Distributed Kernel Representations for Variational Sparse Gaussian Processes

Michalis K. Titsias
Department of Informatics,
Athens University of Economics and Business, Greece

Outline

- Variational learning of inducing variables
- Distributed representations of GPs (kernel functions) that can facilitate approximations in large datasets

Gaussian process regression

Inputs $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and outputs $\mathbf{y} = (y_1, \ldots, y_n)$ such that

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Gaussian process regression

Inputs $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and outputs $\mathbf{y} = (y_1, \ldots, y_n)$ such that

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Place GP prior on latent function $f(\mathbf{x})$:

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}')),$$

Gaussian process regression

Inputs $X = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$ and outputs $\mathbf{y} = (y_1, \ldots, y_n)$ such that

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2)$$

Place GP prior on latent function $f(\mathbf{x})$:

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}')),$$

Given that we have n data our current "marginal model" is

$$p(\mathbf{y}|\mathbf{f})p(\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2 I)\mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}_{ff}), \quad [\mathbf{K}_{ff}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$$

where $\mathbf{f} = (f_1, \ldots, f_n)$ are the parameters

Inducing variables

We need approximations for big Data

Inducing variables

We need approximations for big Data

The problem is that $\mathbf{f}$ grows as we collect more data

Inducing variables

We need approximations for big Data

The problem is that $\mathbf{f}$ grows as we collect more data

Idea: Summarize/replace $\mathbf{f}$ by a smaller parameter vector $\mathbf{u}$

Inducing variables

We need approximations for big Data

The problem is that $\mathbf{f}$ grows as we collect more data

Idea: Summarize/replace $\mathbf{f}$ by a smaller parameter vector $\mathbf{u}$

The size of $\mathbf{u}$ must be user-controllable based on current computational resources

- it could grow if the computational capacity increase in future

Inducing variables

Inducing variables $\mathbf{u}$ form a vector of user-controllable size that augments the GP prior:

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fu} \\ \mathbf{K}_{uf} & \mathbf{K}_{uu} \end{bmatrix} \right), \; \mathbf{K}_{fu} = \mathbb{E}[\mathbf{f}\mathbf{u}^T], \; \mathbf{K}_{uu} = \mathbb{E}[\mathbf{u}\mathbf{u}^T]$$

Inducing variables

Inducing variables $\mathbf{u}$ form a vector of user-controllable size that augments the GP prior:

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fu} \\ \mathbf{K}_{uf} & \mathbf{K}_{uu} \end{bmatrix} \right), \; \mathbf{K}_{fu} = \mathbb{E}[\mathbf{f}\mathbf{u}^T], \; \mathbf{K}_{uu} = \mathbb{E}[\mathbf{u}\mathbf{u}^T]$$

$\mathbf{u}$ can be:

- a subset of $\mathbf{f}$
- values of $f(\mathbf{x})$ at arbitrary "pseudo-inputs"
- arbitrary linear functionals, e.g. $u = z_i f(\mathbf{x}_i) + z_j f(\mathbf{x}_j)$

Inducing variables

Inducing variables $\mathbf{u}$ form a vector of user-controllable size that augments the GP prior:

$$p(\mathbf{f}, \mathbf{u}) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ \mathbf{u} \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{fu} \\ \mathbf{K}_{uf} & \mathbf{K}_{uu} \end{bmatrix} \right), \; \mathbf{K}_{fu} = \mathbb{E}[\mathbf{f}\mathbf{u}^T], \; \mathbf{K}_{uu} = \mathbb{E}[\mathbf{u}\mathbf{u}^T]$$

$\mathbf{u}$ can be:

- a subset of $\mathbf{f}$
- values of $f(\mathbf{x})$ at arbitrary "pseudo-inputs"
- arbitrary linear functionals, e.g. $u = z_i f(\mathbf{x}_i) + z_j f(\mathbf{x}_j)$

The augmentation with $\mathbf{u}$ adds some parameters Z

- indices that specify the subset in $\mathbf{f}$, pseudo-inputs, weights etc
- $\mathbf{K}_{fu}$ and $\mathbf{K}_{uu}$ depend on those parameters

Inducing variables

The whole purpose of adding **u** is to help us obtain an approximation to our Bayesian non-parametric model (**without changing its non-parametric nature**) that will scale better computationally

Inducing variables

The whole purpose of adding $\mathbf{u}$ is to help us obtain an approximation to our Bayesian non-parametric model (**without changing its non-parametric nature**) that will scale better computationally

How do we "turn around" $\mathbf{u}$ in order to make it the basis of our approximation? Further, how do we learn the augmentation parameters Z?

Variational learning of inducing variables

Augmented joint

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

Variational learning of inducing variables

Augmented joint

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

Augmented exact posterior

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

Variational learning of inducing variables

Augmented joint

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

Augmented exact posterior

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

Marginal likelihood is invariant to the augmentation parameters Z

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

and the marginal posterior $p(\mathbf{f}|\mathbf{y})$ is also invariant to Z

- **I.e. Z is not model parameter**
- **$\Rightarrow$ we can turn it into variational parameter by lower bounding**

Variational learning of inducing variables

Joint

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

Variational learning of inducing variables

Joint

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

Exact posterior distribution

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

Variational learning of inducing variables

Joint

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

Exact posterior distribution

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

Variational distribution

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

Variational learning of inducing variables

Joint

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

Exact posterior distribution

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

Variational distribution

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

This choice encourages $\mathbf{u}$ to become approximate sufficient statistic

if $p(\mathbf{f}|\mathbf{u}) \approx p(\mathbf{f}|\mathbf{u}, \mathbf{y})$, then $\mathbf{u}$ summarizes well the data

Variational learning of inducing variables

Minimize KL $[q(\mathbf{f}, \mathbf{u})||p(\mathbf{f}, \mathbf{u}|\mathbf{y})]$ or equivalently maximize the bound on the log marginal likelihood

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

Variational learning of inducing variables

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

Variational learning of inducing variables

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

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

Variational learning of inducing variables

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

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

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

Variational learning of inducing variables

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

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

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

Substitute $q(\mathbf{f},\mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$:

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

Variational learning of inducing variables

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

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

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

Substitute $q(\mathbf{f},\mathbf{u}) = p(\mathbf{f}|\mathbf{u})q(\mathbf{u})$:

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

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

Variational learning of inducing variables

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \left[\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} + \log \frac{p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{u}$$

Variational learning of inducing variables

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \left[\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} + \log \frac{p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{u}$$

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \left[\log e^{\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f}} + \log \frac{p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{u}$$

Variational learning of inducing variables

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \left[\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} + \log \frac{p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{u}$$

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \left[\log e^{\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f}} + \log \frac{p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{u}$$

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \log \frac{e^{\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f}} p(\mathbf{u})}{q(\mathbf{u})} d\mathbf{u}$$

Variational learning of inducing variables

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \left[\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f} + \log \frac{p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{u}$$

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \left[\log e^{\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f}} + \log \frac{p(\mathbf{u})}{q(\mathbf{u})} \right] d\mathbf{u}$$

$$\log p(\mathbf{y}) \geq \int q(\mathbf{u}) \log \frac{e^{\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f}} p(\mathbf{u})}{q(\mathbf{u})} d\mathbf{u}$$

Maximize over $q(\mathbf{u})$:

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

Variational learning of inducing variables

Bound:

For arbitrary GP model:

$$p(\mathbf{y}) \geq \int G(\mathbf{y}, \mathbf{u}) p(\mathbf{u}) d\mathbf{u}, \quad G(\mathbf{y}, \mathbf{u}) = e^{\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f}}$$

For GP regression:

$$\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{ff} + \sigma^2 I) \geq \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf} + \sigma^2 I) e^{-\frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}\right)}$$

Variational learning of inducing variables

Bound:

For arbitrary GP model:

$$p(\mathbf{y}) \geq \int G(\mathbf{y}, \mathbf{u}) p(\mathbf{u}) d\mathbf{u}, \quad G(\mathbf{y}, \mathbf{u}) = e^{\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f}}$$

For GP regression:

$$\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{ff} + \sigma^2 I) \geq \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf} + \sigma^2 I) e^{-\frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}\right)}$$

Monotonicity property: If we have inducing variables $\mathbf{u}$ and add an extra u_i the bound can only increase

$$\int G(\mathbf{y}, \mathbf{u}, u_i) p(\mathbf{u}, u_i) d\mathbf{u} \geq \int G(\mathbf{y}, \mathbf{u}) p(\mathbf{u}) d\mathbf{u}$$

Variational learning of inducing variables

Bound:

For arbitrary GP model:

$$p(\mathbf{y}) \geq \int G(\mathbf{y}, \mathbf{u}) p(\mathbf{u}) d\mathbf{u}, \quad G(\mathbf{y}, \mathbf{u}) = e^{\int p(\mathbf{f}|\mathbf{u}) \log p(\mathbf{y}|\mathbf{f}) d\mathbf{f}}$$

For GP regression:

$$\mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{ff} + \sigma^2 I) \geq \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf} + \sigma^2 I) e^{-\frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}\right)}$$

Monotonicity property: If we have inducing variables $\mathbf{u}$ and add an extra u_i the bound can only increase

$$\int G(\mathbf{y}, \mathbf{u}, u_i) p(\mathbf{u}, u_i) d\mathbf{u} \geq \int G(\mathbf{y}, \mathbf{u}) p(\mathbf{u}) d\mathbf{u}$$

Computation of the bound and the approximate GP prediction scale as $O(nm^2)$ where m is the number of inducing variables

Variational learning of inducing variables

For GP regression the bound has an interesting form:

$$\mathcal{F}(Z,\theta) = \log \mathcal{N}(\mathbf{y}|\mathbf{0}, \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf} + \sigma^2 I) - \frac{1}{2\sigma^2}\mathrm{tr}\left(\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{uf}\right)$$

Variational learning of inducing variables

The approximate posterior/predictive Gaussian process:

$$
\begin{aligned}
q(\mathbf{f}_*) &= \int p(\mathbf{f}_*|\mathbf{f},\mathbf{u})q(\mathbf{f},\mathbf{u})d\mathbf{f}d\mathbf{u} \\
&= \int p(\mathbf{f}_*|\mathbf{f},\mathbf{u})p(\mathbf{f}|\mathbf{u})q(\mathbf{u})d\mathbf{f}d\mathbf{u} \\
&= \int p(\mathbf{f}_*|\mathbf{u})q(\mathbf{u})d\mathbf{u}
\end{aligned}
$$

where we used the consistency $\int p(\mathbf{f}_*|\mathbf{f},\mathbf{u})p(\mathbf{f}|\mathbf{u})d\mathbf{f} = p(\mathbf{f}|\mathbf{u})$

Variational learning of inducing variables

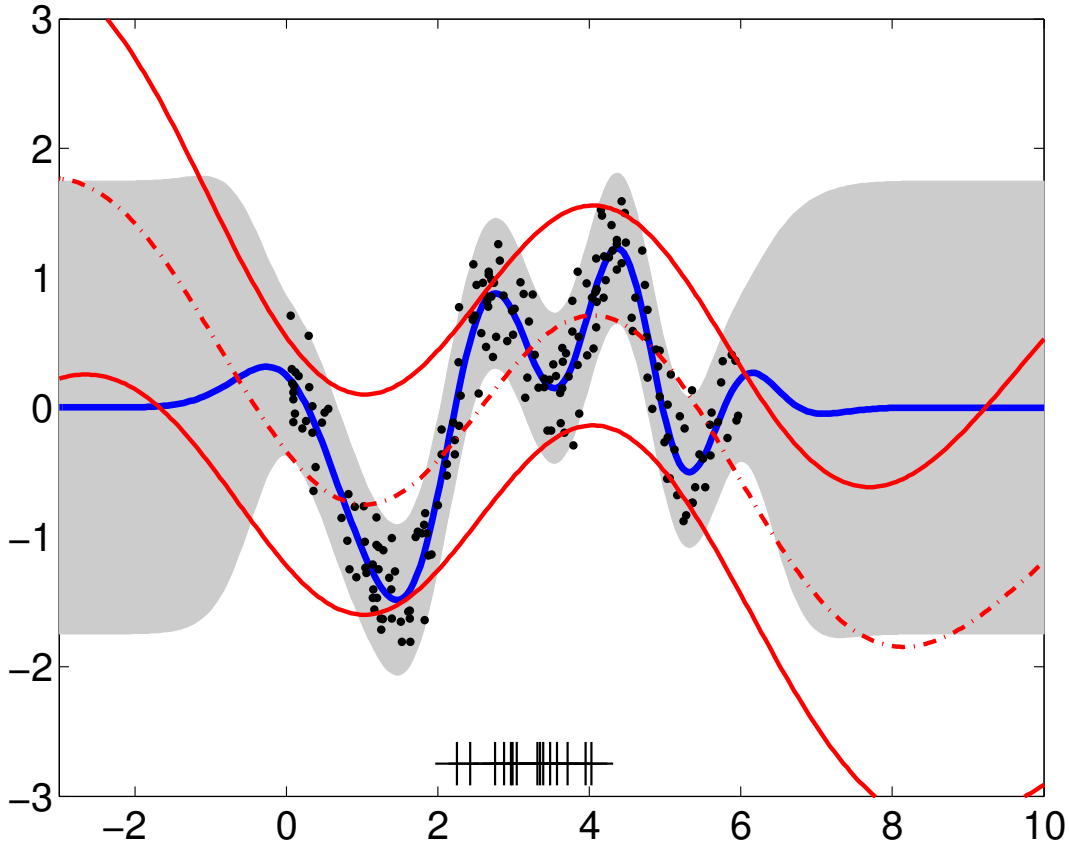
The approximate posterior/predictive Gaussian process:

$$
\begin{aligned}
q(\mathbf{f}_*) &= \int p(\mathbf{f}_*|\mathbf{f},\mathbf{u})q(\mathbf{f},\mathbf{u})d\mathbf{f}d\mathbf{u} \\
&= \int p(\mathbf{f}_*|\mathbf{f},\mathbf{u})p(\mathbf{f}|\mathbf{u})q(\mathbf{u})d\mathbf{f}d\mathbf{u} \\
&= \int p(\mathbf{f}_*|\mathbf{u})q(\mathbf{u})d\mathbf{u}
\end{aligned}
$$

where we used the consistency $\int p(\mathbf{f}_*|\mathbf{f},\mathbf{u})p(\mathbf{f}|\mathbf{u})d\mathbf{f} = p(\mathbf{f}|\mathbf{u})$

The approximation can be thought of been **restricted not to explore freely the information in the training data**. But it maintains fully the non-parametric nature of the model

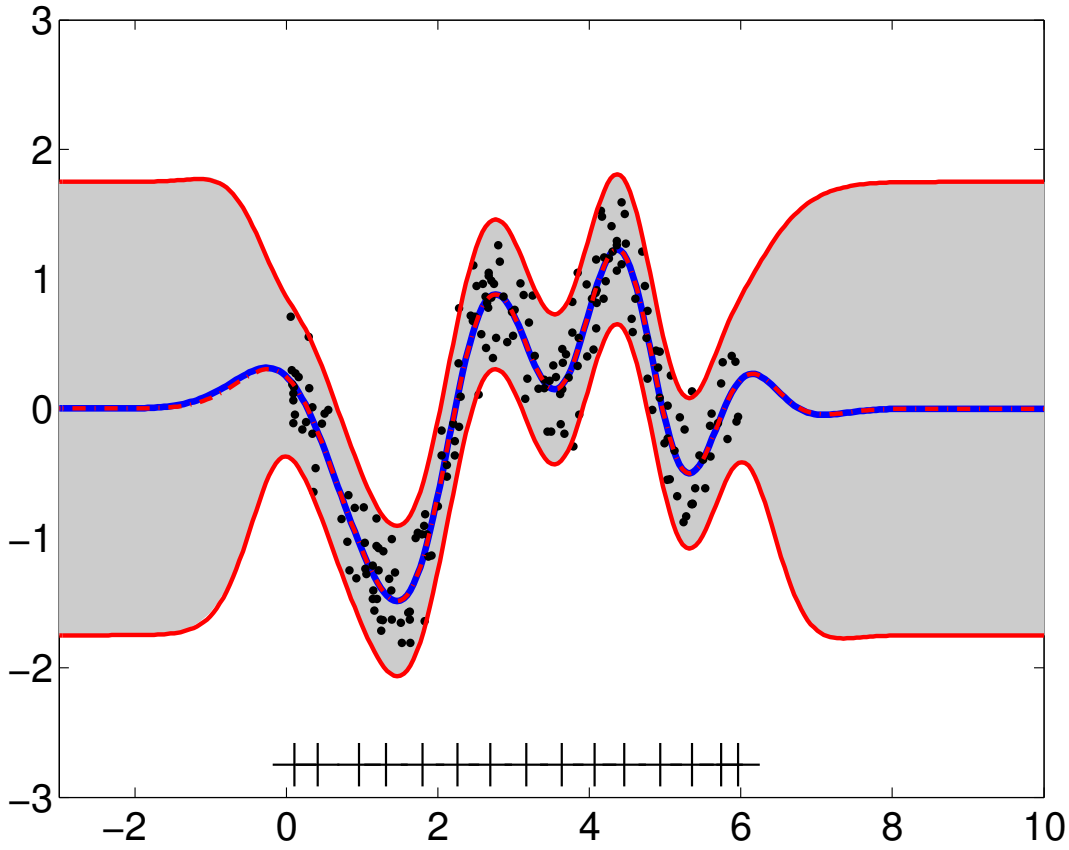
Variational learning of inducing variables



Full GP that scales as $O(n^3) = O(200^3)$
Variational approximation that scales as $O(nm^2) = O(200 \times 15^2)$ **at initialization**

- The crosses $(+)$ are the initial values of the inducing inputs Z

Variational learning of inducing variables



Full GP that scales as $O(n^3) = O(200^3)$

Variational approximation that scales as $O(nm^2) = O(200 \times 15^2)$ **after having maximized the bound**

Distributed representation of GPs

However for very complex functions we will need a large number of inducing variables to get good approximations

Distributed representation of GPs

However for very complex functions we will need a large number of inducing variables to get good approximations

One can say that the problem is that inducing variables try to approximate the GP globally

Distributed representation of GPs

However for very complex functions we will need a large number of inducing variables to get good approximations

One can say that the problem is that inducing variables try to approximate the GP globally

But the actual problem could be that we represent the GP function as a global model

Distributed representation of GPs

However for very complex functions we will need a large number of inducing variables to get good approximations

One can say that the problem is that inducing variables try to approximate the GP globally

But the actual problem could be that we represent the GP function as a global model

If we could construct distributed representations of a given GP

- then our approximations (based on inducing variables or by other means) could computationally become more distributed

Distributed representation of GPs

So the precise problem we wish to address here (and the framework to approach it) is the following:

Assume a GP with pre-specified/known kernel function $k(\mathbf{x}, \mathbf{x}')$

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Distributed representation of GPs

So the precise problem we wish to address here (and the framework to approach it) is the following:

Assume a GP with pre-specified/known kernel function $k(\mathbf{x}, \mathbf{x}')$

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Question: How can we find a distributed representation of this GP that will greatly facilitate our approximations?

Distributed representation of GPs

So the precise problem we wish to address here (and the framework to approach it) is the following:

Assume a GP with pre-specified/known kernel function $k(\mathbf{x}, \mathbf{x}')$

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Question: How can we find a distributed representation of this GP that will greatly facilitate our approximations?

Key idea: Use the divisibility property of Gaussian random quantities (sum of GPs are GPs!)

Distributed representation of GPs

So the precise problem we wish to address here (and the framework to approach it) is the following:

Assume a GP with pre-specified/known kernel function $k(\mathbf{x}, \mathbf{x}')$

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Question: How can we find a distributed representation of this GP that will greatly facilitate our approximations?

Key idea: Use the divisibility property of Gaussian random quantities (sum of GPs are GPs!)

Construct a set of GP function $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_J(\mathbf{x})$ so that their sum has the same probability law as $f(\mathbf{x})$:

$$f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots + f_J(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Distributed representation of GPs

So the precise problem we wish to address here (and the framework to approach it) is the following:

Assume a GP with pre-specified/known kernel function $k(\mathbf{x}, \mathbf{x}')$

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Question: How can we find a distributed representation of this GP that will greatly facilitate our approximations?

Key idea: Use the divisibility property of Gaussian random quantities (sum of GPs are GPs!)

Construct a set of GP function $f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_J(\mathbf{x})$ so that their sum has the same probability law as $f(\mathbf{x})$:

$$f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots + f_J(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

There are many ways to consctruct these f_js $\Rightarrow$ but to facilitate approximations we need these functions to be (in some sense) local!

Distributed representation of GPs

But let's assume we have some suitable functions f_js. How can we can distribute the approximations?

The initial GP model has joint $p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$

Distributed representation of GPs

But let's assume we have some suitable functions f_js. How can we can distribute the approximations?

The initial GP model has joint $p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$

The equivalent distibuted representation has joint

$$p(\mathbf{y}, \mathbf{f}_1, \ldots, \mathbf{f}_J) = p(\mathbf{y}|\sum_{j=1}^{J} \mathbf{f}_j) \times \prod_{j=1}^{J} p(\mathbf{f}_j)$$

Distributed representation of GPs

But let's assume we have some suitable functions f_js. How can we can distribute the approximations?

The initial GP model has joint $p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$

The equivalent distibuted representation has joint

$$p(\mathbf{y}, \mathbf{f}_1, \ldots, \mathbf{f}_J) = p(\mathbf{y}|\sum_{j=1}^{J} \mathbf{f}_j) \times \prod_{j=1}^{J} p(\mathbf{f}_j)$$

Introduce a separate set of m inducing variables $\mathbf{u}_j$ to approximate the correspoding f_j

Distributed representation of GPs

But let's assume we have some suitable functions f_js. How can we can distribute the approximations?

The initial GP model has joint $p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f})$

The equivalent distibuted representation has joint

$$p(\mathbf{y}, \mathbf{f}_1, \ldots, \mathbf{f}_J) = p(\mathbf{y}|\sum_{j=1}^{J} \mathbf{f}_j) \times \prod_{j=1}^{J} p(\mathbf{f}_j)$$

Introduce a separate set of m inducing variables $\mathbf{u}_j$ to approximate the correspoding f_j

Assume a factorized $\prod_{j=0}^{J} p(\mathbf{f}_j|\mathbf{u}_j)q(\mathbf{u}_j)$ and compute the bound

$$\int \prod_{j=1}^{J} p(\mathbf{f}_j|\mathbf{u}_j)q(\mathbf{u}_j) \log p(\mathbf{y}|\sum_{j=0}^{J} \mathbf{f}_j) - \sum_{j=1}^{J} \mathrm{KL}[q(\mathbf{u}_j)||p(\mathbf{u}_j)]$$

which has complexity $O(n * J * m^2)$ (and we are using $J * m$ inducing variables!)

Distributed representation of GPs

Distributed representation of a GP

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots + f_J(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Distributed representation of GPs

Distributed representation of a GP

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

$$f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots + f_J(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}'))$$

Essentially we need a way to divide the kernel function so that

$$k(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{J} k_j(\mathbf{x}, \mathbf{x}')$$

and each k_j is positive definite and hopefully local (variance drops to zero outside from a certain input region)

Distribute the kernel in 1-D

GP regression problem in 1-D with covariance function

$$k(x, x') = \sigma_f^2 e^{-\frac{1}{2\ell^2}(x-x')^2}$$

Distribute the kernel in 1-D

GP regression problem in 1-D with covariance function

$$k(x, x') = \sigma_f^2 e^{-\frac{1}{2\ell^2}(x-x')^2}$$

Explicitly write this as a feature dot product

$$k(x, x') = \int_{-\infty}^{\infty} \phi(x, z)\phi(x', z)dz$$

where $\phi(x, z) = \frac{\sigma_f}{\left(\frac{\pi\ell^2}{2}\right)^{\frac{1}{4}}} e^{-\frac{1}{\ell^2}(x-z)^2}$

Distribute the kernel in 1-D

GP regression problem in 1-D with covariance function

$$k(x, x') = \sigma_f^2 e^{-\frac{1}{2\ell^2}(x-x')^2}$$

Explicitly write this as a feature dot product

$$k(x, x') = \int_{-\infty}^{\infty} \phi(x, z)\phi(x', z)dz$$

where $\phi(x, z) = \frac{\sigma_f}{\left(\frac{\pi\ell^2}{2}\right)^{\frac{1}{4}}} e^{-\frac{1}{\ell^2}(x-z)^2}$

Divide the real axis $a_1 = -\infty < a_2 < a_3 < \ldots < a_J < a_{J+1} = \infty$ and write the kernel

$$k(x, x') = \sum_{j=1}^{J} \int_{a_j}^{a_{j+1}} \phi(x, z)\phi(x', z)dz = \sum_{j=1}^{J} k_j(x, x')$$

Distribute the kernel in 1-D

Divide the real axis $a_1 = -\infty < a_2 < a_3 < \ldots < a_J < a_{J+1} = \infty$ and write the kernel

$$k(x, x') = \sum_{j=1}^{J} \int_{a_j}^{a_{j+1}} \phi(x, z)\phi(x', z)dz = \sum_{j=1}^{J} k_j(x, x')$$

(each $k_j(x, x')$ is positive definite because of the dot product)

Distribute the kernel in 1-D

Divide the real axis $a_1 = -\infty < a_2 < a_3 < \ldots < a_J < a_{J+1} = \infty$ and write the kernel

$$k(x, x') = \sum_{j=1}^{J} \int_{a_j}^{a_{j+1}} \phi(x, z)\phi(x', z)dz = \sum_{j=1}^{J} k_j(x, x')$$

(each $k_j(x, x')$ is positive definite because of the dot product)

Each $k_j(x, x')$ is

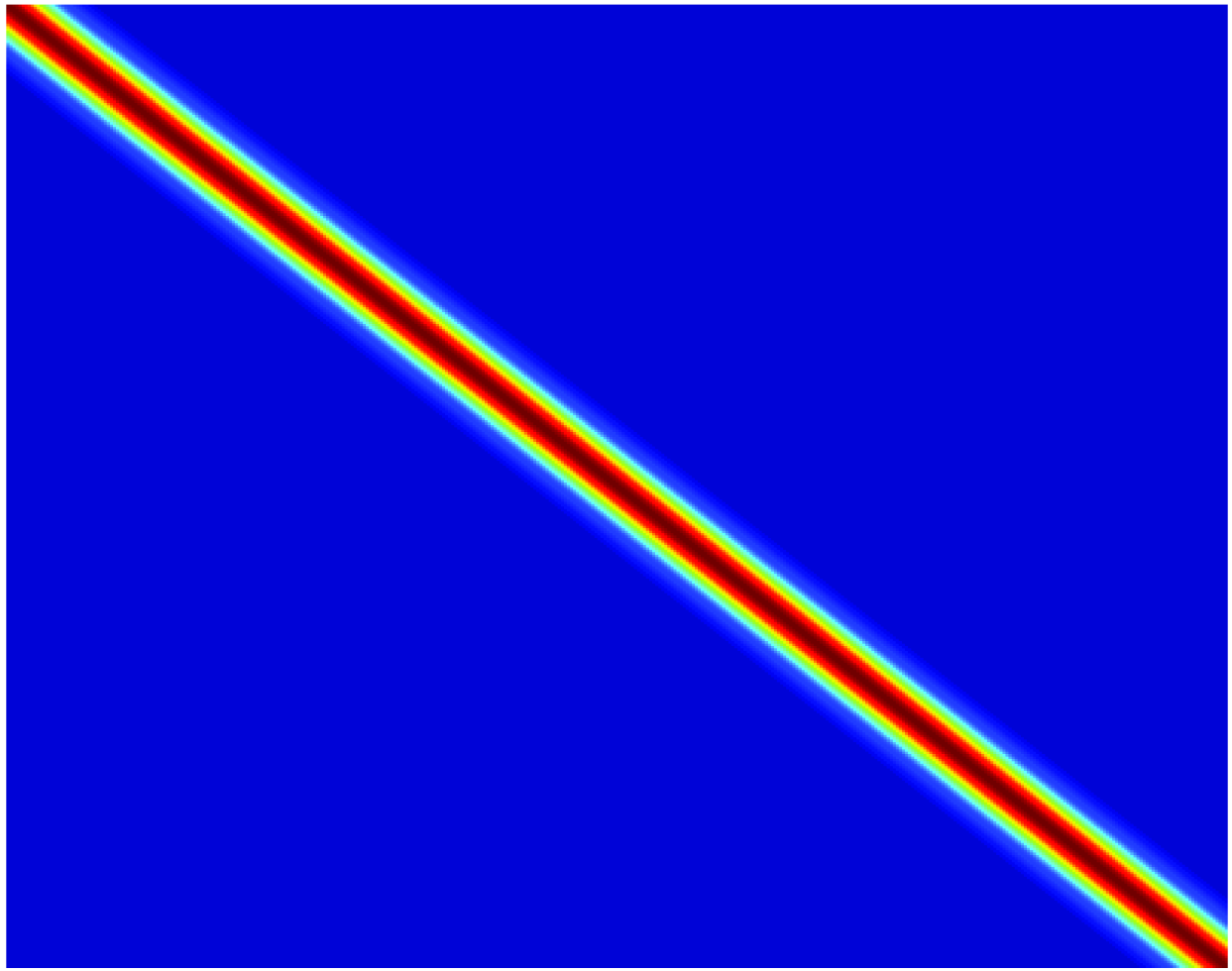
$$k_j(x, x') = k(x, x') \times \frac{\text{erfc}(s_j) - \text{erfc}(s_{j+1})}{2}$$

where the complementary error function is $\text{erfc}(x) = \frac{2}{\sqrt{\pi}} \int_x^{\infty} e^{-s^2} ds$ and

$$s_j = \frac{a_j - \bar{x}}{\ell/\sqrt{2}}, \; s_{j+1} = \frac{a_{j+1} - \bar{x}}{\ell/\sqrt{2}}, \quad \bar{x} = \frac{x + x'}{2}$$

Distribute the kernel in 1-D

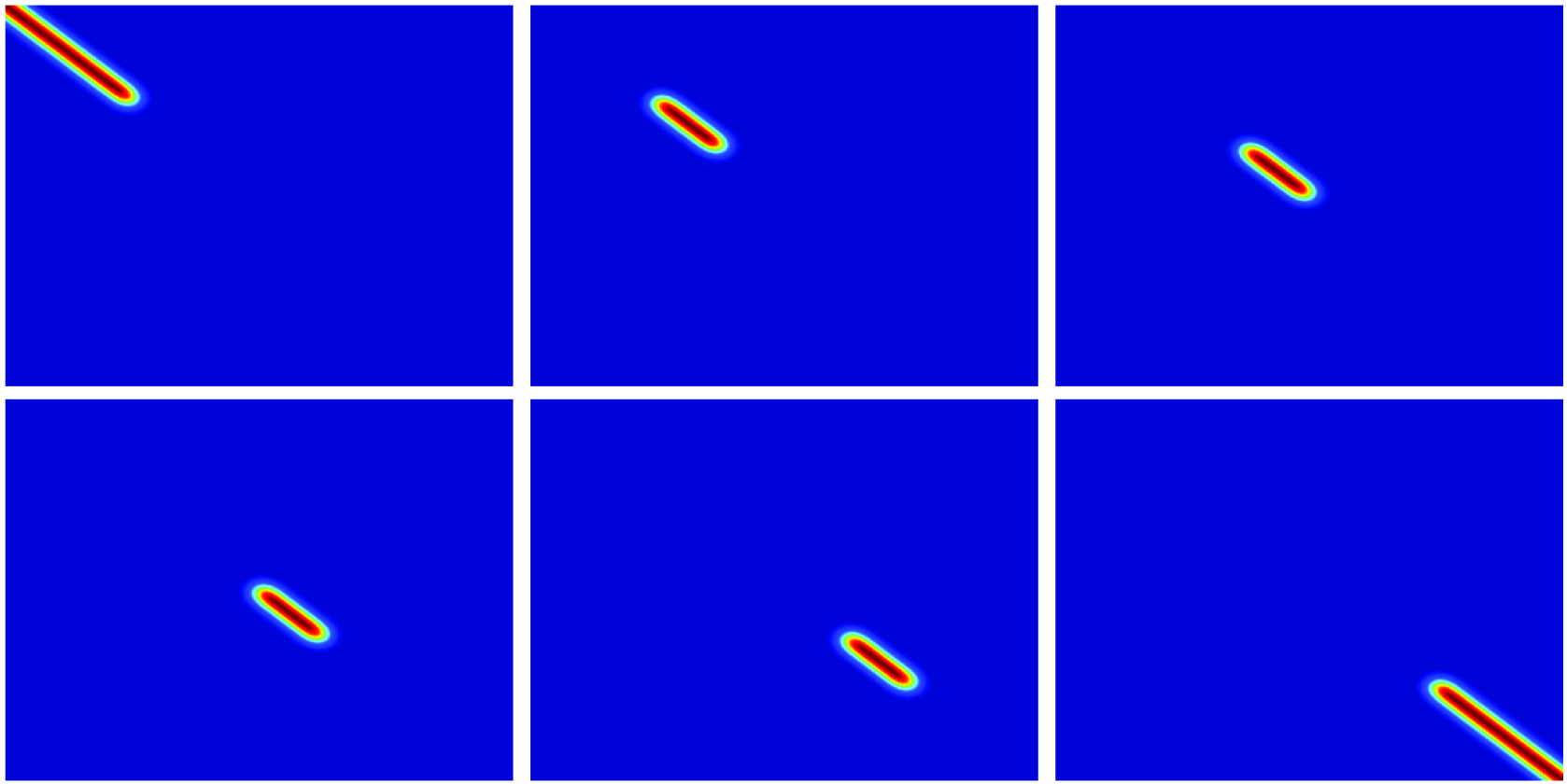
Example



Assume $k(x, x') = e^{-\frac{1}{2}(x-x')^2}$. Figure shows the corresponding kernel matrix in the range $[-20, 20]$

Distribute the kernel in 1-D

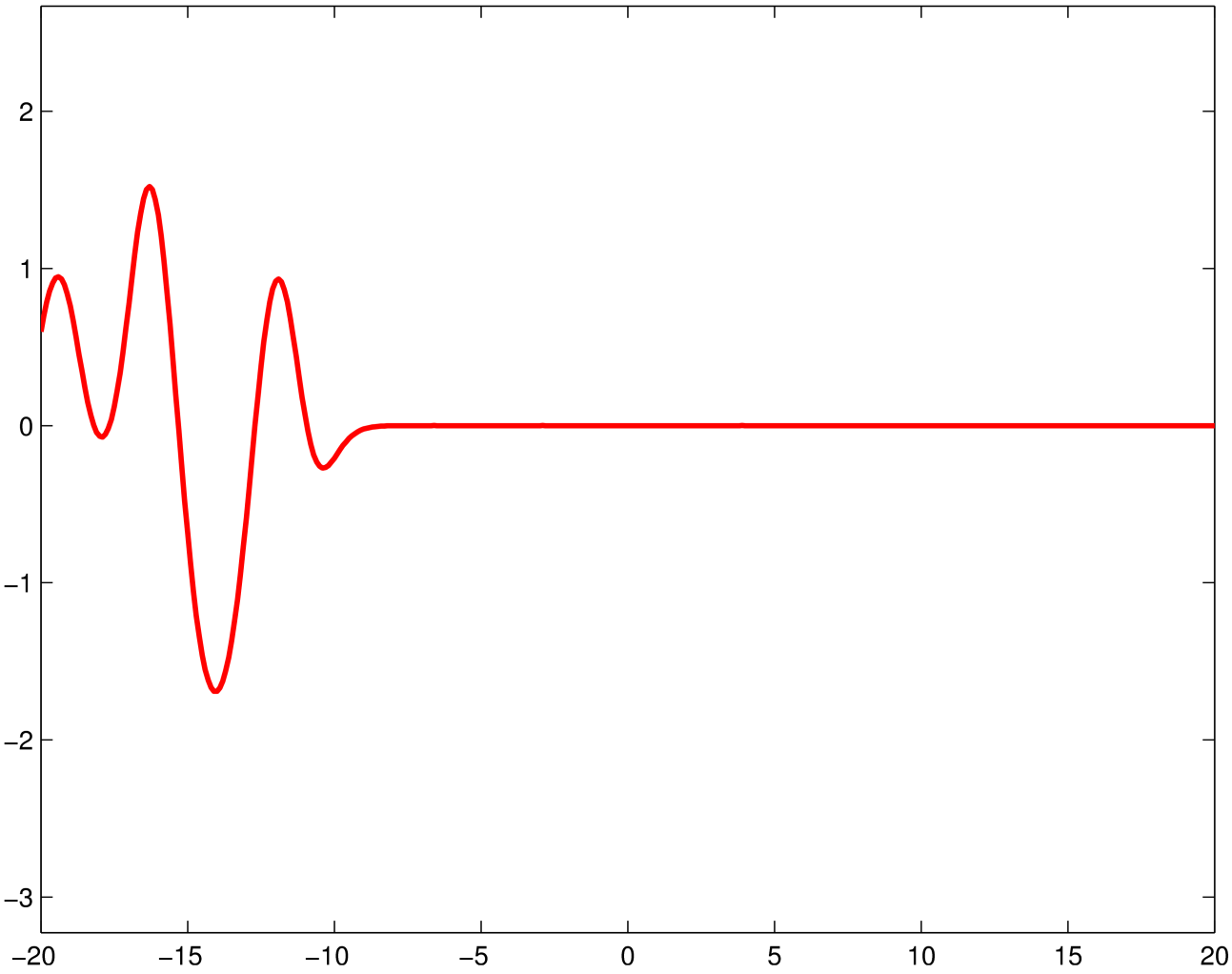
Example



Do the kernel splitting based on $-\infty < -10 < -5 < 0 < 5 < 10 < \infty$.
Figure shows all 6 local kernel matrices

Distribute the kernel in 1-D

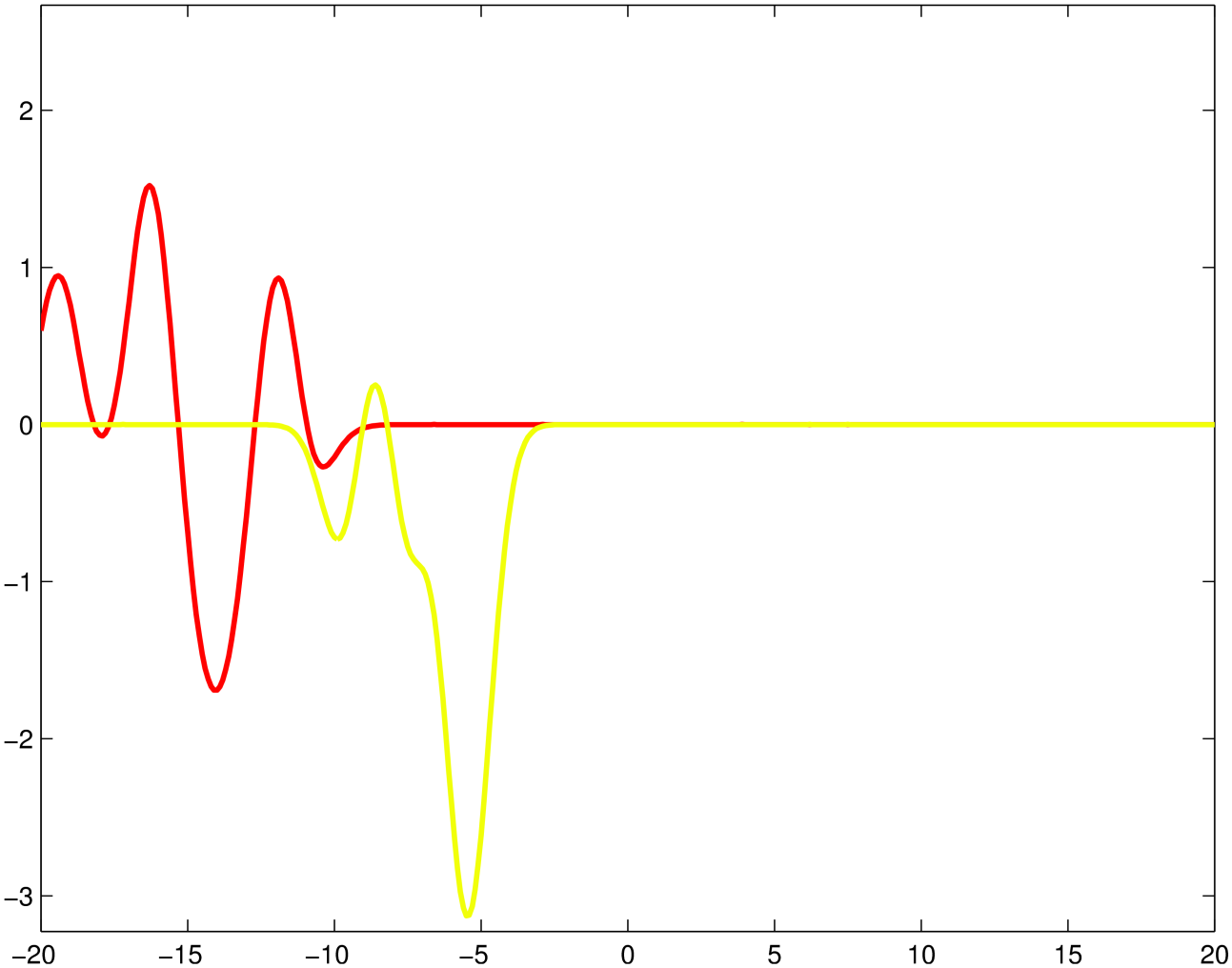
Example



Sample from the local GP associated with the segment $(-\infty, -10]$

Distribute the kernel in 1-D

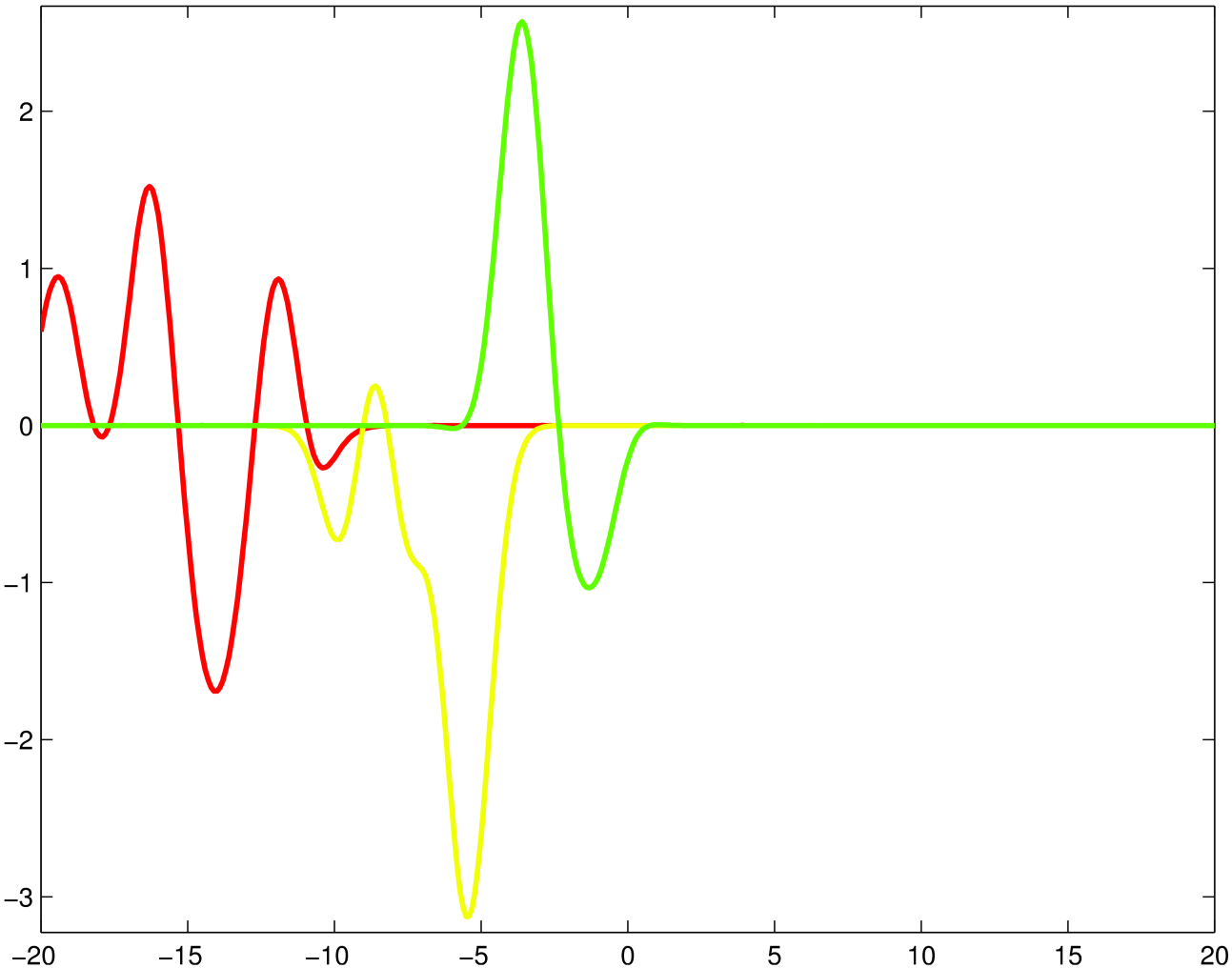
Example



A sample from the next local GP

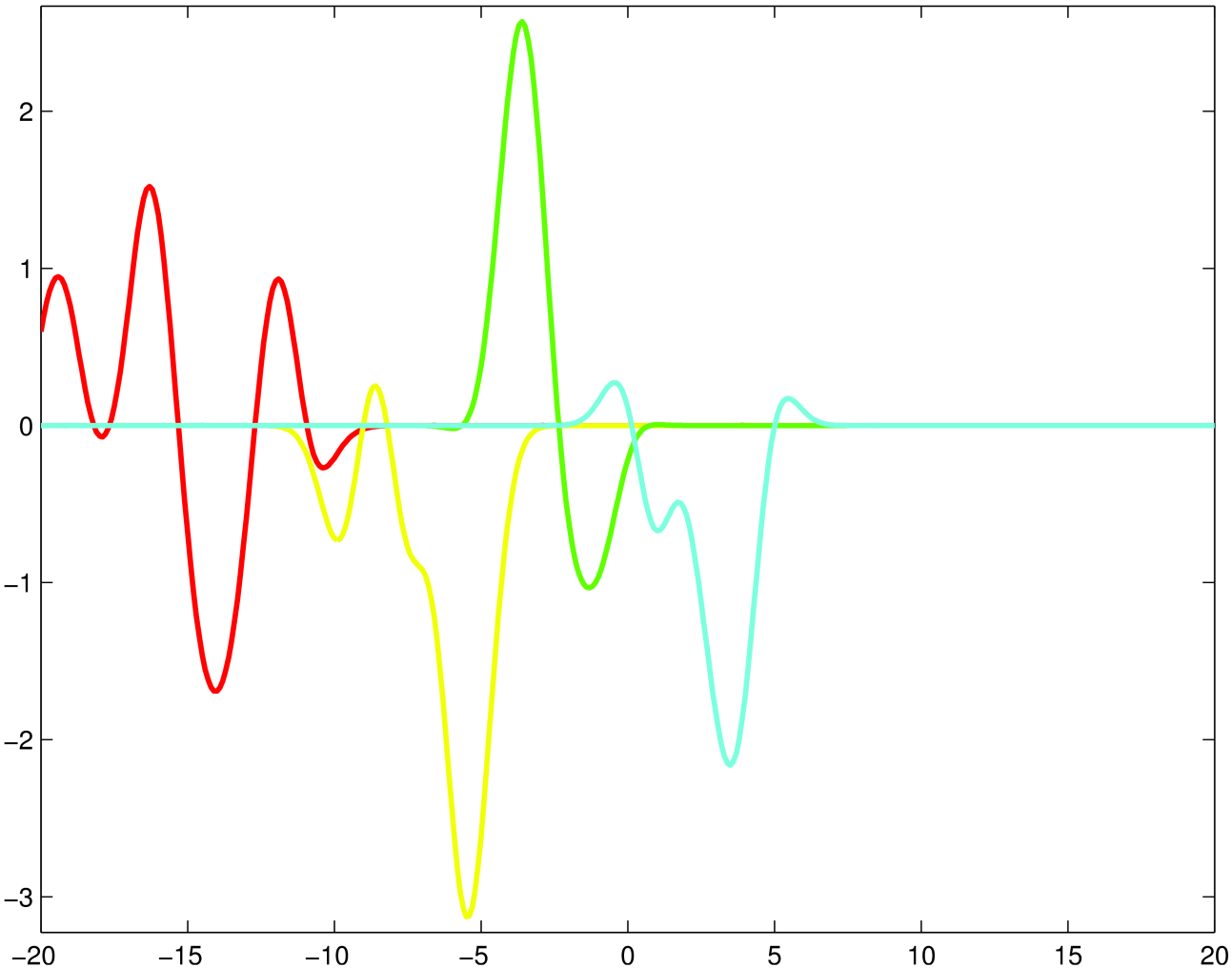
Distribute the kernel in 1-D

Example



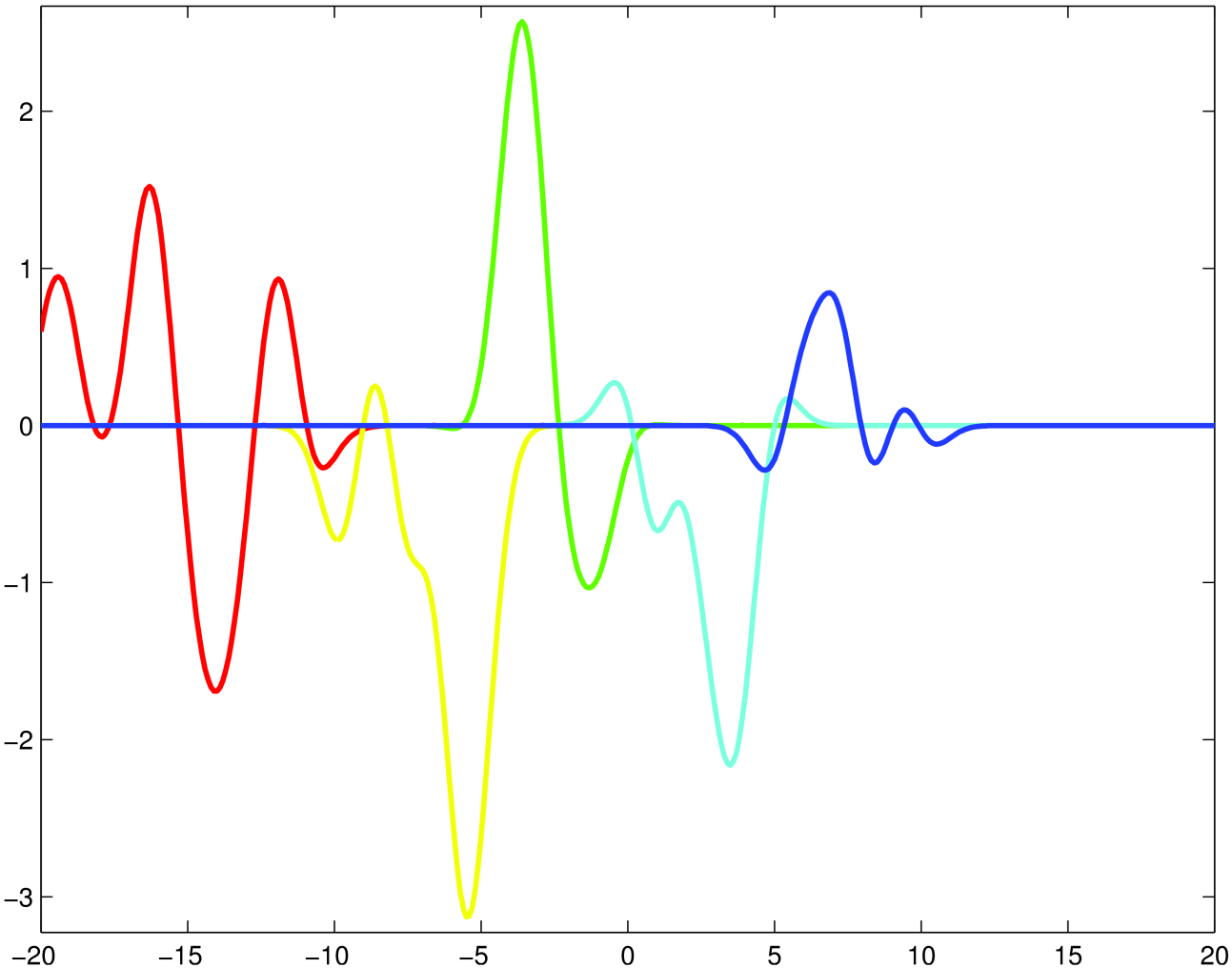
Distribute the kernel in 1-D

Example



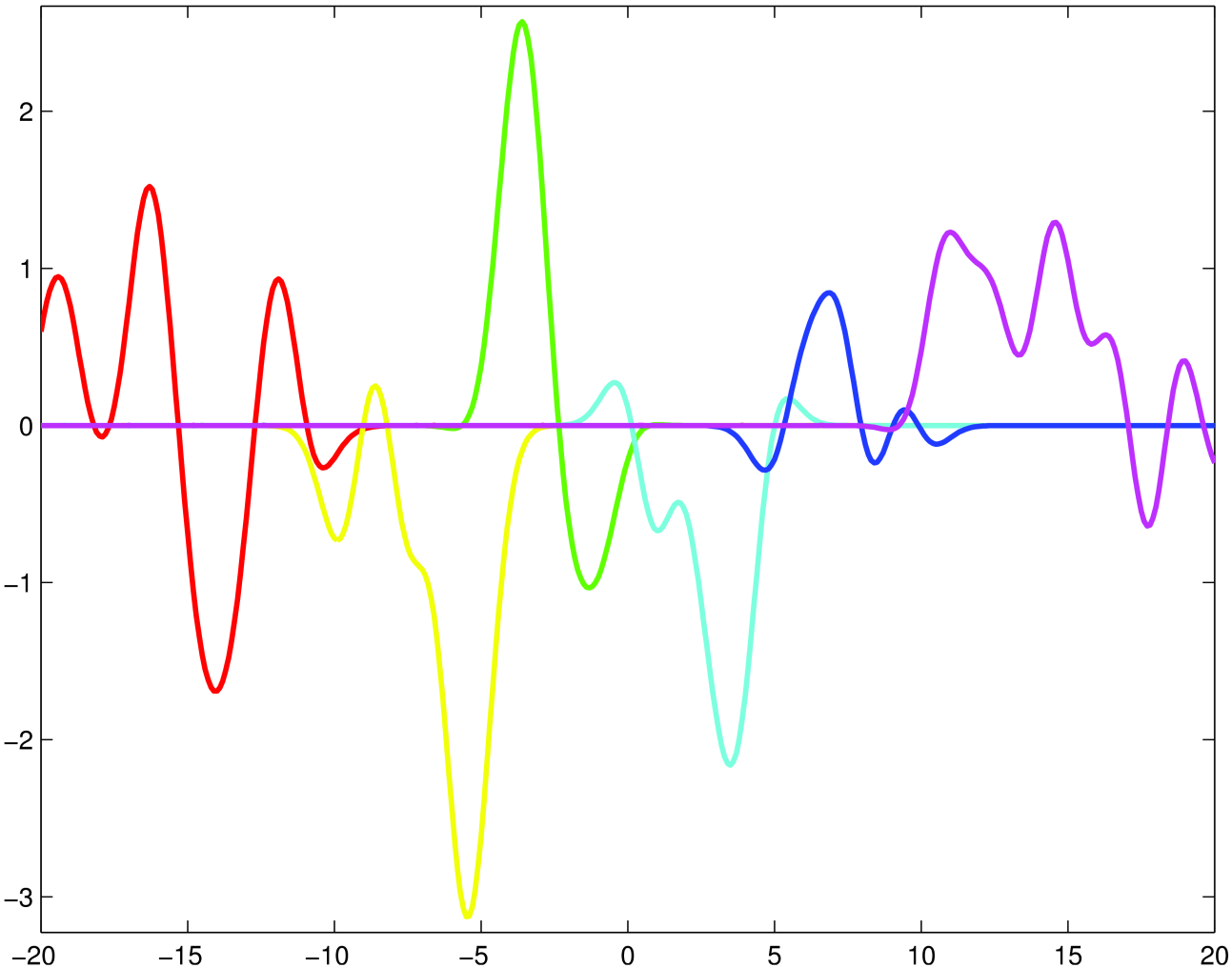
Distribute the kernel in 1-D

Example



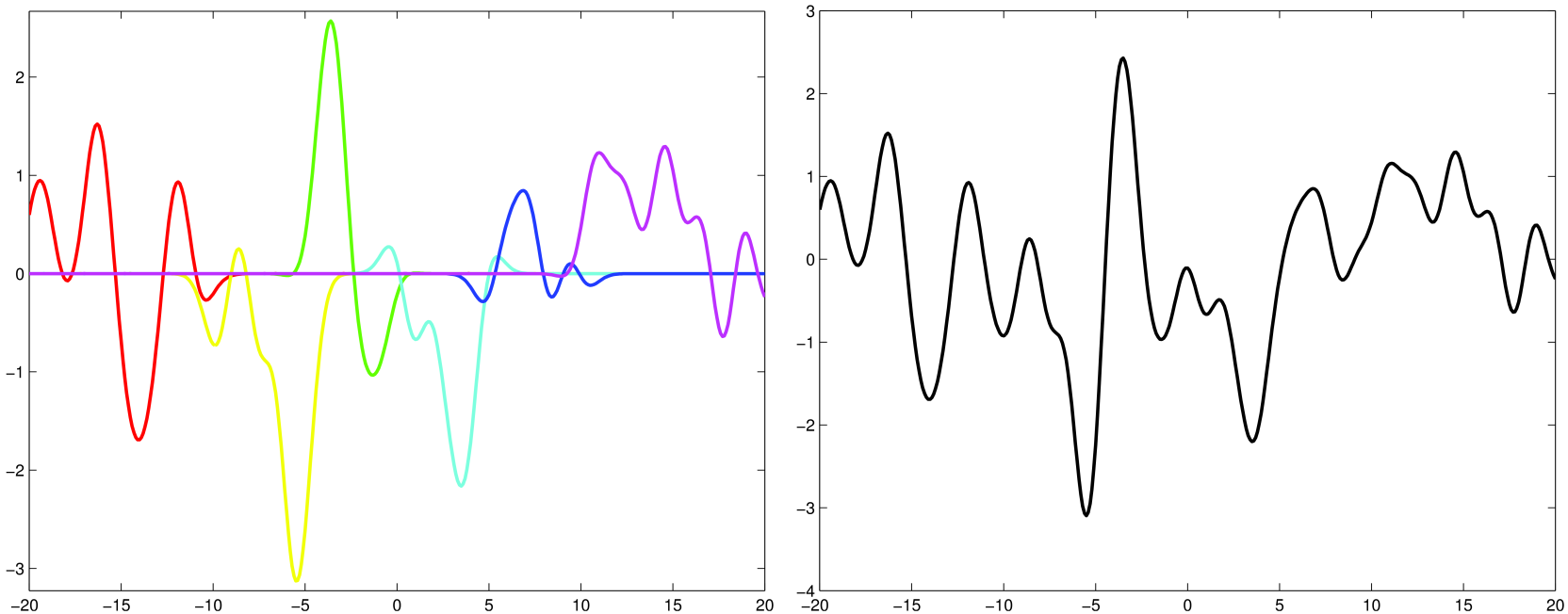
Distribute the kernel in 1-D

Example



Distribute the kernel in 1-D

Example



The sum of these local GP functions is a GP function (black line) with kernel $k(x, x') = e^{-\frac{1}{2}(x-x')^2}$

Distribute the kernel in high dimensions

GP regression with inputs $\mathbf{x} \in \mathbb{R}^D$ and covariance function

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \prod_{d=1}^{D} k_d(x_d, x_d')$$

and let's assume $k(x_d, x_d') = e^{-\frac{1}{2\ell_d^2}(x_d - x_d')^2}$

Distribute the kernel in high dimensions

GP regression with inputs $\mathbf{x} \in \mathbb{R}^D$ and covariance function

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \prod_{d=1}^{D} k_d(x_d, x_d')$$

and let's assume $k(x_d, x_d') = e^{-\frac{1}{2\ell_d^2}(x_d - x_d')^2}$

Pick a certain input dimension d (here you will need a criterion e.g. like information gain used in decision trees) to do the split:

$$\begin{aligned} k(\mathbf{x}, \mathbf{x}') &= \sigma_f^2 \left(\prod_{d' \neq d} k_d(x_d, x_d') \right) \times k_d(x_d, x_d') \\ &= \sigma_f^2 \left(\prod_{d' \neq d} k_d(x_d, x_d') \right) \times \sum_{j=1}^{J} k_{dj}(x_d, x_d') \\ &= \sum_{j=1}^{J} \sigma_f^2 \left(\prod_{d' \neq d} k_d(x_d, x_d') \right) k_{dj}(x_d, x_d') = \sum_{j=1}^{J} k_j(\mathbf{x}, \mathbf{x}') \end{aligned}$$

Discussion

The divisibility property $f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots$ of GPs could allow to distribute approximations

Discussion

The divisibility property $f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots$ of GPs could allow to distribute approximations

This requires the use of a factorized variational approximations over the f_js and can fit nicely with the current variational sparse GP framework

- $\Rightarrow$ there will be a separate localized set of inducing variables to approximater each f_j

Discussion

The divisibility property $f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots$ of GPs could allow to distribute approximations

This requires the use of a factorized variational approximations over the f_js and can fit nicely with the current variational sparse GP framework

- $\Rightarrow$ there will be a separate localized set of inducing variables to approximater each f_j

These f_js need to be somehow local. We have presented one way to achieve locality but perhaps there exist other possibilities

Discussion

The divisibility property $f_1(\mathbf{x}) + f_2(\mathbf{x}) + \ldots$ of GPs could allow to distribute approximations

This requires the use of a factorized variational approximations over the f_js and can fit nicely with the current variational sparse GP framework

- $\Rightarrow$ there will be a separate localized set of inducing variables to approximater each f_j

These f_js need to be somehow local. We have presented one way to achieve locality but perhaps there exist other possibilities

Current issues: How exactly the bound will be maximized in a computionally distributed manner