# Why Doubly Stochastic Variational Inference ??

Faezeh Yazdi

October 18, 2019

# What Problem?

- A set of $N$ obervations $\mathbf{y} = (y_1, ..., y_N)^T$ where $y_i \in \mathbb{R}$
- At $N$ design locations $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N)^T$ where $\mathbf{x}_i \in \mathbb{R}^d$
- $f : \mathbb{R}^d \to \mathbb{R}$ is responsible for generating $\mathbf{y}$ given $\mathbf{X}$

How to approximate $\hat{f}(\mathbf{x}^*)$ at new location $\mathbf{x}^*$ ??

- Fitting a GP/ a DGP
- Computational complexity is $O(N^3)$

A problem for GPs and even more in DGPs (MORE Data needed)

# How complex DGP Models?

- Architecture of a DGP model with $L$ layers corresponds to this graphical model :

$$\mathbf{X} \xrightarrow{\mathbf{f}^1} (\mathbf{F}^1 + \epsilon_1) \xrightarrow{\mathbf{f}^2} \ldots \xrightarrow{\mathbf{f}^{L-1}} (\mathbf{F}^{L-1} + \epsilon_{L-1}) \xrightarrow{\mathbf{f}^L} (\mathbf{F}^L + \epsilon_L) = \mathbf{y}$$

- A nested structure of GPs considering the relationship between the inputs and the final output as a functional composition of GPs :

$$\mathbf{y} = \mathbf{f}^L(\mathbf{f}^{L-1}(\ldots\mathbf{f}^2((\mathbf{f}^1(\mathbf{X}) + \epsilon_1)) + \epsilon_2\ldots) + \epsilon_{L-1}) + \epsilon_L$$

- Latent nodes $(\mathbf{F}^l + \epsilon_l) \in \mathbb{R}^{N \times d}$ for $l = 1, 2, \ldots, L-1$
- $\mathbf{f}^l(.)$ is a multi-output GP for $l = 1, 2, \ldots, L-1$
- $\epsilon_l \sim N(0, {\sigma_{\epsilon_l}}^2 \mathbf{I})$

Computationally Difficult for Large N
Collection of lots of Parameters (Latent Variables) to Estimate
Large Parameter Estimation Space

# How can do Inference on DGPs?

- MCMC to get $\pi(\hat{f}(\mathbf{x}^*)|\mathbf{X}, \mathbf{y})$
- VB approches to get $\overbrace{\pi(\hat{f}(\mathbf{x}^*)|\mathbf{X}, \mathbf{y})}$

  BI into an optimization problem over variational parameters

  - Kulback-Liebler (KL) divergence
  - Inducing Points
  - Minibatches (subsampling)
- Doubly Stochastic Variational Inference (DSVI)
  - Doing Inference quickly
  - Using TensorFlow
    Creating a computational graph, which takes multidimensional matrices called "Tensors" and does computations on them
  - Using GPflow
    A package for building GP models in python, using TensorFlow.

# Other Features of DSVI

- A variational algorithm for inference in DGP models that does not force independence or Gaussianity between the layers
- The variational posterior has the same structure as the full model (it maintains the correlations between layers)
- Using sparse VI to simplify the correlations within layers
- This method of inference does not require us to parameterize noisy corruptions separately from the outputs of each GP.
- They absorb the noise into the kernel

$$k_{noisy}(x_i, x_j) = k(x_i, x_j) + \sigma^2 \delta_{ij}$$

where $\delta_{ij}$ is the Kronecker delta, and $\sigma^2$ is the noise variance between layers

- Complexity of each generative GP mapping is reduced from $O(N^3)$ to $O(NM^2)$

$$X \xrightarrow[z^0 \to U^1]{f^1} F^1 + \varepsilon_1 \xrightarrow[]{f^2} \cdots \xrightarrow[z^{L-2} \to U^{L-1}]{f^{L-1}} F^{L-1} + \varepsilon_{L-1} \xrightarrow[z^{L-1} \to U^L]{f^L} F^L + \varepsilon = Y$$

$f^1 \sim GP(0, k_{X,X})$

$f^2 \sim GP(0, k_{F^1+\varepsilon_1, F^1+\varepsilon_1})$

$\vdots$

$f^L \sim GP(0, k_{F^{L-1}+\varepsilon_{L-1}, F^{L-1}+\varepsilon_{L-1}})$

Full model ( In Lawrence paper )

$$p(Y, \{F^l, U^l\}_{l=1}^L, \{F^l+\varepsilon\}_{l=1}^L) = \underbrace{p(F^1+\varepsilon_1 | F^1)\, p(F^1 | U^1, X)\, p(U^1 | z^0)}_{\text{first layer}} \times \prod_{l=2}^{L} p(F^l+\varepsilon | F^l)\, p(F^l | U^l, F^{l-1}+\varepsilon)\, p(U^l | z^{l-1})$$

Variational posterior ( In Lawrence paper )

$$Q = p(F^1 | U^1, X)\, q(U^1) \times \prod_{l=2}^{L} p(F^l | U^l, F^{l-1}_{f-i}+\varepsilon)\, q(U^l)\, \boxed{q(F^{l-1}+\varepsilon)}$$

$$\downarrow$$

$$\prod_{q=1}^{d} N(\mu_{i,q}, S_{i,q})$$

$$X \xrightarrow[z^0 \to U^1]{f^1} F^1 \xrightarrow[]{f^2} \cdots \xrightarrow[z^{L-2} \to U^{L-1}]{f^{L-1}} F^{L-1} \xrightarrow[z^{L-1} \to U^L]{f^L} F^L + \varepsilon = Y$$

$f^1 \sim GP(0, k^{noisy}_{X,X})$

$f^2 \sim GP(0, k^{noisy}_{F^1, F^1})$

$\vdots$

$f^L \sim GP(0, k^{noisy}_{F^{L-1}, F^{L-1}})$

Full model ( In Salimbani Inference )

$$p(Y, \{F^l, U^l\}_{l=1}^L) = p(Y | F^L) \prod_{l=1}^{L} p(F^l | U^l, F^{l-1}, z^{l-1})\, p(U^l | z^{l-1})$$

Variational posterior ( In Salimbani Inference )

$$Q(\{F^l, U^l\}_{l=1}^L) = \prod_{l=1}^{L} p(F^l | U^l, F^{l-1}, z^{l-1})\, q(U^l)$$

# Why Doubly?

The evidence lower bound of the DGP in this inference method is

$$\mathcal{L}_{DGP} = \sum_{i=1}^{N} \mathbb{E}_{q(\mathbf{f}_i^L)}[\log p(\mathbf{y}_n|\mathbf{f}_n^L)] - \sum_{l=1}^{L} \mathbf{KL}[q(\mathbf{U}^l)||p(\mathbf{U}^l;\mathbf{Z}^{l-1})]$$

Approximate ELBO using two sources of stochasticity

- Approximate the expectation with a Monte Carlo sample from the variational posterior
- Samples from the marginals at the top layer are obtained from marginals of all the other layers
- The bound factorizes over the data we achieve scalability through sub-sampling the data.
- This bound has complexity $O(NM^2Ld)$ to evaluate

# Easy to Set up the Problem !

Let's start with a simple example (Fitting a single layer DGP) :

- Randomly sample 500 points $x \in [0, 1]^2$
- Generate $Y$ from $f(x) = Sin(2\pi x_1) + Cos(2\pi x_2)$
- Randomly sample 100 inducing points $z \in [0, 1]^2$
- kernel=RBF(d, lengthscales=0.2, variance=1.)
- mdgp = DGP(X, Y, Z, Kernel, Gaussianlik(), minibatchsize=None)
- AdamOptimizer(0.001).minimize(mdgp, maxiter=2000)
- samples, m,v = mgp.predict.y(Xp)

- Adam ("Adaptive moments") optimizer is an adaptive learning rate optimization algorithm that's been designed specifically for training deep neural networks.
- adaptive learning rate optimization algorithm wich means, it computes individual learning rate for different parameters
- Adam uses estimations of first and second moments of gradients to adapt the learning rate for each parameter

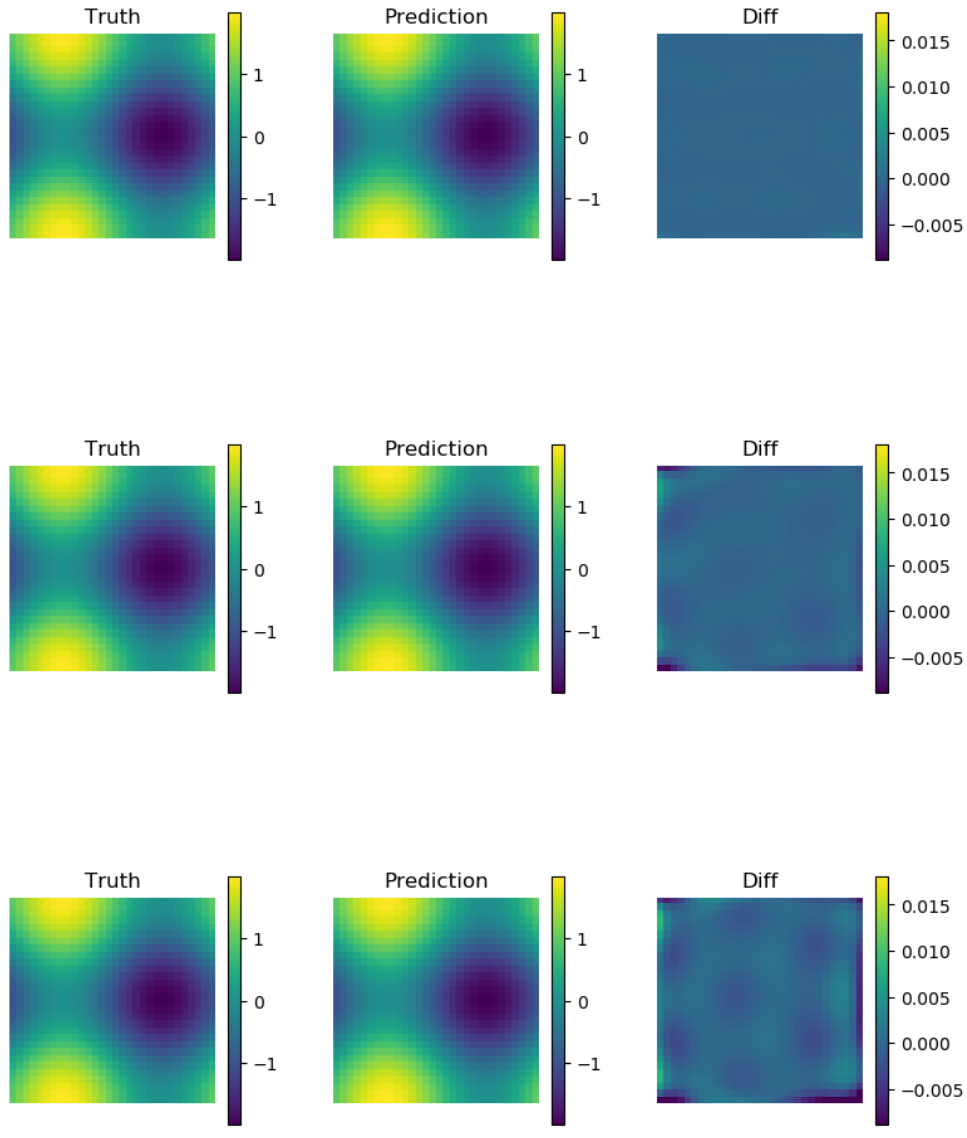


Figure: From top to bottom: Prediction using GP, One single layer DGP and SVGP

Table: Results

| Model | RMSPE | Opt Time |
|---|---|---|
| GP | 0.00015139 | 1.79 s |
| DGP(L=1) | 0.00111246 | 21.74 s |
| Sparse GP | 0.00208732 | 17.00 s |