Training Deep Gaussian Processes using Stochastic Expectation Propagation and Probabilistic Backpropagation

Thang D. Bui, José Miguel Hernández-Lobato, Yingzhen Li, Daniel Hernández-Lobato and Richard E. Turner

Deep Gaussian processes

Generative model:
\[ f_i \sim GP(0, k_{i,i}) \]
\[ h_{i,n} = f_i(f_{i-1}(...f_1(x_n)))) \]
\[ y_n = g(h_{i,n}) = f_i(f_{i-1}(...f_1(x_n))) + \epsilon_n \]

Deep GPs are:
+ multi-layer generalisation of Gaussian processes
+ equivalent to deep neural networks with infinitely wide hidden layers

Advantages:
+ Deep GPs are deep and nonparametric and can,
+ discover useful input warping or dimensionality compression and expansion
→ automatic, nonparametric Bayesian kernel design
+ give a non-Gaussian functional mapping
→ automatic, nonparametric Bayesian kernel design
+ repair the damage done by using sparse approximations,
+ retain uncertainty over latent mappings and representations.

Open theoretical questions:
+ architecture: number of layers, hidden dimensions, covariance functions,
+ learnability/identifiability/prior knowledge,
+ efficient inference and learning.

Generative model:
\[ p(u_i) = N(u_i; 0, K_{u,i}) \]
\[ p(h_i | u_i, h_{i-1}) = N(h_i; A(h_{i-1})u_i, B(h_{i-1}) + \sigma_i^2) \]

where:
\[ A(h_{i-1}) = K_{h_{i-1}; h_{i-1}}^{-1} K_{h_{i-1}; u_i} \]
\[ B(h_{i-1}) = K_{h_{i-1}; h_{i-1}} - K_{h_{i-1}; K_{h_{i-1}; u_i}} K_{h_{i-1}; u_i}^{-1} K_{h_{i-1}; h_{i-1}} \]

Approximate inference using stochastic Expectation Propagation:

1. Sparsify the model using the FITC approximation:
Generative model:
\[ p(u_i) = N(u_i; 0, K_{u,i}) \]
\[ p(h_i | u_i, h_{i-1}) = N(h_i; A(h_{i-1})u_i, B(h_{i-1}) + \sigma_i^2) \]

where:
\[ A(h_{i-1}) = K_{h_{i-1}; h_{i-1}}^{-1} K_{h_{i-1}; u_i} \]
\[ B(h_{i-1}) = K_{h_{i-1}; h_{i-1}} - K_{h_{i-1}; K_{h_{i-1}; u_i}} K_{h_{i-1}; u_i}^{-1} K_{h_{i-1}; h_{i-1}} \]

2. Approximate inference using stochastic Expectation Propagation:

Approx. posterior
\[ q(\theta) \propto p(\theta) \prod_i g_i(\theta) \]

Deletion
\[ q_\theta(\theta) \propto q(\theta) g_i(\theta) \]

Incorporating data
\[ q_\theta(\theta) \propto q_\theta(\theta)p(y_i | \theta) \]

Moment-matching
\[ KL(q_\theta(\theta) | q(\theta)) \rightarrow g_i(\theta) \]

Inclusion
\[ q_\theta(\theta) \propto q_\theta(\theta)g_\theta(\theta) \]

Update
\[ q_\theta(\theta) \rightarrow q_\theta(\theta) \rightarrow g_\theta(\theta) \]

Memory complexity
\[ O(NLM^2) \]

3. Approx. moment-matching using Probabilistic Backpropagation:

Shortcut for the moment matching step:
\[ m = m ^ \dagger \]
\[ V = V ^ \dagger \]
\[ V ^ \dagger \rightarrow V ^ \dagger \]

where:
\[ Z = f_{u_i : L} p(y | x, u_i : L) g_i(q_i(u_i : L)) \]

We compute Z and its gradients using the probabilistic backpropagation algorithm, which propagates a moment-matched Gaussian through the network, then computes the gradients using chain rule in the backward step.

4. Hyperparameter optimisation using stochastic gradients:
+ Optimise the EP energy, but do not wait for EP inner loop to converge
+ Use the median trick and ADF for initialisation
+ Use Theano to compute the gradients of Z
+ Use minibatch-based stochastic optimisation, we use Adam

Comparison to other methods:
- FITC-DGP-2
- FITC-DGP-1
- FITC-GP
- BNN-Dropout
- BNN-PBP
- BNN-VI

We compared two different variants of Deep GPs with GPs and Bayesian neural networks on several regression tasks. Deep GPs using our proposed inference technique outperforms other models/methods.

Experimental results

<table>
<thead>
<tr>
<th>Dataset</th>
<th>N</th>
<th>D</th>
<th>GP, 50</th>
<th>DGP, 1.5</th>
<th>DGP, 2.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boston</td>
<td>506</td>
<td>13</td>
<td>3.09 ± 0.63</td>
<td>2.85 ± 0.65</td>
<td>2.47 ± 0.49</td>
</tr>
<tr>
<td>Concrete</td>
<td>1000</td>
<td>8</td>
<td>5.24 ± 0.55</td>
<td>5.91 ± 1.85</td>
<td>5.21 ± 0.90</td>
</tr>
<tr>
<td>Energy 1</td>
<td>768</td>
<td>8</td>
<td>0.50 ± 0.10</td>
<td>0.77 ± 0.59</td>
<td>0.48 ± 0.05</td>
</tr>
<tr>
<td>Energy 2</td>
<td>768</td>
<td>8</td>
<td>1.60 ± 0.13</td>
<td>1.78 ± 0.43</td>
<td>1.37 ± 0.23</td>
</tr>
<tr>
<td>Kinship</td>
<td>8192</td>
<td>8</td>
<td>0.04 ± 0.00</td>
<td>0.07 ± 0.04</td>
<td>0.02 ± 0.00</td>
</tr>
<tr>
<td>Naval 1</td>
<td>11934</td>
<td>16</td>
<td>0.02 ± 0.01</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>Naval 2</td>
<td>11934</td>
<td>16</td>
<td>0.01 ± 0.00</td>
<td>0.00 ± 0.00</td>
<td>0.00 ± 0.00</td>
</tr>
<tr>
<td>Power</td>
<td>9568</td>
<td>4</td>
<td>3.19 ± 0.18</td>
<td>3.35 ± 0.21</td>
<td>2.95 ± 0.30</td>
</tr>
<tr>
<td>Red wine</td>
<td>1588</td>
<td>11</td>
<td>0.48 ± 0.06</td>
<td>0.62 ± 0.05</td>
<td>0.54 ± 0.11</td>
</tr>
<tr>
<td>White wine</td>
<td>4898</td>
<td>11</td>
<td>0.37 ± 0.04</td>
<td>0.49 ± 0.09</td>
<td>0.34 ± 0.07</td>
</tr>
<tr>
<td>Wine</td>
<td>2066</td>
<td>31</td>
<td>95.87 ± 18.03</td>
<td>74.86 ± 13.66</td>
<td>70.58 ± 15.58</td>
</tr>
</tbody>
</table>

Summary and future work

Our work proposes an approximate inference scheme for Deep GPs, that
+ extends probabilistic backpropagation for Bayesian neural networks
+ combines inducing point based sparse GP approximation with the memory efficient Stochastic Expectation Propagation
+ is fast and easy to implement
+ obtains state of the art regression results.

Current work includes:
+ parallel implementation
+ large scale experiment on big datasets
+ comparison to variational free-energy schemes
+ extending to classification and latent variable models
+ investigating various network architectures.

For more, see http://arxiv.org/abs/1511.03405