Tensor Train Decomposition for Fast Learning in Large Scale Gaussian Process Models

Dmitry Kropotov

Joint work with Pavel Izmailov and Alexander Novikov

Bayesgroup seminar

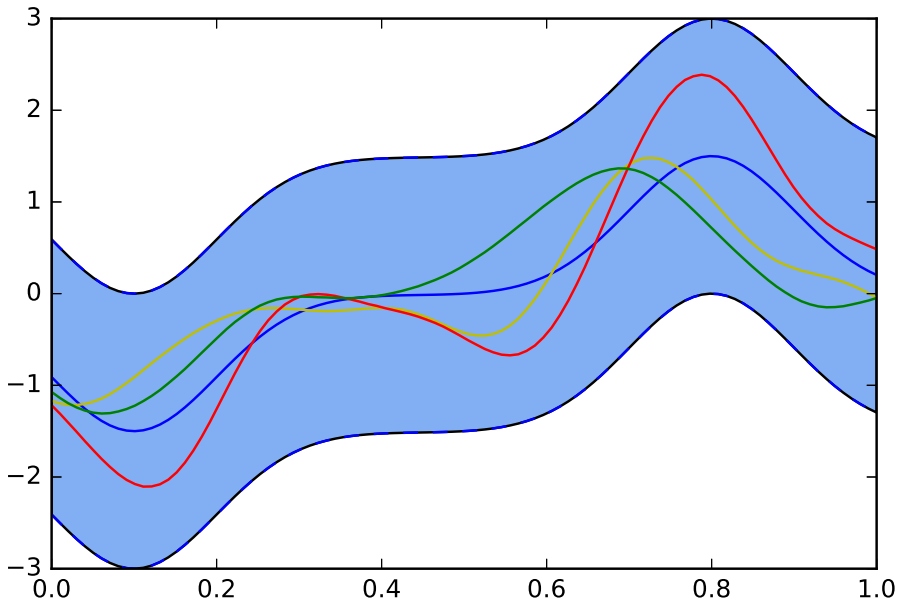
Gaussian Process

Gaussian Process (GP) gives a distribution over real-valued functions.

$$f(\boldsymbol{x}) \sim \mathrm{GP}(m(\cdot), K(\cdot,\cdot)),$$

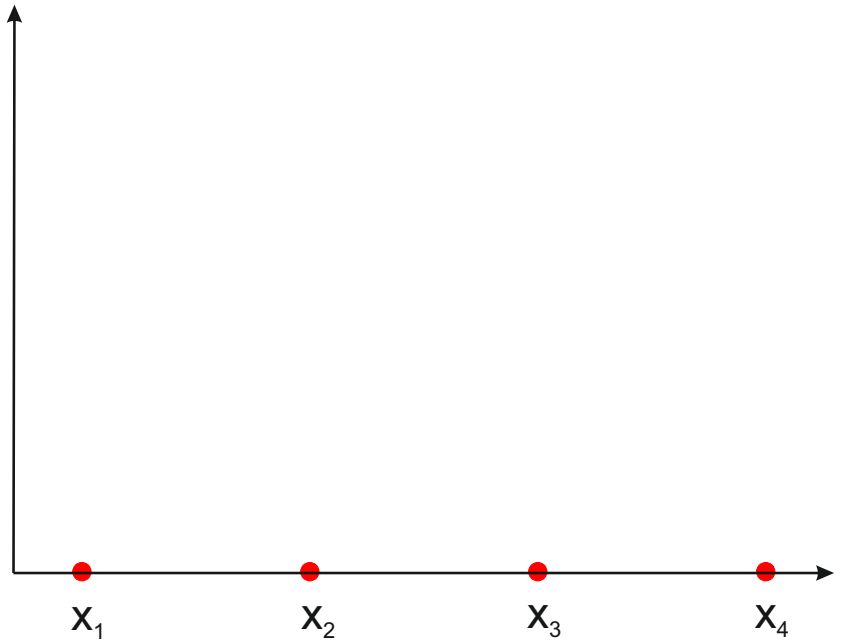
$m(\boldsymbol{x})$ – mean function,

$K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ – covariance function.

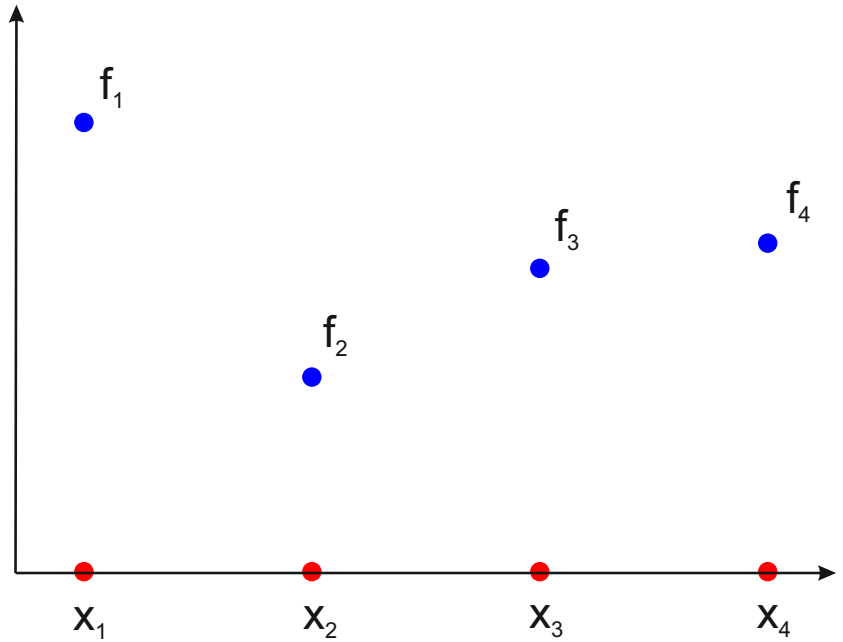


Gaussian Process

$$
\begin{aligned}
&f(\boldsymbol{x}) \sim \mathrm{GP}(m(\cdot), K(\cdot,\cdot)) \;\Leftrightarrow \\
&\boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_n)] \sim \mathcal{N}(\boldsymbol{f}|\boldsymbol{m}, K), \\
&\boldsymbol{m} = [m(\boldsymbol{x}_1), m(\boldsymbol{x}_2), \ldots, m(\boldsymbol{x}_n)], \\
&K \in \mathbb{R}^{n\times n}, \; K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j) \; \forall i,j = 1,\ldots,n.
\end{aligned}
$$

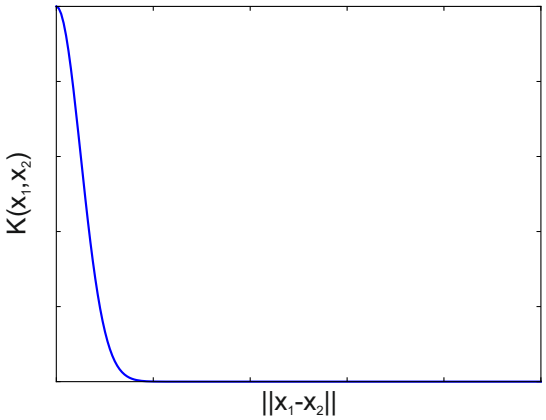


$$f(\boldsymbol{x}) \sim \mathrm{GP}(m(\cdot), K(\cdot,\cdot)) \;\Leftrightarrow$$
$$\boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_n)] \sim \mathcal{N}(\boldsymbol{f}|\boldsymbol{m}, K),$$
$$\boldsymbol{m} = [m(\boldsymbol{x}_1), m(\boldsymbol{x}_2), \ldots, m(\boldsymbol{x}_n)],$$
$$K \in \mathbb{R}^{n\times n},\; K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)\;\forall i,j = 1,\ldots,n.$$

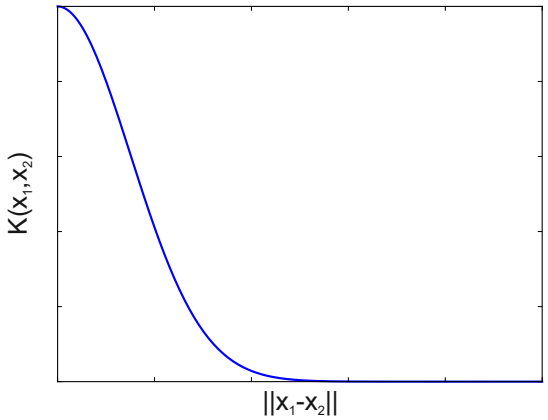


Covariance function

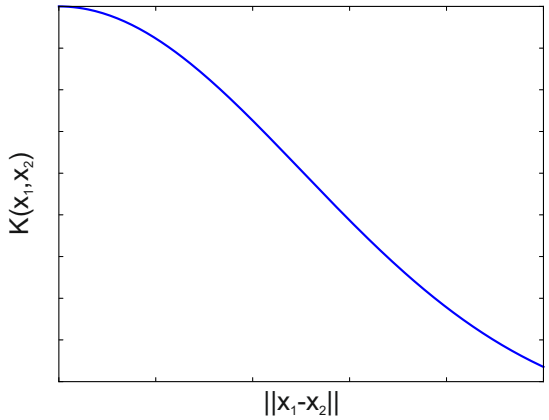
Squared exponential covariance function: $K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{2\theta^2}\right)$



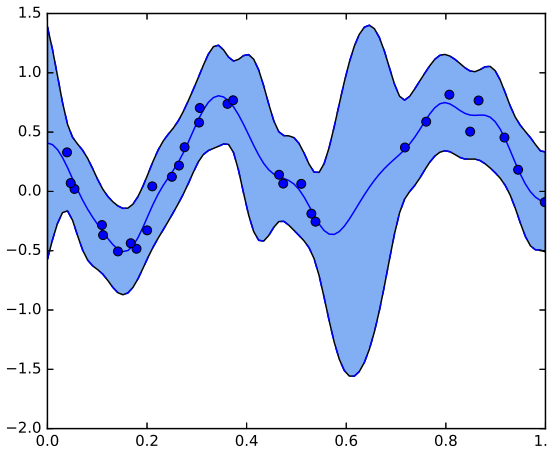
$\theta = 0.05$

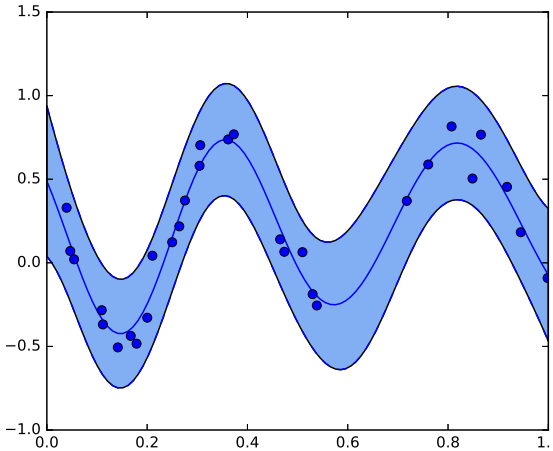


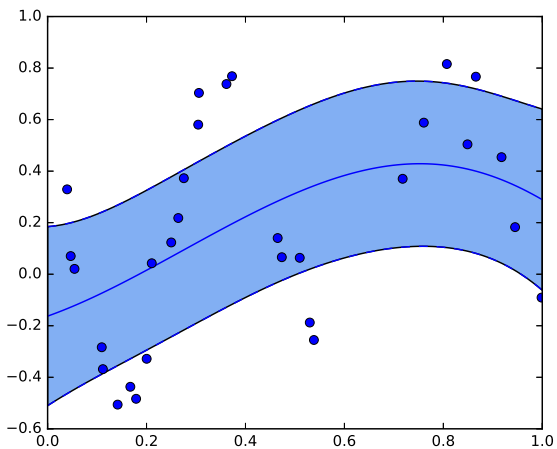
$\theta = 0.15$



$\theta = 0.5$







Gaussian Process models for regression and classification

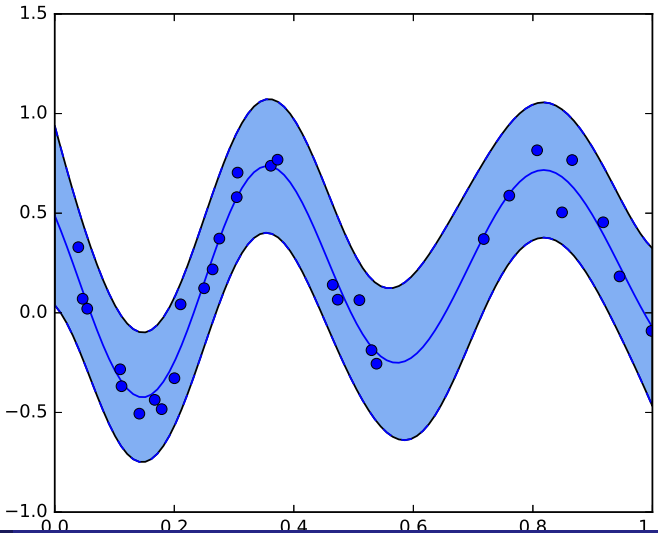
Suppose we have a dataset $(\boldsymbol{y}, X) = \{y_i, \boldsymbol{x}_i\}_{i=1}^n$, $\boldsymbol{x}_i \in \mathbb{R}^d$ – feature vectors, y_i – target values.

$$p(\boldsymbol{y}, \boldsymbol{f}|X, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|X, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i|f_i)p(\boldsymbol{f}|X, \boldsymbol{\theta}),$$

$$p(\boldsymbol{f}|X, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, K(X, X; \boldsymbol{\theta})) \text{ – GP},$$

$$p(y_i|f_i) = \mathcal{N}(y_i|f_i, \sigma^2) \text{ – regression},$$

$$p(y_i|f_i) = \frac{1}{1 + \exp(-y_i f_i)} \text{ – classification with 2 classes.}$$



Gaussian Process model for multiclass classification

Dataset $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$, $\boldsymbol{x}_i \in \mathbb{R}^d$ – feature vectors, $y_i \in \{1, 2, \ldots, C\}$ – target values.

Let's introduce a separate GP $\boldsymbol{f}^c$ for each class $c = 1, \ldots, C$ with shared covariance function $K(\cdot, \cdot; \boldsymbol{\theta}^c)$, but different parameters $\boldsymbol{\theta}^c$.

$$p(\boldsymbol{y}, \boldsymbol{f}^1, \ldots, \boldsymbol{f}^C | X, \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^C) = \prod_{i=1}^n p(y_i | f_i^1, \ldots, f_i^C) \prod_{c=1}^C p(\boldsymbol{f}^c | X, \boldsymbol{\theta}^c),$$

$$p(\boldsymbol{f}^c | X, \boldsymbol{\theta}^c) = \mathcal{N}(\boldsymbol{f}^c | \boldsymbol{0}, K(X, X; \boldsymbol{\theta}^c)) \text{ – GP},$$

$$p(y_i | f_i^1, \ldots, f_i^C) = \frac{\exp(f^{y_i})}{\sum_{c=1}^C \exp(y_i^c)} \text{ – softmax likelihood.}$$

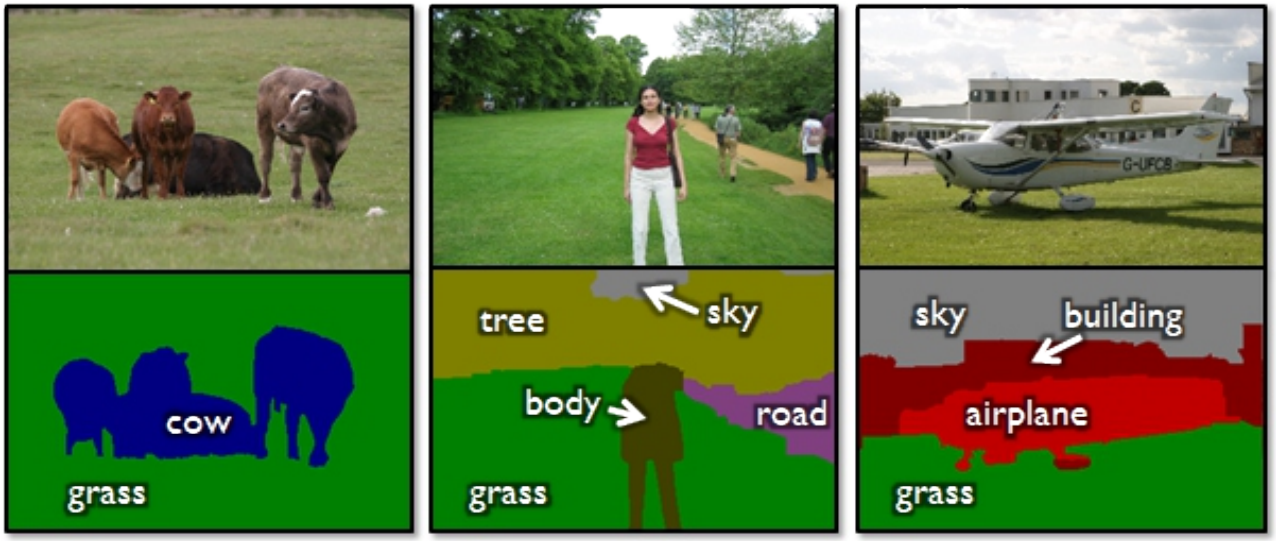
Prediction: $y(\boldsymbol{x}_*) = \arg \max_{c \in \{1, \ldots, C\}} (f^c(\boldsymbol{x}_*))$.

Structured prediction problem

Dataset $\{X_i, y_i\}_{i=1}^n$, X_i – feature vectors, $y_i \in Y$ – target values.

Object X:

Segmentation y:



MRF prediction:

$$y(X^*) = \arg\max_{y \in Y} \left[\sum_p \psi_{un}(y_p, X_p^*) + \sum_{(p,q) \in \mathcal{E}} \psi_{bin}(y_p, y_q, X_p^*, X_q^*) \right]$$

Gaussian Process model for structured prediction

Dataset $\{y_i, X_i\}_{i=1}^n$, X_i – feature vectors, $y_i \in Y$ – target values.

$\boldsymbol{f}_{un}^c$ – separate GP for unary potential for each class $c = 1, \ldots, C$;

$\boldsymbol{f}_{bin}$ – one common GP for pairwise potentials with covariance function $K(y_p^1, y_q^1; y_p^2, y_q^2) = I[y_p^1 = y_p^2 \& y_q^1 = y_q^2]$.

Probabilistic model:

$$p(\boldsymbol{y}, \boldsymbol{f}_{un}^1, \ldots, \boldsymbol{f}_{un}^C, \boldsymbol{f}_{bin} | X, \boldsymbol{\theta}^1, \ldots, \boldsymbol{\theta}^C) =$$

$$= \prod_{i=1}^n p(y_i | \boldsymbol{f}_{un}^1, \ldots, \boldsymbol{f}_{un}^C, \boldsymbol{f}_{bin}) \prod_{c=1}^C p(\boldsymbol{f}_{un}^c | X, \boldsymbol{\theta}^c) p(\boldsymbol{f}_{bin}).$$

CRF likelihood:

$$p(y_i | \boldsymbol{f}_{un}^1, \ldots, \boldsymbol{f}_{un}^C, \boldsymbol{f}_{bin}) = \frac{\exp(\sum_p f_{un}^{y_{p,i}}(X_{p,i}) + \sum_{(p,q)\in\mathcal{E}} f_{bin}(y_{p,i}, y_{q,i}))}{\sum_{\hat{y}\in Y} \exp(\sum_p f_{un}^{\hat{y}_p}(X_{p,i}) + \sum_{(p,q)\in\mathcal{E}} f_{bin}(\hat{y}_p, \hat{y}_q))}$$

Prediction: $y(X^*) = \arg\max_{y\in Y} \left[\sum_p f_{un}^{y_p}(X_p^*) + \sum_{(p,q)\in\mathcal{E}} f_{bin}(y_p, y_q)\right]$.

GP models training and testing

GP probabilistic model:

$$p(\boldsymbol{y}, \boldsymbol{f}|X, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|X, \boldsymbol{\theta}) = \prod_{i=1}^{n} p(y_i|f_i)p(\boldsymbol{f}|X, \boldsymbol{\theta}).$$

Training:

$$\log p(\boldsymbol{y}|X, \boldsymbol{\theta}) \geq \mathbb{E}_{q(\boldsymbol{f})}[\log p(\boldsymbol{y}, \boldsymbol{f}|X, \boldsymbol{\theta}) - \log q(\boldsymbol{f})] \rightarrow \max_{\boldsymbol{\theta}, q(\cdot)},$$

$$q(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}, \Sigma) \approx p(\boldsymbol{f}|\boldsymbol{y}, X, \boldsymbol{\theta}).$$

Testing:

$$p(\boldsymbol{f}_{test}|\boldsymbol{y}, \boldsymbol{\theta}, X, X_{test}) = \int p(\boldsymbol{f}_{test}|\boldsymbol{f}, \boldsymbol{\theta}, X, X_{test})p(\boldsymbol{f}|\boldsymbol{y}, \boldsymbol{\theta}, X)d\boldsymbol{f} \approx$$

$$\approx \int p(\boldsymbol{f}_{test}|\boldsymbol{f}, \boldsymbol{\theta}, X, X_{test})q(\boldsymbol{f})d\boldsymbol{f}.$$

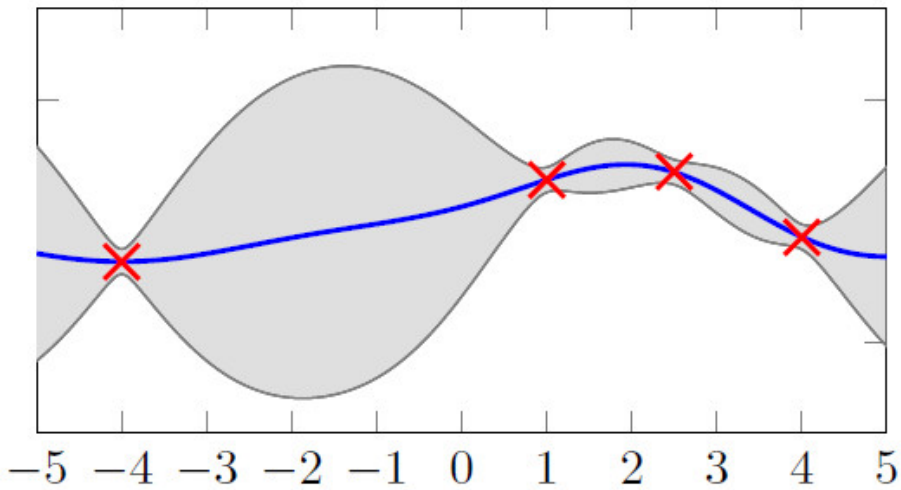
GP properties

Pros:

- Automatically adjust all parameters during training;
- Provides variance during prediction;

Cons:

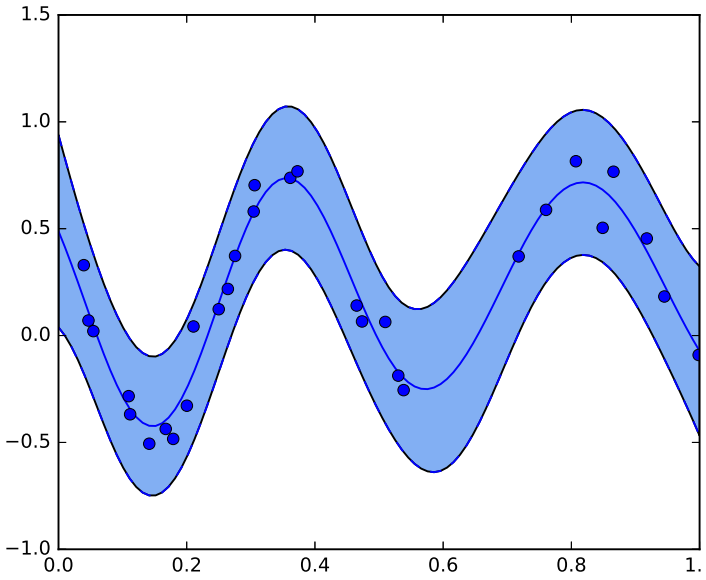
- Training scales as $O(n^3)$, where n – training sample size;
- Do not allow very complex dependencies like deep neural nets.

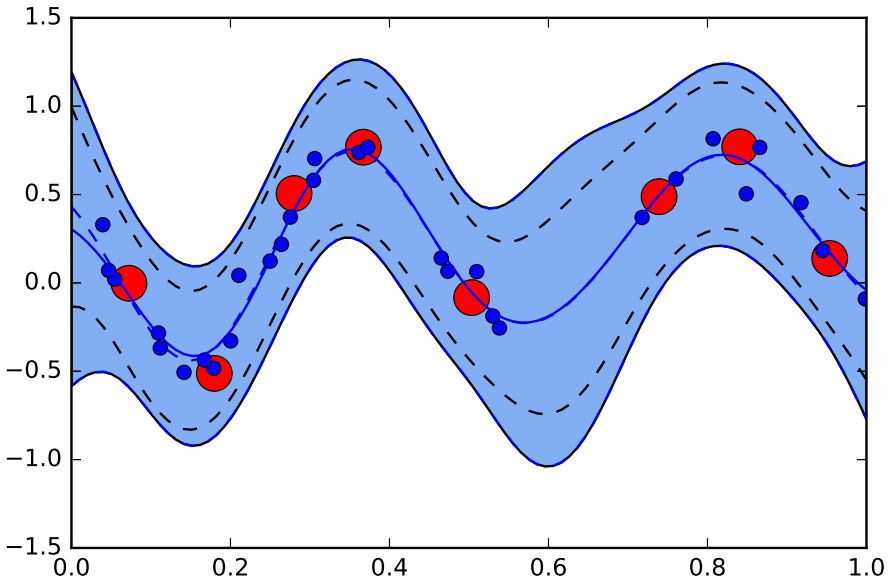


Gaussian Process with inducing inputs

Let's introduce some new points $Z = \{\boldsymbol{z}_i\}_{i=1}^m$ and suppose we know GP values at these points $\boldsymbol{u}$. Key assumption:

$$p(\boldsymbol{f}|\boldsymbol{y}, \boldsymbol{u}, X, Z) \approx p(\boldsymbol{f}|\boldsymbol{u}, X, Z).$$





Gaussian Process with inducing inputs

Augmented model:

$$p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}),$$
$$p(\boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}) = \mathcal{N}([\boldsymbol{f}, \boldsymbol{u}]|[\mathbf{0}_n, \mathbf{0}_m], K_{n+m,n+m}).$$

Augmented model coincides with previous one:

$$\int p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta})d\boldsymbol{u} = p(\boldsymbol{y}, \boldsymbol{f}|X, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{f})\mathcal{N}(\boldsymbol{f}|\mathbf{0}_n, K_{nn}).$$

Gaussian Process with inducing inputs

Augmented model:

$$p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}),$$
$$p(\boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}) = \mathcal{N}([\boldsymbol{f}, \boldsymbol{u}]|[\mathbf{0}_n, \mathbf{0}_m], K_{n+m,n+m}).$$

Augmented model coincides with previous one:

$$\int p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta})d\boldsymbol{u} = p(\boldsymbol{y}, \boldsymbol{f}|X, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{f})\mathcal{N}(\boldsymbol{f}|\mathbf{0}_n, K_{nn}).$$

Training:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u})}[\log p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}) - \log q(\boldsymbol{f}, \boldsymbol{u})] \to \max_{\boldsymbol{\theta}, q(\boldsymbol{f},\boldsymbol{u})},$$

$$q(\boldsymbol{f}, \boldsymbol{u}) \approx p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y}, X, Z, \boldsymbol{\theta}) = p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{y}, X, Z, \boldsymbol{\theta})p(\boldsymbol{u}|\boldsymbol{y}, X, Z, \boldsymbol{\theta}).$$

Gaussian Process with inducing inputs

Augmented model:

$$p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}),$$
$$p(\boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}) = \mathcal{N}([\boldsymbol{f}, \boldsymbol{u}]|[\mathbf{0}_n, \mathbf{0}_m], K_{n+m,n+m}).$$

Augmented model coincides with previous one:

$$\int p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta})d\boldsymbol{u} = p(\boldsymbol{y}, \boldsymbol{f}|X, \boldsymbol{\theta}) = p(\boldsymbol{y}|\boldsymbol{f})\mathcal{N}(\boldsymbol{f}|\mathbf{0}_n, K_{nn}).$$

Training:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u})}[\log p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta}) - \log q(\boldsymbol{f}, \boldsymbol{u})] \to \max_{\boldsymbol{\theta}, q(\boldsymbol{f},\boldsymbol{u})},$$

$$q(\boldsymbol{f}, \boldsymbol{u}) \approx p(\boldsymbol{f}, \boldsymbol{u}|\boldsymbol{y}, X, Z, \boldsymbol{\theta}) = p(\boldsymbol{f}|\boldsymbol{u}, \boldsymbol{y}, X, Z, \boldsymbol{\theta})p(\boldsymbol{u}|\boldsymbol{y}, X, Z, \boldsymbol{\theta}).$$

Using key assumption we can choose $q(\boldsymbol{f}, \boldsymbol{u})$ as follows:

$$q(\boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u}, X, Z, \boldsymbol{\theta})q(\boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u}, X, Z, \boldsymbol{\theta})\mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \Sigma).$$

Gaussian Process with inducing inputs

Family for approximate posterior:

$$q(\boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u}, X, Z, \boldsymbol{\theta})q(\boldsymbol{u}).$$

Training:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u})} \log \frac{p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta})}{q(\boldsymbol{f}, \boldsymbol{u})} =$$

Gaussian Process with inducing inputs

Family for approximate posterior:

$$q(\boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{f}|\boldsymbol{u}, X, Z, \boldsymbol{\theta})q(\boldsymbol{u}).$$

Training:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u})} \log \frac{p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}|X, Z, \boldsymbol{\theta})}{q(\boldsymbol{f}, \boldsymbol{u})} =$$

$$= \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u})} \log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u}, X, Z, \boldsymbol{\theta})p(\boldsymbol{u}|Z, \boldsymbol{\theta})}{p(\boldsymbol{f}|\boldsymbol{u}, X, Z, \boldsymbol{\theta})q(\boldsymbol{u})} =$$

$$= \mathbb{E}_{q(\boldsymbol{f},\boldsymbol{u})} \log p(\boldsymbol{y}|\boldsymbol{f}) + \mathbb{E}_{q(\boldsymbol{u})} \log \frac{p(\boldsymbol{u}|Z, \boldsymbol{\theta})}{q(\boldsymbol{u})} =$$

$$= \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i) - \mathrm{KL}(q(\boldsymbol{u})\|p(\boldsymbol{u}|Z, \boldsymbol{\theta})).$$

Gaussian Process with inducing inputs

Optimization criterion:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i) - \mathrm{KL}(q(\boldsymbol{u})\|p(\boldsymbol{u}|Z, \boldsymbol{\theta})) \to \max_{\boldsymbol{\theta}, \boldsymbol{\mu}, \Sigma}.$$

First term:

$$\begin{aligned} q(\boldsymbol{f}) &= \int q(\boldsymbol{f}, \boldsymbol{u}) d\boldsymbol{u} = \int p(\boldsymbol{f}|\boldsymbol{u}, X, Z) q(\boldsymbol{u}) d\boldsymbol{u} = \\ &= \int \mathcal{N}(\boldsymbol{f}|K_{nm}K_{mm}^{-1}\boldsymbol{u}, K_{nn} - K_{nm}K_{mm}^{-1}K_{mn}) \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \Sigma) d\boldsymbol{u} = \\ &= \mathcal{N}(\boldsymbol{f}|K_{nm}K_{mm}^{-1}\boldsymbol{\mu}, K_{nn} + K_{nm}K_{mm}^{-1}(\Sigma - K_{mm})K_{mm}^{-1}K_{mn}). \end{aligned}$$

Gaussian Process with inducing inputs

Optimization criterion:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i) - \mathrm{KL}(q(\boldsymbol{u})\|p(\boldsymbol{u}|Z, \boldsymbol{\theta})) \to \max_{\boldsymbol{\theta}, \boldsymbol{\mu}, \Sigma}.$$

First term:

$$q(\boldsymbol{f}) = \int q(\boldsymbol{f}, \boldsymbol{u}) d\boldsymbol{u} = \int p(\boldsymbol{f}|\boldsymbol{u}, X, Z) q(\boldsymbol{u}) d\boldsymbol{u} =$$

$$= \int \mathcal{N}(\boldsymbol{f}|K_{nm}K_{mm}^{-1}\boldsymbol{u}, K_{nn} - K_{nm}K_{mm}^{-1}K_{mn})\mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \Sigma) d\boldsymbol{u} =$$

$$= \mathcal{N}(\boldsymbol{f}|K_{nm}K_{mm}^{-1}\boldsymbol{\mu}, K_{nn} + K_{nm}K_{mm}^{-1}(\Sigma - K_{mm})K_{mm}^{-1}K_{mn}).$$

$$q(f_i) = \mathcal{N}(f_i|\boldsymbol{k}_i^T K_{mm}^{-1}\boldsymbol{\mu}, k_{ii} + \boldsymbol{k}_i^T K_{mm}^{-1}(\Sigma - K_{mm})K_{mm}^{-1}\boldsymbol{k}_i),$$

$$\boldsymbol{k}_i = \{k(\boldsymbol{x}_i, \boldsymbol{z}_j)\}_{j=1}^{m}; \quad k_{ii} = k(\boldsymbol{x}_i, \boldsymbol{x}_i).$$

We don't need to work with K_{nn}, only with its diagonal!

Optimization criterion:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i) - \mathrm{KL}(q(\boldsymbol{u})\|p(\boldsymbol{u}|Z, \boldsymbol{\theta})) \to \max_{\boldsymbol{\theta}, \boldsymbol{\mu}, \Sigma}.$$

Second term:

$$\mathrm{KL}(q(\boldsymbol{u}\|p(\boldsymbol{u}|Z, \boldsymbol{\theta})) = \mathrm{KL}(\mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \Sigma)\|\mathcal{N}(\boldsymbol{u}|\mathbf{0}_m, K_{mm})) =$$
$$= -\frac{m}{2} - \frac{1}{2}\log\det K_{mm}^{-1}\Sigma + \frac{1}{2}\mathrm{tr}K_{mm}^{-1}\Sigma + \frac{1}{2}\boldsymbol{\mu}^T K_{mm}^{-1}\boldsymbol{\mu}.$$

Total costs for all $q(f_i)$ and the second term:

Gaussian Process with inducing inputs

Optimization criterion:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i) - \mathrm{KL}(q(\boldsymbol{u})\|p(\boldsymbol{u}|Z, \boldsymbol{\theta})) \to \max_{\boldsymbol{\theta}, \boldsymbol{\mu}, \Sigma}.$$

Second term:

$$\mathrm{KL}(q(\boldsymbol{u}\|p(\boldsymbol{u}|Z, \boldsymbol{\theta})) = \mathrm{KL}(\mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \Sigma)\|\mathcal{N}(\boldsymbol{u}|\mathbf{0}_m, K_{mm})) =$$
$$= -\frac{m}{2} - \frac{1}{2}\log\det K_{mm}^{-1}\Sigma + \frac{1}{2}\mathrm{tr}K_{mm}^{-1}\Sigma + \frac{1}{2}\boldsymbol{\mu}^T K_{mm}^{-1}\boldsymbol{\mu}.$$

Total costs for all $q(f_i)$ and the second term:

$$O(nm^2 + m^3).$$

Computational issues

We need to estimate $\mathbb{E}_{q(f_i)} \log p(y_i|f_i)$.

In case of regression:

$$p(y_i|f_i) = \mathcal{N}(y_i|f_i, \sigma^2),$$
$$\mathbb{E}_{\mathcal{N}(f_i|m_i,s_i^2)} \log p(y_i|f_i) = \log \mathcal{N}(f_i|m_i, \sigma^2) - \frac{1}{2}s_i^2.$$

In case of classification:

$$p(y_i|f_i) = \frac{1}{1+\exp(-y_i f_i)},$$
$$\mathbb{E}_{\mathcal{N}(f_i|m_i,s_i^2)} \log p(y_i|f_i) = \mathbb{E}_{\mathcal{N}(\xi|0,1)} \log p(y_i|s_i\xi + m_i).$$

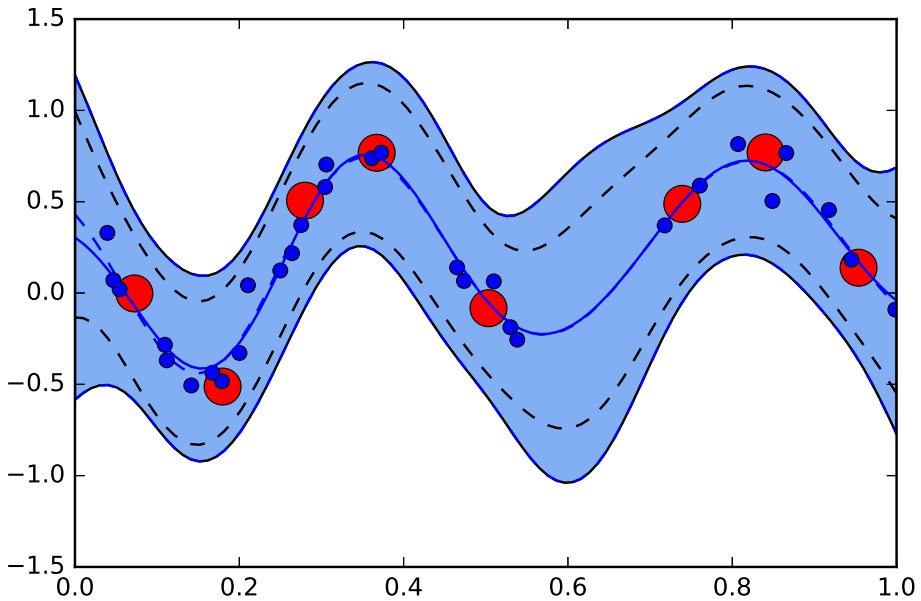
Expectation w.r.t. standard normal distribution can be estimated using Gauss-Hermite quadrature.

Gaussian Process with inducing inputs

- Stochastic optimization with one epoch complexity $O(nm^2 + m^3)$;
- Can optimize w.r.t. positions of inducing points Z, but usually take them fixed, e.g. as cluster centres.

Limitations:

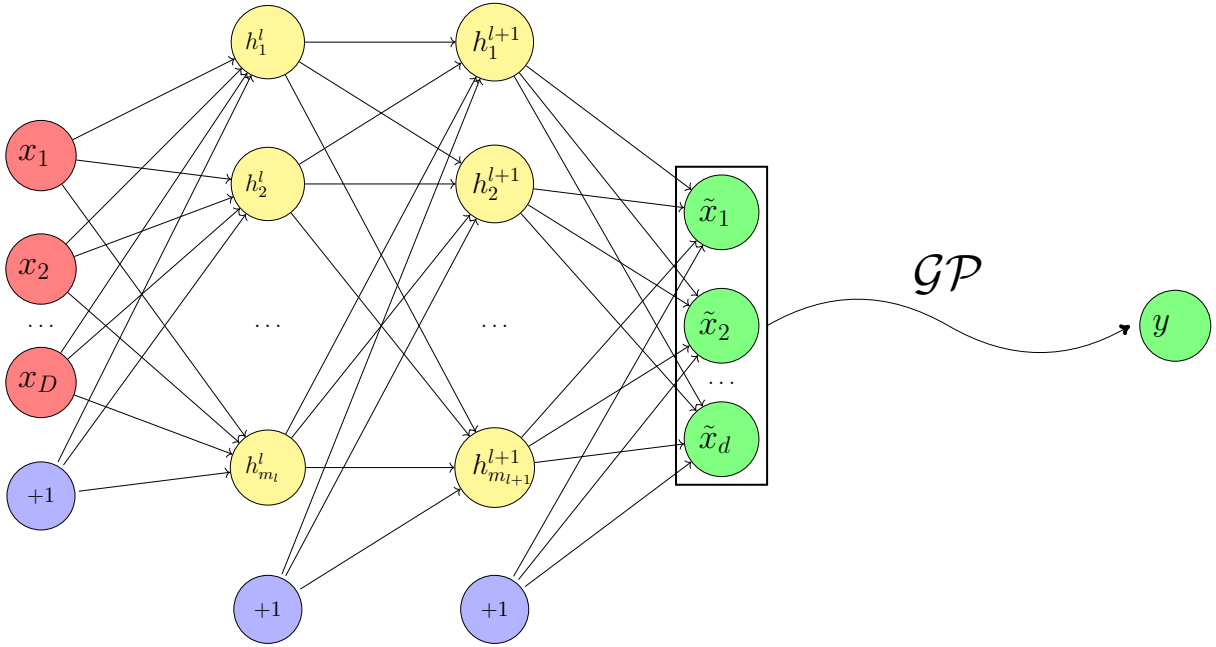
- Can't handle many inducing inputs;
- Current model is not deep.



GP with neural net

Use new covariance function:

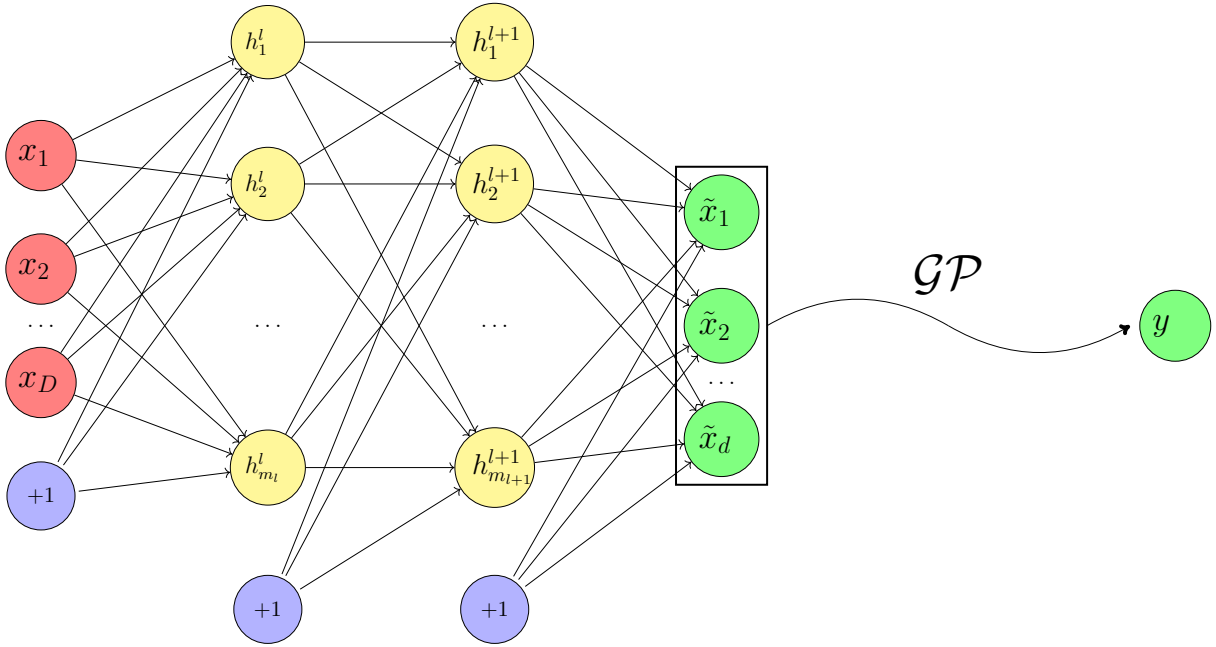
$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\mathrm{net}(\boldsymbol{x}; \boldsymbol{\eta}), \mathrm{net}(\boldsymbol{y}; \boldsymbol{\eta})).$$



GP with neural net

Use new covariance function:

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\mathrm{net}(\boldsymbol{x}; \boldsymbol{\eta}), \mathrm{net}(\boldsymbol{y}; \boldsymbol{\eta})).$$



It is not clear where to put inducing inputs!

Kronecker product

For two matrices $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^{p \times q}$ their Kronecker product is $np \times mq$ matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1m}B \\ \dots & \ddots & \dots \\ a_{n1}B & \dots & a_{nm}B \end{bmatrix}.$$

Properties:

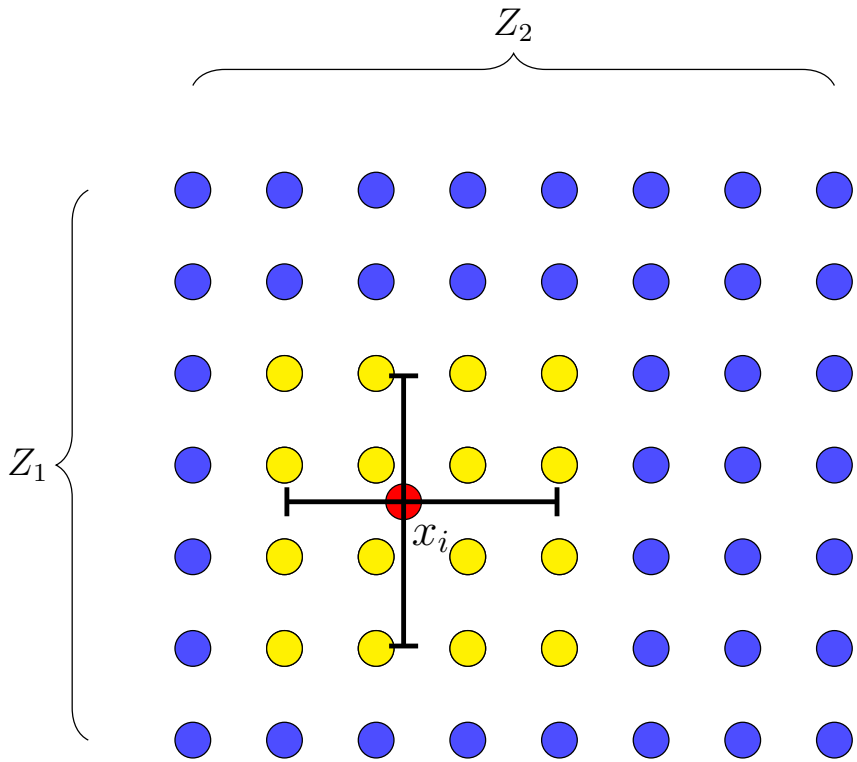
- $(A_1 \otimes A_2 \otimes \cdots \otimes A_r)^{-1} = A_1^{-1} \otimes A_2^{-1} \otimes \cdots \otimes A_r^{-1}$;
- $\det(A_1 \otimes A_2 \otimes \cdots \otimes A_r) = \det(A_1)^{c_1} \det(A_2)^{c_2} \dots \det(A_r)^{c_r}$, where $A_i \in \mathbb{R}^{k_i \times k_i}$, $c_i = \prod_{j \neq i} k_j$.

Inducing points on a regular grid

Let's put inducing inputs Z on a regular grid:

$$Z = Z^1 \times Z^2 \times \cdots \times Z^d,$$

where $Z^i \in \mathbb{R}^{m_i}$, $m = \prod_{i=1}^{d} m_i$.



Inducing points on a regular grid

Suppose that covariance function can be split over dimensions:

$$k(\boldsymbol{x}, \boldsymbol{y}) = k^1(x_1, y_1)k^2(x_2, y_2)\dots k^d(x_d, y_d).$$

E.g. for squared exponential:

$$k(\boldsymbol{x}, \boldsymbol{y}) = A\exp(-B\|\boldsymbol{x} - \boldsymbol{y}\|^2) =$$
$$\underbrace{A^{1/d}\exp(-B(x_1 - y_1)^2)}_{k^1(x_1,y_1)}\underbrace{A^{1/d}\exp(-B(x_2 - y_2)^2)}_{k^2(x_2,y_2)}\dots\underbrace{A^{1/d}\exp(-B(x_d - y_d)}_{k^d(x_d,y_d)}$$

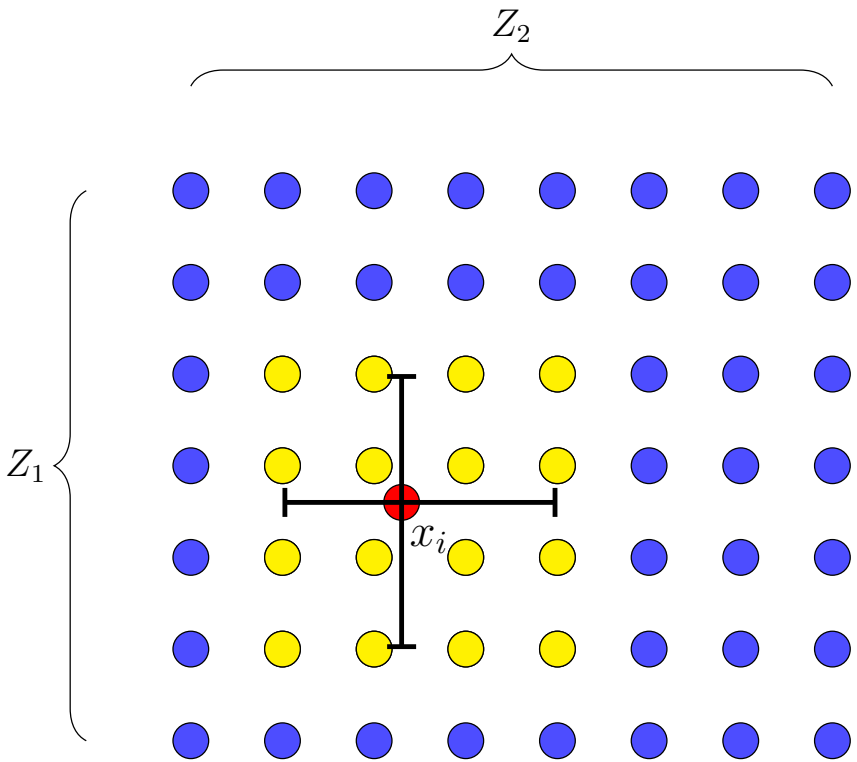
Then covariance matrix is given as Kronecker product:

$$K_{mm} = K^1_{m_1m_1} \otimes K^2_{m_2m_2} \otimes \cdots \otimes K^d_{m_dm_d},$$
$$K^i_{m_im_i} = K^i(Z^i, Z^i) \in \mathbb{R}^{m_i \times m_i}.$$

Cubic convolution interpolation [Keys, 1981]

We need to estimate covariance between training and inducing inputs K_{mn}. In case of cubic convolution interpolation we have:

$$K_{mn} \approx K_{mm} W, \; \boldsymbol{k}_i \approx K_{mm} \boldsymbol{w}_i,$$
$$\boldsymbol{w}_i = \boldsymbol{w}_i^1 \otimes \boldsymbol{w}_i^2 \otimes \cdots \otimes \boldsymbol{w}_i^d.$$



GP with inducing inputs

Optimization criterion:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i) - \mathrm{KL}(q(\boldsymbol{u})\|p(\boldsymbol{u}|Z, \boldsymbol{\theta})) \to \max_{\boldsymbol{\theta}, \boldsymbol{\mu}, \Sigma}.$$

If $\boldsymbol{k}_i = K_{mm}\boldsymbol{w}_i$, then

$$q(f_i) = \mathcal{N}(f_i|\boldsymbol{k}_i^T K_{mm}^{-1}\boldsymbol{\mu}, k_{ii} + \boldsymbol{k}_i^T K_{mm}^{-1}(\Sigma - K_{mm})K_{mm}^{-1}\boldsymbol{k}_i) =$$
$$= \mathcal{N}(f_i|\boldsymbol{w}_i^T\boldsymbol{\mu}, k_{ii} + \boldsymbol{w}_i^T(\Sigma - K_{mm})\boldsymbol{w}_i).$$

Second term:

$$\mathrm{KL}(q(\boldsymbol{u}\|p(\boldsymbol{u}|Z, \boldsymbol{\theta})) =$$
$$= -\frac{m}{2} - \frac{1}{2}\log\det K_{mm}^{-1}\Sigma + \frac{1}{2}\mathrm{tr}K_{mm}^{-1}\Sigma + \frac{1}{2}\boldsymbol{\mu}^T K_{mm}^{-1}\boldsymbol{\mu}.$$

Tensor Train format [Oseledets, 2011]

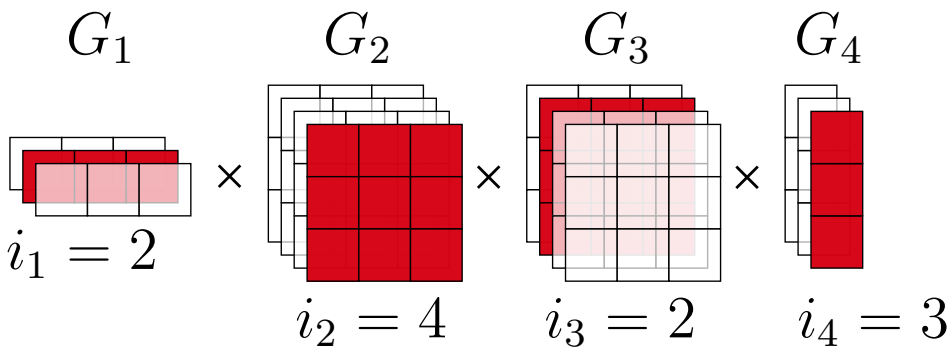
Tensor Train format (TT format) gives a compact representation of multidimensional tensors. If $\mathcal{A} \in \mathbb{R}^{m_1 \times m_2 \times \ldots \times m_d}$, then

$$A(i_1, i_2, \ldots, i_d) = G[i_1] G[i_2] \ldots G[i_d],$$
$$G[i_k] \in \mathbb{R}^{r_k \times r_{k+1}}, \; r_1 = r_{d+1} = 1.$$

Here $G[i_k]$ are TT cores and r_k – TT ranks.

TT format allows many linear algebra operations to perform efficiently.

$$\mathcal{A}(2, 4, 2, 3) =$$



TT-GP [Izmailov et al., 2017]

A family for variational distribution $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \Sigma)$:

- $\boldsymbol{\mu}$ in TT format with fixed rank r;
- $\Sigma = \Sigma^1 \otimes \Sigma^2 \otimes \cdots \otimes \Sigma^d$.

Optimization criterion:

$$\log p(\boldsymbol{y}|X, Z, \boldsymbol{\theta}) \geq \sum_{i=1}^{n} \mathbb{E}_{q(f_i)} \log p(y_i|f_i) - \frac{m}{2} + \frac{1}{2} \log \det K_{mm}^{-1}\Sigma - \\ - \frac{1}{2}\mathrm{tr} K_{mm}^{-1}\Sigma - \frac{1}{2}\boldsymbol{\mu}^T K_{mm}^{-1}\boldsymbol{\mu},$$

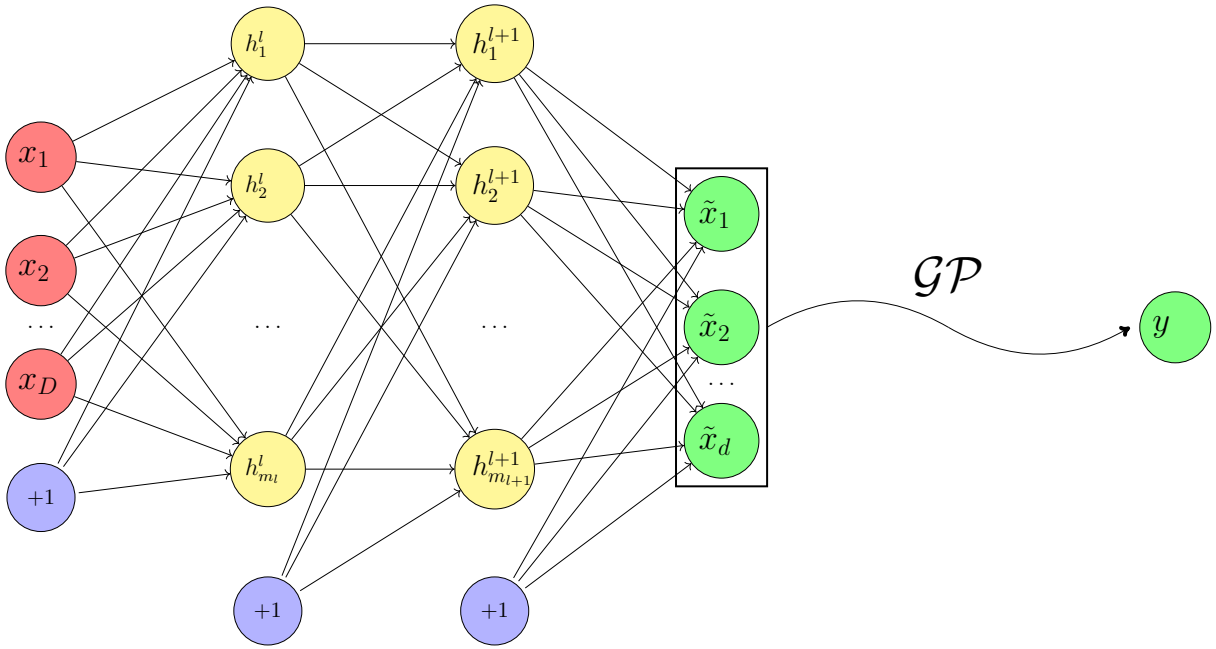
$$q(f_i) = \mathcal{N}(f_i|\boldsymbol{w}_i^T\boldsymbol{\mu}, k_{ii} + \boldsymbol{w}_i^T(\Sigma - K_{mm})\boldsymbol{w}_i).$$

If m_0 is number of inducing inputs per dimension, then all calculations can be performed in $O(ndm_0r^2 + dm_0r^3 + dm_0^3)$.

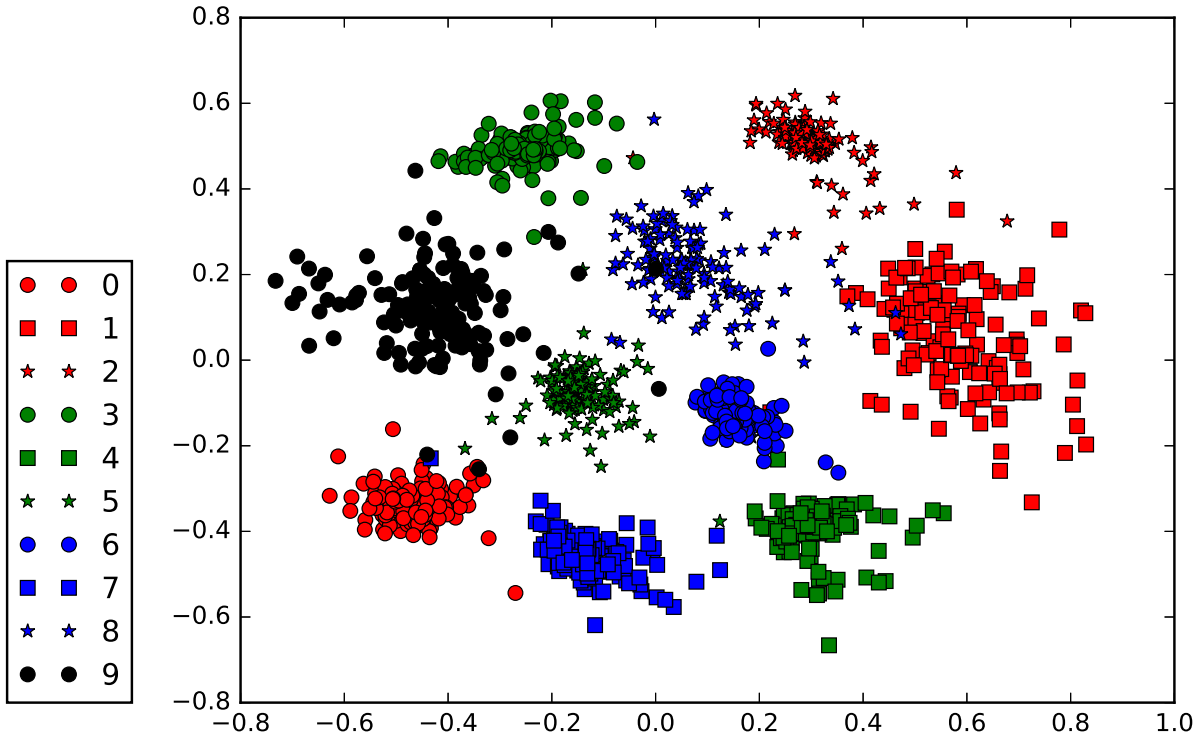
TT-GP with deep net

TT-GP can be easily combined with deep net covariance function

$$k(\boldsymbol{x}, \boldsymbol{y}) = k(\text{net}(\boldsymbol{x}; \boldsymbol{\eta}), \text{net}(\boldsymbol{y}; \boldsymbol{\eta})).$$



Representation for Digits



Experimental results

Dataset			SVI-GP/KLSP-GP			TT-GP				
Name	n	D	acc.	m	time	acc.	m	r	d	time
Powerplant	7654	4	0.94	200	10	0.95	35^4	30	–	5
Protein	36584	9	0.50	200	45	0.56	30^9	25	–	40
YearPred	463K	90	0.30	1000	597	0.32	10^6	10	6	105
Airline	6M	8	0.665	–	–	0.694	20^8	15	–	5200
Svmguide1	3089	4	0.967	200	4	0.969	20^4	15	–	1
EEG	11984	14	0.915	1000	18	0.908	12^{10}	15	10	10
covtype bin	465K	54	0.817	1000	320	0.852	10^6	10	6	172

Experiments with deep networks:

Dataset				SV-DKL	DNN		TT-GP		
Name	n	D	C	acc.	acc.	time	acc.	d	time
Airline	6M	8	2	0.781	0.780	1055	0.784	2	1375
CIFAR-10	50K	$32\times32\times3$	10	0.770	0.915	166	0.909	9	220
MNIST	60K	28×28	10	0.992	0.993	23	0.994	10	64