CS-E4895 Gaussian Processes
Lecture 7: Large-scale Gaussian Processes

Arno Solin

Aalto University

Monday 20.3.2023

based on slides by Zhenwen Dai, Michael Riis Andersen, Ti John, and Arno Solin

Roadmap for today

1. What is the computational issue?
2. Exploiting structure in data and/or GP prior
3. Solving the linear system approximately (with Conjugate Gradients)
4. Local approximations: Split problem into smaller chunks
5. Global approximations: Inducing points
6. Variational inference for sparse GPs
7. Recap

Section 1

What is the computational issue?

Gaussian process regression: Operations

Learning hyperparameters: Maximize (log) marginal likelihood:

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

Test point prediction (mean, variance):

$$\begin{aligned} p(f_* \mid \boldsymbol{y}) &= \mathcal{N}\left(f_* \mid \mu_*, \sigma_*^2\right) \\ \mu_* &= \boldsymbol{k}_{f_* f} \left(\boldsymbol{K}_{ff} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{y} \\ \sigma_*^2 &= k_{f_* f_*} - \boldsymbol{k}_{f_* f} \left(\boldsymbol{K}_{ff} + \sigma^2 \boldsymbol{I}\right)^{-1} \boldsymbol{k}_{f_* f}^\top \end{aligned}$$

Computational complexity of Gaussian process regression

Data set with N observations, computing posterior for 1 test point:

$$\mu_* = \boldsymbol{k}_{f_* f} \left(\boldsymbol{K}_{ff} + \sigma^2 \boldsymbol{I} \right)^{-1} \boldsymbol{y}$$
$$\sigma_*^2 = k_{f_* f_*} - \boldsymbol{k}_{f_* f} \left(\boldsymbol{K}_{ff} + \sigma^2 \boldsymbol{I} \right)^{-1} \boldsymbol{k}_{f_* f}^\top$$

What is computational complexity? What is the dominating operation?

- Matrix–vector multiplication (mvm):
  for $\boldsymbol{A} \in \mathbb{R}^{N \times M}$ and $\boldsymbol{b} \in \mathbb{R}^M$, computing $\boldsymbol{A}\boldsymbol{b}$ costs $\mathcal{O}(NM)$
- Matrix inverse: for $\boldsymbol{C} \in \mathbb{R}^{N \times N}$, computing $\boldsymbol{C}^{-1}$ costs $\mathcal{O}\left(N^3\right)$
- $\boldsymbol{\alpha} = \left(\boldsymbol{K}_{ff} + \sigma^2 \boldsymbol{I} \right)^{-1} \boldsymbol{y}$ scales as $\mathcal{O}\left(N^3\right)$, $\mu_* = \boldsymbol{k}_{f_* f} \boldsymbol{\alpha}$ scales as $\mathcal{O}(N)$

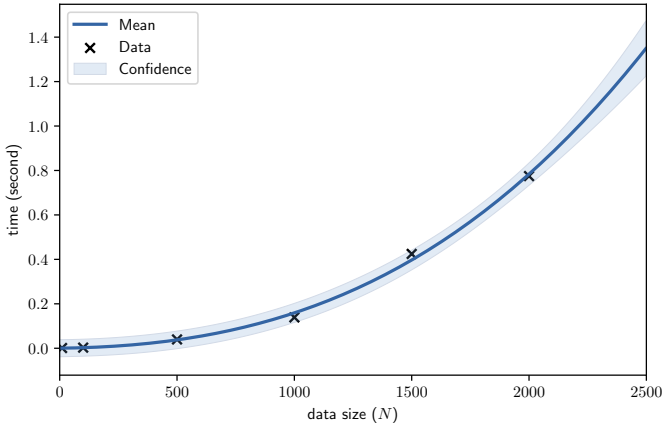
Log marginal likelihood

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

- Cholesky decomposition: $\mathbf{L} = \texttt{chol}(\mathbf{K} + \sigma^2 \mathbf{I})$, such that $\mathbf{K} + \sigma^2 \mathbf{I} = \mathbf{L}\mathbf{L}^\top$.
- $\mathcal{O}(N^3)$ computational complexity

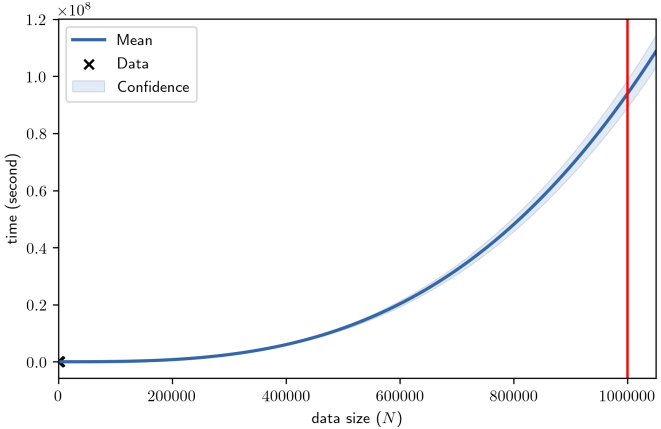
Empirical analysis of computational time

N	10	100	500	1000	1500	2000
time	1.3ms	8.5ms	28ms	0.12s	0.29s	0.76s



What if we have 1 million data points?

Predicted computational time: 9.4×10^7 seconds ≈ 2.98 years.



$N \leq 1000$: Fine, $N \leq 10000$: Slow, but possible, $N > 10000$: Prohibitively slow

Biggest bottlenecks

```
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
```

- constructing $\mathbf{K}$
- computing Cholesky of $\mathbf{K}$

How can we address the bottleneck?

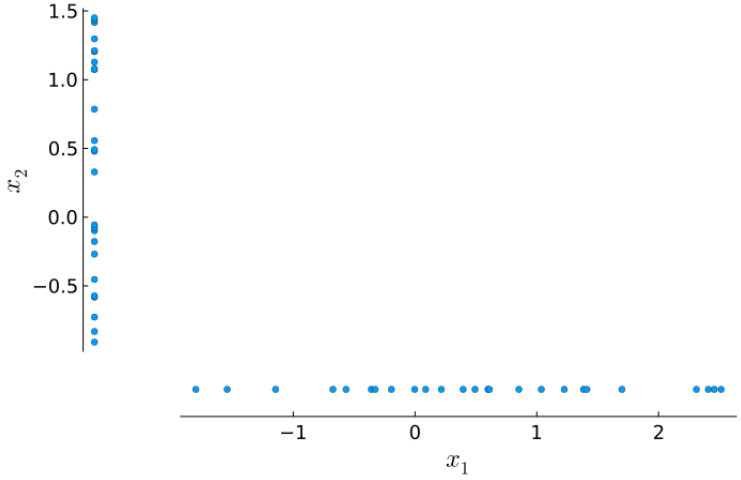
The naïve $\mathcal{O}(n^3)$ computational bottleneck ($\mathcal{O}(n^2)$ memory) can be tackled by

- **Exploiting structure in the data**
  (data on grid, inputs are in 1D, ...)
- **Exploiting structure in the GP prior**
  (GP prior is stationary, separable over input dimensions, ...)
- Solving the linear system approximately
  (conjugate-gradient solvers)
- Split problem into smaller chunks
  (local experts, subset of data, divide & conquer, ...)
- Approximate the problem
  (Nyström, low-rank, inducing points, ...)
- Approximate the problem solution
  (SVGP = sparse (and stochastic) variational methods)

Section 2

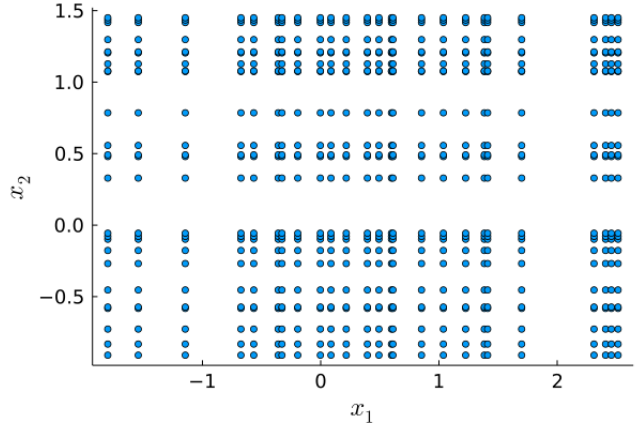
Exploiting structure in data and/or GP prior

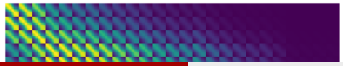
Exploiting structure: Inputs on grid



Exploiting structure: Inputs on grid

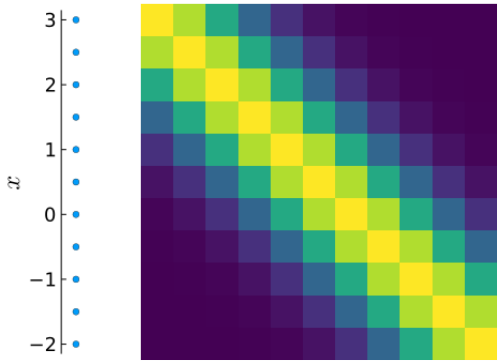
and *separable* kernel: $k(\boldsymbol{x}, \boldsymbol{x}') = \prod_{d=1}^{D} k_d(x_d, x'_d)$





Exploiting structure: Inputs on *regular* grid

and *stationary* kernel: $k(x, x') = k(x - x')$



Toeplitz structure: $K_{i,j} = K_{i+1,j+1} = k(x_i - x_j)$

efficient mvm via FFT: computation $\mathcal{O}(N + M \log M)$, storage $\mathcal{O}(N + M)$.

What if inputs are not exactly on grid?

Interpolate (e.g., cubically) kernel matrix between grid points: "KISS-GP"

- Can use very fine grids: very accurate
- Works well in small dimensions (up to 4)

What about higher dimensions?

More structure:

- Additive kernel
- Product kernel

Time series, graphs: sparse precision matrix, state space formulation

What about arbitrary kernels?

Section 3

Solving the linear system approximately (with Conjugate Gradients)

Matrix inverse as quadratic optimization

Rewrite matrix inverse

$$\boldsymbol{v} = \hat{K}^{-1}\boldsymbol{y}, \qquad \hat{K} = \mathbf{K} + \sigma^2\mathbf{I}$$

as linear system:

$$\hat{K}\boldsymbol{v} - \boldsymbol{y} = 0$$

Solve as quadratic optimization problem:

$$\boldsymbol{v}^* = \arg\min_{\boldsymbol{v}} \boldsymbol{v}^\top \hat{K}\boldsymbol{v} - \boldsymbol{v}^\top \boldsymbol{y}$$

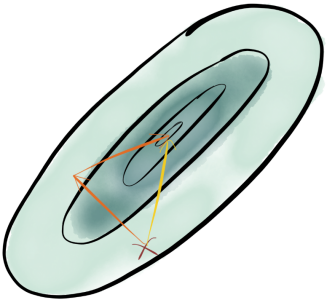
Conjugate Gradients

Efficiently solve

$$\boldsymbol{v}^* = \arg\min_{\boldsymbol{v}} \boldsymbol{v}^\top \hat{K} \boldsymbol{v} - \boldsymbol{v}^\top \boldsymbol{y}$$

using Conjugate Gradients (CG)

- Iterative method
- Each step is $\mathcal{O}(N^2)$
- Recovers exact solution after N steps $\rightarrow \mathcal{O}(N^3)$
- *Approximate* solution in much fewer steps! ("easier" problems: less steps)



Conjugate Gradient (CG)

Figure taken from Davies (2015)

Convergence and preconditioning

Condition number: ratio of largest to smallest eigenvalue,

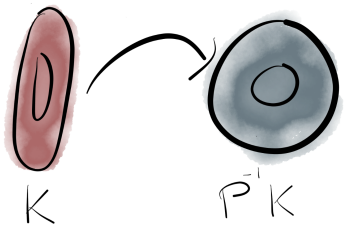
$$\kappa(\hat{K}) = \frac{\lambda_{\text{max}}(\hat{K})}{\lambda_{\text{min}}(\hat{K})}$$

High condition numbers: numerically unstable, slow convergence

Improve by preconditioning: Instead of $\hat{K}\boldsymbol{v} - \boldsymbol{y} = 0$, solve

$$P^{-1}\hat{K}\boldsymbol{v} - P^{-1}\boldsymbol{y} = 0$$

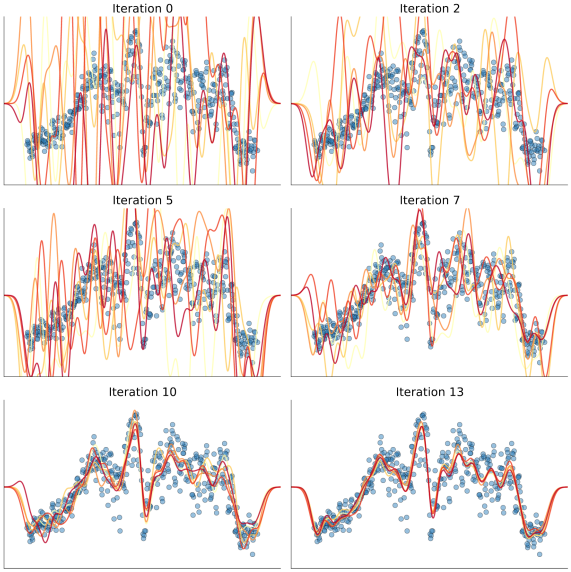
If $P^{-1} = \hat{K}^{-1}$, then $\kappa(P^{-1}\hat{K}) = 1$ and we solve in one step.



Preconditioning

Figure taken from Davies (2015)

CG example


Iteration 0
Iteration 2
Iteration 5
Iteration 7
Iteration 10
Iteration 13

Section 4

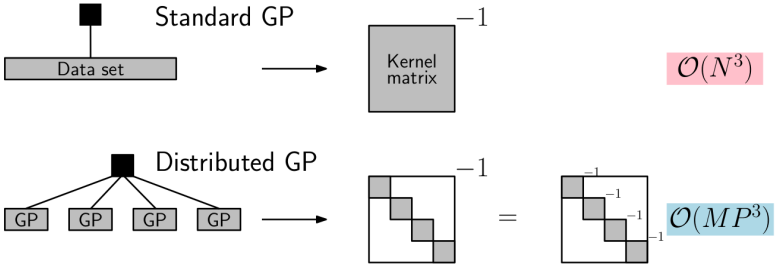
Local approximations: Split problem into smaller chunks

1. What is the computational issue?
2. Exploiting structure in data and/or GP prior
3. Solving the linear system approximately (with Conjugate Gradients)
4. Local approximations: Split problem into smaller chunks
5. Global approximations: Inducing points
6. Variational inference for sparse GPs
7. Recap

Combining "local experts"

- Split input domain/dataset (size N) into M subsets (size $P \approx N/M$)
- Fit M independent GP models (shared kernel hyperparameters): $\mathcal{O}(P^3)$ each
- Aggregate predictions
  - $+$ Mixture of experts
  - $\times$ product of experts, e.g.

$$p(f_* \,|\, \boldsymbol{x}_*, \mathcal{D}) \propto \prod_{k=1}^{M} p_k(f_* \,|\, \boldsymbol{x}_*, \mathcal{D}^{(k)})$$



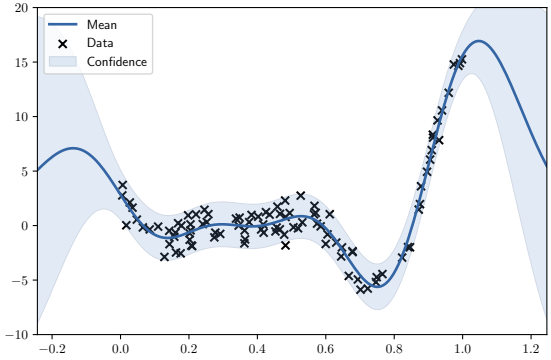
Section 5

Global approximations: Inducing points

1. What is the computational issue?
2. Exploiting structure in data and/or GP prior
3. Solving the linear system approximately (with Conjugate Gradients)
4. Local approximations: Split problem into smaller chunks
5. Global approximations: Inducing points
   - Data redundancy
   - Nyström approximation
   - Pseudo data / inducing points
6. Variational inference for sparse GPs
7. Recap

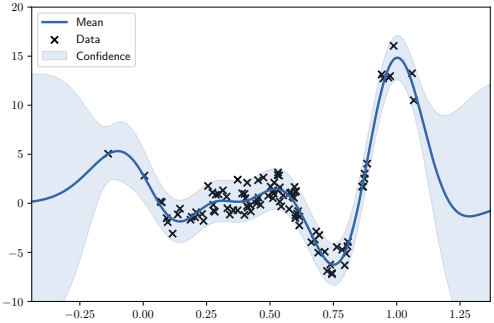
Big data (?)

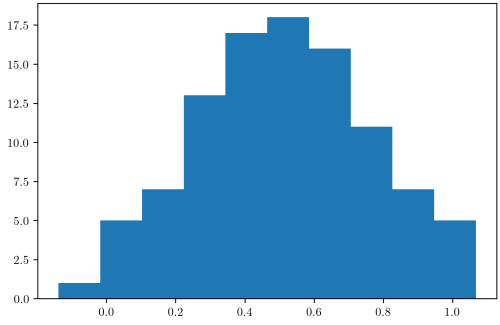
- Lots of data $\neq$ complex function
- In real world problems, we often collect a lot of data for modeling relatively simple relations.



Data subsampling?

- Real data generally not evenly distributed.
- Typically a lot of data on common cases, very few data on rare cases.





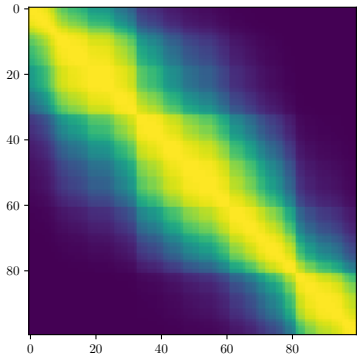
Sparse Gaussian processes

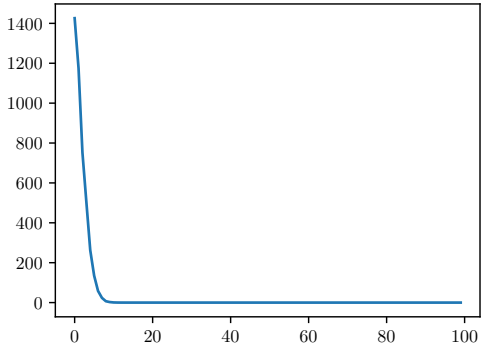
"Sparse GP" refers to various different approximations:

- Nyström approximation
- Fully independent training conditional (FITC)
- Variational sparse Gaussian process

Covariance matrix of redundant data

- With redundant data, the covariance matrix becomes low rank.





- What about low rank approximation?

1. What is the computational issue?
2. Exploiting structure in data and/or GP prior
3. Solving the linear system approximately (with Conjugate Gradients)
4. Local approximations: Split problem into smaller chunks
5. Global approximations: Inducing points
   - Data redundancy
   - Nyström approximation
   - Pseudo data / inducing points
6. Variational inference for sparse GPs
7. Recap

Low-rank approximation

- Recall GP marginal log-likelihood:

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

  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 the Nyström approximation by Williams and Seeger

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 $\tilde{\mathbf{K}}$.

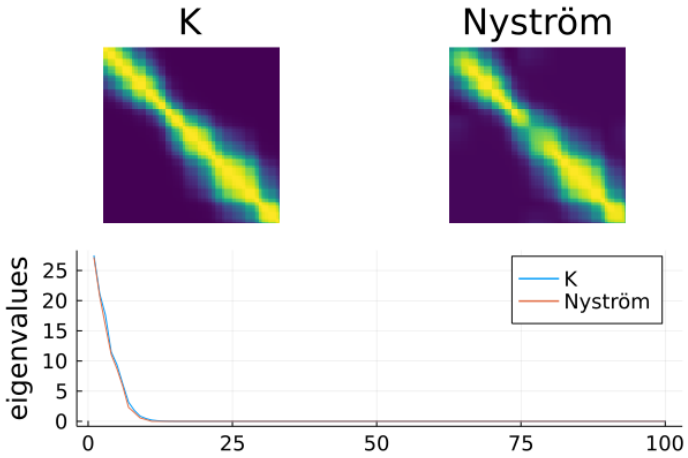
$$\tilde{\mathbf{K}} = \mathbf{K}_z \mathbf{K}_{zz}^{-1} \mathbf{K}_z^\top, \text{ where } \mathbf{K}_z = \mathbf{K}(\mathbf{X}, \mathbf{Z}) \text{ and } \mathbf{K}_{zz} = \mathbf{K}(\mathbf{Z}, \mathbf{Z}).$$

- 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

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

Nyström approximation example

Nyström approximation using 10 random data points:



Efficient computation using Woodbury formula

Naive formulation: still $\mathcal{O}(N^3)$!

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

Apply the Woodbury matrix identity:

$$\begin{aligned}(UC \quad V \quad +A)^{-1} \quad &= A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} \\ (\mathbf{K}_z\mathbf{K}_{zz}^{-1}\mathbf{K}_z^\top + \sigma^2\mathbf{I})^{-1} &= \sigma^{-2}\mathbf{I} - \sigma^{-4}\mathbf{K}_z(\mathbf{K}_{zz} + \sigma^{-2}\mathbf{K}_z^\top\mathbf{K}_z)^{-1}\mathbf{K}_z^\top\end{aligned}$$

- 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

- Approximation directly on the covariance matrix
- Randomly selected subset of data points
- Approximation becomes exact if the whole data set is taken, *i.e.*, $\mathbf{K}\mathbf{K}^{-1}\mathbf{K}^{\top} = \mathbf{K}$.

1. What is the computational issue?
2. Exploiting structure in data and/or GP prior
3. Solving the linear system approximately (with Conjugate Gradients)
4. Local approximations: Split problem into smaller chunks
5. Global approximations: Inducing points
   - Data redundancy
   - Nyström approximation
   - Pseudo data / inducing points
6. Variational inference for sparse GPs
7. Recap

Inducing point methods

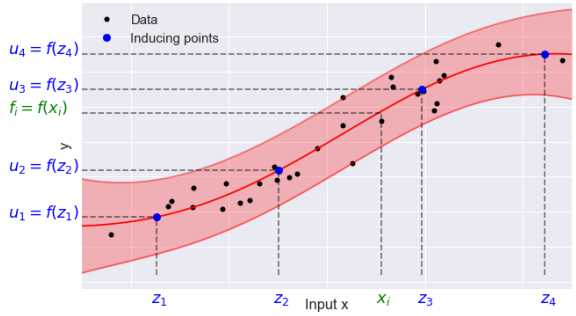
Main idea: "represent" the information from the full dataset using a smaller "virtual" dataset

- Recall our GP model:

$$p(\boldsymbol{y}, \boldsymbol{f}) = p(\boldsymbol{y} \mid \boldsymbol{f})p(\boldsymbol{f}), \quad \text{where} \quad \boldsymbol{f} = [f(\boldsymbol{x}_1), f(\boldsymbol{x}_2), \ldots, f(\boldsymbol{x}_N)]$$

- We will now introduce a set of *inducing points* $\{\boldsymbol{z}_m\}_{m=1}^M$
- They live in the same space as the input points, i.e. $\boldsymbol{x}_i, \boldsymbol{z}_j \in \mathbb{R}^D$
- Let u_m denote the value of the function f evaluated at each $\boldsymbol{z}_m$, i.e. $u_m = f(\boldsymbol{z}_m)$
- ... and $\boldsymbol{u} = [f(\boldsymbol{z}_1), f(\boldsymbol{z}_2), \ldots, f(\boldsymbol{z}_M)]$

Inducing point methods



- Goal: choose set of inducing points such that it contains same information as full dataset
- Remember: Both $u_j = f(\boldsymbol{z}_j)$ and $f_i = f(\boldsymbol{x}_i)$ are random variables
- How can we learn $\boldsymbol{u}$?
  Next step: Formulate joint model $p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u})$

Section 6

Variational inference for sparse GPs

1. What is the computational issue?
2. Exploiting structure in data and/or GP prior
3. Solving the linear system approximately (with Conjugate Gradients)
4. Local approximations: Split problem into smaller chunks
5. Global approximations: Inducing points
6. Variational inference for sparse GPs
   - Recap: Variational inference
   - Variational inference for inducing variables
   - Collapsed bound for Gaussian likelihood
   - Mini-batch learning
   - Mini-batch learning for GPs
   - Numerical considerations

Variational inference: The big picture

Recipe for approximating intractable distribution $p \in \mathcal{P}$

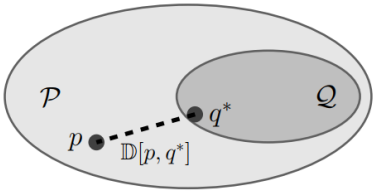
1. Define some "simple" family of distributions $\mathcal{Q}$.
2. Define some way to compute a "distance" $\mathbb{D}[p, q]$ between intractable distribution p and each distribution $q \in \mathcal{Q}$

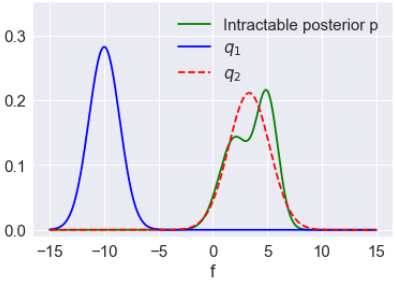
$$\mathbb{D}[p, q_1] > \mathbb{D}[p, q_2]$$

3. Search for $q \in \mathcal{Q}$ such that $\mathbb{D}[p, q]$ is minimized

$$q^* = \arg\min_{q \in \mathcal{Q}} \mathbb{D}[p, q]$$

4. Use q^* as an approximation of p





Here we will always choose $\mathcal{Q}$ to be the set of multivariate Gaussian distributions.

How to "measure distances" between distributions?

Here: *Kullback-Leibler divergence*

$$\mathbb{D}[p, q] := \mathrm{KL}[q \,\|\, p] = \int q(\boldsymbol{f}) \log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f})} \,\mathrm{d}\boldsymbol{f} = \mathbb{E}_q \left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f})} \right]$$

Important properties:

1. Non-symmetric: $\mathrm{KL}[q \,\|\, p] \neq \mathrm{KL}[p \,\|\, q]$
2. Positive: $\mathrm{KL} \geq 0$ (Gibbs' inequality)
3. Minimum: $\mathrm{KL}[q \,\|\, p] = 0 \iff q \equiv p$.

Variational inference: Minimizing $\mathrm{KL}[q(\boldsymbol{f}) \,\|\, p(\boldsymbol{f} \mid \boldsymbol{y})]$ by bounding

$$\mathrm{KL}[q(\boldsymbol{f}) \,\|\, p(\boldsymbol{f} \mid \boldsymbol{y})] = \mathrm{KL}[q(\boldsymbol{f}) \,\|\, p(\boldsymbol{f})] - \int q(\boldsymbol{f}) \big[\log p(\boldsymbol{y} \mid \boldsymbol{f})\big] \mathrm{d}\boldsymbol{f} + \log p(\boldsymbol{y})$$

$$\log p(\boldsymbol{y}) = \underbrace{\int q(\boldsymbol{f}) \big[\log p(\boldsymbol{y} \mid \boldsymbol{f})\big] \mathrm{d}\boldsymbol{f} - \mathrm{KL}[q(\boldsymbol{f}) \,\|\, p(\boldsymbol{f})]}_{\mathcal{L}[q]} + \underbrace{\mathrm{KL}[q(\boldsymbol{f}) \,\|\, p(\boldsymbol{f} \mid \boldsymbol{y})]}_{\geq 0}$$

$$\geq \int q(\boldsymbol{f}) \big[\log p(\boldsymbol{y} \mid \boldsymbol{f})\big] \mathrm{d}\boldsymbol{f} - \mathrm{KL}[q(\boldsymbol{f}) \,\|\, p(\boldsymbol{f})] = \mathcal{L}[q]$$

- $\log p(\boldsymbol{y})$ is a constant
- $\mathcal{L}[q]$ does not depend on $p(\boldsymbol{f} \mid \boldsymbol{y})$
- $\mathcal{L}[q] \leq \log p(\boldsymbol{y})$, so $\mathcal{L}[q]$ is a *lower bound* on the marginal log likelihood
- Maximizing $\mathcal{L}[q]$ is equivalent to minimizing $\mathrm{KL}[q(\boldsymbol{f}) \,\|\, p(\boldsymbol{f} \mid \boldsymbol{y})]$

Key take-away: we can fit variational approximation q by optimizing $\mathcal{L}$

Variational inference: ELBO

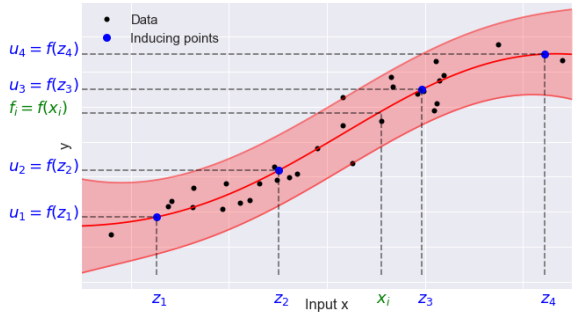
$$\log p(\boldsymbol{y}) \geq \mathcal{L}[q] = \underbrace{\int q(\boldsymbol{f}) \big[\log p(\boldsymbol{y} \mid \boldsymbol{f})\big] \mathrm{d}\boldsymbol{f}}_{\text{data fit}} - \underbrace{\mathrm{KL}[q(\boldsymbol{f}) \,\|\, p(\boldsymbol{f})]}_{\text{regularization}}$$

$\mathcal{L}[q]$ often called the *Evidence Lower Bound* (ELBO)

- To approximate $p(\boldsymbol{f} \mid \boldsymbol{y})$, use $q(\boldsymbol{f}) = \mathcal{N}(\boldsymbol{f} \mid \boldsymbol{m}, \boldsymbol{S})$
- Define $\boldsymbol{\lambda} = \{\boldsymbol{m}, \boldsymbol{S}\}$, then we can write $\mathcal{L}[q] = \mathcal{L}[\boldsymbol{\lambda}]$
- In practice, we optimize $\mathcal{L}[\boldsymbol{\lambda}]$ using gradient-based methods

1. What is the computational issue?
2. Exploiting structure in data and/or GP prior
3. Solving the linear system approximately (with Conjugate Gradients)
4. Local approximations: Split problem into smaller chunks
5. Global approximations: Inducing points
6. Variational inference for sparse GPs
   - Recap: Variational inference
   - Variational inference for inducing variables
   - Collapsed bound for Gaussian likelihood
   - Mini-batch learning
   - Mini-batch learning for GPs
   - Numerical considerations

Joint model over function values and inducing variables



- Goal: choose set of inducing points such that it contains same information as full dataset
- Remember: Both $u_j = f(\boldsymbol{z}_j)$ and $f_i = f(\boldsymbol{x}_i)$ are random variables
- Before: $p(\boldsymbol{y}, \boldsymbol{f}) = p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f})$.
  Now: joint model $p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f}, \boldsymbol{u})$

Inducing point methods: the joint model

- The augmented model

$$p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f}, \boldsymbol{u})$$

- We recover the original model by marginalizing over $\boldsymbol{u}$:

$$p(\boldsymbol{y}, \boldsymbol{f}) = \int p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f}, \boldsymbol{u}) \, \mathrm{d}\boldsymbol{u} = p(\boldsymbol{y} \mid \boldsymbol{f}) \int p(\boldsymbol{f}, \boldsymbol{u}) \, \mathrm{d}\boldsymbol{u} = p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f})$$

- Let's decompose the "augmented" model as follows

$$p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f} \mid \boldsymbol{u}) p(\boldsymbol{u})$$

Inducing point methods: the joint model II

- The joint model

$$p(\boldsymbol{y}, \boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{y} \mid \boldsymbol{f}) p(\boldsymbol{f} \mid \boldsymbol{u}) p(\boldsymbol{u})$$

- Using Gaussian conditional densities (lecture #1):

$$\begin{aligned} p(\boldsymbol{y} \mid \boldsymbol{f}) &= \mathcal{N}\left(\boldsymbol{y} \mid \boldsymbol{f}, \sigma^2 \boldsymbol{I}\right) \\ p(\boldsymbol{f} \mid \boldsymbol{u}) &= \mathcal{N}\left(\boldsymbol{f} \mid \boldsymbol{K}_{nm} \boldsymbol{K}_{mm}^{-1} \boldsymbol{u}, \tilde{\boldsymbol{K}}\right), \quad \tilde{\boldsymbol{K}} = \boldsymbol{K}_{nn} - \boldsymbol{K}_{nm} \boldsymbol{K}_{mm}^{-1} \boldsymbol{K}_{mn} \\ p(\boldsymbol{u}) &= \mathcal{N}\left(\boldsymbol{u} \mid \boldsymbol{0}, \boldsymbol{K}_{mm}\right) \end{aligned}$$

- Covariance of inducing points: $[\boldsymbol{K}_{mm}]_{ij} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)$
- Cross-covariance between inducing points and training: $[\boldsymbol{K}_{mn}]_{ij} = k(\boldsymbol{z}_i, \boldsymbol{x}_j)$
- Covariance of training points: $[\boldsymbol{K}_{nn}]_{ij} = k(\boldsymbol{z}_i, \boldsymbol{x}_j)$

Variational Sparse Gaussian Process

- Typical variational lower bound of a marginal likelihood:

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

- Let's define a *special* variational posterior (make use of structure in our problem):

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

- Plug it into the lower bound:

$$\begin{aligned}\mathcal{L} &= \int_{\mathbf{f},\mathbf{u}} p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u}) \log \frac{p(\mathbf{y} \mid \mathbf{f}) \cancel{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})} p(\mathbf{u} \mid \mathbf{Z})}{\cancel{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})} q(\mathbf{u})} \\ &= \langle \log p(\mathbf{y} \mid \mathbf{f}) \rangle_{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u})} - \mathrm{KL}[q(\mathbf{u}) \,\|\, p(\mathbf{u} \mid \mathbf{Z})]\end{aligned}$$

Likelihood term

- Remember: $p(\boldsymbol{y} \mid \boldsymbol{f}) = \prod_{i=1}^{N} p(y_i \mid f_i)$
- Let's have a closer look at the first term

$$\begin{aligned}
\mathbb{E}_{q(\boldsymbol{u},\boldsymbol{f})}\left[\log p(\boldsymbol{y} \mid \boldsymbol{f})\right] &= \mathbb{E}_{q(\boldsymbol{u},\boldsymbol{f})}\left[\log \prod_{i=1}^{N} p(y_i \mid f_i)\right] = \sum_{i=1}^{N} \mathbb{E}_{q(\boldsymbol{u},\boldsymbol{f})}\left[\log p(y_i \mid f_i)\right] \\
&= \sum_{i=1}^{N} \int\int q(\boldsymbol{u}, \boldsymbol{f}) \log p(y_i \mid f_i)\, \mathrm{d}\boldsymbol{u}\, \mathrm{d}\boldsymbol{f} \\
&= \sum_{i=1}^{N} \int\int p(f_i \mid \boldsymbol{u}) \mathcal{N}\left(\boldsymbol{u} \mid \boldsymbol{m}, \boldsymbol{S}\right) \log p(y_i \mid f_i)\, \mathrm{d}\boldsymbol{u}\, \mathrm{d}f_i \\
&= \sum_{i=1}^{N} \int\int p(f_i \mid \boldsymbol{u}) \mathcal{N}\left(\boldsymbol{u} \mid \boldsymbol{m}, \boldsymbol{S}\right)\, \mathrm{d}\boldsymbol{u} \log p(y_i \mid f_i)\, \mathrm{d}f_i
\end{aligned}$$

Decomposing the likelihood term

- Let's define the univariate distribution

$$q(f_i) \equiv \int p(f_i \mid \boldsymbol{u}) \mathcal{N}(\boldsymbol{u} \mid \boldsymbol{m}, \boldsymbol{S}) \, \mathrm{d}\boldsymbol{u} = \mathcal{N}\left(f_i \mid \boldsymbol{k}_{im} \boldsymbol{K}_{mm}^{-1} \boldsymbol{m}, \tilde{K}_{ii} + \boldsymbol{k}_{im} \boldsymbol{K}_{mm}^{-1} \boldsymbol{S} \boldsymbol{K}_{mm}^{-1} \boldsymbol{k}_{mi}\right)$$

- Variational expectations term:

$$\begin{aligned}
\mathbb{E}_{q(\boldsymbol{u}, \boldsymbol{f})}\left[\log p(\boldsymbol{y} \mid \boldsymbol{f})\right] &= \sum_{i=1}^{N} \iint p(f_i \mid \boldsymbol{u}) \mathcal{N}(\boldsymbol{u} \mid \boldsymbol{m}, \boldsymbol{S}) \, \mathrm{d}\boldsymbol{u} \log p(y_i \mid f_i) \, \mathrm{d}f_i \\
&= \sum_{i=1}^{N} \int q(f_i) \log p(y_i \mid f_i) \, \mathrm{d}f_i
\end{aligned}$$

- As in the previous lecture:
  - Decomposes into a sum over 1D integrals
  - Solved analytically for some likelihoods
  - Fast to approximate using numerical integration for others

Sparse variational posterior for **Gaussian** likelihood

- Variational posterior:

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

- Lower bound:

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

- 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 $q(\mathbf{u})$:

$$(\boldsymbol{m}^*, \mathbf{S}^*) = \arg\max_{\boldsymbol{m},\mathbf{S}} \mathcal{L}(\boldsymbol{m}, \mathbf{S}).$$

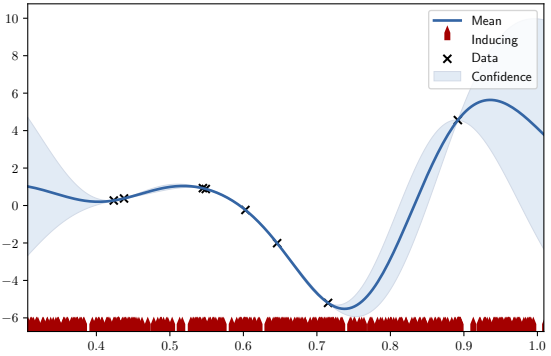
- Make the bound as tight as possible by plugging in $\boldsymbol{m}^*$ and $\mathbf{S}^*$:

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

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

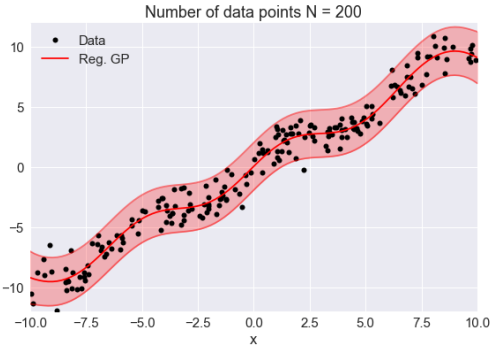
Variational sparse GP

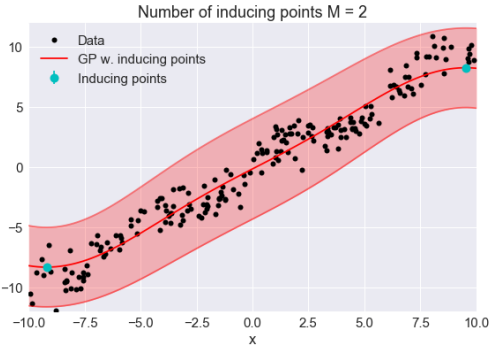
- Note that $\mathcal{L}$ is not a valid log-pdf, $\int_{\mathbf{y}} \exp(\mathcal{L}(\mathbf{y})) \leq 1$, due to the trace term.
- As inducing points are variational parameters, optimizing the inducing inputs $\mathbf{Z}$ always leads to a better bound.
- The model does not "overfit" with too many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



Example: Number of inducing points

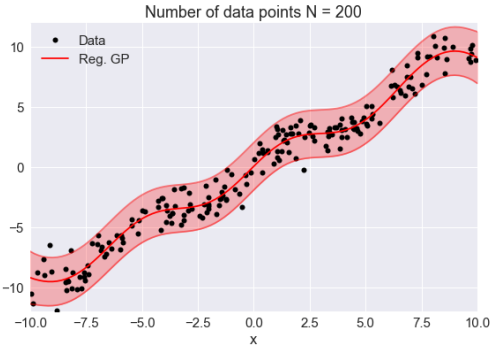
We can think of number of inducing points as parameter that trades off speed for accuracy

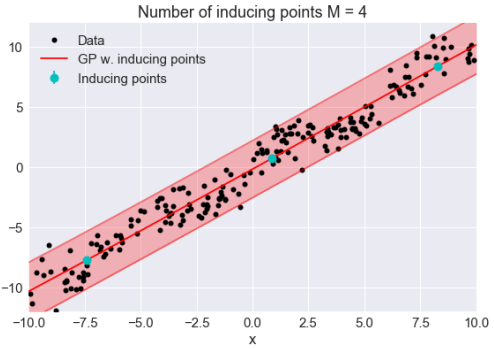




Example: Number of inducing points

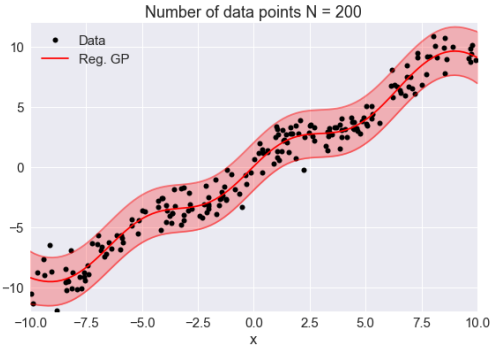
We can think of number of inducing points as parameter that trades off speed for accuracy

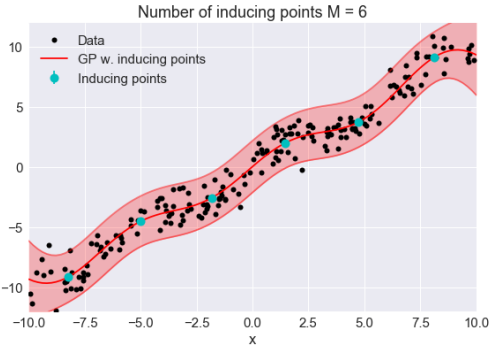




Example: Number of inducing points

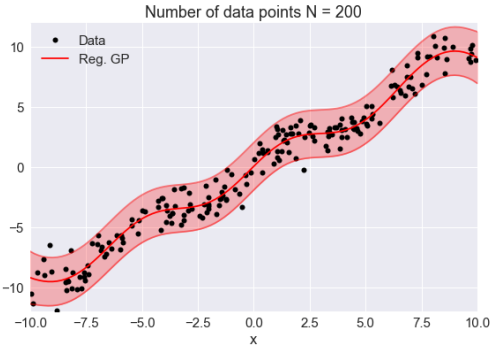
We can think of number of inducing points as parameter that trades off speed for accuracy

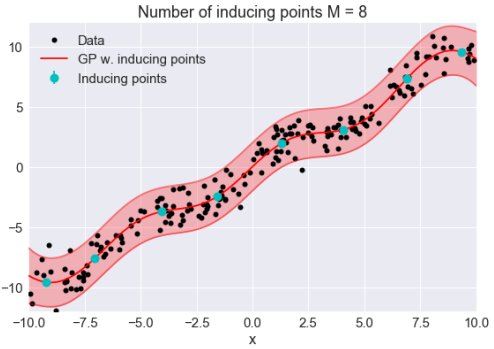




Example: Number of inducing points

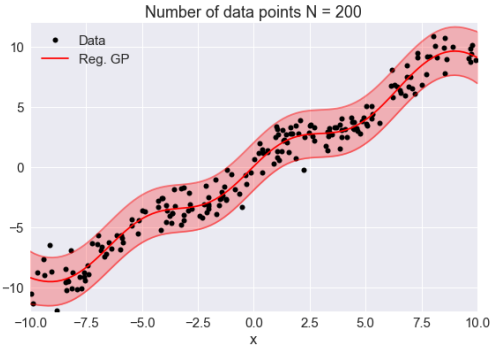
We can think of number of inducing points as parameter that trades off speed for accuracy

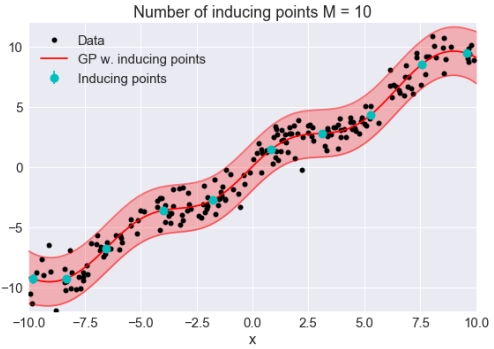




Example: Number of inducing points

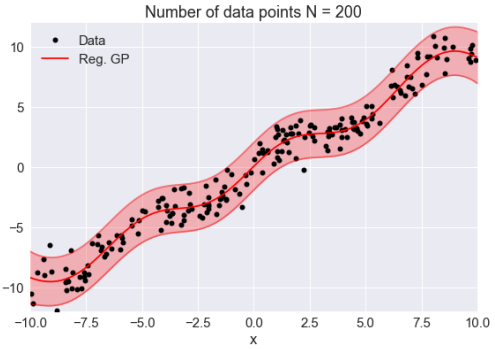
We can think of number of inducing points as parameter that trades off speed for accuracy

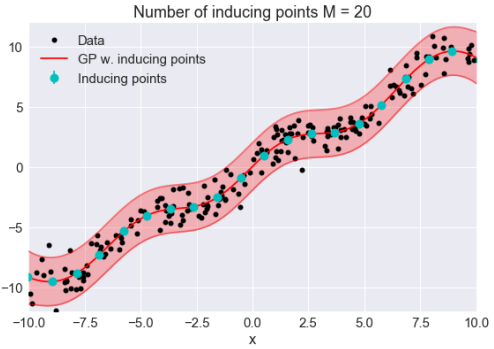




Example: Number of inducing points

We can think of number of inducing points as parameter that trades off speed for accuracy





Limitations of sparse GPs

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.

1. What is the computational issue?
2. Exploiting structure in data and/or GP prior
3. Solving the linear system approximately (with Conjugate Gradients)
4. Local approximations: Split problem into smaller chunks
5. Global approximations: Inducing points
6. Variational inference for sparse GPs
   - Recap: Variational inference
   - Variational inference for inducing variables
   - Collapsed bound for Gaussian likelihood
   - Mini-batch learning
   - Mini-batch learning for GPs
   - Numerical considerations

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,

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

- Stochastic gradient descent (SGD) updates the parameters by

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

- 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:

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

- The variational lower bound of sparse GP:

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

- Not an expectation w.r.t. the data!

"Uncollapsed" lower bound

- Let's revisit original ("uncollapsed") variational lower bound of sparse GP:

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

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

"Uncollapsed" lower bound

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

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

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

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

Stochastic Variational GP (SVGP)

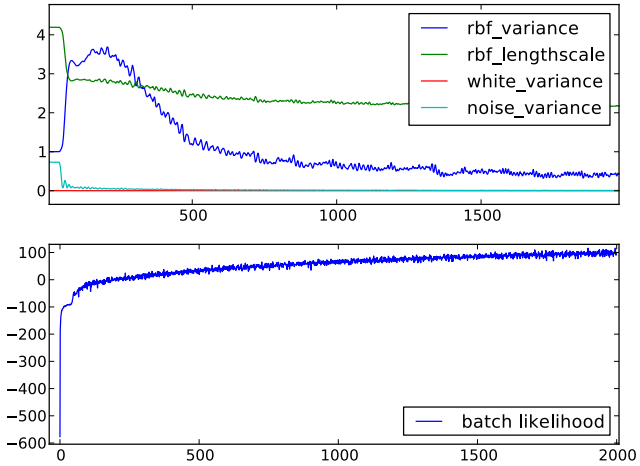
- The resulting lower bound can be written as the sum over the data,

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

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

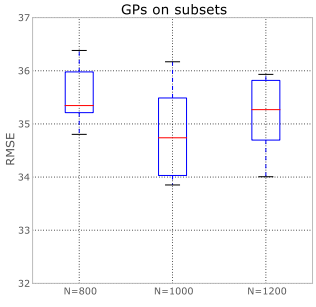
$$\theta_{t+1} = \theta_t - \eta \frac{\mathrm{d}\mathcal{L}_{\mathsf{MB}}}{\mathrm{d}\theta}.$$

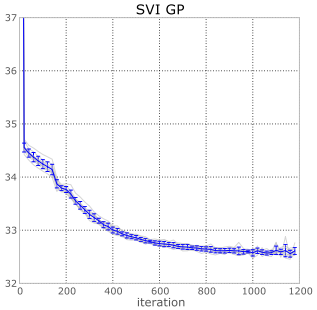
Example: 2D synthetic data

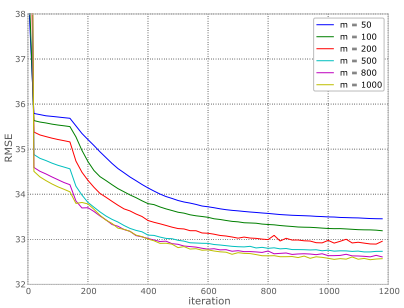


Example: Airline delay data

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







Example from the paper

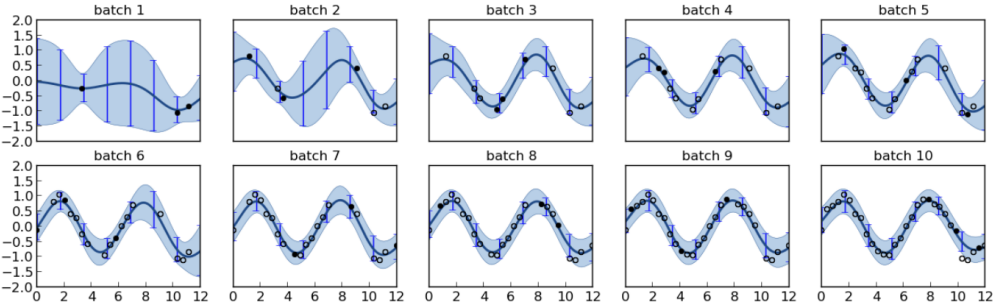


Figure 2: Stochastic variational inference on a trivial GP regression problem. Each pane shows the posterior of the GP after a batch of data, marked as solid points. Previoulsy seen (and discarded) data are marked as empty points, the distribution $q(\mathbf{u})$ is represented by vertical errorbars.

(from Hensman et al: Gaussian processes for big data)

Avoid direct inverses

- Kernel matrix may have high condition number (particularly under squared exponential kernel)

$\rightarrow$ direct inverse numerically unstable

- Instead: Cholesky and (triangular) solve
- Sparse inducing points help

Jitter

- Kernel matrix is positive definite . . . or is it?
- Might not be *numerically* positive definite (near-zero but negative eigenvalues)

→ Cholesky failures. . .

- Solution: add small diagonal *jitter* (or *nugget*) to kernel matrix (e.g., $10^{-6}\mathbf{I}$)
- Corresponds to "observing" function values under (very small) noise to make up for finite machine precision

Whitening

- Prior $p(\boldsymbol{u})$ strongly correlates the elements of $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}, \mathbf{S})$
  $\rightarrow$ Optimization over $\boldsymbol{m}$ and $\mathbf{S}$ is challenging

- **Whitening**: reparameterization
  - Define $\boldsymbol{u} = \mathbf{L}\boldsymbol{v}$(+prior mean)
    where $\mathbf{L} = \texttt{chol}(\mathbf{K})$ or $\mathbf{K} = \mathbf{L}\mathbf{L}^\top$
  - Prior $p(\boldsymbol{v}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, completely uncorrelated
  - Optimize over $q(\boldsymbol{v}) = \mathcal{N}(\boldsymbol{m}', \mathbf{S}')$ instead
  - Approximate posterior $q(\boldsymbol{u}) = \mathcal{N}(\mathbf{L}\boldsymbol{m}', \mathbf{L}\mathbf{S}'\mathbf{L}^\top)$

- (similar idea to preconditioning in CG)

How do we initialize all these parameters?

- Kernel hyperparameters: as for exact GP regression
  see `https://tinyurl.com/guide2gp` (scale your data!)
- Approximate posterior $q(\boldsymbol{u})$: at the prior $\mathcal{N}(\mathbf{0}, \mathbf{K})$
  whitening: initialize $q(\boldsymbol{v}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Inducing point locations Z:
  - Low dimensions: grid
  - (Randomly shuffled!) subset of data
  - Cluster centers (e.g., k-means)
  - Determinantal point process
- Random restarts

Inducing points method summary

- The inducing point approximation allows us to
  - ... scale Gaussian processes to big data
  - ... use non-Gaussian likelihoods
- It reduces the computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(M^3)$, where $M \ll N$
- It's implemented in most GP toolboxes, e.g. GPy (numpy) and gpflow (tensorflow)

Section 7

Recap

Recap on scaling to large data

The naïve $\mathcal{O}(n^3)$ computational bottleneck ($\mathcal{O}(n^2)$ memory) can be tackled by

- Exploiting structure in the data
  (data on grid, inputs are in 1D, ...)
- Exploiting structure in the GP prior
  (GP prior is stationary, separable over input dimensions, ...)
- Solving the linear system approximately
  (conjugate-gradient solvers)
- Split problem into smaller chunks
  (local experts, subset of data, divide & conquer, ...)
- Approximate the problem
  (Nyström, low-rank, inducing points, ...)
- Approximate the problem solution
  (SVGP = sparse (and stochastic) variational methods)

End of today's lecture

- Computational bottlenecks in exact GP regression (most methods typically apply to general likelihoods)
- The limitations we discussed on the first lecture are actually not that severe
- We will look at one more approach for speeding up GPs later in the course