Training Deep Gaussian Processes with Sampling

Keyon Vafa
Columbia University; keyon.vafa@columbia.edu

Summary

- Deep Gaussian Proceses (deep GPs) are the composition of Gaussian Processes (GPs).
- We propose a stochastic gradient training scheme that uses sampling and pseudo data.
- Experiments show that deep GPs are well-suited to fit non-stationary functions.

Deep Gaussian Processes

A **deep Gaussian Process** (deep GP) is the composition of Gaussian Processes (GPs):

$$\mathbf{f}^{(1:L)}(\mathbf{x}) = \mathbf{f}^{(L)}(\mathbf{f}^{(L-1)}(\dots \mathbf{f}^{(2)}(\mathbf{f}^{(1)}(\mathbf{x}))\dots))$$

$$\text{where } f_d^{(l)} \sim \mathcal{GP}\left(0, k_d^{(l)}(\mathbf{x}, \mathbf{x}')\right) \text{ for } f_d^{(l)} \in \mathbf{f}^{(l)}.$$

Qualities of a deep GP:

- Standard GPs require non-stationary kernels to model non-stationary functions.
- Deep GPs compose various kernels, can thus capture non-stationarity.
- They can be viewed as deep neural networks with alternating infinite-dimensional hidden layers.

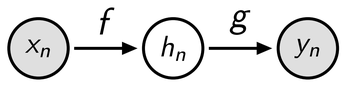

Figure 1: A simple two-layer deep GP where every layer has one dimension.

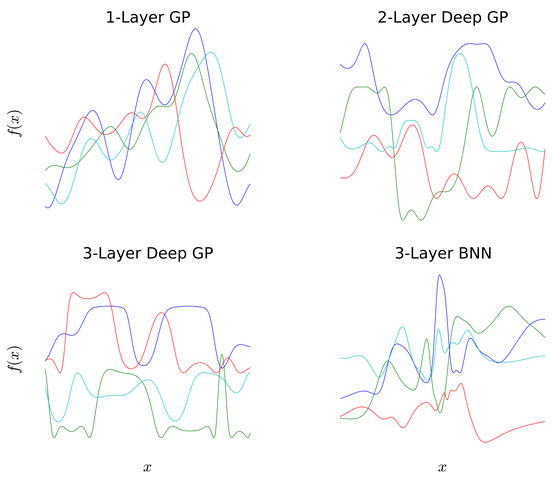

Figure 2: Draws from a GP, two deep GPs, and a BNN.

Inference

It is intractable to evaluate $P(\mathbf{y}|\mathbf{X}, \theta)$, so we propose a stochastic gradient descent approximate algorithm, which uses two central ideas:

- We **sample** predictive means and covariances to approximate the marginal likelihood, and use automatic differentiation techniques and the reparameterization trick to evaluate gradients.
- We use the **Fully Independent Training Conditional (FITC) approximation**, which incorporates training samples that may not be in the original data (pseudo data), at each layer.

As an example, for the 2-layer deep GP in Figure **??**, we would like to learn:

- The pseudo data for each layer.
- The kernel parameters for f and g.

To evaluate the log-likelihood at our current parameter estimates,

1. Sample values from the hidden layer, $\mathbf{H}$, given our inputs, $\mathbf{X}$, using the FITC approximation.
2. Use each sample $\tilde{\mathbf{H}}^j$ to approximate the FITC GP log-likelihood of the outputs, $\mathbf{y}$, at the last level.
3. Use automatic differentiation and the reparameterization trick to evaluate gradients.

Advantages of this method:

- It allows for automatic differentiation, giving it a "black-box" nature.
- Unlike existing methods, it can extend to arbitrary kernels.
- It is intuitive to understand and implement.

Experiments

For a noisy step function, we train various GP architectures and compare them to a Bayesian Neural Network (BNN).

- Deep GPs capture non-stationarity by composing two GPs with varying kernels.

Experiments (continued)

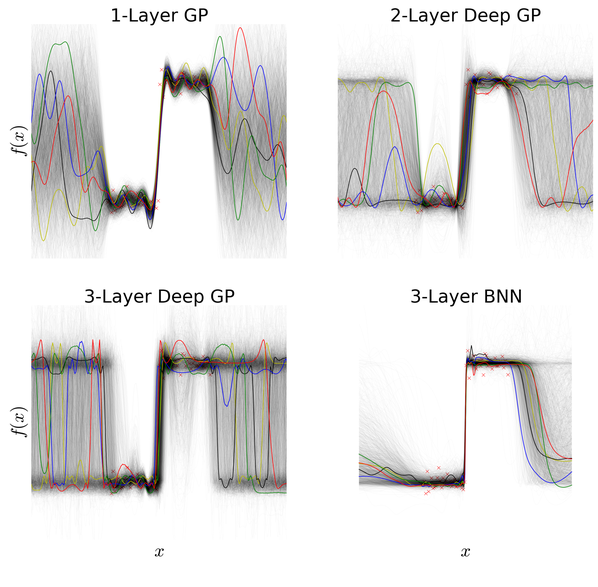

Figure 3: The red 'x's denote the data points, and the black curves denote 2000 posterior draws for each model, 5 of which are randomly colored for emphasis.

- The uncertainty outside the input space for deep GPs better captures our true beliefs than the BNN.

We run an experiment comparing prediction quality on noisy step function for 1-, 2-, and 3-layer deep GPs (where each hidden layer has one unit) with varying sizes of our data set.

- Occasionally deeper models yield poor predictive performance.
- Overfitting not an issue. Instead, optimization becomes more difficult for deeper models, as they are prone to poor local optima.
- We performed 10 random restarts for each setting, choosing the trial with the largest train log-likelihood.

Number of Data Points	1 Layer	2 Layers	3 Layers
50	.44	−.03	**1.27**
100	.23	**.54**	−.36
200	−.11	.71	**.80**

Figure 4: Test log-likelihood per data in the step function experiment.

Experiments (continued)

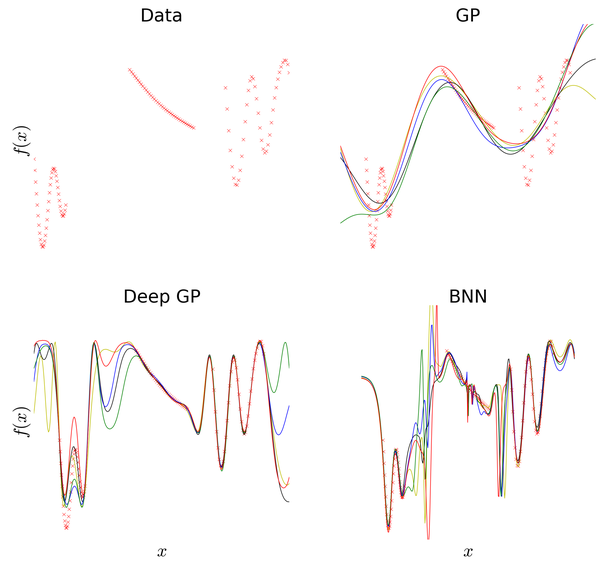

Figure 5: Predictive draws for toy non-stationary data.

Additionally, we create toy non-stationary data and fit it with various architectures. In Figure **??**, we see

- Single-layer GP can only recover a single length scale, so the curvature is constant.
- Deep GP and BNN yield predictive draws that are highly-varying in the outside regions, and mostly flat in the intermediate region.
- BNN draws and uncertainty are spikier than deep GP.

Conclusion

- Deep GPs combine the deep architecture of BNNs with the probabilistic properties of standard GPs.
- While our proposed algorithm is easy to implement and can extend to arbitrary kernels, it is difficult to optimize.
- When deeper models successfully train, they achieve a higher predictive performance than shallower models.