Distributed Variational Inference in Sparse Gaussian Process Regression and Latent Variable Models

Yarin Gal • Mark van der Wilk • Carl E. Rasmussen

yg279@cam.ac.uk

June 25th, 2014
Outline

Gaussian process regression and latent variable models

Why do we want to scale these?

Distributed inference

Utility in scaling-up GPs

New horizons in big data
Gaussian processes (GPs) are a powerful tool for probabilistic inference over functions.

- GP regression captures non-linear functions
  - Can be seen as an infinite limit of single layer neural networks
- GP latent variable models are an unsupervised version of regression, used for manifold learning
  - Can be seen as a non-linear generalisation of PCA
GPs offer:

- uncertainty estimates,
- robustness to over-fitting,
- and principled ways for tuning hyper-parameters
GP latent variable models are used for tasks such as...

- Dimensionality reduction
- Face reconstruction
- Human pose estimation and tracking
- Matching silhouettes
- Animation deformation and segmentation
- WiFi localisation
- State-of-the-art results for face recognition
GP latent variable models are used for tasks such as...

- Dimensionality reduction
- Face reconstruction
- Human pose estimation and tracking
- Matching silhouettes
- Animation deformation and segmentation
- WiFi localisation
- State-of-the-art results for face recognition
GP latent variable models are used for tasks such as...

- Dimensionality reduction
- Face reconstruction
- Human pose estimation and tracking
- Matching silhouettes
- Animation deformation and segmentation
- WiFi localisation
- State-of-the-art results for face recognition
GP latent variable models are used for tasks such as:

- Dimensionality reduction
- Face reconstruction
- Human pose estimation and tracking
- Matching silhouettes
- Animation deformation and segmentation
- WiFi localisation
- State-of-the-art results for face recognition
GP latent variable models

GP latent variable models are used for tasks such as...

- Dimensionality reduction
- Face reconstruction
- Human pose estimation and tracking
- Matching silhouettes
- Animation deformation and segmentation
- WiFi localisation
- State-of-the-art results for face recognition
GP regression

Regression setting:

- Training dataset with $N$ inputs $X \in \mathbb{R}^{N \times Q}$ ($Q$ dimensional)
- Corresponding $D$ dimensional outputs $F_n = f(X_n)$
- We place a Gaussian process prior over the space of functions
  
  $$f \sim \mathcal{GP}(\text{mean } \mu(x), \text{covariance } k(x, x'))$$

- This implies a joint Gaussian distribution over function values:
  
  $$p(F|X) = \mathcal{N}(F; \mu(X), K), \quad K_{ij} = k(x_i, x_j)$$

- $Y$ consists of noisy observations, making the functions $F$ latent:
  
  $$p(Y|F) = \mathcal{N}(Y; F, \beta^{-1}I_n)$$
GP regression

Regression setting:

- Training dataset with \( N \) inputs \( X \in \mathbb{R}^{N \times Q} \) (\( Q \) dimensional)
- Corresponding \( D \) dimensional outputs \( F_n = f(X_n) \)
- We place a \textit{Gaussian process prior} over the space of functions
  \[ f \sim \mathcal{GP}(\text{mean } \mu(x), \text{covariance } k(x, x')) \]
- This implies a joint Gaussian distribution over function values:
  \[ p(F|X) = \mathcal{N}(F; \mu(X), K), \quad K_{ij} = k(x_i, x_j) \]
- \( Y \) consists of noisy observations, making the functions \( F \) latent:
  \[ p(Y|F) = \mathcal{N}(Y; F, \beta^{-1}I_n) \]
GP regression

Regression setting:

- Training dataset with $N$ inputs $X \in \mathbb{R}^{N \times Q}$ ($Q$ dimensional)
- Corresponding $D$ dimensional outputs $F_n = f(X_n)$
- We place a Gaussian process prior over the space of functions
  \[ f \sim \mathcal{GP}({\text{mean } \mu(x), \text{covariance } k(x, x')}) \]
- This implies a joint Gaussian distribution over function values:
  \[ p(F|X) = \mathcal{N}(F; \mu(X), K), \quad K_{ij} = k(x_i, x_j) \]
- $Y$ consists of noisy observations, making the functions $F$ latent:
  \[ p(Y|F) = \mathcal{N}(Y; F, \beta^{-1}I_n) \]
GP regression

Regression setting:

► Training dataset with $N$ inputs $X \in \mathbb{R}^{N \times Q}$ ($Q$ dimensional)

► Corresponding $D$ dimensional outputs $F_n = f(X_n)$

► We place a Gaussian process prior over the space of functions

$$f \sim \mathcal{GP}\left(\text{mean } \mu(x), \text{covariance } k(x, x')\right)$$

► This implies a joint Gaussian distribution over function values:

$$p(F|X) = \mathcal{N}(F; \mu(X), K), \quad K_{ij} = k(x_i, x_j)$$

► $Y$ consists of noisy observations, making the functions $F$ latent:

$$p(Y|F) = \mathcal{N}(Y; F, \beta^{-1} I_n)$$
GP regression

Regression setting:

- Training dataset with $N$ inputs $X \in \mathbb{R}^{N \times Q}$ ($Q$ dimensional)
- Corresponding $D$ dimensional outputs $F_n = f(X_n)$
- We place a Gaussian process prior over the space of functions $f \sim \mathcal{GP}$(mean $\mu(x)$, covariance $k(x, x')$)
- This implies a joint Gaussian distribution over function values:
  $$p(F|X) = \mathcal{N}(F; \mu(X), K), \quad K_{ij} = k(x_i, x_j)$$
- $Y$ consists of noisy observations, making the functions $F$ latent:
  $$p(Y|F) = \mathcal{N}(Y; F, \beta^{-1}I_n)$$
Latent variable models setting:

- Infer both the inputs, which are now latent, and the latent function mappings at the same time

- Model identical to regression, with a prior over now latents $X$

  $X_n \sim \mathcal{N}(X_n; 0, I), \quad F(X_n) \sim \mathcal{GP}(0, k(X, X)), \quad Y_n \sim \mathcal{N}(F_n, \beta^{-1} I)$

- In approximate inference we look for variational lower bound to:

  $$p(Y) = \int p(Y|F)p(F|X)p(X)d(F, X)$$

- This leads to Gaussian approximation to the posterior over $X$

  $$q(X) \approx p(X|Y)$$
Latent variable models setting:

- Infer both the inputs, which are now latent, and the latent function mappings at the same time

- Model identical to regression, with a prior over now latents $X$

  \[ X_n \sim \mathcal{N}(X_n; 0, I), \quad F(X_n) \sim \mathcal{GP}(0, k(X, X)), \quad Y_n \sim \mathcal{N}(F_n, \beta^{-1} I) \]

- In approximate inference we look for variational lower bound to:

  \[
  p(Y) = \int p(Y|F)p(F|X)p(X)d(F, X)
  \]

- This leads to Gaussian approximation to the posterior over $X$

  \[ q(X) \approx p(X|Y) \]
Latent variable models setting:

- Infer both the inputs, which are now latent, and the latent function mappings at the same time
- Model identical to regression, with a prior over now latents $X$

$$X_n \sim \mathcal{N}(X_n; 0, I), \quad F(X_n) \sim \mathcal{GP}(0, k(X, X)), \quad Y_n \sim \mathcal{N}(F_n, \beta^{-1} I)$$

- In approximate inference we look for variational lower bound to:

$$p(Y) = \int p(Y|F)p(F|X)p(X)d(F, X)$$

- This leads to Gaussian approximation to the posterior over $X$

$$q(X) :\approx p(X|Y)$$
Latent variable models setting:

- Infer both the inputs, which are now latent, and the latent function mappings at the same time.

- Model identical to regression, with a prior over now latents $X$

\[
X_n \sim \mathcal{N}(X_n; 0, I), \quad F(X_n) \sim \mathcal{GP}(0, k(X, X)), \quad Y_n \sim \mathcal{N}(F_n, \beta^{-1} I)
\]

- In approximate inference we look for variational lower bound to:

\[
p(Y) = \int p(Y|F)p(F|X)p(X)d(F, X)
\]

- This leads to Gaussian approximation to the posterior over $X$

\[
q(X) \approx p(X|Y)
\]
Why do we want to scale GPs?

- Naive models are often used with big data (linear regression, ridge regression, random forests, etc.)
- These don’t offer many of the desirable properties of GPs (non-linearity, robustness, uncertainty, etc.)
- Scaling GP regression and latent variable models allows for non-linear regression, density estimation, data imputation, dimensionality reduction, etc. on big datasets
Problem – time and space complexity

- Evaluating $p(Y|X)$ directly is an expensive operation
- Involves the inversion of the $n$ by $n$ matrix $K$
- requiring $O(n^3)$ time complexity
Solution – sparse approximation!

- A collection of $M$ “inducing inputs” – a set of points in the same input space with corresponding values in the output space.
- These summarise the characteristics of the function using less points than the training data.
- Given the dataset, we want to learn an optimal subset of inducing inputs.
- Requires $O(nm^2 + m^3)$ time complexity.

[Quiñonero-Candela and Rasmussen, 2005]
Solution – sparse approximation!

- A collection of $M$ “inducing inputs” – a set of points in the same input space with corresponding values in the output space.

- These summarise the characteristics of the function using less points than the training data.

- Given the dataset, we want to learn an optimal subset of inducing inputs.

- Requires $\mathcal{O}(nm^2 + m^3)$ time complexity.

[Quiñonero-Candela and Rasmussen, 2005]
Solution – sparse approximation!

- A collection of $M$ “inducing inputs” – a set of points in the same input space with corresponding values in the output space.

- These summarise the characteristics of the function using less points than the training data.

- Given the dataset, we want to learn an optimal subset of inducing inputs.

- Requires $O(nm^2 + m^3)$ time complexity.

[Quiñonero-Candela and Rasmussen, 2005]
Solution – sparse approximation!

- A collection of $M$ “inducing inputs” – a set of points in the same input space with corresponding values in the output space.

- These summarise the characteristics of the function using less points than the training data.

- Given the dataset, we want to learn an optimal subset of inducing inputs.

- Requires $O(nm^2 + m^3)$ time complexity.

[Quiñonero-Candela and Rasmussen, 2005]
Sparse approximation

Sparse approximation in pictures:

Regression on 5000 points dataset
Sparse approximation in pictures:

- We can summarise the data using a small number of points.

Regression on 500 points subset (in red)
Sparse approximation in pictures:

- We can summarise the data using a small number of points

Regression on 50 points subset (in red)
Distributed Inference in GPs
Why do we want distributed inference?

Usual datasets used with full GPs $[\mathcal{O}(n^3)]$
Why do we want distributed inference?

Usual datasets used with Sparse GPs $[\mathcal{O}(nm^2 + m^3), \ m << n]$
Why do we want distributed inference?

Big data
Why do we want distributed inference?

Distributed Sparse GPs – \( \mathcal{O}(\frac{nm^2}{T} + m^3) = \mathcal{O}(n + m^3) \),
for \( T = m^2 \) nodes, \( m << n \)
Distributing the inference

- The data points become independent of one another given the inducing inputs
- We can write the evidence lower bound as:

\[
\log p(Y) \geq \sum_{i=1}^{n} \int q(u)q(X_i)p(F_i|X_i, u) \log p(Y_i|F_i)d(F_i, X_i, u) \\
\quad - KL(q(u)||p(u)) - KL(q(X)||p(X))
\]

with inducing inputs \( u \) and approximating distributions \( q(\cdot) \)
- We can analytically integrate out \( q(u) \) and still keep a factorised form
- We can compute each term in the factorised form independently of the others with the Map-Reduce framework.
Distributing the inference

- The data points become independent of one another given the inducing inputs

- We can write the evidence lower bound as:

\[
\log p(Y) \geq \sum_{i=1}^{n} \int q(u)q(X_i)p(F_i|X_i, u) \log p(Y_i|F_i)d(F_i, X_i, u) \\
- KL(q(u)||p(u)) - KL(q(X)||p(X))
\]

with inducing inputs \( u \) and approximating distributions \( q(\cdot) \)

- We can analytically integrate out \( q(u) \) and still keep a factorised form

- We can compute each term in the factorised form independently of the others with the Map-Reduce framework.
Distributing the inference

- The data points become independent of one another given the inducing inputs

- We can write the evidence lower bound as:

\[
\log p(Y) \geq \sum_{i=1}^{n} \int q(u)q(X_i)p(F_i|X_i,u) \log p(Y_i|F_i)d(F_i,X_i,u) \\
- KL(q(u)||p(u)) - KL(q(X)||p(X))
\]

with inducing inputs \( u \) and approximating distributions \( q(\cdot) \)

- We can analytically integrate out \( q(u) \) and still keep a factorised form

- We can compute each term in the factorised form independently of the others with the Map-Reduce framework.
Distributing the inference

- The data points become independent of one another given the inducing inputs.

- We can write the evidence lower bound as:

\[
\log p(Y) \geq \sum_{i=1}^{n} \int q(u)q(X_i)p(F_i|X_i, u) \log p(Y_i|F_i)d(F_i, X_i, u) \\
- KL(q(u)||p(u)) - KL(q(X)||p(X))
\]

with inducing inputs \( u \) and approximating distributions \( q(\cdot) \).

- We can analytically integrate out \( q(u) \) and still keep a factorised form.

- We can compute each term in the factorised form independently of the others with the Map-Reduce framework.
Map-Reduce framework

[http://mohamednabeel.blogspot.co.uk/]
The inference procedure should:

- distribute the computational load evenly across nodes,
- scale favourably with the number of nodes,
- and have low overhead in the global steps.
Characteristics of distributed inference

Load balancing - 30 cores

Distribution of computational load
Characteristics of distributed inference

Time scaling with data

Suggested inference

GPy

Scalability with the number of nodes
Negligible overhead in the global steps (constant time – $O(m^3)$)
Utility in scaling-up GPs

- We want to predict flight delays from various flight-record characteristics (flight date and time, flight distance, etc.)

- Can we improve on GP prediction using increasing amounts of data?

- We use different subset sizes of data: 7K, 70K, and 700K
Utility in scaling-up GPs

<table>
<thead>
<tr>
<th>Size</th>
<th>7K</th>
<th>70K</th>
<th>700K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist GP</td>
<td>33.56</td>
<td>33.11</td>
<td>32.95</td>
</tr>
</tbody>
</table>

Root mean square error (RMSE) on flight dataset 7K-700K

- With more data we can learn better inducing inputs!

ARD parameters for flight 700K
Utility in scaling-up GPs

GP latent variable model on the full MNIST dataset (60K, 784 dim.):

▶ Used a density model for each digit

▶ No pre-processing (the model is non-specialised)

▶ Trained the models on 10K and all 60K points

<table>
<thead>
<tr>
<th>Size</th>
<th>10K</th>
<th>60K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist GP</td>
<td>8.98%</td>
<td>5.95%</td>
</tr>
</tbody>
</table>

Classification error on a subset and full MNIST

▶ Improvement of 3.03 percentage points

▶ Training on the full MNIST dataset took 20 minutes for the longest running model
Utility in scaling-up GPs

GP latent variable model on the full MNIST dataset (60K, 784 dim.):

- Used a density model for each digit
- No pre-processing (the model is non-specialised)
- Trained the models on 10K and all 60K points

<table>
<thead>
<tr>
<th>Size</th>
<th>10K</th>
<th>60K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist GP</td>
<td>8.98%</td>
<td>5.95%</td>
</tr>
</tbody>
</table>

Classification error on a subset and full MNIST:

- Improvement of 3.03 percentage points
- Training on the full MNIST dataset took 20 minutes for the longest running model
GP latent variable model on the full MNIST dataset (60K, 784 dim.):

- Used a density model for each digit
- No pre-processing (the model is non-specialised)
- Trained the models on 10K and all 60K points

<table>
<thead>
<tr>
<th>Size</th>
<th>10K</th>
<th>60K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist GP</td>
<td>8.98%</td>
<td>5.95%</td>
</tr>
</tbody>
</table>

Classification error on a subset and full MNIST

- Improvement of 3.03 percentage points
- Training on the full MNIST dataset took 20 minutes for the longest running model
GP latent variable model on the full MNIST dataset (60K, 784 dim.):

- Used a density model for each digit
- No pre-processing (the model is non-specialised)
- Trained the models on 10K and all 60K points

<table>
<thead>
<tr>
<th>Size</th>
<th>10K</th>
<th>60K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist GP</td>
<td>8.98%</td>
<td>5.95%</td>
</tr>
</tbody>
</table>

Classification error on a subset and full MNIST

- Improvement of 3.03 percentage points

- Training on the full MNIST dataset took 20 minutes for the longest running model
Utility in scaling-up GPs

GP latent variable model on the full MNIST dataset (60K, 784 dim.):

- Used a density model for each digit
- No pre-processing (the model is non-specialised)
- Trained the models on 10K and all 60K points

<table>
<thead>
<tr>
<th>Size</th>
<th>10K</th>
<th>60K</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dist GP</td>
<td>8.98%</td>
<td>5.95%</td>
</tr>
</tbody>
</table>

Classification error on a subset and full MNIST

- Improvement of 3.03 percentage points
- Training on the full MNIST dataset took 20 minutes for the longest running model
New horizons in big data

But these models give us much more...

- The MNIST trained models are density estimation models
- They allow us to perform image imputation,
- Generate new digits by sampling from the posterior, etc.
Furthermore, real big data is complex and non-linear – and naive models may under-perform on it

- Back to flight regression –

- Flight 2M dataset compared to common approaches in big data:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mean</th>
<th>Linear</th>
<th>Ridge</th>
<th>RF</th>
<th>Dist GP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight 2M</td>
<td>38.92</td>
<td>37.65</td>
<td>37.65</td>
<td>37.33</td>
<td>35.31</td>
</tr>
</tbody>
</table>

RMSE of regression over flight data with 2M points

- These are just error rates – we can do much more with GPs
  - robust, offer uncertainty bounds, etc.
New horizons in big data

Furthermore, real big data is complex and non-linear – and naive models may under-perform on it

- Back to flight regression –
- Flight 2M dataset compared to common approaches in big data:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mean</th>
<th>Linear</th>
<th>Ridge</th>
<th>RF</th>
<th>Dist GP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight 2M</td>
<td>38.92</td>
<td>37.65</td>
<td>37.65</td>
<td>37.33</td>
<td>35.31</td>
</tr>
</tbody>
</table>

RMSE of regression over flight data with 2M points

- These are just error rates – we can do much more with GPs
  - robust, offer uncertainty bounds, etc.
New horizons in big data

Furthermore, real big data is complex and non-linear – and naive models may under-perform on it

➤ Back to flight regression –

➤ Flight 2M dataset compared to common approaches in big data:

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Mean</th>
<th>Linear</th>
<th>Ridge</th>
<th>RF</th>
<th>Dist GP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flight 2M</td>
<td>38.92</td>
<td>37.65</td>
<td>37.65</td>
<td>37.33</td>
<td>35.31</td>
</tr>
</tbody>
</table>

RMSE of regression over flight data with 2M points

➤ These are just error rates – we can do much more with GPs
  ➤ robust, offer uncertainty bounds, etc.
Conclusions

- We showed that the inference scales well with data and computational resources
- We demonstrated the utility in scaling GPs to big data
- The results show that GPs perform better than many common models often used for big data
Conclusions

- Developing the inference we wrote an introductory tutorial [Gal and van der Wilk, 2014] with detailed derivations

- The code developed is open source\(^1\)
  - 300 lines of Python with detailed and documented examples

- Pointers between equations in the tutorial and in code

\[^1\text{See } \url{https://github.com/markvdw/GParML}\]