Scaling up Gaussian processes for real-world data

Zhenwen Dai

Spotify

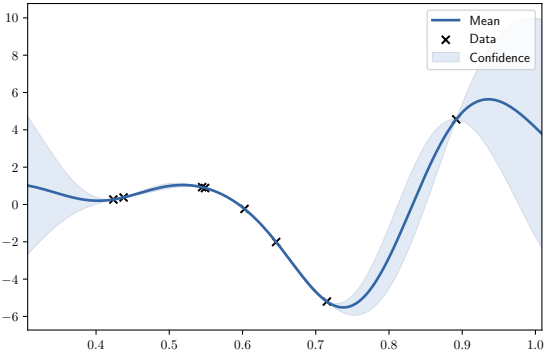
2020-3-19 @ Gaussian Processes Cambridge

Gaussian process regression

Input and Output Data:

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

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



Scale of real-world data

Why using Gaussian process / probabilistic models?

- "Big" data?
- New users and new content.
- Actively data collection and exploration.

Why using Gaussian process / probabilistic models?

- "Big" data?
- New users and new content.
- Actively data collection and exploration.

Does it mean that we are free from scalability issues?

- No, in practice, we often need to handle thousands or millions of data points.

Behind a Gaussian process fit

- Prediction on a test point given the observed data and the model parameters:

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

  where $\mathbf{K}$ is the covariance matrix,

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

  For example, the RBF kernel is $k(\mathbf{x}_i, \mathbf{x}_j) = \gamma \exp\left(-\frac{1}{2l^2}(\mathbf{x}_i - \mathbf{x}_j)^\top(\mathbf{x}_i - \mathbf{x}_j)\right)$.
- The slowest component is the inversion $(\mathbf{K}+\sigma^2\mathbf{I})^{-1}$, which is $O(N^3)$.

How to determine the parameters?

- Maximum likelihood estimate of the parameters.

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

- Posterior inference for $p(\theta|\mathbf{y}, \mathbf{X})$
  - Variational Inference: optimize a parametric distribution $q(\theta; \psi)$ such that

$$\psi^* = \arg\min_{\psi} \mathsf{KL}\left(q(\theta; \psi) \,\|\, p(\theta|\mathbf{y}, \mathbf{X})\right)$$

  - MCMC: draw samples from $p(\theta|\mathbf{y}, \mathbf{X})$:

$$\theta_1, \ldots, \theta_T \sim p(\theta|\mathbf{y}, \mathbf{X})$$

The log-likelihood implementation

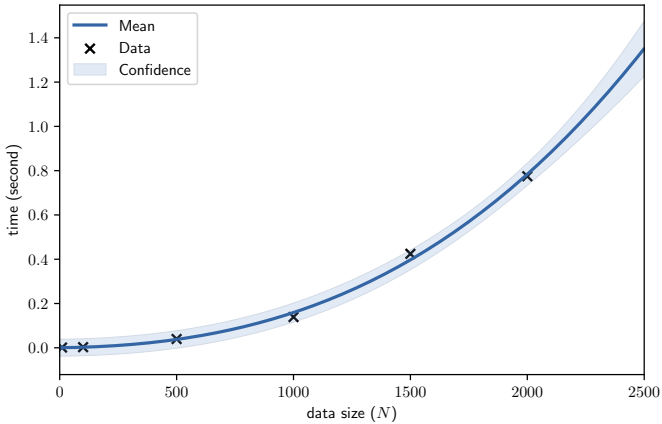
- The log-pdf of multi-variate normal distribution:

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

- 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 $O(N^3)$.

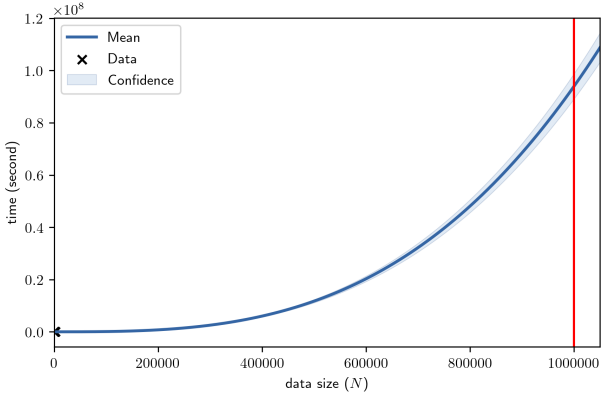
Empirical analysis of computational time

- 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.



That is only one iteration!

What about waiting for faster computers?

- Computational time $= \frac{\text{amount of work}}{\text{computer speed}}$
- 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.
- If we double the size of data, it takes 11.4 years to catch up.

What about waiting for faster computers?

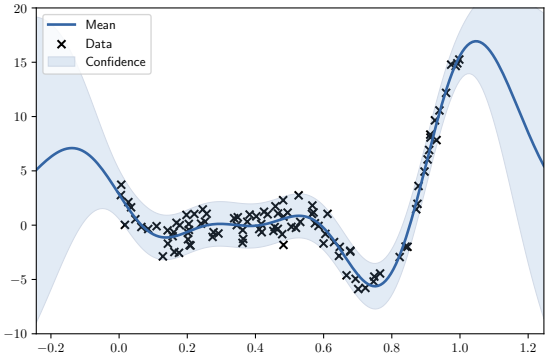
- Computational time $= \frac{\text{amount of work}}{\text{computer speed}}$
- If the computer speed increase at the pace of 20% year over year:
  - After 10 years, it will take about 176 days.
  - After 50 years, it will take about 2.9 hours.
- If we double the size of data, it takes 11.4 years to catch up.
- What about multi-core CPUs or GPU?
  It takes 8TB to store a 1M x 1M covariance matrix in momery.

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.

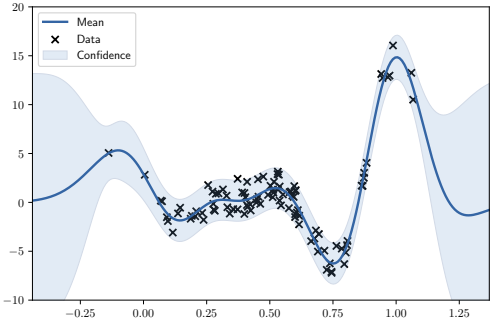
Big data (?)

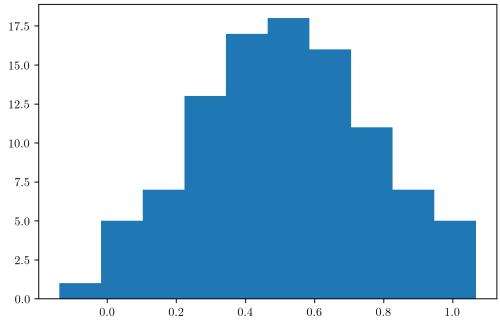
- lots of data $\neq$ complex function
- In real world problems, we often collect a lot of data for modeling relatively simple relations.



Data subsampling?

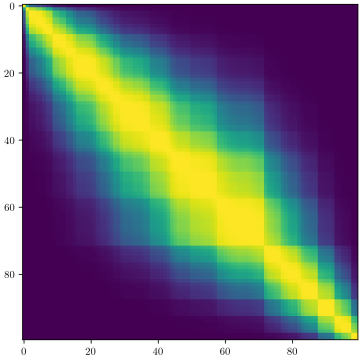
- Real data often do not evenly distributed.
- We tend to get a lot of data on common cases and very few data on rare cases.

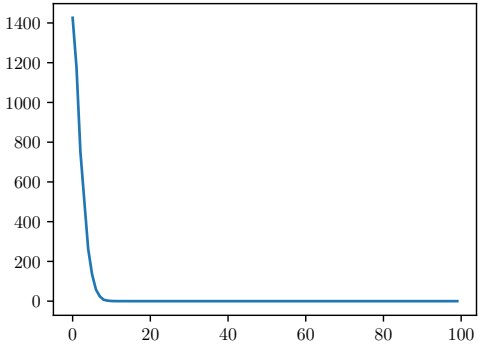




Covariance matrix of redundant data

- 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

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

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 above approach is called Nyström approximation by Williams and Seeger (2001).
- 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

When the exact model/inference is intractable, typically there are two types of approaches:

- Approximate the original model with a simpler one such that inference becomes tractable, like Nyström approximation, FITC.
- Keep the original model but derive an approximate inference method which is often *not* able to return the true answer, like variational inference.

Model Approximation vs. Approximate Inference

A problem with model approximation is that

- when an approximated model requires some tuning, e.g., for hyper-parameters, it is unclear how to improve it based on training data.
- In the case of FITC, we know the model is correct if $\mathbf{Z} = \mathbf{X}$, however, optimizing $\mathbf{Z}$ will not necessarily lead to a better location.
- In fact, optimizing $\mathbf{Z}$ can lead to overfitting. (Quiñonero-Candela and Rasmussen, 2005)

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, also known as evidence lower bound (ELBO), 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 $q(\mathbf{u})$:

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

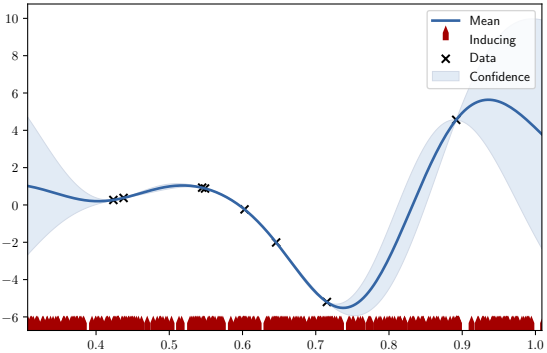
- Make the bound as tight as possible by plugging in μ^* and Σ^*:

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

- The 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.



Distributed Training & MiniBatch Training

What if the data does not fit into the memory of a single machine?

Parallel Sparse Gaussian Process

- Beyond Approximate the inference method, maybe we could exploit parallelization.
- For Gaussian process, it turns out to be very hard, because parallel Cholesky decomposition is very difficult.
- Dai et al. (2014) and Gal et al. (2014) proposes a parallel inference method for sparse GP.

Data Parallelism

- Consider a training set: $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$.
- Assume there are C computational cores/machines.
- A data parallelism algorithm divides the data set into C partitions as evenly as possible: $\mathcal{D} = \bigcup_{c=1}^{C} \mathcal{D}_c$.
- The parallelism happens in the way that the function running on each core only requiring the data from the local partition.

Data Parallelism for Sparse GP

The variational lower bound (after applying Woodbury formula) is

$$\mathcal{L} = -\frac{N}{2}\log 2\pi\sigma^2 + \frac{1}{2}\log\frac{|\mathbf{K}_{uu}|}{|\mathbf{K}_{uu}+\sigma^{-2}\mathbf{\Phi}|} - \frac{1}{2\sigma^2}\mathbf{y}^\top\mathbf{y} + \frac{1}{2\sigma^4}\mathbf{y}^\top\mathbf{K}_{fu}(\mathbf{K}_{uu}+\mathbf{\Phi})^{-1}\mathbf{K}_{fu}^\top\mathbf{y} - \frac{1}{2\sigma^2}\phi + \frac{1}{2\sigma^2}\mathsf{tr}\left(\mathbf{K}_{uu}^{-1}\mathbf{\Phi}\right)$$

where $\mathbf{\Phi} = \mathbf{K}_{fu}^\top\mathbf{K}_{fu}$ and $\phi = \mathsf{tr}\left(\mathbf{K}_{ff}\right)$.

Data Parallelism for Sparse GP

- The lower bound is not fully distributable like in the simple example.
- All the terms involving data can be written as a sum across data points:

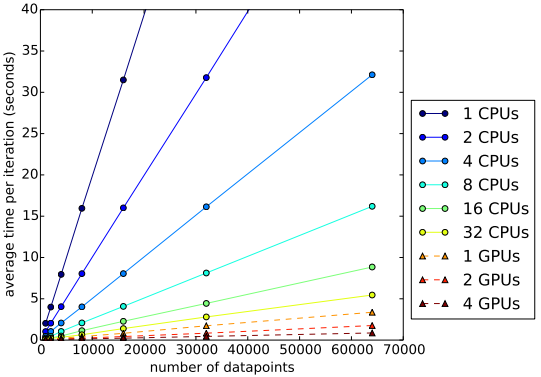
$$\mathbf{y}^\top \mathbf{y} = \sum_{n=1}^{N} y_n^2, \quad \mathbf{y}^\top \mathbf{K}_{fu} = \sum_{n=1}^{N} y_n \mathbf{K}_{f_n u}, \quad \mathbf{\Phi} = \sum_{n=1}^{N} \mathbf{K}_{f_n u}^\top \mathbf{K}_{f_n u}$$

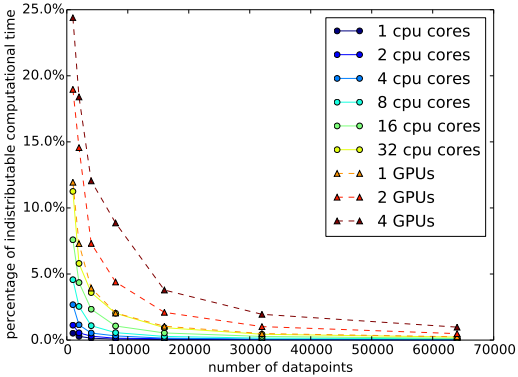
$$\phi = \sum_{n=1}^{N} \mathbf{K}_{f_n f_n}, \text{where } \mathbf{K}_{f_n u} = \mathbf{K}(\mathbf{x}_n, \mathbf{Z}), \quad \mathbf{K}_{f_n f_n} = \mathbf{K}(\mathbf{x}_n, \mathbf{x}_n).$$

Data Parallelism for Sparse GP

1. **[local]** Compute all the data related terms locally: $\mathbf{y}_c^\top \mathbf{y}_c$, $\mathbf{y}_c^\top \mathbf{K}_{f_c u}$, $\mathbf{\Phi}_c$ and ϕ_c.
2. **[global]** Aggregate all the local terms and compute the lower bound $\mathcal{L}$ on one node.
3. **[global]** Compute the gradient of the bound w.r.t. the model parameters.
4. **[global]** Compute the gradient w.r.t. the local terms $\partial \mathcal{L}/\partial \mathbf{K}_{f_c u}$, $\partial \mathcal{L}/\partial \mathbf{\Phi}_c$ and $\partial \mathcal{L}/\partial \phi_c$ and broadcast to individual nodes.
5. **[local]** Compute the gradient contribution of the local terms and aggregate the local gradients into the final gradient.
6. **[global]** Take a gradient step and repeat Step 1.

Data Parallelism for Sparse GP





Stochastic gradient descent and mini-batch training

SGD enables training with a subset of data points,

$$\theta = \arg\min_{\theta} \int f(x)p(x)\mathrm{d}x,$$

$$\theta_{t+1} = \theta_t - \eta_t \frac{1}{C} \nabla_\theta f(x_i), \quad x_i \sim p(x).$$

Applying SGD to variational sparse GP?

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

Stochastic variational Gaussian process

Hensman et al. (2013) proposed that the "uncollapsed" form of the variational lower bound can be written as such a summation:

$$\mathcal{L} = \sum_n \left\langle \log p(y_n|f_n) \right\rangle_{p(f_n|\mathbf{u},\mathbf{X},\mathbf{Z})q(\mathbf{u})} - \mathsf{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u}|\mathbf{Z})\right).$$

This allows us to have a stochastic gradient estimation by subsampling the data:

$$\nabla\tilde{\mathcal{L}} = \sum_i \left\langle \log p(y_i|f_i) \right\rangle_{p(f_i|\mathbf{u},\mathbf{X},\mathbf{Z})q(\mathbf{u})} - \frac{N}{C}\mathsf{KL}\left(q(\mathbf{u}) \,\|\, p(\mathbf{u}|\mathbf{Z})\right).$$

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

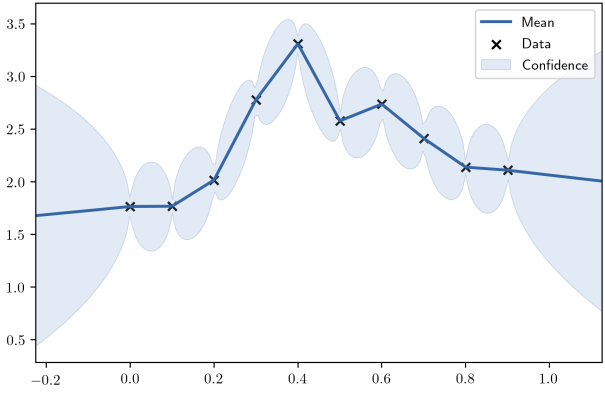
- Of course, not.
- A time series example

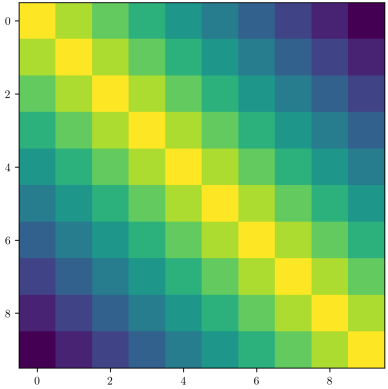
$$y = f(t) + \epsilon$$

- The data are collected with even time interval continuously.

A time series example: 10 data points

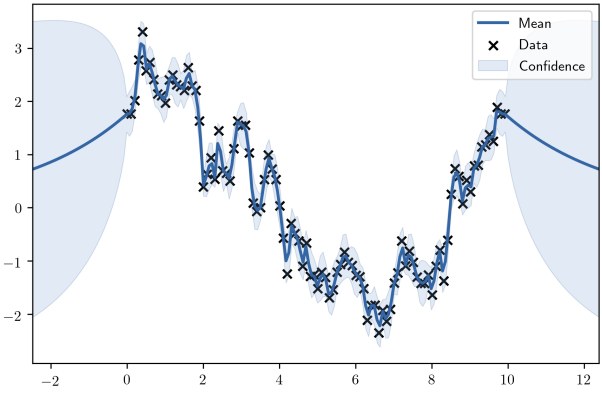
When we observe until $t = 1.0$:

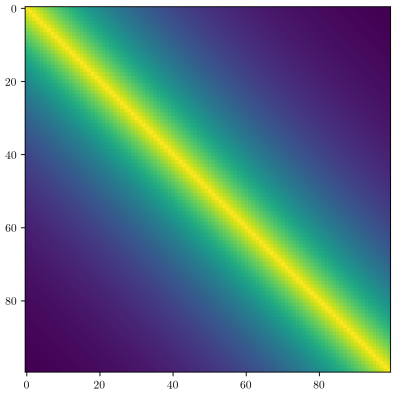




A time series example: 100 data points

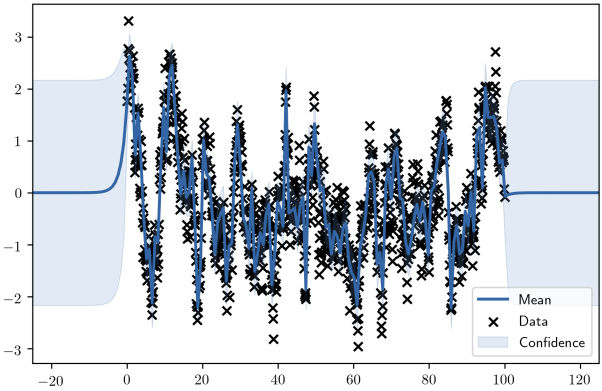
When we observe until $t = 10.0$:

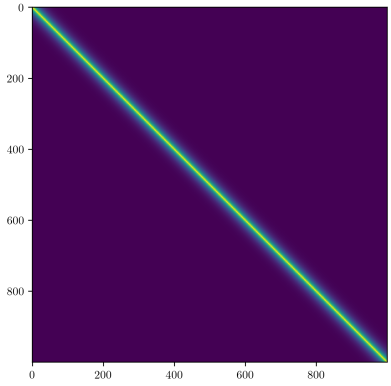




A time series example: 1000 data points

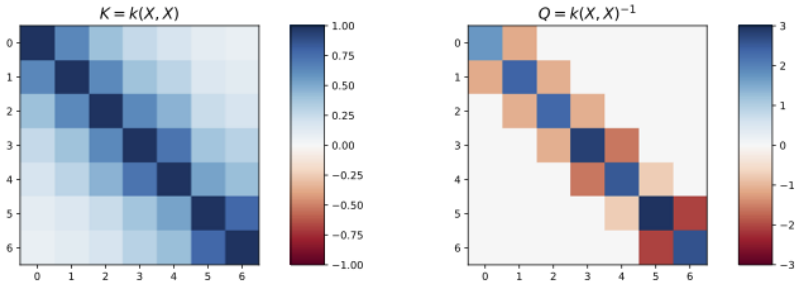
When we observe until $t = 100.0$:





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 (**?**).

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 $O(N)$.

Conjugate Gradient Methods

- The covariance matrix inversion can be formulated as $\mathbf{A}\mathbf{x} = \mathbf{b}$. This can also be solved as a quadratic problem:

$$\min_{\mathbf{x}} \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} - \mathbf{b}^\top \mathbf{x}.$$

- Conjugate Gradient (CG) is a method to iteratively solve the quadratic problem. It returns the exact solution after n iterations. The result before the Nth iteration can be used as an approximated solution.
- CG usually works well for sparse matrices.
- CG with a pre-conditioner has been used to GP inference in (Gardner et al., 2018).

Random Fourier Features

Bochner Theorem

A stationary function $k(\mathbf{x}, \mathbf{x}') = \tilde{k}(|\mathbf{x} - \mathbf{x}'|)$ is positive definite if and only if $\tilde{k}$ can be represented as $\tilde{k}(t) = \int_{\mathbb{R}} e^{i\omega t} \mathrm{d}\mu(\omega)$ where μ is a finite positive measure.

Therefore, stationary kernels can be approximated by their Fourier basis functions.

Thank you!

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

Dai, Z., Damianou, A., Hensman, J., and Lawrence, N. D. (2014). Gaussian process models with parallelization and gpu acceleration. In *NIPS workshop Software Engineering for Machine Learning*.

Gal, Y., van der Wilk, M., and Rasmussen, C. E. (2014). Distributed variational inference in sparse gaussian process regression and latent variable models. In *Advances in Neural Information Processing Systems 27*, pages 3257–3265.

Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., and Wilson, A. G. (2018). Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*.

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, page 282–290.

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate gaussian process regression. *Journal of Machine Learning Research*, 6:1939—-1959.

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

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

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