{w}^\mathsf{T}(\mathbf{x} - \mathbf{x}')\} s(\mathbf{w}) d\mathbf{w}\,,$$

$$s(\mathbf{w}) = \frac{1}{(2\pi)^D} \int \exp\{i\mathbf{w}^\mathsf{T}\boldsymbol{\tau}\} C(\boldsymbol{\tau}, \mathbf{0}) d\boldsymbol{\tau}\,.$$

$s(\mathbf{w})$ is called the spectral density of the covariance function.

Covariances as Expectations of Cosines

Due to Bochner's theorem, the covariance can be written as:

$$\begin{aligned} C(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{s(\mathbf{w})}\left[\exp\{-i\mathbf{w}^\mathsf{T}(\mathbf{x} - \mathbf{x}')\}\right] \\ &= 2\mathbb{E}_{s(\mathbf{w}), b\sim U[0,2\pi]}\left[\cos\left(\mathbf{w}^\mathsf{T}\mathbf{x} + b\right)\cos\left(\mathbf{w}^\mathsf{T}\mathbf{x}' + b\right)\right] . \end{aligned}$$

Covariances as Expectations of Cosines

Due to Bochner's theorem, the covariance can be written as:

$$
\begin{aligned}
C(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{s(\mathbf{w})}\left[\exp\{-i\mathbf{w}^{\mathsf{T}}(\mathbf{x}-\mathbf{x}')\}\right] \\
&= 2\mathbb{E}_{s(\mathbf{w}), b\sim U[0,2\pi]}\left[\cos\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}+b\right)\cos\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}'+b\right)\right] .
\end{aligned}
$$

The expectation can be approximated by a Monte Carlo average!

Covariances as Expectations of Cosines

Due to Bochner's theorem, the covariance can be written as:

$$\begin{aligned} C(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{s(\mathbf{w})}\left[\exp\{-i\mathbf{w}^{\mathsf{T}}(\mathbf{x} - \mathbf{x}')\}\right] \\ &= 2\mathbb{E}_{s(\mathbf{w}), b\sim U[0,2\pi]}\left[\cos\left(\mathbf{w}^{\mathsf{T}}\mathbf{x} + b\right)\cos\left(\mathbf{w}^{\mathsf{T}}\mathbf{x}' + b\right)\right] . \end{aligned}$$

The expectation can be approximated by a Monte Carlo average!

We can reduce the variance of the estimator by generating M samples:

$$C(\mathbf{x}, \mathbf{x}') \approx \frac{2}{M}\sum_{m=1}^{M} cos\left(\mathbf{w}_m^{\mathsf{T}}\mathbf{x} + b_m\right)\cos\left(\mathbf{w}_m^{\mathsf{T}}\mathbf{x}' + b_m\right) = \phi(\mathbf{x})^{\mathsf{T}}\phi(\mathbf{x}')$$

with $\phi(\mathbf{x}) = \sqrt{\frac{2}{M}}\cos\left(\mathbf{W}^{\mathsf{T}}\mathbf{x} + \mathbf{b}\right)$ a random M feature expansion.

Covariances as Expectations of Cosines

Due to Bochner's theorem, the covariance can be written as:

$$\begin{aligned} C(\mathbf{x}, \mathbf{x}') &= \mathbb{E}_{s(\mathbf{w})}\left[\exp\{-i\mathbf{w}^\mathsf{T}(\mathbf{x}-\mathbf{x}')\}\right] \\ &= 2\mathbb{E}_{s(\mathbf{w}),b\sim U[0,2\pi]}\left[\cos\left(\mathbf{w}^\mathsf{T}\mathbf{x}+b\right)\cos\left(\mathbf{w}^\mathsf{T}\mathbf{x}'+b\right)\right]. \end{aligned}$$

The expectation can be approximated by a Monte Carlo average!

We can reduce the variance of the estimator by generating M samples:

$$C(\mathbf{x}, \mathbf{x}') \approx \frac{2}{M}\sum_{m=1}^{M} cos\left(\mathbf{w}_m^\mathsf{T}\mathbf{x}+b_m\right)\cos\left(\mathbf{w}_m^\mathsf{T}\mathbf{x}'+b_m\right) = \phi(\mathbf{x})^\mathsf{T}\phi(\mathbf{x}')$$

with $\phi(\mathbf{x}) = \sqrt{\frac{2}{M}}\cos\left(\mathbf{W}^\mathsf{T}\mathbf{x}+\mathbf{b}\right)$ a random M feature expansion.

For the squared exponential covariance function $s(\mathbf{w})$ is Gaussian!

Approximate Covariance Function

The covariance matrix can be simply approximated as:

$$\boldsymbol{\Sigma} \approx \tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathsf{T}}$$

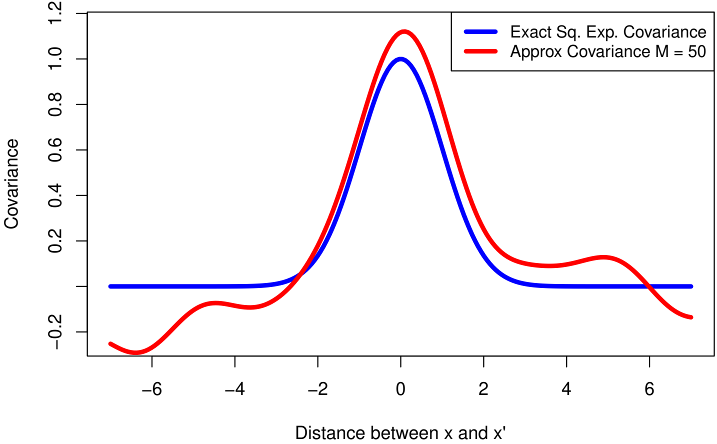
and hence $\mathbf{I}\sigma^2 + \tilde{\boldsymbol{\Sigma}}$ can be inverted with cost $\mathcal{O}(NM^2)$.

Approximate Covariance Function

The covariance matrix can be simply approximated as:

$$\boldsymbol{\Sigma} \approx \tilde{\boldsymbol{\Sigma}} = \boldsymbol{\Phi}\boldsymbol{\Phi}^{\mathsf{T}}$$

and hence $\mathbf{I}\sigma^2 + \tilde{\boldsymbol{\Sigma}}$ can be inverted with cost $\mathcal{O}(NM^2)$.



Predictive Distribution

We want to compute the value of $f^{\star}$ at a new $\mathbf{x}^{\star}$:

Predictive Distribution

We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

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

Predictive Distribution

We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

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

All prior covariances are now approximated using dot products with the random features computed before!

Predictive Distribution

We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

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

All prior covariances are now approximated using dot products with the random features computed before!

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

Predictive Distribution

We want to compute the value of $f^\star$ at a new $\mathbf{x}^\star$:

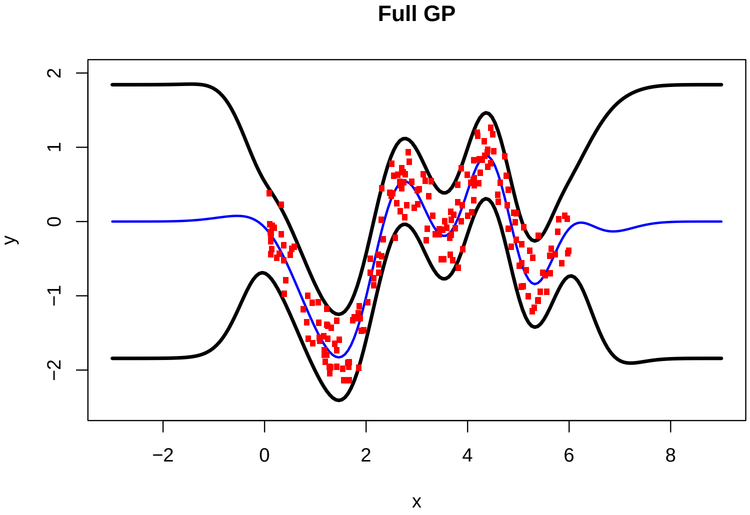
$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \tilde{\Sigma} & \tilde{\Sigma}_{\mathbf{f}\mathbf{f}^\star} \\ \tilde{\Sigma}_{\mathbf{f}^\star\mathbf{f}}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

All prior covariances are now approximated using dot products with the random features computed before!

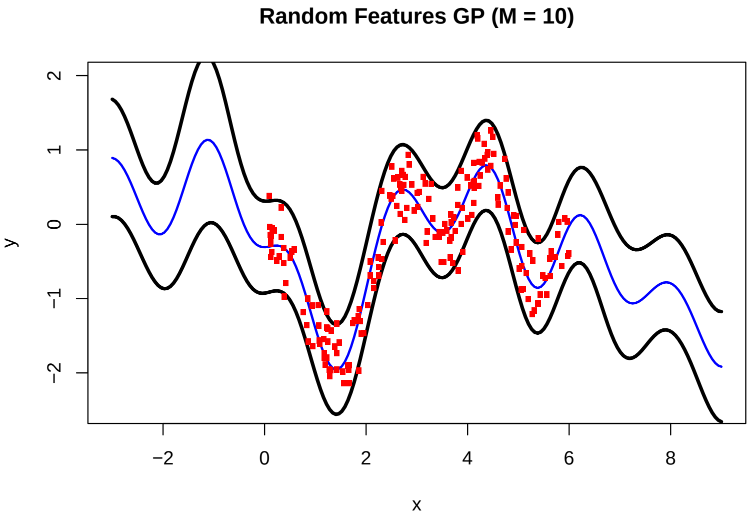
$$p(\mathbf{f}^\star|\mathbf{f}) = \mathcal{N}\left(\mathbf{f}^\star \middle| \tilde{\Sigma}_{\mathbf{f}^\star\mathbf{f}} \tilde{\Sigma}_{\mathbf{ff}}^{-1} \mathbf{f}, \tilde{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} - \tilde{\Sigma}_{\mathbf{f}^\star\mathbf{f}} \tilde{\Sigma}_{\mathbf{ff}}^{-1} \tilde{\Sigma}_{\mathbf{f}^\star\mathbf{f}}^{\mathrm{T}}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

The computational cost is $\mathcal{O}(NM^2)$!

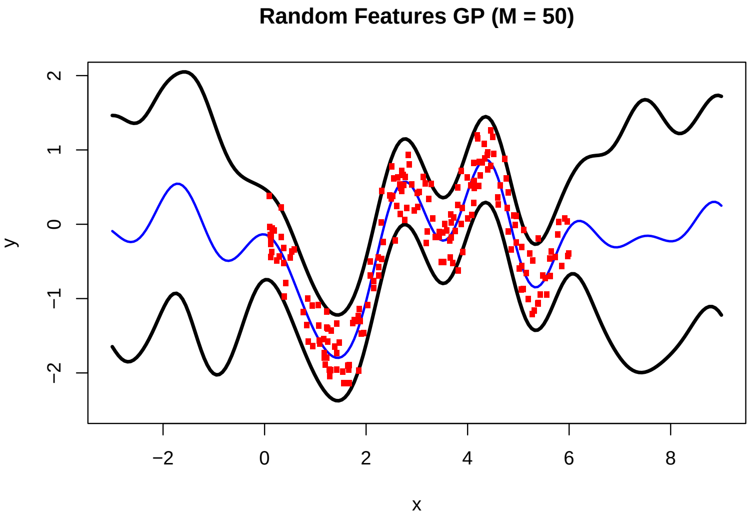
Random Features: Illustrative Example



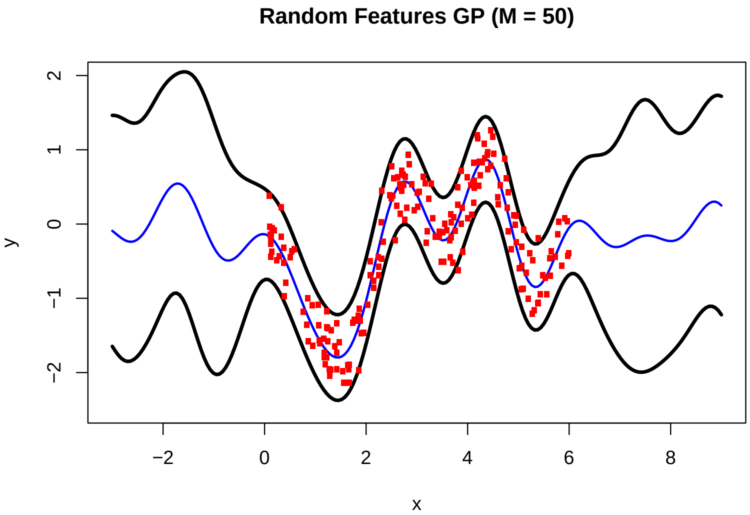
Random Features: Illustrative Example



Random Features: Illustrative Example



Random Features: Illustrative Example



In regions with no data the approximation may be wiggling a lot!

Summary of Random Features Approximation

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.

Summary of Random Features Approximation

- Reduces the cost to $\mathcal{O}(M^2N)$ with $M \ll N$.
- For small M it can give bad results due to the wiggling effect of cosine features.

Summary of Random Features Approximation

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- For small M it can give bad results due to the wiggling effect of cosine features.
- Guaranteed to be exact only for $M \to \infty$.

Summary of Random Features Approximation

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- For small M it can give bad results due to the wiggling effect of cosine features.
- Guaranteed to be exact only for $M \to \infty$.
- It is restricted to stationary covariance functions.

Summary of Random Features Approximation

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- For small M it can give bad results due to the wiggling effect of cosine features.
- Guaranteed to be exact only for $M \to \infty$.
- It is restricted to stationary covariance functions.
- Very simple to implement!

Summary of Random Features Approximation

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- For small M it can give bad results due to the wiggling effect of cosine features.
- Guaranteed to be exact only for $M \to \infty$.
- It is restricted to stationary covariance functions.
- Very simple to implement!
- Equivalent to a neural network with a hidden layer with M units and cosine activations, and a Bayesian linear model in the last layer!

Run the cells of the notebook to fit a sparse GP using the Random Features approximation and complete task 4!

Full Independent Training Conditional (FITC)

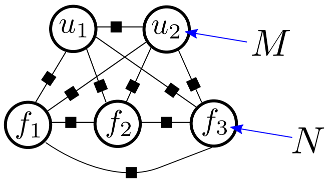
1. Extend model with $M \ll N$ inducing points and outputs at $\overline{\mathbf{X}}$.

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

Full Independent Training Conditional (FITC)

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

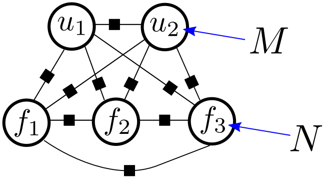
$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{ff}} & \boldsymbol{\Sigma}_{\mathbf{fu}} \\ \boldsymbol{\Sigma}_{\mathbf{uf}} & \boldsymbol{\Sigma}_{\mathbf{uu}}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



Full Independent Training Conditional (FITC)

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

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



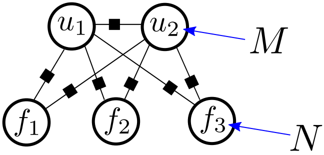
2. Introduce conditional independences:

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

Full Independent Training Conditional (FITC)

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

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



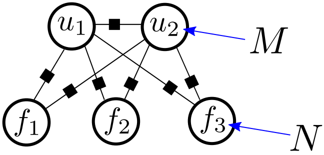
2. Introduce conditional independences:

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

Full Independent Training Conditional (FITC)

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

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



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 $\mathbf{f}$.

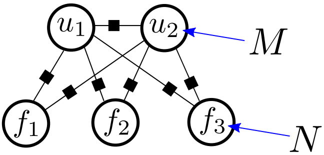
$$p(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{u})p(\mathbf{u})d\mathbf{u} = \prod_{i=1}^{N} p(f_i|\mathbf{u})p(\mathbf{u})d\mathbf{u} = \mathcal{N}(\mathbf{f}|0, \tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}})$$

where $\tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}} = \mathbf{D} + \mathbf{Q}_{\mathbf{ff}}$ with $\mathbf{D}$ diagonal and $\mathbf{Q}_{\mathbf{ff}} = \boldsymbol{\Sigma}_{\mathbf{fu}}\boldsymbol{\Sigma}_{\mathbf{uu}}^{-1}\boldsymbol{\Sigma}_{\mathbf{uf}}$ 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:

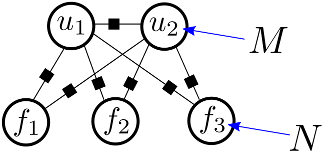
$$p(\mathbf{f}, \mathbf{f}^\star) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{f}^\star \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}} & \mathbf{Q}_{\mathbf{ff}^\star} \\ \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}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

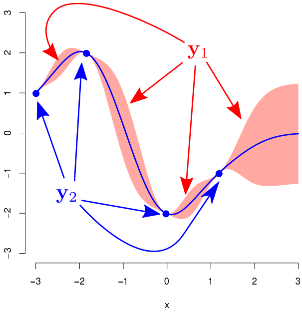


Full Independent Training Conditional (FITC)

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

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





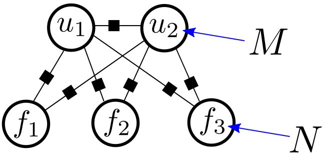
$$p(\mathbf{y}_1, \mathbf{y}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}, \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^\mathsf{T} & \mathbf{B} \end{bmatrix} \right),$$

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

Full Independent Training Conditional (FITC)

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

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

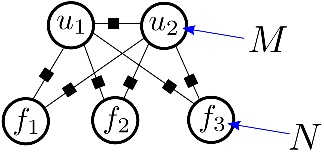


$$p(\mathbf{f}^\star|\mathbf{f}) = \mathcal{N}\left(\mathbf{f}^\star \middle| \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}\tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}}^{-1}\mathbf{f}, \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} - \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}\tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}}^{-1}\mathbf{Q}_{\mathbf{f}^\star\mathbf{f}}^\top\ri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

Full Independent Training Conditional (FITC)

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

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



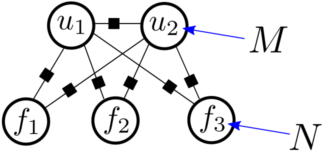
$$p(\mathbf{f}^\star|\mathbf{f}) = \mathcal{N}\left(\mathbf{f}^\star \middle| \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}}^{-1} \mathbf{f}, \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} - \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}}^{-1}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

Due to the structure in $\tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}}$ 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:

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



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

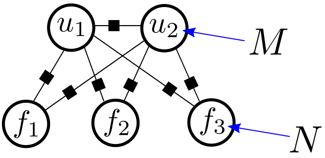
Due to the structure in $\tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}}$ all computations have cost in $\mathcal{O}(NM^2)$.

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:

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



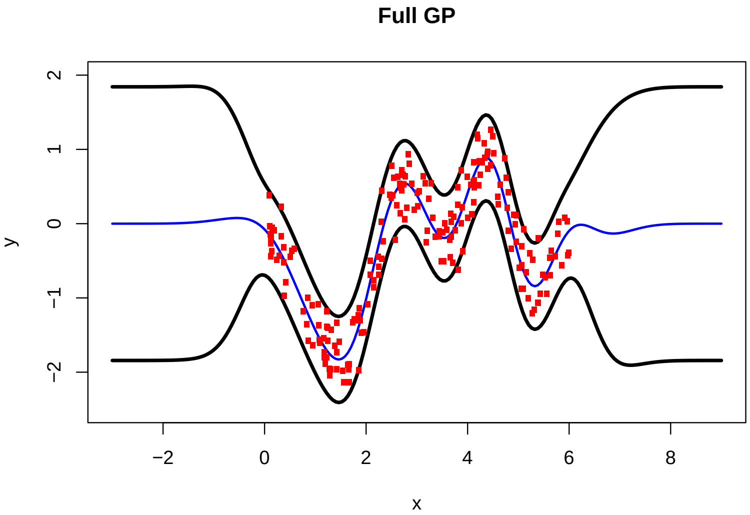
$$p(\mathbf{f}^\star|\mathbf{f}) = \mathcal{N}\left(\mathbf{f}^\star \middle| \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}}^{-1} \mathbf{f}, \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} - \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \tilde{\boldsymbol{\Sigma}}_{\mathbf{ff}}^{-1}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

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

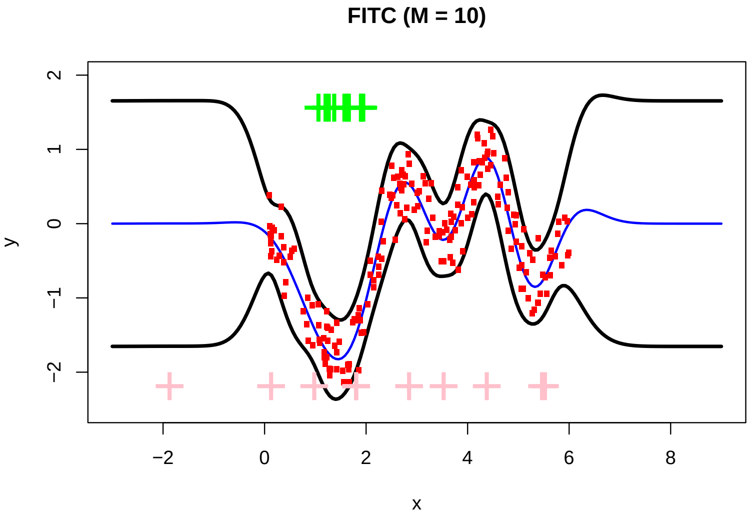
6. How do we find the location of the inducing points $\overline{\mathbf{X}}$?

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

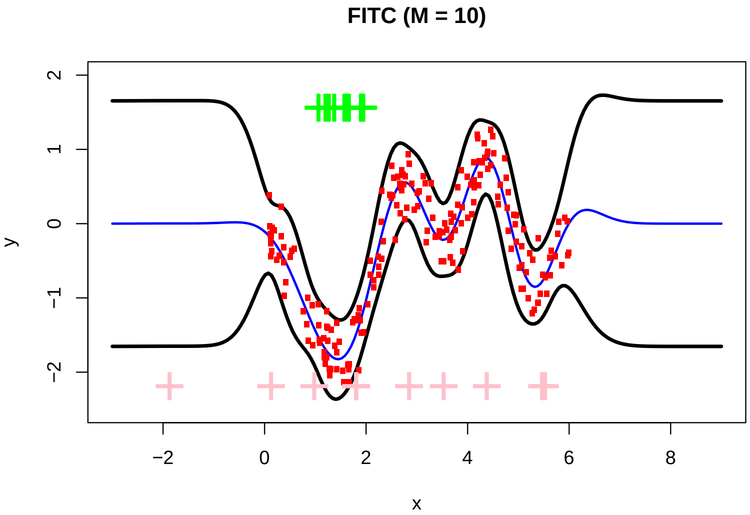
FITC: Illustrative Example



FITC: Illustrative Example



FITC: Illustrative Example



The inducing points cover the regions where the function changes!

Summary of FITC

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.

Summary of FITC

- Reduces the cost to $\mathcal{O}(M^2N)$ with $M \ll N$.
- The optimized inducing points spread over the input space where the latent function changes.

Summary of FITC

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- The optimized inducing points spread over the input space where the latent function changes.
- Guaranteed to be exact if $M = N$ and the inducing points are not optimized and located at the training points.

Summary of FITC

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- The optimized inducing points spread over the input space where the latent function changes.
- Guaranteed to be exact if $M = N$ and the inducing points are not optimized and located at the training points.
- It can be understood as considering heteroscedastic (input dependent) noise!

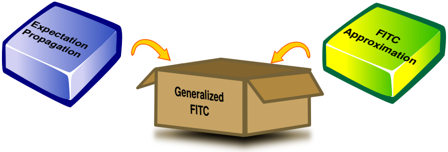
Run the cells of the notebook to fit a sparse GP using the FITC approximation and complete task 5!

Generalized FITC

Combines FITC with the use of Expectation Propagation to address binary classification problems!

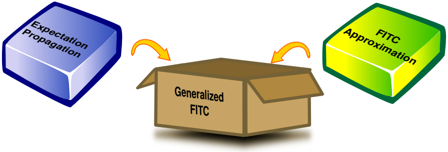
Generalized FITC

Combines FITC with the use of Expectation Propagation to address binary classification problems!



Generalized FITC

Combines FITC with the use of Expectation Propagation to address binary classification problems!

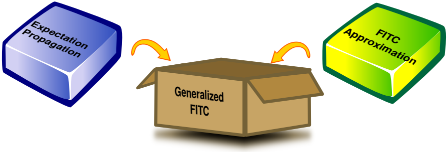


Assumes $y_i \in \{-1, 1\}$ and a probit likelihood:

$$p(y_i|f(\mathbf{x}_i)) = \phi(y_i f(\mathbf{x}_i)) \,, \quad \phi(\cdot) \equiv \text{The c.d.f. of a standard Gaussian.}$$

Generalized FITC

Combines FITC with the use of Expectation Propagation to address binary classification problems!



Assumes $y_i \in \{-1, 1\}$ and a probit likelihood:

$$p(y_i|f(\mathbf{x}_i)) = \phi(y_i f(\mathbf{x}_i)), \quad \phi(\cdot) \equiv \text{The c.d.f. of a standard Gaussian.}$$

Approximates with a **Gaussian distribution** the intractable posterior:

$$p(\mathbf{f}|\mathbf{y}) = \frac{\prod_{i=1}^{N} \phi(y_i f(\mathbf{x}_i))\mathcal{N}(\mathbf{f}|\mathbf{0}, \tilde{\boldsymbol{\Sigma}})}{p(\mathbf{y})},$$

where $\tilde{\boldsymbol{\Sigma}}$ is the **approximate FITC covariance matrix**.

Introduction to Expectation Propagation

Approximates an intractable distribution p by a parametric distribution q.

Introduction to Expectation Propagation

Approximates an intractable distribution p by a parametric distribution q.

It is based on the minimization of the KL-divergence, $\mathrm{KL}(p||q)$:

$$\int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} = \mathrm{KL}(q|p) \geq 0\,.$$

Introduction to Expectation Propagation

Approximates an intractable distribution p by a parametric distribution q.

It is based on the minimization of the KL-divergence, $\mathrm{KL}(p||q)$:

$$\int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} = \mathrm{KL}(q|p) \geq 0 \,.$$

q is restricted to belong to a family of distributions **closed under the product and ratio operation**: The **exponential family**.

Introduction to Expectation Propagation

Approximates an intractable distribution p by a parametric distribution q.

It is based on the minimization of the KL-divergence, $\mathrm{KL}(p||q)$:

$$\int p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} d\mathbf{x} = \mathrm{KL}(q|p) \geq 0 \,.$$

q is restricted to belong to a family of distributions **closed under the product and ratio operation**: The **exponential family**.

The exponential family:

$$q(\mathbf{x}) = \exp\left(\boldsymbol{\eta}^\mathsf{T}\mathbf{u}(\mathbf{x}) - g(\boldsymbol{\eta})\right) \,, \qquad g(\boldsymbol{\eta}) = \log \int \exp\left(\boldsymbol{\eta}^\mathsf{T}\mathbf{u}(\mathbf{x})\right) d\mathbf{x}$$

where $\boldsymbol{\eta}$ is a vector of natural parameters of q, $\mathbf{u}(\mathbf{x})$ are the sufficient statistics and $g(\boldsymbol{\eta})$ is a **log partition function**.

Examples of Distributions in the Exponential Family

Gaussian:

$$\mathcal{N}(x|\mu, \sigma^2) = 1/\sqrt{2\pi\sigma^2} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

Examples of Distributions in the Exponential Family

Gaussian:

$$\mathcal{N}(x|\mu, \sigma^2) = 1/\sqrt{2\pi\sigma^2}\exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

Exponential form:

$$\mathcal{N}(x|\mu, \sigma^2) = \exp\left(\boldsymbol{\eta}^\mathsf{T}\mathbf{u}(x) - g(\boldsymbol{\eta})\right)$$

$$\boldsymbol{\eta} = (\mu/\sigma^2, 1.0/\sigma^2)^\mathsf{T}\,, \quad \mathbf{u}(x) = (x, -0.5x^2)^\mathsf{T}\,, \quad g(\boldsymbol{\eta}) = -\frac{1}{2}\log\frac{2\pi}{\eta_2} + \frac{\eta_1^2}{2\eta_2}\,.$$

Examples of Distributions in the Exponential Family

Gaussian:

$$\mathcal{N}(x|\mu,\sigma^2) = 1/\sqrt{2\pi\sigma^2}\,\exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

Exponential form:

$$\mathcal{N}(x|\mu,\sigma^2) = \exp\Big(\boldsymbol{\eta}^\mathsf{T}\mathbf{u}(x) - g(\boldsymbol{\eta})\Big)$$

$$\boldsymbol{\eta} = (\mu/\sigma^2, 1.0/\sigma^2)^\mathsf{T}\,, \quad \mathbf{u}(x) = (x, -0.5x^2)^\mathsf{T}\,, \quad g(\boldsymbol{\eta}) = -\frac{1}{2}\log\frac{2\pi}{\eta_2} + \frac{\eta_1^2}{2\eta_2}\,.$$

Most parametric distributions belong to the exponential family!

Product and Ratio of Gaussians

Consider these two Gaussian distributions:

$$p_1(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left\{ -\frac{1}{2}\sigma_1^2 (x - \mu_1)^2 \right\} ,$$

$$p_2(x) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left\{ -\frac{1}{2}\sigma_2^2 (x - \mu_2)^2 \right\} .$$

Product and Ratio of Gaussians

Consider these two Gaussian distributions:

$$p_1(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left\{-\frac{1}{2}\sigma_1^2(x-\mu_1)^2\right\} ,$$
$$p_2(x) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left\{-\frac{1}{2}\sigma_2^2(x-\mu_2)^2\right\} .$$

- $p_1(x)p_2(x)$ is Gaussian with natural parameters $\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2$.

Product and Ratio of Gaussians

Consider these two Gaussian distributions:

$$p_1(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left\{-\frac{1}{2}\sigma_1^2(x-\mu_1)^2\right\} ,$$

$$p_2(x) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left\{-\frac{1}{2}\sigma_2^2(x-\mu_2)^2\right\} .$$

- $p_1(x)p_2(x)$ is Gaussian with natural parameters $\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2$.
- The log-normalization constant of $p_1(x)p_2(x)$ is $g(\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2) - g(\boldsymbol{\eta}_1) - g(\boldsymbol{\eta}_2)$.

Product and Ratio of Gaussians

Consider these two Gaussian distributions:

$$p_1(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left\{-\frac{1}{2}\sigma_1^2(x-\mu_1)^2\right\} ,$$
$$p_2(x) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left\{-\frac{1}{2}\sigma_2^2(x-\mu_2)^2\right\} .$$

- $p_1(x)p_2(x)$ is Gaussian with natural parameters $\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2$.
- The log-normalization constant of $p_1(x)p_2(x)$ is $g(\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2) - g(\boldsymbol{\eta}_1) - g(\boldsymbol{\eta}_2)$.
- $p_1(x)/p_2(x)$ is Gaussian with natural parameters $\boldsymbol{\eta}_1 - \boldsymbol{\eta}_2$.

Product and Ratio of Gaussians

Consider these two Gaussian distributions:

$$p_1(x) = \frac{1}{\sqrt{2\pi\sigma_1^2}} \exp\left\{-\frac{1}{2}\sigma_1^2(x-\mu_1)^2\right\} ,$$
$$p_2(x) = \frac{1}{\sqrt{2\pi\sigma_2^2}} \exp\left\{-\frac{1}{2}\sigma_2^2(x-\mu_2)^2\right\} .$$

- $p_1(x)p_2(x)$ is Gaussian with natural parameters $\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2$.
- The log-normalization constant of $p_1(x)p_2(x)$ is $g(\boldsymbol{\eta}_1 + \boldsymbol{\eta}_2) - g(\boldsymbol{\eta}_1) - g(\boldsymbol{\eta}_2)$.
- $p_1(x)/p_2(x)$ is Gaussian with natural parameters $\boldsymbol{\eta}_1 - \boldsymbol{\eta}_2$.
- The log-normalization constant of $p_1(x)/p_2(x)$ is $g(\boldsymbol{\eta}_1 - \boldsymbol{\eta}_2) - g(\boldsymbol{\eta}_1) + g(\boldsymbol{\eta}_2)$.

KL-Divergence Minimization

Consider the KL-divergence between p and q (q in the exponential family):

$$\mathrm{KL}(p||q) = -\int p(\mathbf{x}) \log \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x} = g(\boldsymbol{\eta}) - \boldsymbol{\eta}^{\mathsf{T}} \mathbb{E}_p[\mathbf{u}(\mathbf{x})] + \mathrm{Const}\,.$$

KL-Divergence Minimization

Consider the KL-divergence between p and q (q in the exponential family):

$$\mathrm{KL}(p||q) = -\int p(\mathbf{x}) \log \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x} = g(\boldsymbol{\eta}) - \boldsymbol{\eta}^{\mathrm{T}} \mathbb{E}_p[\mathbf{u}(\mathbf{x})] + \mathrm{Const}\,.$$

When **minimizing** with respect to the natural parameters $\boldsymbol{\eta}$ of q:

$$\frac{\partial \mathrm{KL}(p||q)}{\partial \boldsymbol{\eta}} = 0 \Longleftrightarrow \frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_p[\mathbf{u}(\mathbf{x})]\,,$$

KL-Divergence Minimization

Consider the KL-divergence between p and q (q in the exponential family):

$$\mathrm{KL}(p||q) = -\int p(\mathbf{x}) \log \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x} = g(\boldsymbol{\eta}) - \boldsymbol{\eta}^{\mathrm{T}} \mathbb{E}_p[\mathbf{u}(\mathbf{x})] + \mathrm{Const}\,.$$

When **minimizing** with respect to the natural parameters $\boldsymbol{\eta}$ of q:

$$\frac{\partial \mathrm{KL}(p||q)}{\partial \boldsymbol{\eta}} = 0 \Longleftrightarrow \frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_p[\mathbf{u}(\mathbf{x})]\,,$$

Furthermore, it is possible to show that:

$$\frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_q[\mathbf{u}(\mathbf{x})]\,.$$

KL-Divergence Minimization

Consider the KL-divergence between p and q (q in the exponential family):

$$\mathrm{KL}(p||q) = -\int p(\mathbf{x}) \log \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x} = g(\boldsymbol{\eta}) - \boldsymbol{\eta}^{\mathsf{T}} \mathbb{E}_p[\mathbf{u}(\mathbf{x})] + \mathrm{Const}\,.$$

When **minimizing** with respect to the natural parameters $\boldsymbol{\eta}$ of q:

$$\frac{\partial \mathrm{KL}(p||q)}{\partial \boldsymbol{\eta}} = 0 \Longleftrightarrow \frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_p[\mathbf{u}(\mathbf{x})]\,,$$

Furthermore, it is possible to show that:

$$\frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_q[\mathbf{u}(\mathbf{x})]\,.$$

$\mathrm{KL}(p||q)$ is minimized by matching expected sufficient statistics.

KL-Divergence Minimization

Consider the KL-divergence between p and q (q in the exponential family):

$$\mathrm{KL}(p||q) = -\int p(\mathbf{x}) \log \left\{ \frac{q(\mathbf{x})}{p(\mathbf{x})} \right\} d\mathbf{x} = g(\boldsymbol{\eta}) - \boldsymbol{\eta}^{\mathrm{T}} \mathbb{E}_p[\mathbf{u}(\mathbf{x})] + \mathrm{Const}\,.$$

When **minimizing** with respect to the natural parameters $\boldsymbol{\eta}$ of q:

$$\frac{\partial \mathrm{KL}(p||q)}{\partial \boldsymbol{\eta}} = 0 \iff \frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_p[\mathbf{u}(\mathbf{x})]\,,$$

Furthermore, it is possible to show that:

$$\frac{\partial g(\boldsymbol{\eta})}{\partial \boldsymbol{\eta}} = \mathbb{E}_q[\mathbf{u}(\mathbf{x})]\,.$$

KL($p||q$) is minimized by matching expected sufficient statistics.

If q is Gaussian, then we have to match $\mathbb{E}_q[\mathbf{x}] = \mathbb{E}_p[\mathbf{x}]$ and $\mathbb{E}_q[\mathbf{x}\mathbf{x}^{\mathrm{T}}] = \mathbb{E}_p[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$.

Joint Approximation

EP approximates this joint distribution by **a product of simpler factors**:

$$p(\mathbf{f}, \mathbf{y}) = \prod_{i=1}^{N} \phi_i(y_i f(\mathbf{x}_i) \mathcal{N}(\mathbf{f}|\mathbf{0}, \tilde{\mathbf{\Sigma}}) = \prod_i t_i(\mathbf{f}) \approx \prod_i \tilde{t}_i(\mathbf{f}) \,,$$

where each $\tilde{t}_i$ approximates the corresponding t_i. Each $\tilde{t}_i$ must **belong to the exponential family** but **need not be normalized**.

Joint Approximation

EP approximates this joint distribution by **a product of simpler factors**:

$$p(\mathbf{f}, \mathbf{y}) = \prod_{i=1}^{N} \phi_i(y_i f(\mathbf{x}_i) \mathcal{N}(\mathbf{f}|\mathbf{0}, \tilde{\mathbf{\Sigma}}) = \prod_i t_i(\mathbf{f}) \approx \prod_i \tilde{t}_i(\mathbf{f}),$$

where each $\tilde{t}_i$ approximates the corresponding t_i. Each $\tilde{t}_i$ must **belong to the exponential family** but **need not be normalized**.

The exponential family is closed under the product and $\prod_i \tilde{t}_i$ can be easily normalized to compute an approximate distribution:

$$p(\mathbf{f}|\mathbf{y}) = \frac{1}{p(\mathbf{y})} \prod_i t_i(\mathbf{f}) \approx \frac{1}{Z} \prod_i \tilde{t}_i(\mathbf{f}) = q(\mathbf{f}),$$

where $Z = \int \prod_i \tilde{t}_i(\mathbf{f}) d\mathbf{f}$ can be used to **approximate** $p(\mathbf{y})$.

Joint Approximation

EP approximates this joint distribution by **a product of simpler factors**:

$$p(\mathbf{f}, \mathbf{y}) = \prod_{i=1}^{N} \phi_i(y_i f(\mathbf{x}_i) \mathcal{N}(\mathbf{f}|\mathbf{0}, \tilde{\mathbf{\Sigma}}) = \prod_i t_i(\mathbf{f}) \approx \prod_i \tilde{t}_i(\mathbf{f})\,,$$

where each $\tilde{t}_i$ approximates the corresponding t_i. Each $\tilde{t}_i$ must **belong to the exponential family** but **need not be normalized**.

The exponential family is closed under the product and $\prod_i \tilde{t}_i$ can be easily normalized to compute an approximate distribution:

$$p(\mathbf{f}|\mathbf{y}) = \frac{1}{p(\mathbf{y})} \prod_i t_i(\mathbf{f}) \approx \frac{1}{Z} \prod_i \tilde{t}_i(\mathbf{f}) = q(\mathbf{f})\,,$$

where $Z = \int \prod_i \tilde{t}_i(\mathbf{f}) d\mathbf{f}$ can be used to **approximate** $p(\mathbf{y})$.

Therefore q has the same form as the approximate factors!

Approximate Factors

How do we determine each approximate factor $\tilde{t}_i$?

Approximate Factors

How do we determine each approximate factor $\tilde{t}_i$?

We would like to **minimize** $\mathrm{KL}(p||q)$, but this is **intractable**!

Approximate Factors

How do we determine each approximate factor $\tilde{t}_i$?

We would like to **minimize** $\mathrm{KL}(p||q)$, but this is **intractable**!

EP minimizes the KL divergence **between pairs** of t_i and $\tilde{t}_i$. This has the risk that **the product** may not be a good approximation. EP tries to **circumvent** this by an iterative procedure.

Approximate Factors

How do we determine each approximate factor $\tilde{t}_i$?

We would like to **minimize** $\mathrm{KL}(p||q)$, but this is **intractable**!

EP minimizes the KL divergence **between pairs** of t_i and $\tilde{t}_i$. This has the risk that **the product** may not be a good approximation. EP tries to **circumvent** this by an iterative procedure.

Suppose we wish to refine $\tilde{t}_j$. We first remove this factor from the product:

$$q^{\setminus j}(\mathbf{f}) \propto \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \propto q(\mathbf{f}) / \tilde{t}_j(\mathbf{f}) \,,$$

Approximate Factors

How do we determine each approximate factor $\tilde{t}_i$?

We would like to **minimize** $\mathrm{KL}(p||q)$, but this is **intractable**!

EP minimizes the KL divergence **between pairs** of t_i and $\tilde{t}_i$. This has the risk that **the product** may not be a good approximation. EP tries to **circumvent** this by an iterative procedure.

Suppose we wish to refine $\tilde{t}_j$. We first remove this factor from the product:

$$q^{\setminus j}(\mathbf{f}) \propto \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \propto q(\mathbf{f}) / \tilde{t}_j(\mathbf{f}) \,,$$

Then, $\tilde{t}_j$ is updated to minimize the KL-divergence between:

$$q_{\text{new}}(\mathbf{f}) \propto \tilde{t}_j(\mathbf{f}) q^{\setminus j}(\mathbf{f}) \,, \quad \hat{p}_j(\mathbf{f}) = \frac{1}{Z_j} t_j(\mathbf{f}) q^{\setminus j}(\mathbf{f}) \,, \quad Z_j = \int t_j(\mathbf{f}) q^{\setminus j}(\mathbf{f}) d\mathbf{f} \,,$$

where $q^{\setminus j}$ is fixed. This ensures that $\tilde{t}_j$ is accurate where $q^{\setminus j}$ is high.

Approximate Factors

In practice, $\tilde{t}_j$ is found by first **minimizing** with respect to q_{new}:

$$\mathrm{KL}\left(\frac{t_j(\mathbf{f}) q^{\setminus j}(\mathbf{f})}{Z_j} \middle| \, q_{\text{new}}(\mathbf{f}) \right) .$$

Approximate Factors

In practice, $\tilde{t}_j$ is found by first **minimizing** with respect to q_{new}:

$$\mathrm{KL}\left(\left. \frac{t_j(\mathbf{f}) q^{\setminus j}(\mathbf{f})}{Z_j} \right| q_{\text{new}}(\mathbf{f}) \right) .$$

This is done by **matching expected sufficient statistics**. As q is Gaussian, we only have to match the mean and the variance.

Approximate Factors

In practice, $\tilde{t}_j$ is found by first **minimizing** with respect to q_{new}:

$$\text{KL}\left(\frac{t_j(\mathbf{f})q^{\setminus j}(\mathbf{f})}{Z_j} \middle| q_{\text{new}}(\mathbf{f}) \right) .$$

This is done by **matching expected sufficient statistics**. As q is Gaussian, we only have to match the mean and the variance.

It is required that the moments of $\hat{p}_j(\mathbf{f}) = 1/Z_j f_j(\mathbf{f}) q^{\setminus j}(\mathbf{f})$ are tractable.

Approximate Factors

In practice, $\tilde{t}_j$ is found by first **minimizing** with respect to q_{new}:

$$\mathrm{KL}\left(\frac{t_j(\mathbf{f}) q^{\setminus j}(\mathbf{f})}{Z_j} \middle| q_{\text{new}}(\mathbf{f}) \right) .$$

This is done by **matching expected sufficient statistics**. As q is Gaussian, we only have to match the mean and the variance.

It is required that the moments of $\hat{p}_j(\mathbf{f}) = 1/Z_j f_j(\mathbf{f}) q^{\setminus j}(\mathbf{f})$ are tractable.

The refined factor $\tilde{t}_j$ is set in practice to be:

$$\tilde{t}_j(\mathbf{f}) = Z_j \frac{q_{\text{new}}(\mathbf{f})}{q^{\setminus j}(\mathbf{f})} , \qquad \text{with} \quad \tilde{t}_j(\mathbf{f}) q^{\setminus j}(\mathbf{f}) \propto q_{\text{new}} ,$$

which ensures that $\tilde{t}_j(\mathbf{f}) q^{\setminus j}(\mathbf{f})$ and $t_j(\mathbf{f}) q^{\setminus j}(\mathbf{f})$ **integrate the same**.

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

EP Algorithm in General: Computes q and an approximation to $p(\mathbf{y})$.

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

EP Algorithm in General: Computes q and an approximation to $p(\mathbf{y})$.

1. Initialize q and each $\tilde{t}_i$ to be uniform.

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

EP Algorithm in General: Computes q and an approximation to $p(\mathbf{y})$.

1. Initialize q and each $\tilde{t}_i$ to be uniform.
2. Repeat until convergence of the $\tilde{t}_i$:

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

EP Algorithm in General: Computes q and an approximation to $p(\mathbf{y})$.

1. Initialize q and each $\tilde{t}_i$ to be uniform.
2. Repeat until convergence of the $\tilde{t}_i$:
   1. Choose a factor $\tilde{t}_j$ to refine.

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

EP Algorithm in General: Computes q and an approximation to $p(\mathbf{y})$.

1. Initialize q and each $\tilde{t}_i$ to be uniform.
2. Repeat until convergence of the $\tilde{t}_i$:
    1. Choose a factor $\tilde{t}_j$ to refine.
    2. Remove $\tilde{t}_j$ from q by division $q^{\setminus j} \propto q/\tilde{t}_j$.

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

EP Algorithm in General: Computes q and an approximation to $p(\mathbf{y})$.

1. Initialize q and each $\tilde{t}_i$ to be uniform.
2. Repeat until convergence of the $\tilde{t}_i$:
    1. Choose a factor $\tilde{t}_j$ to refine.
    2. Remove $\tilde{t}_j$ from q by division $q^{\setminus j} \propto q/\tilde{t}_j$.
    3. Compute Z_j and $\hat{p}_j$ and find q_{new} by minimizing $\text{KL}(\hat{p}_j || q_{\text{new}})$.

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

EP Algorithm in General: Computes q and an approximation to $p(\mathbf{y})$.

1. Initialize q and each $\tilde{t}_i$ to be uniform.
2. Repeat until convergence of the $\tilde{t}_i$:
    1. Choose a factor $\tilde{t}_j$ to refine.
    2. Remove $\tilde{t}_j$ from q by division $q^{\setminus j} \propto q/\tilde{t}_j$.
    3. Compute Z_j and $\hat{p}_j$ and find q_{new} by minimizing $\text{KL}(\hat{p}_j || q_{\text{new}})$.
    4. Compute and store the new factor $\tilde{t}_j = Z_j q_{\text{new}} / q^{\setminus j}$.

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

EP Algorithm in General: Computes q and an approximation to $p(\mathbf{y})$.

1. Initialize q and each $\tilde{t}_i$ to be uniform.
2. Repeat until convergence of the $\tilde{t}_i$:
   1. Choose a factor $\tilde{t}_j$ to refine.
   2. Remove $\tilde{t}_j$ from q by division $q^{\backslash j} \propto q/\tilde{t}_j$.
   3. Compute Z_j and $\hat{p}_j$ and find q_{new} by minimizing $\text{KL}(\hat{p}_j || q_{\text{new}})$.
   4. Compute and store the new factor $\tilde{t}_j = Z_j q_{\text{new}} / q^{\backslash j}$.
3. Evaluate the approximation to the model evidence:

$$p(\mathbf{y}) \approx Z = \int \prod_j \tilde{t}_j(\mathbf{f}) d\mathbf{f} \,.$$

Full Algorithm

Several passes are made trough the factors until they **converge**. The **model evidence** is approximated by the **normalizing constant** of q.

EP Algorithm in General: Computes q and an approximation to $p(\mathbf{y})$.

1. Initialize q and each $\tilde{t}_i$ to be uniform.
2. Repeat until convergence of the $\tilde{t}_i$:
    1. Choose a factor $\tilde{t}_j$ to refine.
    2. Remove $\tilde{t}_j$ from q by division $q^{\setminus j} \propto q/\tilde{t}_j$.
    3. Compute Z_j and $\hat{p}_j$ and find q_{new} by minimizing $\text{KL}(\hat{p}_j || q_{\text{new}})$.
    4. Compute and store the new factor $\tilde{t}_j = Z_j q_{\text{new}} / q^{\setminus j}$.
3. Evaluate the approximation to the model evidence:

$$p(\mathbf{y}) \approx Z = \int \prod_j \tilde{t}_j(\mathbf{f}) d\mathbf{f} \, .$$

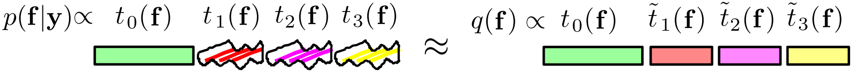
The FITC prior results in a total cost of $\mathcal{O}(NM^2)$!

Graphical Illustration

Approximates $p(\mathbf{f}|\mathbf{y}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} t_j(\mathbf{f})$ with $q(\mathbf{f}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} \tilde{t}_j(\mathbf{t})$

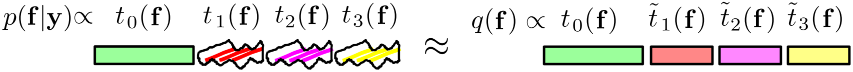
Graphical Illustration

Approximates $\boxed{p(\mathbf{f}|\mathbf{y}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} t_j(\mathbf{f})}$ with $\boxed{q(\mathbf{f}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} \tilde{t}_j(\mathbf{t})}$



Graphical Illustration

Approximates $\boxed{p(\mathbf{f}|\mathbf{y}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} t_j(\mathbf{f})}$ with $\boxed{q(\mathbf{f}) \propto t_0(\mathbf{f}) \prod_{j=1}^{N} \tilde{t}_j(\mathbf{t})}$



The $\tilde{t}_j$ are tuned by minimizing the KL-divergence

$$\mathrm{KL}[\hat{p}_j || q] \quad \text{for } j = 1, \ldots, N, \quad \text{where} \quad \begin{array}{rcl} \hat{p}_j(\mathbf{f}) & \propto & t_j(\mathbf{f}) \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \\ q(\mathbf{f}) & \propto & \tilde{t}_j(\mathbf{f}) \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \end{array}.$$

Graphical Illustration

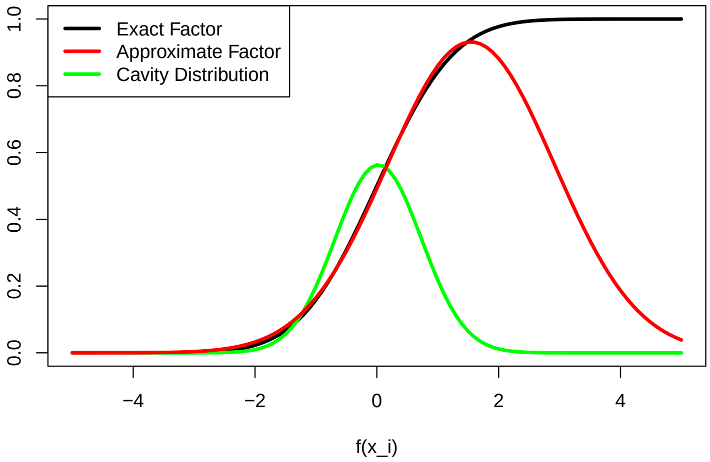
Approximates $\boxed{p(\mathbf{f}|\mathbf{y}) \propto t_0(\mathbf{f}) \prod_{j=1}^N t_j(\mathbf{f})}$ with $\boxed{q(\mathbf{f}) \propto t_0(\mathbf{f}) \prod_{j=1}^N \tilde{t}_j(\mathbf{t})}$

The $\tilde{t}_j$ are tuned by minimizing the KL-divergence

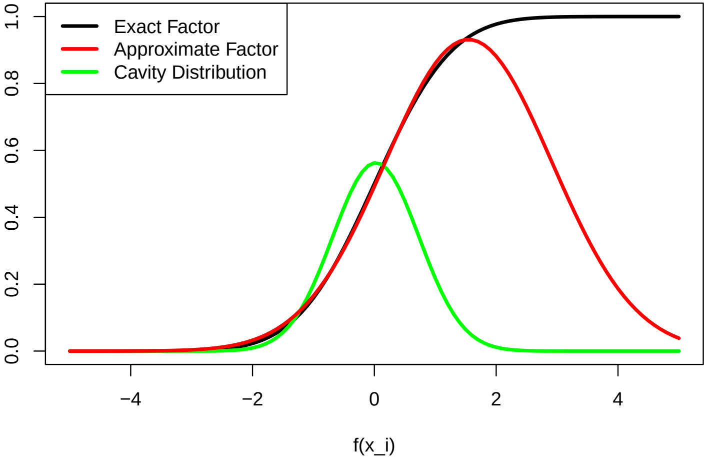
$$\mathrm{KL}[\hat{p}_j || q] \quad \text{for } j = 1, \ldots, N\,, \quad \text{where} \quad \begin{array}{rcl} \hat{p}_j(\mathbf{f}) & \propto & t_j(\mathbf{f}) \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \\ q(\mathbf{f}) & \propto & \tilde{t}_j(\mathbf{f}) \prod_{i \neq j} \tilde{t}_i(\mathbf{f}) \end{array}.$$

If the exact factor already belongs to the exponential family it needs not be approximated!

GFITC: Factor Approximation



GFITC: Factor Approximation



The approximate factor is accurate in regions of high posterior probability as indicated by the cavity distribution!

Hyper-parameters and Inducing Points

They are optimized by maximizing the EP estimate of the log-marginal likelihood $\log Z \approx \log p(\mathbf{y})$**.**

Hyper-parameters and Inducing Points

They are optimized by maximizing the EP estimate of the log-marginal likelihood $\log Z \approx \log p(\mathbf{y})$.

Problem: The parameters θ_i of the approximate factors also depend on the hyper-parameters (including the inducing points)!

Hyper-parameters and Inducing Points

They are optimized by maximizing the EP estimate of the log-marginal likelihood $\log Z \approx \log p(\mathbf{y})$.

Problem: The parameters θ_i of the approximate factors also depend on the hyper-parameters (including the inducing points)!

- Direct dependence of $\log Z$ on the hyper-parameters.

Hyper-parameters and Inducing Points

They are optimized by maximizing the EP estimate of the log-marginal likelihood $\log Z \approx \log p(\mathbf{y})$.

Problem: The parameters θ_i of the approximate factors also depend on the hyper-parameters (including the inducing points)!

- Direct dependence of $\log Z$ on the hyper-parameters.
- Indirect dependence of $\log Z$ on the hyper-parameters via each θ_i.

Hyper-parameters and Inducing Points

They are optimized by maximizing the EP estimate of the log-marginal likelihood $\log Z \approx \log p(\mathbf{y})$.

Problem: The parameters θ_i of the approximate factors also depend on the hyper-parameters (including the inducing points)!

- Direct dependence of $\log Z$ on the hyper-parameters.
- Indirect dependence of $\log Z$ on the hyper-parameters via each θ_i.

If EP converges the gradient of $\log Z$ w.r.t. each θ_i is zero, which allows to easily compute the gradients of $\log Z$!

GFITC: Predictions

We want to compute the value of $\mathbf{f}^{\star}$ at a new $\mathbf{x}^{\star}$:

GFITC: Predictions

We want to compute the value of $\mathbf{f}^\star$ at a new $\mathbf{x}^\star$:

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

GFITC: Predictions

We want to compute the value of $\mathbf{f}^\star$ at a new $\mathbf{x}^\star$:

$$p(\mathbf{f}^\star, \mathbf{f}) = \mathcal{N}\left(\left[\begin{array}{c} \mathbf{f}^\star \\ \mathbf{f} \end{array} \right] \middle| \left[\begin{array}{c} \mathbf{0} \\ \mathbf{0} \end{array} \right], \left[\begin{array}{cc} \mathbf{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} & \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \\ \mathbf{Q}_{\mathbf{f}\mathbf{f}^\star} & \tilde{\mathbf{\Sigma}}_{\mathbf{f}\mathbf{f}} \end{array} \right] \right)$$

The conditional $p(\mathbf{f}^\star|\mathbf{f})$ is:

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

GFITC: Predictions

We want to compute the value of $\mathbf{f}^\star$ at a new $\mathbf{x}^\star$:

$$p(\mathbf{f}^\star, \mathbf{f}) = \mathcal{N}\left(\left[\begin{array}{c} \mathbf{f}^\star \\ \mathbf{f} \end{array} \right] \middle| \left[\begin{array}{c} \mathbf{0} \\ \mathbf{0} \end{array} \right], \left[\begin{array}{cc} \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} & \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \\ \mathbf{Q}_{\mathbf{f}\mathbf{f}^\star} & \tilde{\boldsymbol{\Sigma}}_{\mathbf{f}\mathbf{f}} \end{array} \right] \right)$$

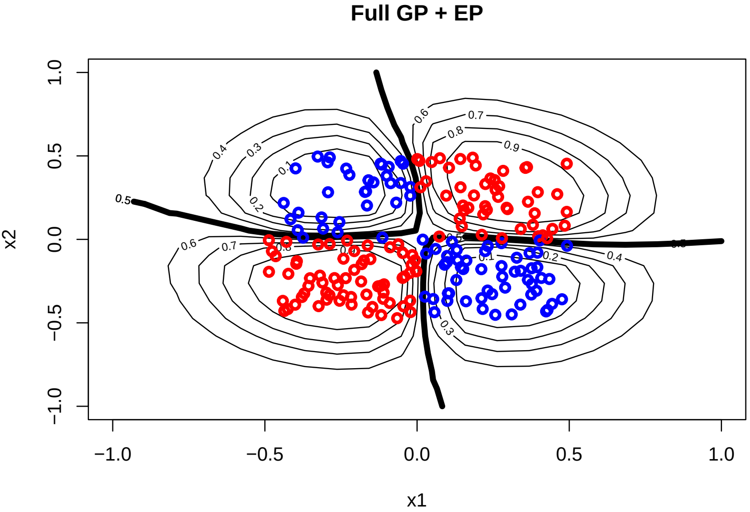
The conditional $p(\mathbf{f}^\star|\mathbf{f})$ is:

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

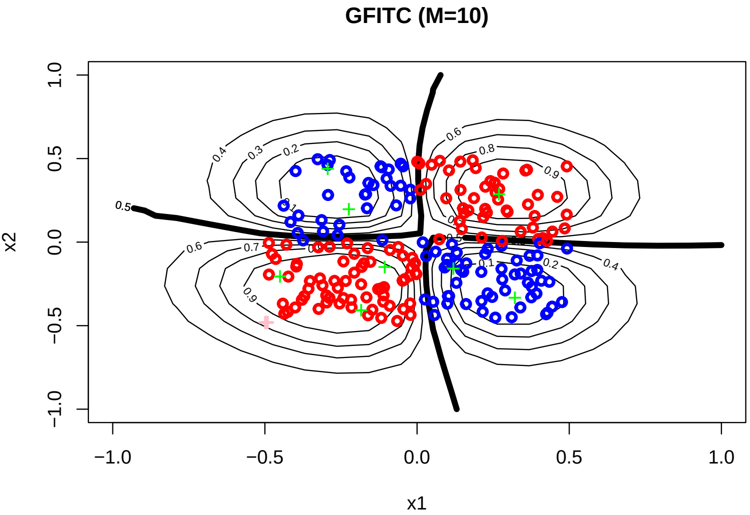
After marginalizing $\mathbf{f}$ w.r.t. $q(\mathbf{f})$, we obtain the predictive distribution:

$$\begin{aligned} p(\mathbf{f}^\star|\mathbf{y}) &= \int p(\mathbf{f}^\star|\mathbf{f}) q(\mathbf{f}) d\mathbf{f} \\ &= \mathcal{N}\left(\mathbf{f}^\star \middle| \, \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \left(\tilde{\boldsymbol{\Sigma}}_{\mathbf{f}\mathbf{f}} + \tilde{\boldsymbol{\Pi}} \right)^{-1} \tilde{\mathbf{y}}, \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} - \mathbf{Q}_{\mathbf{f}^\star\mathbf{f}} \left(\tilde{\boldsymbol{\Sigma}}_{\mathbf{f}\mathbf{f}} + \tilde{\boldsymbol{\Pi}} \right)^{-1} \mathbf{Q}_{\mathbf{f}\mathbf{f}^\star} \right) \end{aligned}$$

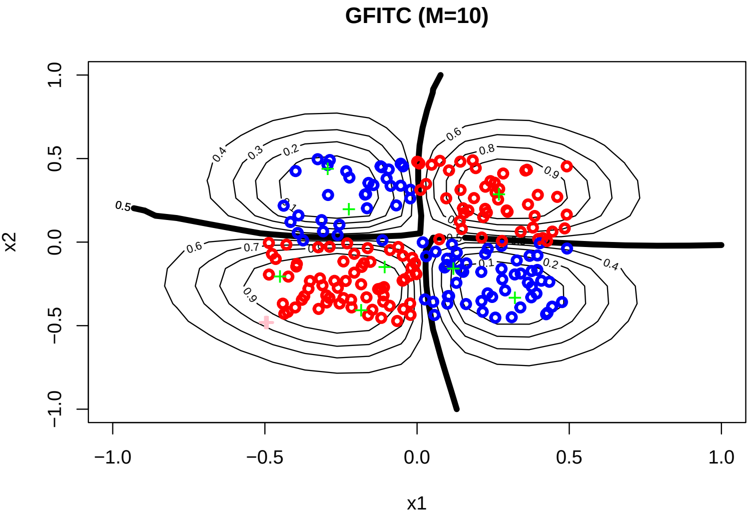
GFITC: Illustrative Example



GFITC: Illustrative Example



GFITC: Illustrative Example



The inducing points spread across the input space!

Variational Free Energy

Previous methods approximate the GP prior using a low rank approximation of Σ, resulting in a cost $\mathcal{O}(NM^2)$.

Variational Free Energy

Previous methods approximate the GP prior using a low rank approximation of Σ, resulting in a cost $\mathcal{O}(NM^2)$.

Variational Free Energy (VFE) Method:

Variational Free Energy

Previous methods approximate the GP prior using a low rank approximation of Σ, resulting in a cost $\mathcal{O}(NM^2)$.

Variational Free Energy (VFE) Method:

- Keeps the GP prior intact and does not introduce any simplification!

Variational Free Energy

Previous methods approximate the GP prior using a low rank approximation of Σ, resulting in a cost $\mathcal{O}(NM^2)$.

Variational Free Energy (VFE) Method:

- Keeps the GP prior intact and does not introduce any simplification!
- Carries out approximate inference to approximate the GP posterior.

Variational Free Energy

Previous methods approximate the GP prior using a low rank approximation of Σ, resulting in a cost $\mathcal{O}(NM^2)$.

Variational Free Energy (VFE) Method:

- Keeps the GP prior intact and does not introduce any simplification!
- Carries out approximate inference to approximate the GP posterior.
- The particular approximate distribution q results in cost $\mathcal{O}(NM^2)$.

Variational Free Energy

Previous methods approximate the GP prior using a low rank approximation of Σ, resulting in a cost $\mathcal{O}(NM^2)$.

Variational Free Energy (VFE) Method:

- Keeps the GP prior intact and does not introduce any simplification!
- Carries out approximate inference to approximate the GP posterior.
- The particular approximate distribution q results in cost $\mathcal{O}(NM^2)$.
- Variational inference is used to tune q.

Variational Free Energy

Previous methods approximate the GP prior using a low rank approximation of Σ, resulting in a cost $\mathcal{O}(NM^2)$.

Variational Free Energy (VFE) Method:

- Keeps the GP prior intact and does not introduce any simplification!
- Carries out approximate inference to approximate the GP posterior.
- The particular approximate distribution q results in cost $\mathcal{O}(NM^2)$.
- Variational inference is used to tune q.

Since the GP prior is not changed it tends to perform better than the previous methods!

Variational Inference

Adjust the parameters of q to match p by minimizing $\mathbf{KL}(q|p) \geq 0$.

Variational Inference

Adjust the parameters of q to match p by minimizing $\mathbf{KL}(q|p) \geq 0$.

$$\mathsf{KL}(q|p) = 0 \Longleftrightarrow q(\mathbf{f}) = p(\mathbf{f})$$

Variational Inference

Adjust the parameters of q to match p by minimizing $\mathbf{KL}(q|p) \geq 0$.

$$\mathrm{KL}(q|p) = 0 \Longleftrightarrow q(\mathbf{f}) = p(\mathbf{f})$$

The expression for the KL divergence between q and p is:

$$\int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f})} d\mathbf{f} \geq 0$$

Variational Inference

Adjust the parameters of q to match p by minimizing $\mathbf{KL}(q|p) \geq 0$.

$$\mathrm{KL}(q|p) = 0 \Longleftrightarrow q(\mathbf{f}) = p(\mathbf{f})$$

The expression for the KL divergence between q and p is:

$$\int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f})} d\mathbf{f} \geq 0$$

$\mathbf{KL}(q|p)$ depends on p, which is assumed to be intractable!

Variational Inference

Adjust the parameters of q to match p by minimizing KL$(q|p) \geq 0$.

$$\mathrm{KL}(q|p) = 0 \Longleftrightarrow q(\mathbf{f}) = p(\mathbf{f})$$

The expression for the KL divergence between q and p is:

$$\int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f})} d\mathbf{f} \geq 0$$

KL$(q|p)$ depends on p, which is assumed to be intractable!

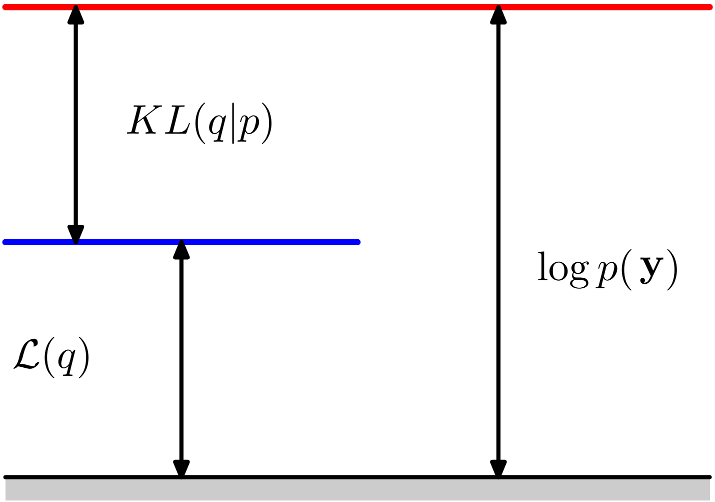
Let the target be $p(\mathbf{f}|\mathbf{y})$. Consider the decomposition of $p(\mathbf{y})$:

$$\log p(\mathbf{y}) = \mathcal{L}(q) + \mathrm{KL}(q|p)\,,$$

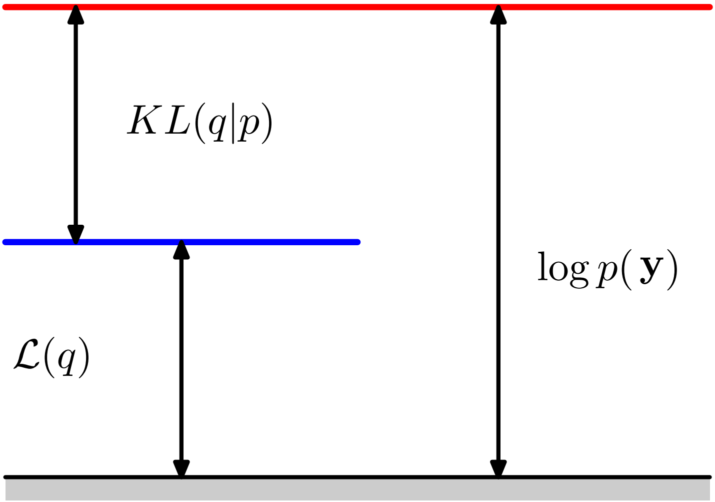
where

$$\mathcal{L}(q) = \int q(\mathbf{f}) \log \frac{p(\mathbf{f},\mathbf{y})}{q(\mathbf{f})} d\mathbf{f}\,, \qquad \mathrm{KL}(q|p) = \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f}|\mathbf{y})} d\mathbf{f}\,.$$

Decomposition of the Marginal Likelihood



Decomposition of the Marginal Likelihood

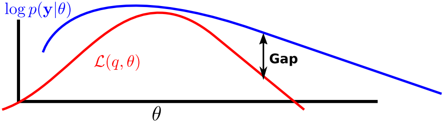


$\mathcal{L}(q)$ can be used to approximate $\log p(\mathbf{y})$ if $\mathbf{KL}(q|p)$ is small!

Variational Free Energy (VFE)

Lower bound the log-likelihood:

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

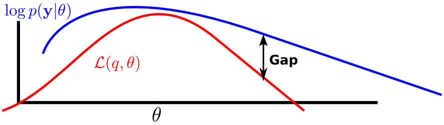


Variational Free Energy (VFE)

Lower bound the log-likelihood:

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

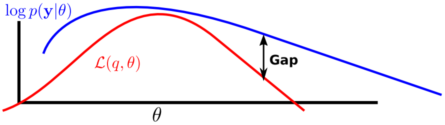
$$= \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) \frac{q(\mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u}$$



Variational Free Energy (VFE)

Lower bound the log-likelihood:

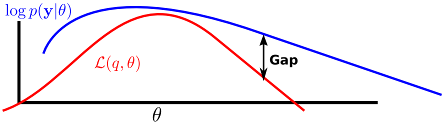
$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta) d\mathbf{f} d\mathbf{u}$$



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

Variational Free Energy (VFE)

Lower bound the log-likelihood:



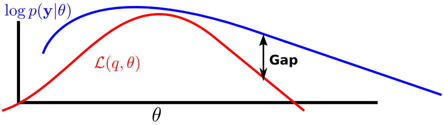
$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y},\mathbf{f},\mathbf{u}|\theta) d\mathbf{f} d\mathbf{u}$$

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

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

Variational Free Energy (VFE)

Lower bound the log-likelihood:



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

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

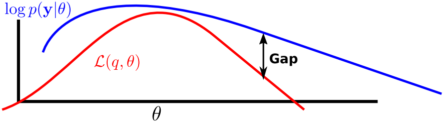
$$\mathcal{L}(q, \theta) = \int q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u}|\theta)}{q(\mathbf{f}, \mathbf{u})} d\mathbf{f} d\mathbf{u} = \log p(\mathbf{y}|\theta) - \mathbf{KL}[q(\mathbf{f}, \mathbf{u})|p(\mathbf{f}, \mathbf{u}|\mathbf{y})]$$

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:

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

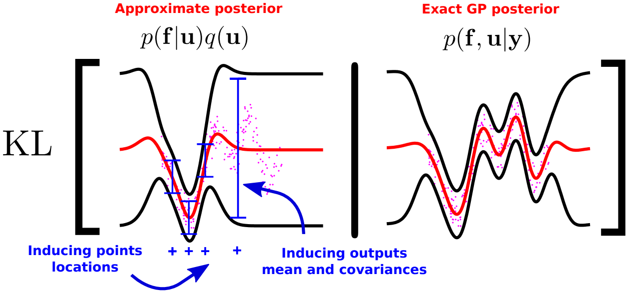
- Fixed
- Tunable

Variational Free Energy (VFE)

Consider the following approximate distribution:

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

- Fixed
- Tunable

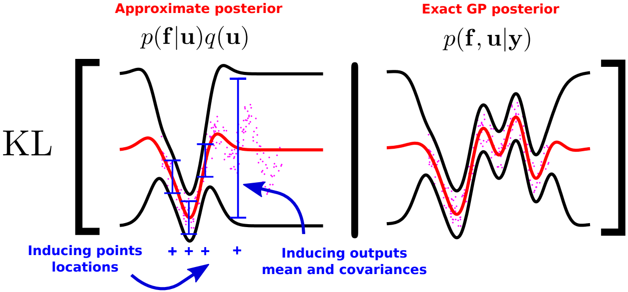


Variational Free Energy (VFE)

Consider the following approximate distribution:

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

- 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:

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

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

Variational Free Energy (VFE)

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

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

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

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

- Mean squared prediction error
- KL between Gaussians

Variational Free Energy (VFE)

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

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

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

- 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:

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

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

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

- Mean squared prediction error (→ $\mathbb{E}_{q(\mathbf{f})}[\log p(\mathbf{y}|\mathbf{f}, \theta)]$)
- KL between Gaussians (→ $\mathbf{KL}[q(\mathbf{u})|p(\mathbf{u})]$)

- 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 $\mathbf{u}$.

VFE: Predictions for Test Instances

We want to compute the value of $\mathbf{f}^\star$ at a new $\mathbf{x}^\star$:

VFE: Predictions for Test Instances

We want to compute the value of $\mathbf{f}^\star$ at a new $\mathbf{x}^\star$:

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

VFE: Predictions for Test Instances

We want to compute the value of $\mathbf{f}^\star$ at a new $\mathbf{x}^\star$:

$$p(\mathbf{f}^\star, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f}^\star \\ \mathbf{u} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{\Sigma_{f^\star f^\star}} & \mathbf{\Sigma_{f^\star u}} \\ \mathbf{\Sigma_{u f^\star}} & \mathbf{\Sigma_{uu}} \end{bmatrix} \right)$$

The conditional $p(\mathbf{f}^\star|\mathbf{u})$ is:

$$p(\mathbf{f}^\star|\mathbf{u}) = \mathcal{N}\left(\mathbf{f}^\star \middle| \mathbf{\Sigma_{f^\star u}} \mathbf{\Sigma_{uu}^{-1}} \mathbf{u}, \mathbf{\Sigma_{f^\star f^\star}} - \mathbf{\Sigma_{f^\star u}} \mathbf{\Sigma_{uu}^{-1}} \mathbf{\Sigma_{u f^\star}} \right)$$

VFE: Predictions for Test Instances

We want to compute the value of $\mathbf{f}^\star$ at a new $\mathbf{x}^\star$:

$$p(\mathbf{f}^\star, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f}^\star \\ \mathbf{u} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} & \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{u}} \\ \boldsymbol{\Sigma}_{\mathbf{u}\mathbf{f}^\star} & \boldsymbol{\Sigma}_{\mathbf{u}\mathbf{u}} \end{bmatrix} \right)$$

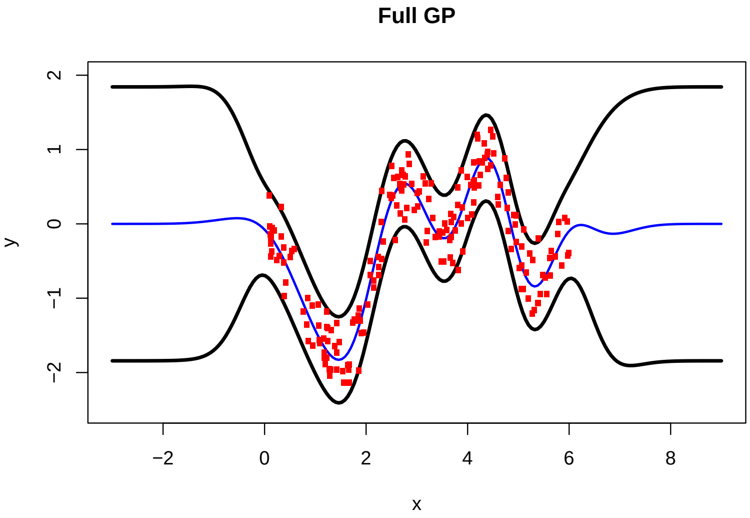
The conditional $p(\mathbf{f}^\star|\mathbf{u})$ is:

$$p(\mathbf{f}^\star|\mathbf{u}) = \mathcal{N}\left(\mathbf{f}^\star \middle| \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{u}}\boldsymbol{\Sigma}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{u}, \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} - \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{u}}\boldsymbol{\Sigma}_{\mathbf{u}\mathbf{u}}^{-1}\boldsymbol{\Sigma}_{\mathbf{u}\mathbf{f}^\star}\right)$$

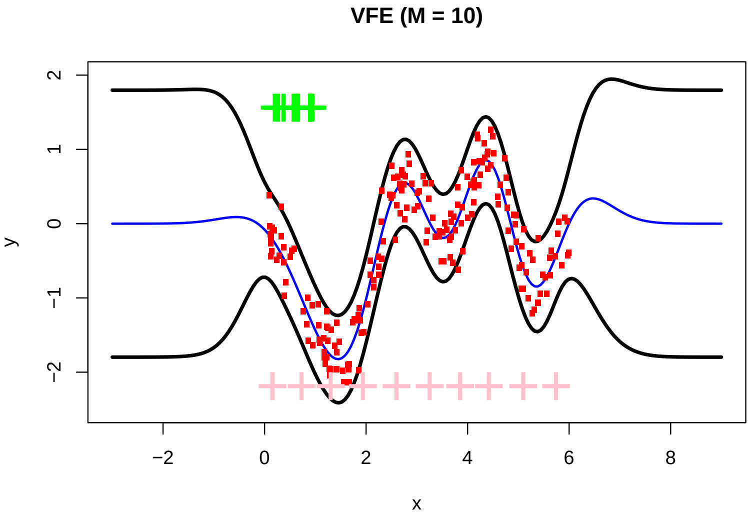
After marginalizing $\mathbf{u}$ w.r.t. $q(\mathbf{u})$, we obtain the predictive distribution:

$$\begin{aligned} p(\mathbf{f}^\star|\mathbf{y}) &= \int p(\mathbf{f}^\star|\mathbf{u})q(\mathbf{u})d\mathbf{u} \\ &= \mathcal{N}\left(\mathbf{f}^\star \middle| \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{u}}\boldsymbol{\Sigma}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{m}, \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{f}^\star} - \boldsymbol{\Sigma}_{\mathbf{f}^\star\mathbf{u}}\left(\boldsymbol{\Sigma}_{\mathbf{u}\mathbf{u}}^{-1} - \boldsymbol{\Sigma}_{\mathbf{u}\mathbf{u}}^{-1}\mathbf{S}\boldsymbol{\Sigma}_{\mathbf{u}\mathbf{u}}^{-1}\right)\boldsymbol{\Sigma}_{\mathbf{u}\mathbf{f}^\star}\right) \end{aligned}$$

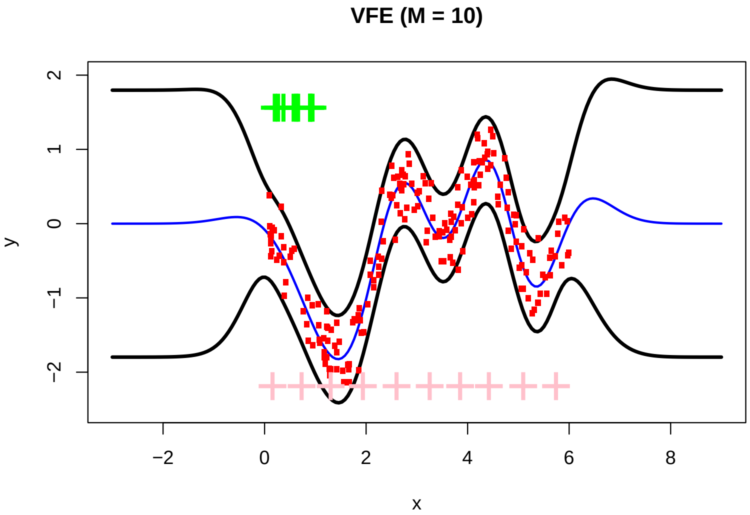
VFE: Illustrative Example



VFE: Illustrative Example

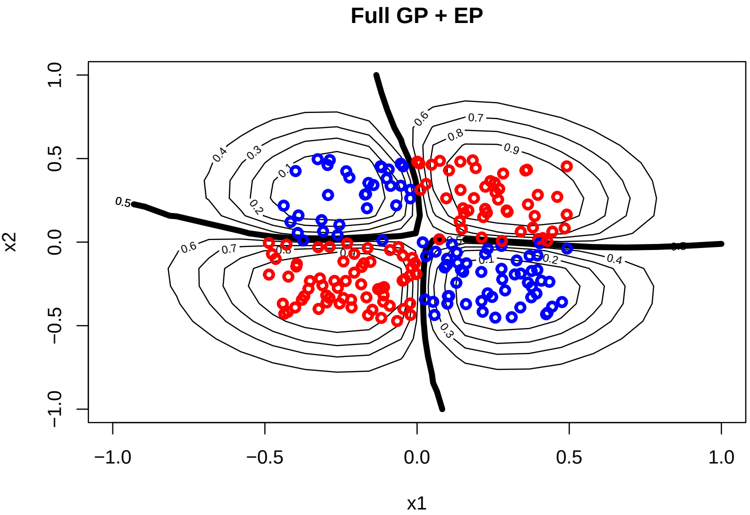


VFE: Illustrative Example

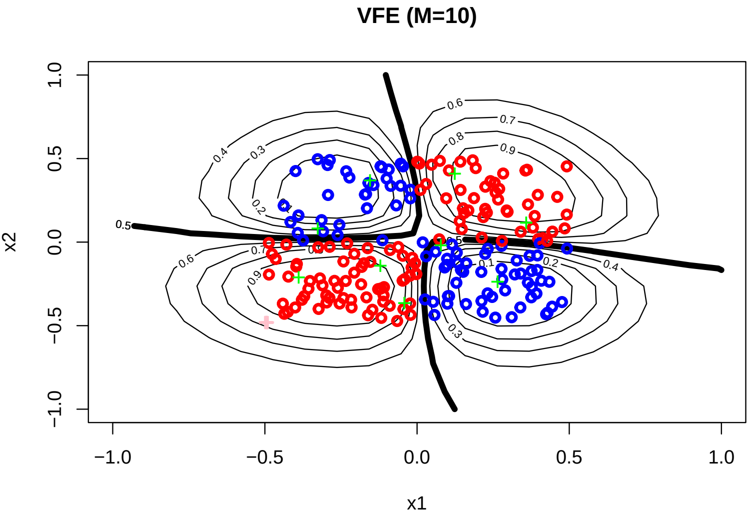


The inducing points cover the regions where the function changes!

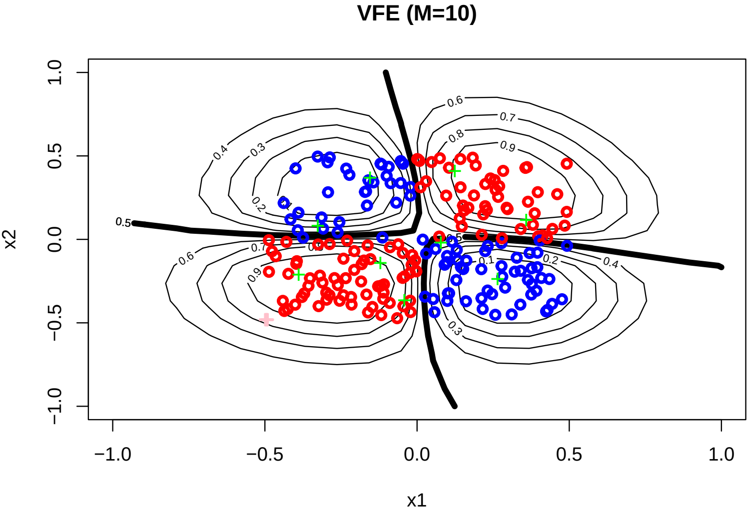
VFE: Illustrative Classification Example



VFE: Illustrative Classification Example



VFE: Illustrative Classification Example



The inducing points spread across the input space!

FITC vs. VFE

Two approaches:

FITC vs. VFE

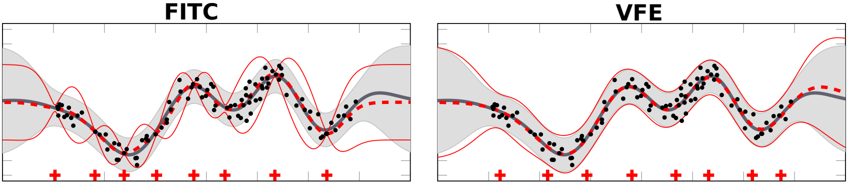
Two approaches:

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

FITC vs. VFE

Two approaches:

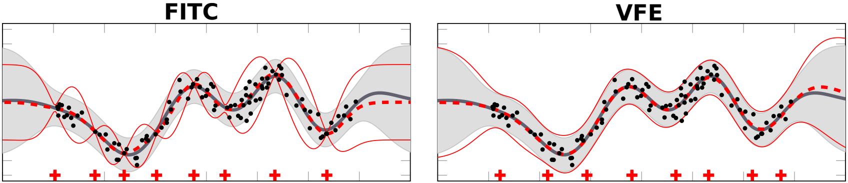
- FITC: optimize the marginal likelihood of an approximate GP model.
- VFE: maximize fidelity to the original exact GP.



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)

Run the cells of the notebook to fit a sparse GP using the VFE approximation and complete task 6!

Whitened Parameterization for VFE

Alternative VFE objective expected to be easier to optimize!

Whitened Parameterization for VFE

Alternative VFE objective expected to be easier to optimize!

Instead of making inference about $\mathbf{u}$, the whitened VFE objective makes inference about:

$$\mathbf{e} \quad \text{such that} \quad \mathbf{u} = \mathbf{L}\mathbf{e}\,, \qquad\qquad \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})\,,$$

with $\mathbf{u}$ the latent process values at the inducing points and $\mathbf{L}^{\mathsf{T}}\mathbf{L} = \mathbf{\Sigma}_{\mathbf{uu}}$.

Whitened Parameterization for VFE

Alternative VFE objective expected to be easier to optimize!

Instead of making inference about $\mathbf{u}$, the whitened VFE objective makes inference about:

$$\mathbf{e} \quad \text{such that} \quad \mathbf{u} = \mathbf{L}\mathbf{e}\,, \qquad\qquad \mathbf{e} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})\,,$$

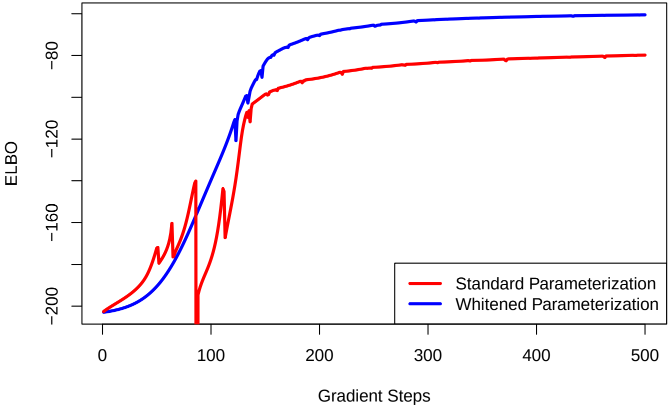
with $\mathbf{u}$ the latent process values at the inducing points and $\mathbf{L}^\text{T}\mathbf{L} = \mathbf{\Sigma}_{\mathbf{uu}}$.

The VFE objective becomes:

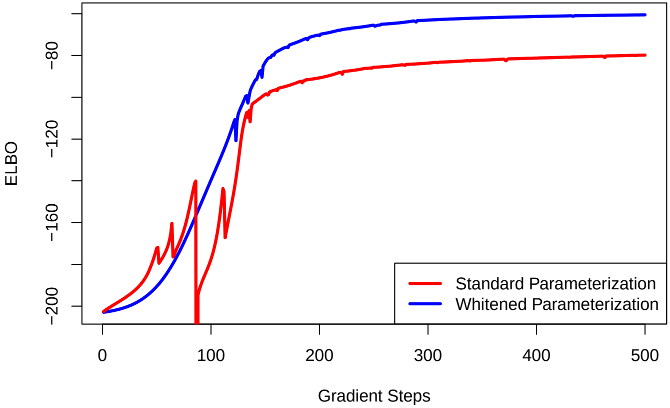
$$\sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{e})p(f(\mathbf{x}_i)|\mathbf{e})} [\log p(y_i|f(\mathbf{x}_i))] - \text{KL}(q(\mathbf{e})|\mathcal{N}(\mathbf{0}, \mathbf{I}))\,,$$

with $p(f(\mathbf{x}_i)|\mathbf{e})$ using the covariances between $f(\mathbf{x}_i)$ and $\mathbf{e}$.

Whitened Parameterization: Illustrative Example



Whitened Parameterization: Illustrative Example



Whitening significantly improves convergence!

Run the cells of the notebook to fit a sparse GP using the VFE approximation with whitening!

Natural Gradient Ascent

Gradient ascent moves in the direction of the gradient $\nabla_{\xi}\mathcal{L}(\xi)$.

Natural Gradient Ascent

Gradient ascent moves in the direction of the gradient $\nabla_{\boldsymbol{\xi}}\mathcal{L}(\boldsymbol{\xi})$.

Formally:

$$\nabla_{\boldsymbol{\xi}}\mathcal{L}(\boldsymbol{\xi}) \propto \lim_{\epsilon \to 0} \frac{1}{\epsilon} \operatorname*{arg\,max}_{\mathbf{d}\; s.t. ||\mathbf{d}|| \leq \epsilon} \mathcal{L}(\boldsymbol{\xi} + \mathbf{d})$$

Natural Gradient Ascent

Gradient ascent moves in the direction of the gradient $\nabla_{\boldsymbol{\xi}}\mathcal{L}(\boldsymbol{\xi})$.

Formally:

$$\nabla_{\boldsymbol{\xi}}\mathcal{L}(\boldsymbol{\xi}) \propto \lim_{\epsilon\to 0} \frac{1}{\epsilon} \operatorname*{arg\,max}_{\mathbf{d}\; s.t. ||\mathbf{d}|| \leq \epsilon} \mathcal{L}(\boldsymbol{\xi} + \mathbf{d})$$

The steepest ascent direction picks d in the ϵ-vicinity of $\boldsymbol{\xi}$ (measured by the Euclidean norm) that maximizes $\mathcal{L}(\cdot)$.

Natural Gradient Ascent

Gradient ascent moves in the direction of the gradient $\nabla_{\boldsymbol{\xi}}\mathcal{L}(\boldsymbol{\xi})$.

Formally:

$$\nabla_{\boldsymbol{\xi}}\mathcal{L}(\boldsymbol{\xi}) \propto \lim_{\epsilon \to 0} \frac{1}{\epsilon} \operatorname*{arg\,max}_{\mathbf{d}\; s.t. ||\mathbf{d}|| \leq \epsilon} \mathcal{L}(\boldsymbol{\xi} + \mathbf{d})$$

The steepest ascent direction picks d in the ϵ-vicinity of $\boldsymbol{\xi}$ (measured by the Euclidean norm) that maximizes $\mathcal{L}(\cdot)$.

If $\boldsymbol{\xi}$ represents the parameters of probability distributions, the Euclidean norm may be problematic!

Illustration with Two Gaussians

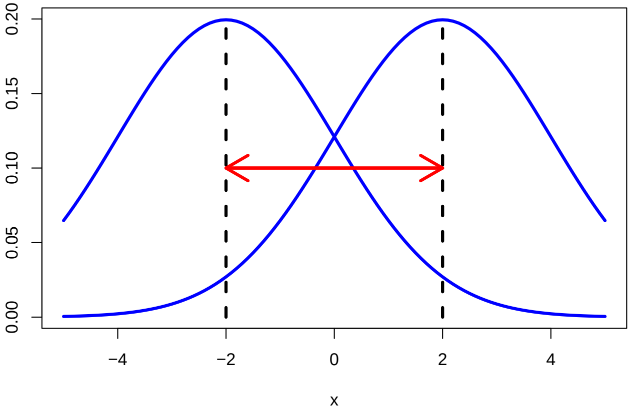


Illustration with Two Gaussians

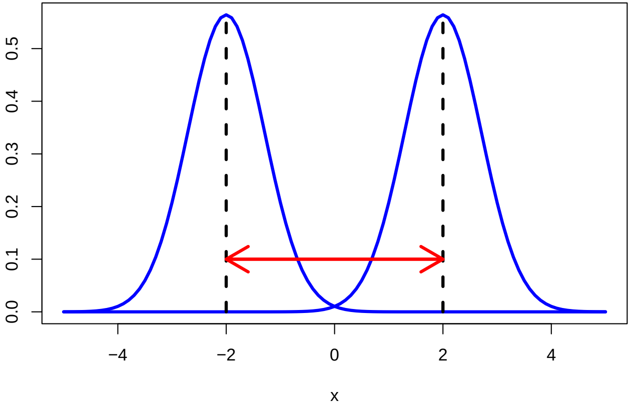
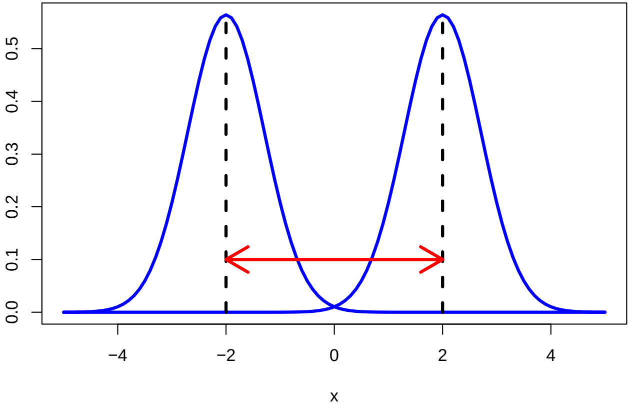
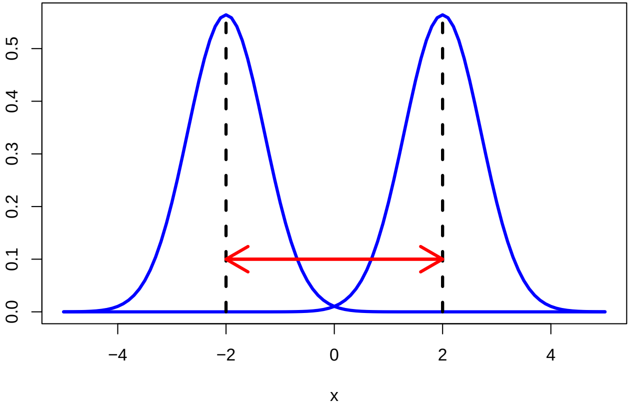


Illustration with Two Gaussians



The Euclidean distance between parameters is 4 in both cases!

Illustration with Two Gaussians



A better alternative is the KL-divergence between distributions!

Natural Gradient Ascent

Considers the KL-divergence as a norm:

$$\nabla_{\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\xi}) \mathbf{F}_{\boldsymbol{\xi}}^{-1} \propto \lim_{\epsilon \to 0} \frac{1}{\epsilon} \underset{\mathbf{d} \, s.t. \mathrm{KL}[q(\mathbf{u}|\boldsymbol{\xi}) | q(\mathbf{u}|\boldsymbol{\xi}+\mathbf{d})] \leq \epsilon}{\arg\max} \mathcal{L}(\boldsymbol{\xi} + \mathbf{d})$$

with $\mathbf{F}_{\boldsymbol{\xi}}$ the Fisher information of q:

$$\mathbf{F}_{\boldsymbol{\xi}} = -\mathbb{E}_{q(\mathbf{u}|\boldsymbol{\xi})} [\nabla_{\boldsymbol{\xi}}^2 \log q(\mathbf{u}|\boldsymbol{\xi})]$$

Natural Gradient Ascent

Considers the KL-divergence as a norm:

$$\nabla_{\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\xi}) \mathbf{F}_{\boldsymbol{\xi}}^{-1} \propto \lim_{\epsilon \to 0} \frac{1}{\epsilon} \underset{\mathbf{d}\; s.t. \mathrm{KL}[q(\mathbf{u}|\boldsymbol{\xi})|q(\mathbf{u}|\boldsymbol{\xi}+\mathbf{d})] \leq \epsilon}{\arg\max} \mathcal{L}(\boldsymbol{\xi} + \mathbf{d})$$

with $\mathbf{F}_{\boldsymbol{\xi}}$ the Fisher information of q:

$$\mathbf{F}_{\boldsymbol{\xi}} = -\mathbb{E}_{q(\mathbf{u}|\boldsymbol{\xi})}[\nabla^2_{\boldsymbol{\xi}} \log q(\mathbf{u}|\boldsymbol{\xi})]$$

Let $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ be the natural and expectation parameters of q, respectively:

$$\mathbf{F}_{\boldsymbol{\eta}} = \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\eta}}, \qquad \mathbf{F}_{\boldsymbol{\xi}} = \left(\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\xi}}\right)^{\mathsf{T}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\eta}} \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\xi}} = \left(\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\xi}}\right)^{\mathsf{T}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\xi}}.$$

Natural Gradient Ascent

Considers the KL-divergence as a norm:

$$\nabla_{\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\xi}) \mathbf{F}_{\boldsymbol{\xi}}^{-1} \propto \lim_{\epsilon \to 0} \frac{1}{\epsilon} \underset{\mathbf{d}\; s.t. \mathrm{KL}[q(\mathbf{u}|\boldsymbol{\xi}) | q(\mathbf{u}|\boldsymbol{\xi}+\mathbf{d})] \leq \epsilon}{\arg\max} \mathcal{L}(\boldsymbol{\xi} + \mathbf{d})$$

with $\mathbf{F}_{\boldsymbol{\xi}}$ the Fisher information of q:

$$\mathbf{F}_{\boldsymbol{\xi}} = -\mathbb{E}_{q(\mathbf{u}|\boldsymbol{\xi})}[\nabla_{\boldsymbol{\xi}}^2 \log q(\mathbf{u}|\boldsymbol{\xi})]$$

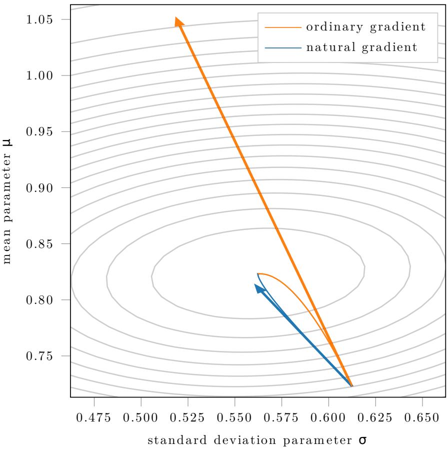
Let $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ be the natural and expectation parameters of q, respectively:

$$\mathbf{F}_{\boldsymbol{\eta}} = \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\eta}}, \qquad \mathbf{F}_{\boldsymbol{\xi}} = \left(\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\xi}}\right)^{\mathsf{T}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\eta}} \frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\xi}} = \left(\frac{\partial \boldsymbol{\eta}}{\partial \boldsymbol{\xi}}\right)^{\mathsf{T}} \frac{\partial \boldsymbol{\theta}}{\partial \boldsymbol{\xi}}.$$

Thus,

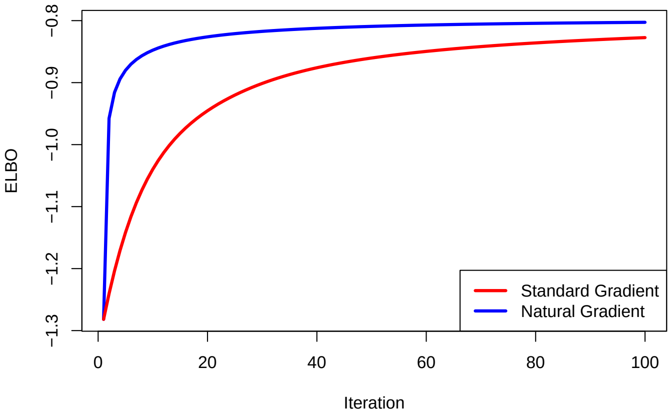
$$\nabla_{\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\xi}) \mathbf{F}_{\boldsymbol{\xi}}^{-1} = \nabla_{\boldsymbol{\xi}} \mathcal{L}(\boldsymbol{\xi}) \frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\theta}} \left(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\eta}}\right)^{\mathsf{T}} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} \left(\frac{\partial \boldsymbol{\xi}}{\partial \boldsymbol{\eta}}\right)^{\mathsf{T}}.$$

Natural Gradient Ascent

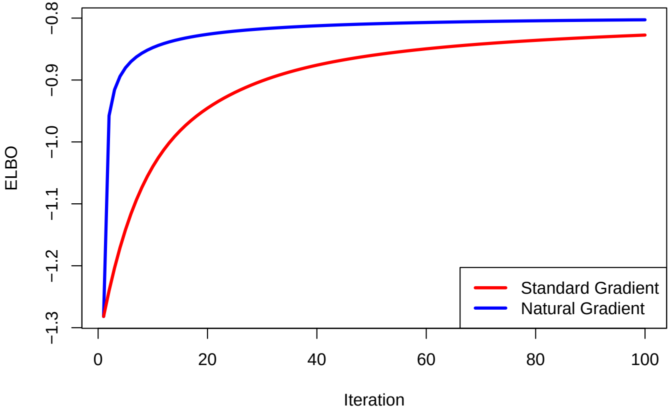


(Salimbeni et al., 2018)

Natural Gradient: Illustrative Example



Natural Gradient: Illustrative Example



The natural gradient achieves a faster convergence!

Run the cells of the notebook to fit a sparse GP using the VFE approximation with natural gradients!

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:

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

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!

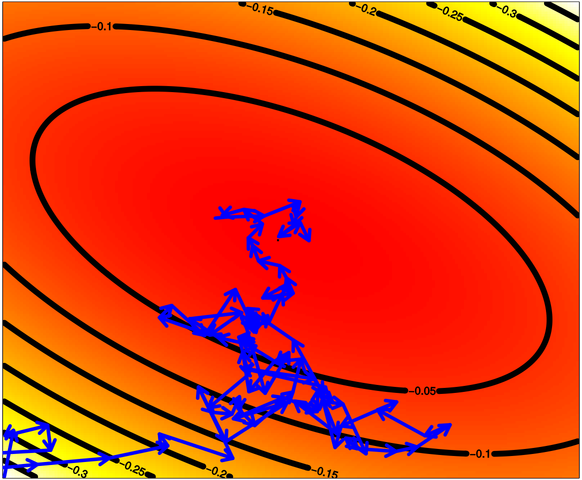
Straight forward to do that in the VFE approach:

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

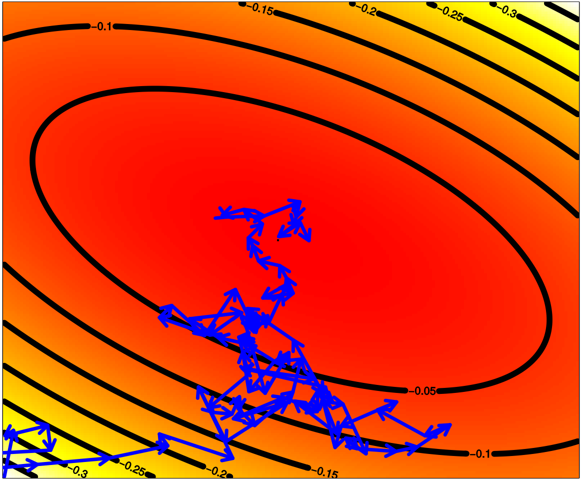
The training cost goes down to $\mathcal{O}(M^3)$ which allows to address datasets with millions of instances!

(Hensman et al., 2013)

GPs for Big Data

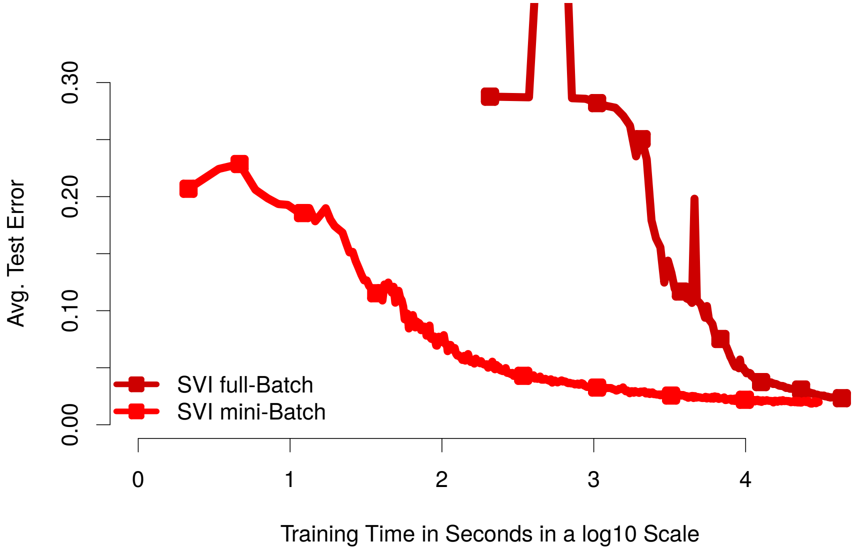


GPs for Big Data



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 about VFE

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.

Summary about VFE

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- The optimized inducing points spread over the input space where the latent function changes.

Summary about VFE

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- The optimized inducing points spread over the input space where the latent function changes.
- Guaranteed to be exact if $M = N$ and the inducing points are not optimized and located at the training points.

Summary about VFE

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- The optimized inducing points spread over the input space where the latent function changes.
- Guaranteed to be exact if $M = N$ and the inducing points are not optimized and located at the training points.
- It does not change the model. It relies on a particular posterior approximation that speeds-up the computations.

Summary about VFE

- Reduces the cost to $\mathcal{O}(M^2N)$ with $M \ll N$.
- The optimized inducing points spread over the input space where the latent function changes.
- Guaranteed to be exact if $M = N$ and the inducing points are not optimized and located at the training points.
- It does not change the model. It relies on a particular posterior approximation that speeds-up the computations.
- It allows for minibatch training which reduces the cost to $\mathcal{O}(M^3)$.

Summary about VFE

- Reduces the cost to $\mathcal{O}(M^2 N)$ with $M \ll N$.
- The optimized inducing points spread over the input space where the latent function changes.
- Guaranteed to be exact if $M = N$ and the inducing points are not optimized and located at the training points.
- It does not change the model. It relies on a particular posterior approximation that speeds-up the computations.
- It allows for minibatch training which reduces the cost to $\mathcal{O}(M^3)$.
- The objective is prone to local optima and difficult to optimize.

Run the cells of the notebook to fit a sparse GP using the VFE approximation with mini-batches!

Run the cells of the notebook to fit a sparse GP using the VFE approximation for classification and carry out task 7!

Sparse GP Conclusions

- Exact GPs have an $\mathcal{O}(N^3)$ computational cost, making them feasible on small datasets with a few thousand instances only.

Sparse GP Conclusions

- Exact GPs have an $\mathcal{O}(N^3)$ computational cost, making them feasible on small datasets with a few thousand instances only.
- Sparse GPs provide an approximate solution with a smaller computational cost that is $\mathcal{O}(NM^2)$ with $M \ll N$.

Sparse GP Conclusions

- Exact GPs have an $\mathcal{O}(N^3)$ computational cost, making them feasible on small datasets with a few thousand instances only.
- Sparse GPs provide an approximate solution with a smaller computational cost that is $\mathcal{O}(NM^2)$ with $M \ll N$.
- The non-parametric property of GP is lost when using sparse approximations. They are no longer more flexible with more data.

Sparse GP Conclusions

- Exact GPs have an $\mathcal{O}(N^3)$ computational cost, making them feasible on small datasets with a few thousand instances only.
- Sparse GPs provide an approximate solution with a smaller computational cost that is $\mathcal{O}(NM^2)$ with $M \ll N$.
- The non-parametric property of GP is lost when using sparse approximations. They are no longer more flexible with more data.
- The methods that approximate the GP prior often introduce a low-rank structure in the covariance matrix.

Sparse GP Conclusions

- Exact GPs have an $\mathcal{O}(N^3)$ computational cost, making them feasible on small datasets with a few thousand instances only.
- Sparse GPs provide an approximate solution with a smaller computational cost that is $\mathcal{O}(NM^2)$ with $M \ll N$.
- The non-parametric property of GP is lost when using sparse approximations. They are no longer more flexible with more data.
- The methods that approximate the GP prior often introduce a low-rank structure in the covariance matrix.
- The best performing method seems to be the VFE method since it does not modify the prior.

Sparse GP Conclusions

- Exact GPs have an $\mathcal{O}(N^3)$ computational cost, making them feasible on small datasets with a few thousand instances only.
- Sparse GPs provide an approximate solution with a smaller computational cost that is $\mathcal{O}(NM^2)$ with $M \ll N$.
- The non-parametric property of GP is lost when using sparse approximations. They are no longer more flexible with more data.
- The methods that approximate the GP prior often introduce a low-rank structure in the covariance matrix.
- The best performing method seems to be the VFE method since it does not modify the prior.
- Some methods allow for stochastic optimization and mini-batch training that further reduce the cost to $\mathcal{O}(M^3)$.

References

- Williams, C., & Seeger, M. (2000). Using the Nyström method to speed up kernel machines. Advances in neural information processing systems, 13.
- Snelson, E., & Ghahramani, Z. (2005). Sparse Gaussian processes using pseudo-inputs. Advances in neural information processing systems, 18.
- Rahimi, A., & Recht, B. (2007). Random features for large-scale kernel machines. Advances in neural information processing systems, 20.
- Naish-Guzman, A., & Holden, S. (2007). The generalized FITC approximation. Advances in neural information processing systems, 20.
- Hernández-Lobato, D., & Hernández-Lobato, J. M. (2016, May). Scalable Gaussian process classification via expectation propagation. In Artificial Intelligence and Statistics (pp. 168-176).
- Hensman, J., Fusi, N., & Lawrence, N. D. (2013). Gaussian processes for big data. Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence.
- Salimbeni, H., Eleftheriadis, S., & Hensman, J. (2018, March). Natural gradients in practice: Non-conjugate variational inference in Gaussian process models. In International Conference on Artificial Intelligence and Statistics (pp. 689-697).
- Wu, L., Miller, A., Anderson, L., Pleiss, G., Blei, D., & Cunningham, J. (2021). Hierarchical inducing point gaussian process for inter-domian observations. International Conference on Artificial Intelligence and Statistics (pp. 2926-2934).