Variational Gaussian Processes

Zhenwen Dai

Spotify

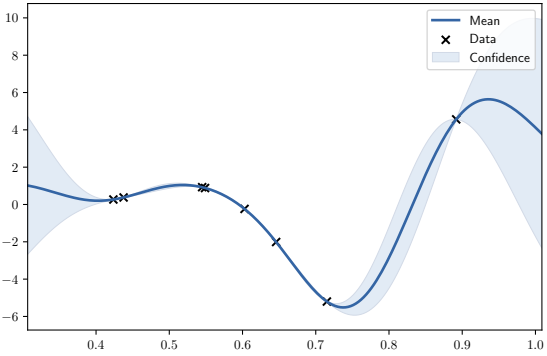
15 September 2020 @GPSS 2020

Gaussian process

Input and Output Data:

$$\mathbf{y} = (y_1, \ldots, y_N), \quad \mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)^\top$$

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



Behind a Gaussian process fit

- Maximum likelihood estimate of the hyper-parameters.

$$\theta^* = \arg\max_\theta \log p(\mathbf{y}|\mathbf{X}, \theta) = \arg\max_\theta \log \mathcal{N}\left(\mathbf{y}|0, \mathbf{K} + \sigma^2\mathbf{I}\right)$$

- Prediction on a test point given the observed data and the optimized hyper-parameters.

$$p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{y}, \mathbf{X}, \theta) = \mathcal{N}\left(\mathbf{f}_*|\mathbf{K}_*(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{y}, \mathbf{K}_{**} - \mathbf{K}_*(\mathbf{K} + \sigma^2\mathbf{I})^{-1}\mathbf{K}_*^\top\right)$$

How to implement the log-likelihood (1)

- Compute the covariance matrix $\mathbf{K}$:

$$\mathbf{K} = \begin{pmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{pmatrix}$$

  where $k(\mathbf{x}_i, \mathbf{x}_j) = \gamma \exp\left(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)^\top (\mathbf{x}_i - \mathbf{x}_j)\right)$

- The complexity is $O(N^2 Q)$.

How to implement the log-likelihood (2)

- Plug in the log-pdf of multi-variate normal distribution:

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{X}) &= \log \mathcal{N}\left(\mathbf{y}|0, \mathbf{K}+\sigma^2\mathbf{I}\right) \\ &= -\frac{1}{2}\log|2\pi(\mathbf{K}+\sigma^2\mathbf{I})| - \frac{1}{2}\mathbf{y}^\top(\mathbf{K}+\sigma^2\mathbf{I})^{-1}\mathbf{y} \\ &= -\frac{1}{2}(||\mathbf{L}^{-1}\mathbf{y}||^2 + N\log 2\pi) - \sum_i \log \mathbf{L}_{ii}\end{aligned}$$

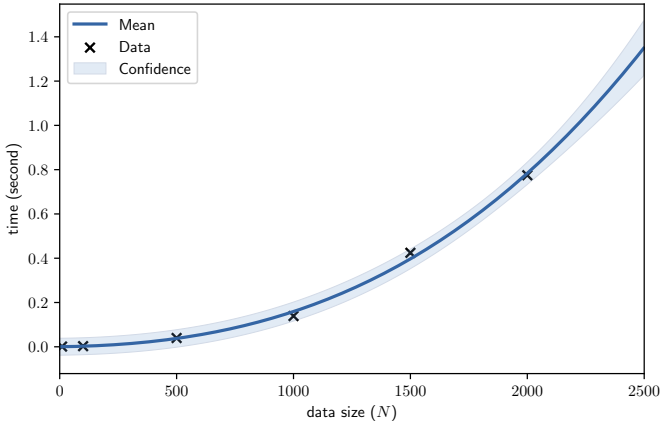
- Take a Cholesky decomposition: $\mathbf{L} = \texttt{chol}(\mathbf{K}+\sigma^2\mathbf{I})$.
- The computational complexity is $O(N^3+N^2+N)$. Therefore, the overall complexity including the computation of $\mathbf{K}$ is $O(N^3)$.

A quick profiling (N=1000, Q=10)

```
Line #    Time(ms)   % Time      Line Contents
 2                            def log_likelihood(kern, X, Y, sigma2):
 3           6.0       0.0        N = X.shape[0]
 4       55595.0      58.7        K = kern.K(X)
 5        4369.0       4.6        Ky = K + np.eye(N)*sigma2
 6       30012.0      31.7        L = np.linalg.cholesky(Ky)
 7        4361.0       4.6        LinvY = dtrtrs(L, Y, lower=1)[0]
 8          49.0       0.1        logL = N*np.log(2*np.pi)/-2.
 9          82.0       0.1        logL += np.square(LinvY).sum()/-2.
10         208.0       0.2        logL += -np.log(np.diag(L)).sum()
11           2.0       0.0        return logL
```

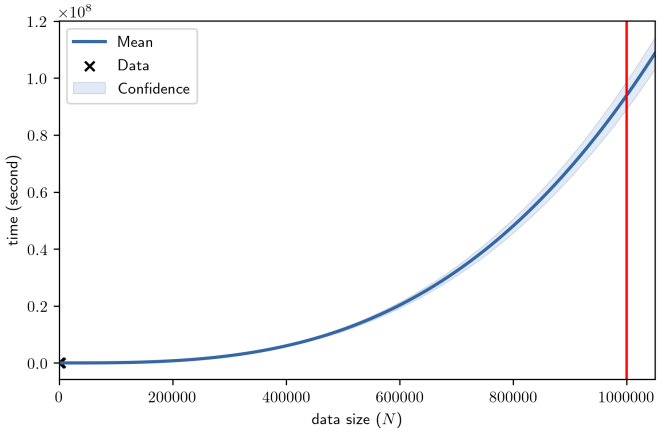
Empirical analysis of computational time

- I collect the run time for $N = \{10, 100, 500, 1000, 1500, 2000\}$.
- They take 1.3ms, 8.5ms, 28ms, 0.12s, 0.29s, 0.76s.



What if we have 1 million data points?

The mean of predicted computational time is 9.4×10^7 seconds ≈ 2.98 years.

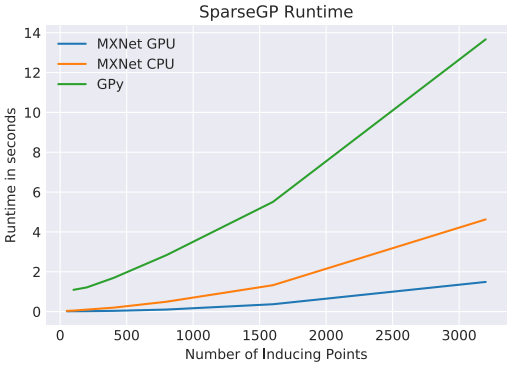


What about waiting for faster computers?

- Computational time = amount of work / computer speed.
- If the computer speed increase at the pace of 20% year over year:
  - After 10 years, it will take about 176 days.
  - After 50 years, it will take about 2.9 hours.

What about parallel computing / GPU?

- Ongoing works about speeding up Cholesky decomposition with multi-core CPU or GPU.
- Main limitation:
  - heavy communication and shared memory.
  - $O(N^2)$ memory consumption

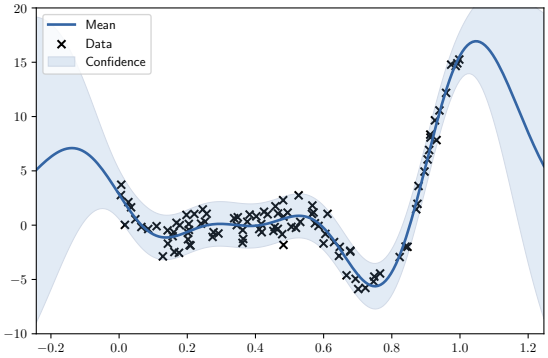


Other approaches

- Apart from speeding up the exact computation, there have been a lot of works on approximation of GP inference.
- These methods often target at some specific scenario and provide good approximation for the targeted scenarios.
- Provide an overview about common approximations.

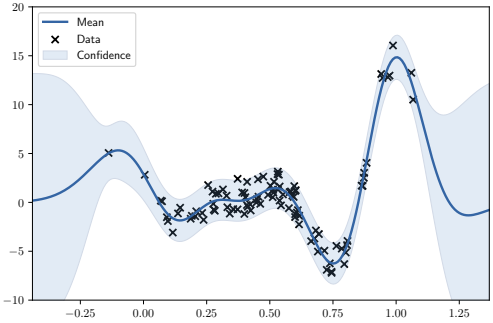
Big data (?)

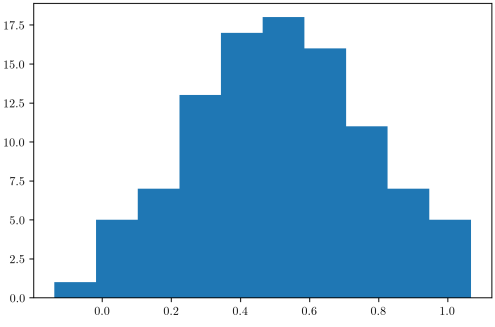
- lots of data $\neq$ complex function
- In real world problems, we often collect a lot of data for modeling relatively simple relations.



Data subsampling?

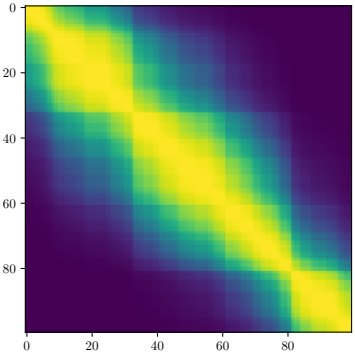
- Real data often do not evenly distributed.
- We tend to get a lot of data on common cases and very few data on rare cases.

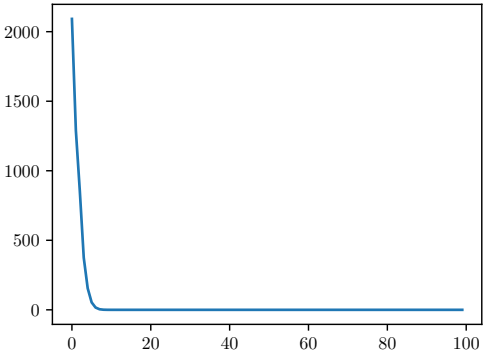




Covariance matrix of redundant data

- With redundant data, the covariance matrix becomes low rank.
- What about low rank approximation?





Low-rank approximation

- Let's recall the log-likelihood of GP:

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

  where $\mathbf{K}$ is the covariance matrix computed from $\mathbf{X}$ according to the kernel function $k(\cdot, \cdot)$ and σ^2 is the variance of the Gaussian noise distribution.
- Assume $\mathbf{K}$ to be low rank.
- This leads to Nyström approximation by Williams and Seeger [Williams and Seeger, 2001].

Approximation by subset

- Let's randomly pick a subset from the training data: $\mathbf{Z} \in \mathbb{R}^{M \times Q}$.
- Approximate the covariance matrix $\mathbf{K}$ by $\tilde{\mathbf{K}}$.

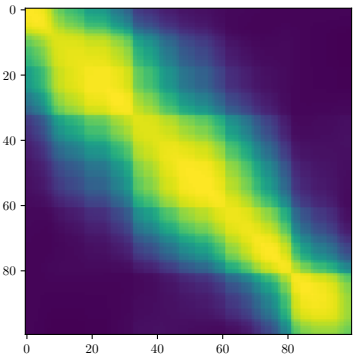
$$\tilde{\mathbf{K}} = \mathbf{K}_z \mathbf{K}_{zz}^{-1} \mathbf{K}_z^\top, \text{ where } \mathbf{K}_z = \mathbf{K}(\mathbf{X}, \mathbf{Z}) \text{ and } \mathbf{K}_{zz} = \mathbf{K}(\mathbf{Z}, \mathbf{Z}).$$

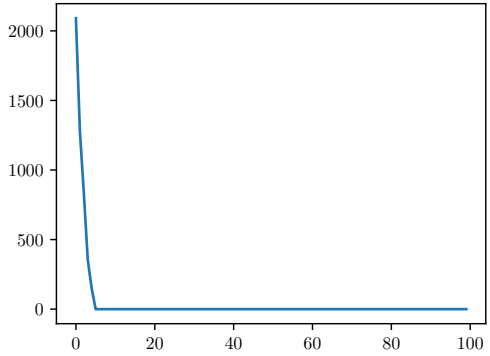
- Note that $\tilde{\mathbf{K}} \in \mathbb{R}^{N \times N}$, $\mathbf{K}_z \in \mathbb{R}^{N \times M}$ and $\mathbf{K}_{zz} \in \mathbb{R}^{M \times M}$.
- The log-likelihood is approximated by

$$\log p(\mathbf{y}|\mathbf{X}, \theta) \approx \log \mathcal{N}\left(\mathbf{y}|0, \mathbf{K}_z \mathbf{K}_{zz}^{-1} \mathbf{K}_z^\top + \sigma^2 \mathbf{I}\right).$$

Nyström approximation example

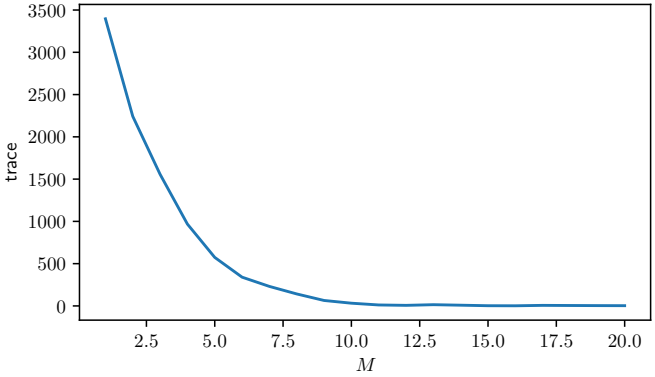
The covariance matrix with Nyström approximation using 5 random data points:





Nyström approximation example

Compute $\text{tr}\left(\mathbf{K} - \tilde{\mathbf{K}}\right)$ with different M.



Efficient computation using Woodbury formula

- The naive formulation does not bring any computational benefits.

$$\tilde{\mathcal{L}} = -\frac{1}{2}\log|2\pi(\tilde{\mathbf{K}} + \sigma^2\mathbf{I})| - \frac{1}{2}\mathbf{y}^\top(\tilde{\mathbf{K}} + \sigma^2\mathbf{I})^{-1}\mathbf{y}$$

- Apply the Woodbury formula:

$$(\mathbf{K}_z\mathbf{K}_{zz}^{-1}\mathbf{K}_z^\top + \sigma^2\mathbf{I})^{-1} = \sigma^{-2}\mathbf{I} - \sigma^{-4}\mathbf{K}_z(\mathbf{K}_{zz} + \sigma^{-2}\mathbf{K}_z^\top\mathbf{K}_z)^{-1}\mathbf{K}_z^\top$$

- Note that $(\mathbf{K}_{zz} + \sigma^{-2}\mathbf{K}_z^\top\mathbf{K}_z) \in \mathbb{R}^{M\times M}$.
- The computational complexity reduces to $O(NM^2)$.

Nyström approximation

- The approximation is directly done on the covariance matrix without the concept of pseudo data.
- The approximation becomes exact if the whole data set is taken, *i.e.*, $\mathbf{K}\mathbf{K}^{-1}\mathbf{K}^{\top} = \mathbf{K}$.
- The subset selection is done randomly.

Gaussian process with Pseudo Data (1)

- Snelson and Ghahramani [2006] proposes the idea of having pseudo data, which is later referred to as Fully independent training conditional (FITC).
- Augment the training data $(\mathbf{X}, \mathbf{y})$ with pseudo data $\mathbf{u}$ at location $\mathbf{Z}$.

$$p\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{u}\end{bmatrix} \mid \begin{bmatrix}\mathbf{X}\\ \mathbf{Z}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\mathbf{y}\\ \mathbf{u}\end{bmatrix} | 0, \begin{bmatrix}\mathbf{K}_{ff} + \sigma^2\mathbf{I} & \mathbf{K}_{fu}\\ \mathbf{K}_{fu}^\top & \mathbf{K}_{uu}\end{bmatrix}\right)$$

where $\mathbf{K}_{ff} = \mathbf{K}(\mathbf{X}, \mathbf{X})$, $\mathbf{K}_{fu} = \mathbf{K}(\mathbf{X}, \mathbf{Z})$ and $\mathbf{K}_{uu} = \mathbf{K}(\mathbf{Z}, \mathbf{Z})$.

Gaussian process with Pseudo Data (2)

- Thanks to the marginalization property of Gaussian distribution,

$$p(\mathbf{y}|\mathbf{X}) = \int_{\mathbf{u}} p(\mathbf{y}, \mathbf{u}|\mathbf{X}, \mathbf{Z}).$$

- Further re-arrange the notation:

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

where $p(\mathbf{u}|\mathbf{Z}) = \mathcal{N}(\mathbf{u}|0, \mathbf{K}_{uu})$,
$p(\mathbf{y}|\mathbf{u}, \mathbf{X}, \mathbf{Z}) = \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu}^{\top} + \sigma^2\mathbf{I}\right)$.

FITC approximation (1)

- So far, $p(\mathbf{y}|\mathbf{X})$ has not been changed, but there is no speed-up.
- $\mathbf{K}_{ff} \in \mathbb{R}^{N \times N}$ in $\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu}^{\top} + \sigma^2\mathbf{I}$.
- The FITC approximation assumes

$$\tilde{p}(\mathbf{y}|\mathbf{u}, \mathbf{X}, \mathbf{Z}) = \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{\Lambda} + \sigma^2\mathbf{I}\right),$$

where $\mathbf{\Lambda} = (\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu}^{\top}) \circ \mathbf{I}$.

FITC approximation (2)

- Marginalize $\mathbf{u}$ from the model definition:

$$\tilde{p}(\mathbf{y}|\mathbf{X},\mathbf{Z}) = \mathcal{N}\left(\mathbf{y}|0, \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu}^{\top} + \mathbf{\Lambda} + \sigma^2\mathbf{I}\right)$$

- Woodbury formula can be applied in the sam way as in Nyström approximation:

$$(\mathbf{K}_z\mathbf{K}_{zz}^{-1}\mathbf{K}_z^{\top} + \mathbf{\Lambda} + \sigma^2\mathbf{I})^{-1} = \mathbf{A} - \mathbf{A}\mathbf{K}_z(\mathbf{K}_{zz} + \mathbf{K}_z^{\top}\mathbf{A}\mathbf{K}_z)^{-1}\mathbf{K}_z^{\top}\mathbf{A},$$

where $\mathbf{A} = (\mathbf{\Lambda} + \sigma^2\mathbf{I})^{-1}$.

FITC approximation (3)

- FITC allows the pseudo data not being a subset of training data.
- The inducing inputs $\mathbf{Z}$ can be optimized via gradient optimization.
- Like Nyström approximation, when taking all the training data as inducing inputs, the FITC approximation is equivalent to the original GP:

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

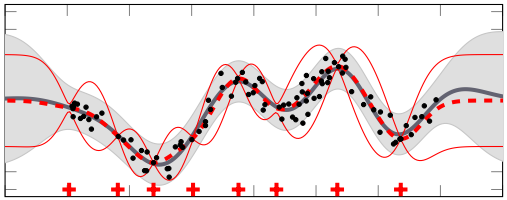
- FITC can be combined easily with expectation propagation (EP).
- Bui et al. [2017] provides an overview and a nice connection with variational sparse GP.

Model Approximation vs. Approximate Inference

FITC approximation changes the model definition.

- A better objective under FITC does not necessarily corresponds to a better approximation to the original GP.
- In fact, optimizing $\mathbf{Z}$ can lead to overfitting. [Quiñonero-Candela and Rasmussen, 2005, Bauer et al., 2016]

FITC (nlml $= 23.16$, $\sigma_n = 1.93 \cdot 10^{-4}$)



Optimal values for the exact GP: nlml $= 34.15$, $\sigma = 0.274$. [Bauer et al., 2016]

Model Approximation vs. Approximate Inference

Variational inference (VI) takes a different approach.

- VI keeps the model definition untouched.
- VI derives a lower bound of the log-marginal likelihood:

$$\log(y) \geq \int q(x) \log \frac{p(y,x)}{q(x)} \mathrm{d}x = \mathcal{L}$$

- Alternatively, it can be written as

$$\mathrm{KL}\left(q(x) \,\|\, p(x|y)\right) = \log p(y) - \mathcal{L}.$$

Variational Sparse Gaussian Process (1)

- Titsias [2009] introduces a variational approach for sparse GP.
- It follows the same concept of pseudo data:

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

where $p(\mathbf{u}|\mathbf{Z}) = \mathcal{N}(\mathbf{u}|0,\mathbf{K}_{uu})$,
$p(\mathbf{y}|\mathbf{u},\mathbf{X},\mathbf{Z}) = \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu}^{\top} + \sigma^2\mathbf{I}\right)$.

Variational Sparse Gaussian Process (2)

- Instead of approximate the model, Titsias [2009] derives a variational lower bound.
- Normally, a variational lower bound of a marginal likelihood looks like

$$\begin{aligned}\log p(\mathbf{y}|\mathbf{X}) &= \log \int_{\mathbf{f},\mathbf{u}} p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{u},\mathbf{X},\mathbf{Z}) p(\mathbf{u}|\mathbf{Z}) \\ &\geq \int_{\mathbf{f},\mathbf{u}} q(\mathbf{f},\mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{u},\mathbf{X},\mathbf{Z}) p(\mathbf{u}|\mathbf{Z})}{q(\mathbf{f},\mathbf{u})}.\end{aligned}$$

Special Variational Posterior

- Titsias [2009] defines an unusual variational posterior:

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u}), \quad \text{where } q(\mathbf{u}) = \mathcal{N}(\mathbf{u}|\mu, \Sigma).$$

- Plug it into the lower bound:

$$\begin{aligned}
\mathcal{L} &= \int_{\mathbf{f},\mathbf{u}} p(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u}) \log \frac{p(\mathbf{y}|\mathbf{f})\cancel{p(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z})}p(\mathbf{u}|\mathbf{Z})}{\cancel{p(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z})}q(\mathbf{u})} \\
&= \langle \log p(\mathbf{y}|\mathbf{f}) \rangle_{p(\mathbf{f}|\mathbf{u},\mathbf{X},\mathbf{Z})q(\mathbf{u})} - \mathsf{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u}|\mathbf{Z})\right) \\
&= \left\langle \log \mathcal{N}\left(\mathbf{y}|\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \sigma^2\mathbf{I}\right) \right\rangle_{q(\mathbf{u})} - \mathsf{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u}|\mathbf{Z})\right)
\end{aligned}$$

Special Variational Posterior

- There is no inversion of any big covariance matrices in the first term:

$$-\frac{N}{2}\log 2\pi\sigma^2 - \frac{1}{2\sigma^2}\left\langle(\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}-\mathbf{y})^\top(\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}-\mathbf{y})\right\rangle_{q(\mathbf{u})}$$

- The overall complexity of the lower bound is $O(NM^2)$.

Tighten the Bound

- Find the optimal parameters of $q(\mathbf{u})$:

$$\mu^*, \Sigma^* = \arg\max_{\mu,\Sigma} \mathcal{L}(\mu, \Sigma).$$

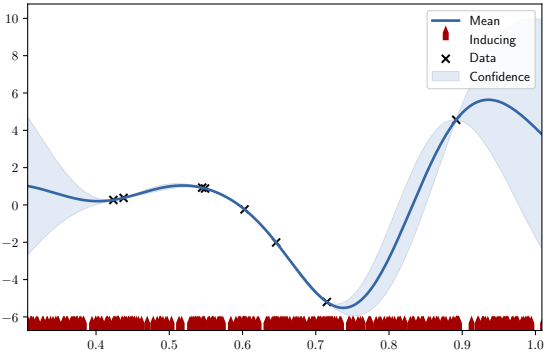
- Make the bound as tight as possible by plugging in μ^* and Σ^*:

$$\mathcal{L} = \log \mathcal{N}\left(\mathbf{y}|0, \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu}^\top + \sigma^2\mathbf{I}\right) - \frac{1}{2\sigma^2}\text{tr}\left(\mathbf{K}_{ff} - \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{K}_{fu}^\top\right).$$

- The 1st term is the same as in the Nyström approximation.
- The overall complexity of the lower bound remains $O(NM^2)$.

Variational sparse GP

- Note that $\mathcal{L}$ is not a valid log-pdf, $\int_{\mathbf{y}} \exp(\mathcal{L}(\mathbf{y})) \leq 1$, due to the trace term.
- As inducing points are variational parameters, optimizing the inducing inputs $\mathbf{Z}$ always leads to a better bound.
- The model does not "overfit" with too many inducing points.

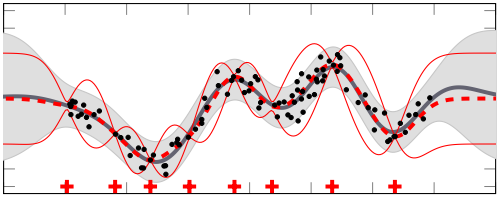


FITC vs. Variational sparse GP

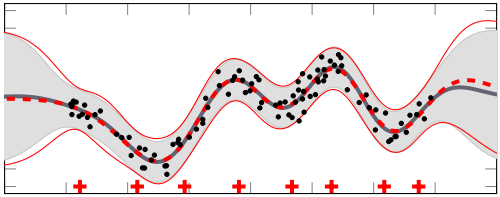
- model approximation vs. approximate inference (see [Bauer et al., 2016])
- Note that, when point estimating hyper-parameters, if the number of inducing points is too small, the model may "under-fit":

$$\mathcal{L} = \log p(y) - \mathsf{KL}\left(q(x) \,\|\, p(x|y)\right).$$

FITC (nlml $= 23.16$, $\sigma_n = 1.93 \cdot 10^{-4}$)



VFE (nlml $= 38.86$, $\sigma_n = 0.286$)



Optimal values for the exact GP: nlml $= 34.15$, $\sigma = 0.274$. [Bauer et al., 2016]

Limitations of Sparse GP

Variational sparse GP has computational complexity $O(NM^2)$.

The computation becomes infeasible under two scenarios:

- The number of data points N is very high, e.g., millions of data points.
- The function is very complex, which requires tens of thousands of inducing points.

Mini-batch Learning (1)

- Mini-batch learning allows DNNs to be trained on millions of data points.
- Given a set of inputs and labels, $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, $(\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)$, the true loss function is defined as

$$c_{\text{true}} = \int l(f_\theta(\mathbf{x}), y) p(\mathbf{x}, y) \mathrm{d}\mathbf{x} \mathrm{d}y \approx \frac{1}{N} \sum_{i=1}^{N} l(f_\theta(\mathbf{x}), y) = c,$$

  where $f_\theta(\cdot)$ is DNN and $l(\cdot, \cdot)$ is the loss function.
- Gradient descent (GD) updates the parameters by

$$\theta_{t+1} = \theta_t - \eta \frac{\mathrm{d}c}{\mathrm{d}\theta}.$$

Mini-batch Learning (2)

- Mini-batch learning approximates the loss by subsampling the data,

$$c_{\mathsf{MB}} = \frac{1}{B} \sum_{\mathbf{x}_i, y_i \sim \tilde{p}(\mathbf{x}, y)} l(f_\theta(\mathbf{x}_i), y_i).$$

- Stochastic gradient descent (SGD) updates the parameters by

$$\theta_{t+1} = \theta_t - \eta \frac{\mathsf{d}c_{\mathsf{MB}}}{\mathsf{d}\theta}.$$

- Can mini-batch learning be applied to GPs as well?

Mini-batch Learning for GPs

- Mini-batch learning relies on the objective being an expectation w.r.t. the data, *i.e.*, $\langle l(f_\theta(\mathbf{x}), y) \rangle_{p(\mathbf{x},y)}$.
- The log-marginal likelihood of GP:

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

- The variational lower bound of sparse GP:

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

"Uncollapsed" Lower Bound

- Hensman et al. [2013] discovers that the "uncollapsed" variational lower bound of sparse GP can be used for mini-batch learning.
- The "uncollapsed" variational lower bound of sparse GP:

$$\mathcal{L} = \langle \log p(\mathbf{y}|\mathbf{f}) \rangle_{p(\mathbf{f}|\mathbf{u},\mathbf{X},\mathbf{Z})q(\mathbf{u})} - \mathrm{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u})\right)$$

- The 2nd term, $\mathrm{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u})\right)$, does not depend on the data.

"Uncollapsed" Lower Bound

- In the 1st term, as $p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma^2\mathbf{I})$,

$$\log p(\mathbf{y}|\mathbf{f}) = \sum_{n=1}^{N} \log \mathcal{N}\left(y_n|f_n, \sigma^2\right)$$

- Denote $q(\mathbf{f}|\mathbf{X}, \mathbf{Z}) = \int p(\mathbf{f}|\mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u}) \mathrm{d}\mathbf{u}$.

$$\begin{aligned}\langle \log p(\mathbf{y}|\mathbf{f}) \rangle_{q(\mathbf{f}|\mathbf{X},\mathbf{Z})} &= \left\langle \sum_{n=1}^{N} \log \mathcal{N}\left(y_n|f_n, \sigma^2\right) \right\rangle_{q(\mathbf{f}|\mathbf{X},\mathbf{Z})} \\ &= \sum_{n=1}^{N} \left\langle \log \mathcal{N}\left(y_n|f_n, \sigma^2\right) \right\rangle_{q(f_n|\mathbf{x}_n,\mathbf{Z})}\end{aligned}$$

Stochastic Variational GP (SVGP)

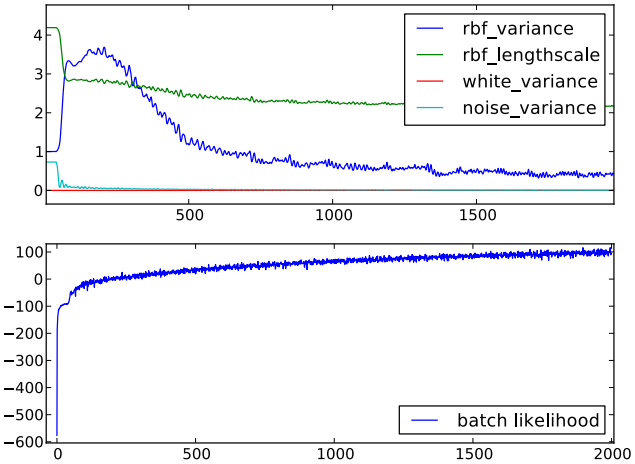
- The resulting lower bound can be written as the sum over the data,

$$\begin{aligned}\mathcal{L} =& \sum_{n=1}^{N} \left\langle \log \mathcal{N}\left(y_n | f_n, \sigma^2\right) \right\rangle_{q(f_n|\mathbf{x}_n,\mathbf{Z})} - \mathsf{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u})\right) \\ \approx& \frac{N}{B} \sum_{\mathbf{x}_i, y_i \sim \tilde{p}(\mathbf{x},y)} \left\langle \log \mathcal{N}\left(y_i | f_i, \sigma^2\right) \right\rangle_{q(f_i|\mathbf{x}_i,\mathbf{Z})} - \frac{N}{B}\mathsf{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u})\right) = \mathcal{L}_{\mathsf{MB}}\end{aligned}$$

- This allows us to do mini-batch learning with SGD,

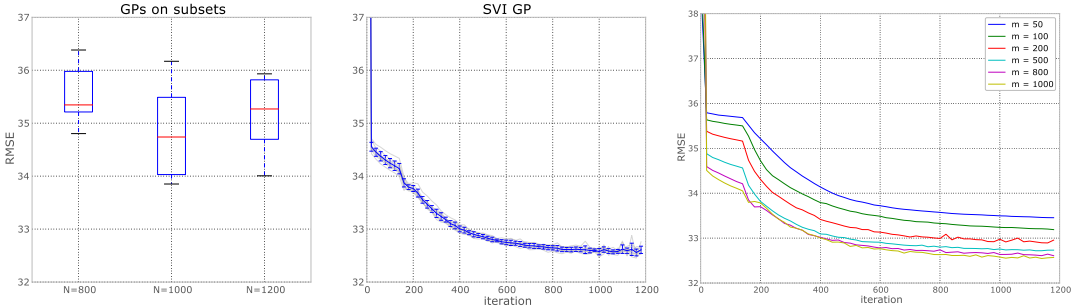
$$\theta_{t+1} = \theta_t - \eta \frac{\mathsf{d}\mathcal{L}_{\mathsf{MB}}}{\mathsf{d}\theta}.$$

2D Synthetic Data



Airline Delay Data

Flight delays for every commercial flight in the USA from January to April 2008.
700,000 train, 100,000 test



The pros and cons of SVGP

Pros

- With mini-batch learning, the computational complexity reduces from $O(NM^2)$ to $O(M^3)$.

Cons

- The variational distribution $q(\mathbf{u})$ needs to be explicitly optimized.
- The number of variational parameters increase from MQ to $(2M + M^2)Q$.
- Optimization relies on SGD methods and the methods like L-BFGS are no longer applicable.
- It can be challenging to initialize $q(\mathbf{u})$.

Non-Gaussian likelihood

So far, we have only discussed GP regression with Gaussian noise distribution.

In practice, many difference noise distributions for modelling real data, e.g.,

- Student-t distribution: data with outliers
- Poisson / Multi-nomial distribution: Integer counts
- Beta distribution: bounded real values
- Bernoulli / Categorical distribution: classification labels

Common approaches

Common choice for approximation inference:

- Exact GP with Laplace approximation
- Expectation Propagation (EP) with sparse GP

Both of them are quite complex to implement and difficult to scale.

SVGP with non-Gaussian likelihood

Let's use a binary classification as an example.

- The outputs are binary. $\mathbf{y} = (y_1, \ldots, y_N)$, $y_i \in \{0, 1\}$.
- The likelihood is a Bernoulli distribution with a Sigmoid link function:

$$p(y_i|f_i) = \sigma(f_i)^{y_i}(1 - \sigma(f_i))^{(1-y_i)}$$

SVGP with non-Gaussian likelihood

- The lower bound of SVGP is

$$\mathcal{L} = \sum_{n=1}^{N} \left\langle \log p(y_n|f_n) \right\rangle_{q(f_n|\mathbf{x}_n,\mathbf{Z})} - \mathsf{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u})\right).$$

- The 2nd term, $\mathsf{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u})\right)$, is closed form.
- The 1st term, $\sum_{n=1}^{N} \left\langle \log p(y_n|f_n) \right\rangle_{q(f_n|\mathbf{x}_n,\mathbf{Z})}$, is the sum of a list of 1D integrals.
- Those integrals are intractable.

Gauss-Hermite Quadrature

Regarding those 1D integrals,

- $q(f_n|\mathbf{x}_n, \mathbf{Z})$ is a 1D Gaussian distribution. See the definition:

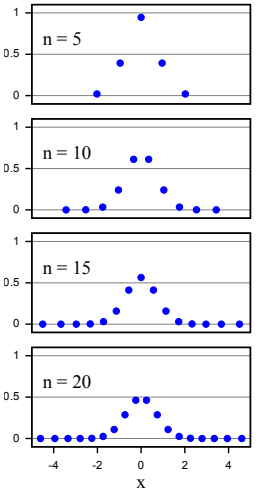
$$q(f_n|\mathbf{x}_n, \mathbf{Z}) = \int p(f_n|\mathbf{u}, \mathbf{x}_n, \mathbf{Z}) q(\mathbf{u}) \mathrm{d}\mathbf{u}.$$

- Gauss-Hermite quadrature can be applied,

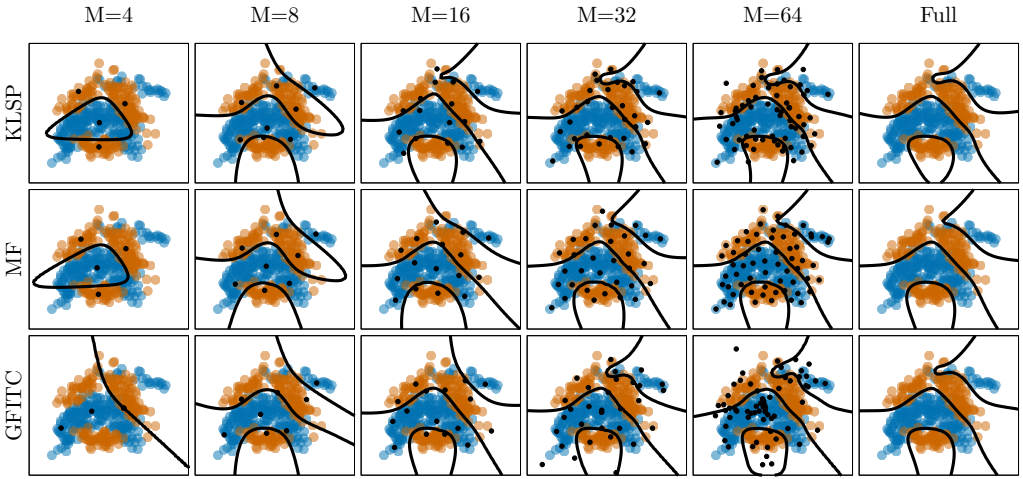
$$\langle \log p(y_n|f_n) \rangle_{q(f_n|\mathbf{x}_n, \mathbf{Z})} \approx \sum_{j=1}^{C} w_j \log p(y_n|f_j),$$

$$w_j = \frac{2^{C-1} C! \sqrt{\pi}}{C^2 [H_{C-1}(f_j)]^2}.$$

- The quadrature result is exact if $\log p(y_n|f_n)$ is a polynomial with its order less than C.



Binary Classification Example



[Hensman et al., 2015]

Beyond 1D GPs

- Multi-class classification is a common example.
- For a C-class classification, $y \in \{1, \ldots, C\}$, a GP is used to model each class,

$$\mathbf{f}_1, \ldots, \mathbf{f}_C \sim GP(0, K(\mathbf{X}, \mathbf{X})).$$

- The common likelihood is a categorical distribution with a soft-max function,

$$p(y_n | f_{n1}, \ldots, f_{nC}) = \prod_{j=1}^{C} g(f_{nj})^{\delta[y_n - j]}, \; g(f_{nj}) = \frac{e^{f_{nj}}}{\sum_{j'=1}^{C} e^{f_{nj'}}}$$

- Gauss-Hermite quadrature is not a good choice due to high dimensionality.

Monte Carlo Sampling

- Monte Carlo sampling can approximate the multi-dimensional integral:

$$\begin{aligned}\langle \log p(y_n|\mathbf{f}_n)\rangle_{q(\mathbf{f}_n|\mathbf{x}_n,\mathbf{Z})} &= \int q(\mathbf{f}_n|\mathbf{x}_n,\mathbf{Z}) \sum_{j=1}^{C} \delta[y_n - j] \log g(f_{nj}) \mathrm{d}\mathbf{f}_n \\ &\approx \frac{1}{T} \sum_{t=1}^{T} \sum_{j=1}^{C} \delta[y_n - j] \log g(f_{tnj})\end{aligned}$$

  where $\mathbf{f}_{tn} \sim q(\mathbf{f}_n|\mathbf{x}_n,\mathbf{Z})$ and $\mathbf{f}_{tn} = (f_{tn1},\ldots,f_{tnC})$.

- *Reparameterization trick* can be used to reduce of the variance of the gradient. Denote $q(f_{nj}|\mathbf{x}_{nj},\mathbf{Z}) = \mathcal{N}(m_{nj},\sigma_{nj}^2)$. A sample can be rewritten as

$$f_{tnj} = m_{nj} + \sigma_{nj}\epsilon_t,\ \epsilon_t \sim \mathcal{N}(0,1).$$

Are big covariance matrices always (almost) low-rank?

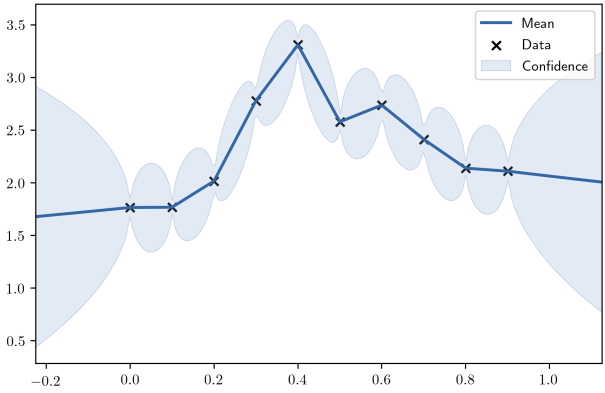
- Of course, not.
- A time series example

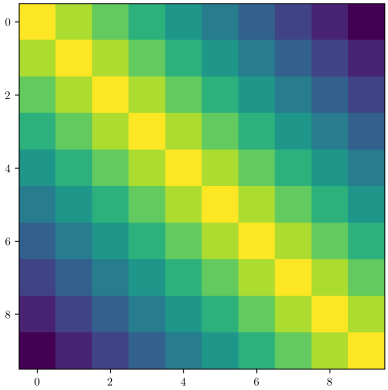
$$y = f(t) + \epsilon$$

- The data are collected with even time interval continuously.

A time series example: 10 data points

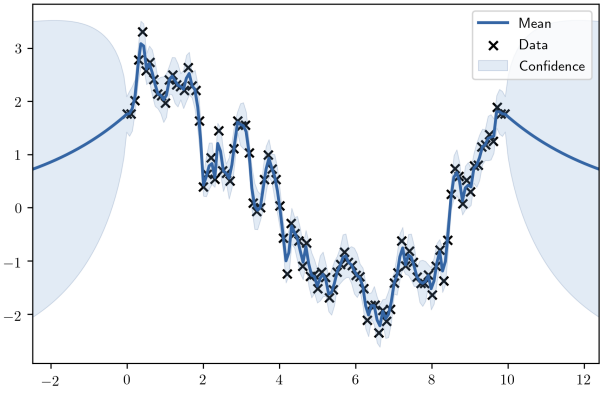
When we observe until $t = 1.0$:

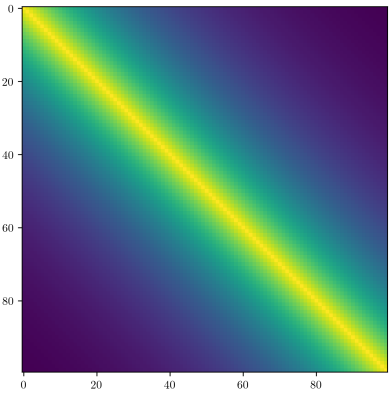




A time series example: 100 data points

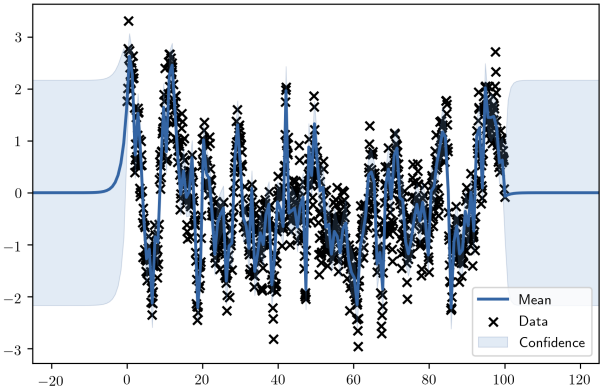
When we observe until $t = 10.0$:

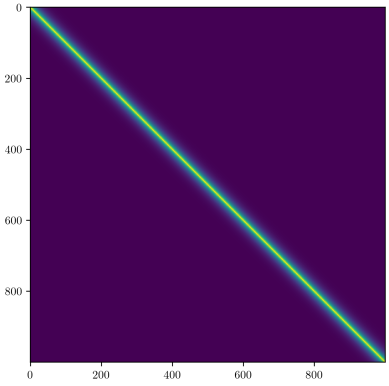




A time series example: 1000 data points

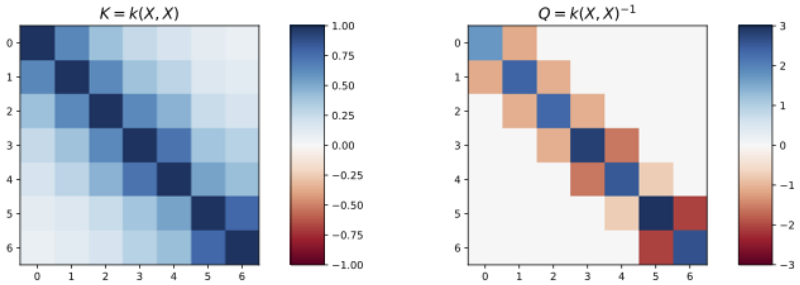
When we observe until $t = 100.0$:





Banded precision matrix

- For the kernels like the Matern family, the precision matrix is banded.
- For example, given a Matern $\frac{1}{2}$ or known as exponential kernel: $k(x, x') = \sigma^2 \exp(-\frac{|x-x'|}{l^2})$.



This slide is taken from Nicolas Durrande [Durrande et al., 2019].

Closed form precision matrix

- The precision matrix of Matern kernels can be computed in closed form.
- The lower triangular matrix from the Cholesky decomposition of the precision matrix is banded as well.

$$\log(\mathbf{y}|\mathbf{X}) = -\frac{1}{2}\log|2\pi(LL^\top)^{-1}| - \frac{1}{2}\mathsf{tr}\left(\mathbf{y}\mathbf{y}^\top LL^\top\right)$$

  where L is the lower triangular matrix from the Cholesky decomposition of the precision matrix Q, $Q = LL^\top$.
- The computational complexity becomes $O(N)$.

Other approximations

- deterministic/stochastic frequency approximation
- distributed approximation
- conjugate gradient methods for covariance matrix inversion

Q & A!

Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen. Understanding probabilistic sparse gaussian process approximations. In *Advances in Neural Information Processing Systems 29*, pages 1533–1541. 2016.

Thang D Bui, Josiah Yan, and Richard E Turner. A unifying framework for gaussian process pseudo-point approximations using power expectation propagation. *Journal of Machine Learning Research*, 18:3649–3720, 2017.

Nicolas Durrande, Vincent Adam, Lucas Bordeaux, Stefanos Eleftheriadis, and James Hensman. Banded matrix operators for gaussian markov models in the automatic differentiation era. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 2780–2789, 2019.

James Hensman, Nicolò Fusi, and Neil D. Lawrence. Gaussian processes for big data. page 282–290, 2013.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable Variational Gaussian Process Classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 351–360, 2015.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939—-1959, 2005.

Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264. 2006.

Michalis Titsias. Variational learning of inducing variables in sparse gaussian processes. In *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, pages 567–574, 2009.

Christopher K. I. Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688. 2001.