Insights into Distributed Variational Inference for Bayesian Deep Learning

Rosa Candela[1], Pietro Michiardi[1], Maurizio Filippone[1]

1 - EURECOM, Sophia Antipolis, France

Distributed Inference

- Motivations:
  - Training of complex models and large datasets needs a distributed implementation
  - Current distributed architectures are not efficient and do not scale well
  - Previous experimental studies only consider simple models
- Parameter-Server architecture with data-parallelism approach:
  - Data and workloads are distributed over worker nodes;
  - Parameter Server maintains globally shared parameters;

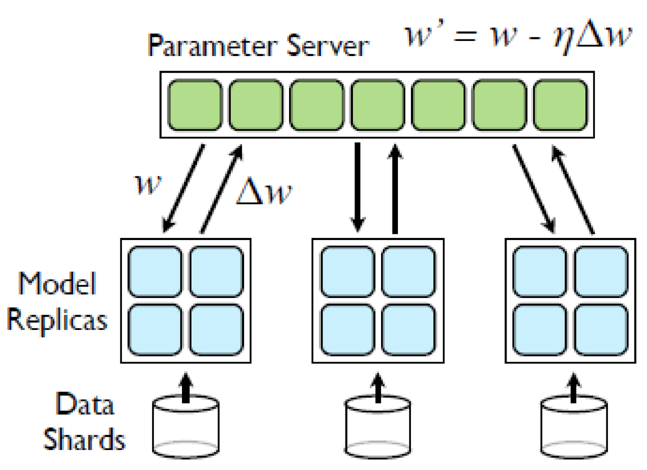


Fig. 1: Parameter-Server architecture

- Synchronization schemes in TensorFlow implementation:

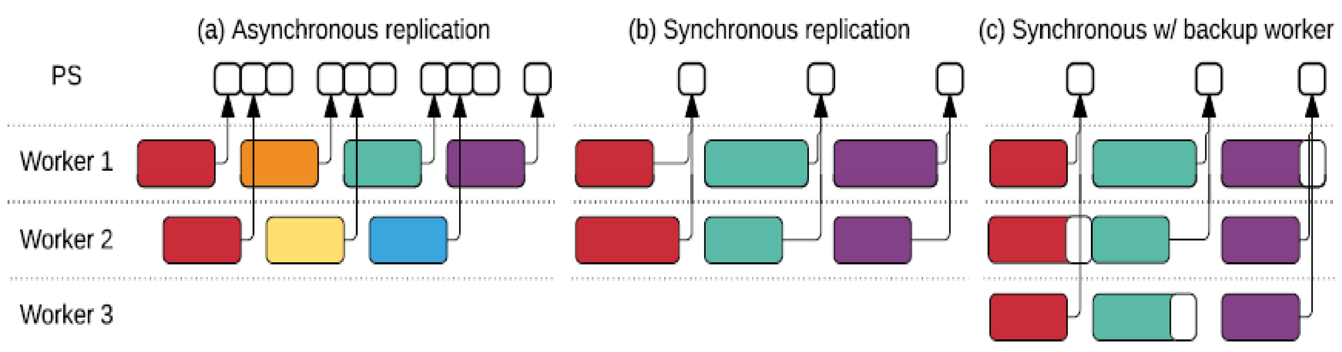


Fig. 2: Different synchronization techniques

Stochastic Variational Inference for DGPs

- Stochastic Variational Inference
  - Intractable posterior over model parameters:

$$p(\theta|Y,X) = \frac{p(Y|X,\theta)p(\theta)}{\int p(Y|X,\theta)p(\theta)d\theta};$$

  - Lower bound on marginal likelihood with mini-batch Stochastic Gradient optimization:

$$\log[p(Y|X,\theta)] \geq \frac{n}{m}\sum_{k\in\mathcal{I}_m} \mathrm{E}_{q(\theta)}(\log[p(y_k|\theta)]) - \mathrm{D}_{\mathrm{KL}}[q(\theta)\|p(\theta|X)],$$

    where $q(\theta)$ approximates $p(\theta|X)$;
  - Estimate the expectation using Monte Carlo:

$$\mathrm{E}_{q(\theta)}(\log[p(y_k|\theta)]) \approx \frac{1}{N_{\mathrm{MC}}}\sum_{r=1}^{N_{\mathrm{MC}}} \log[p(y_k|\tilde{\theta}_r)] \quad \text{with} \quad \tilde{\theta}_r \sim q(\theta);$$

- Deep Gaussian Processes (DGPs)
  - Deep probabilistic models;
  - Composition of functions:

$$\mathbf{f}(\mathbf{x}) = \left(\mathbf{h}^{(N_h-1)}\left(\theta^{(N_h-1)}\right) \circ \ldots \circ \mathbf{h}^{(0)}\left(\theta^{(0)}\right)\right)(\mathbf{x});$$

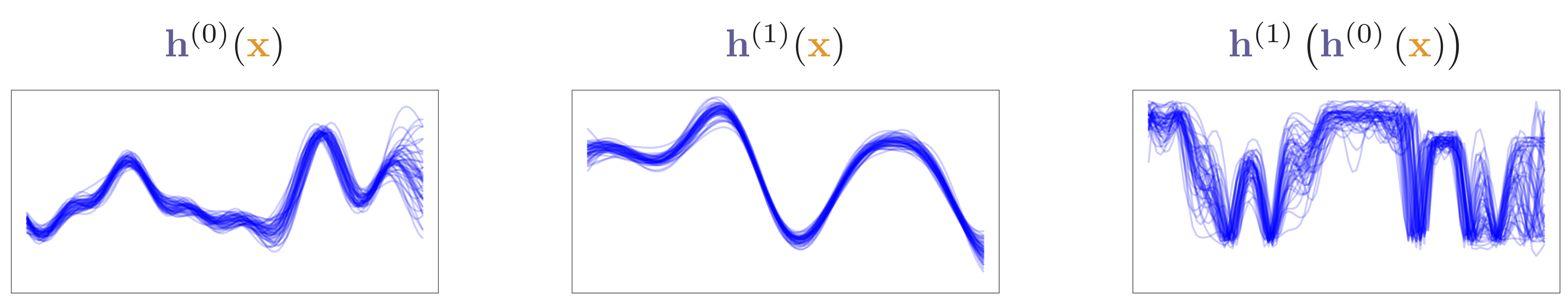


Fig. 3: Illustration of how stochastic processes may be composed.

- DGPs with random feature expansions
  - Example of RBF kernel approximated with trigonometric functions:

$$\Phi_{\mathrm{rbf}} = \sqrt{\frac{\sigma^2}{N_{\mathrm{RF}}}}\left[\cos(F\Omega), \sin(F\Omega)\right],$$

    with

$$F = \Phi W, \quad p\left(\Omega_{\cdot j}|\theta\right) = \mathcal{N}\left(\mathbf{0}, \Lambda^{-1}\right), \quad \Lambda = \mathrm{diag}(l_1^2, \ldots, l_d^2);$$

  - DGPs become equivalent to Deep Neural Networks with low-rank weight matrices.

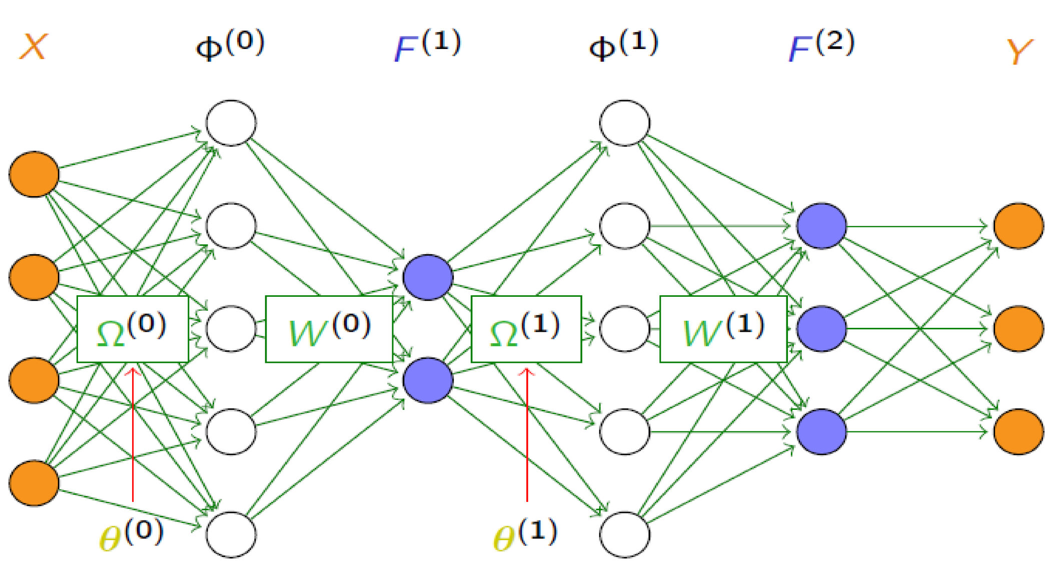


Fig. 4: Diagram of the proposed DGP model with random features.

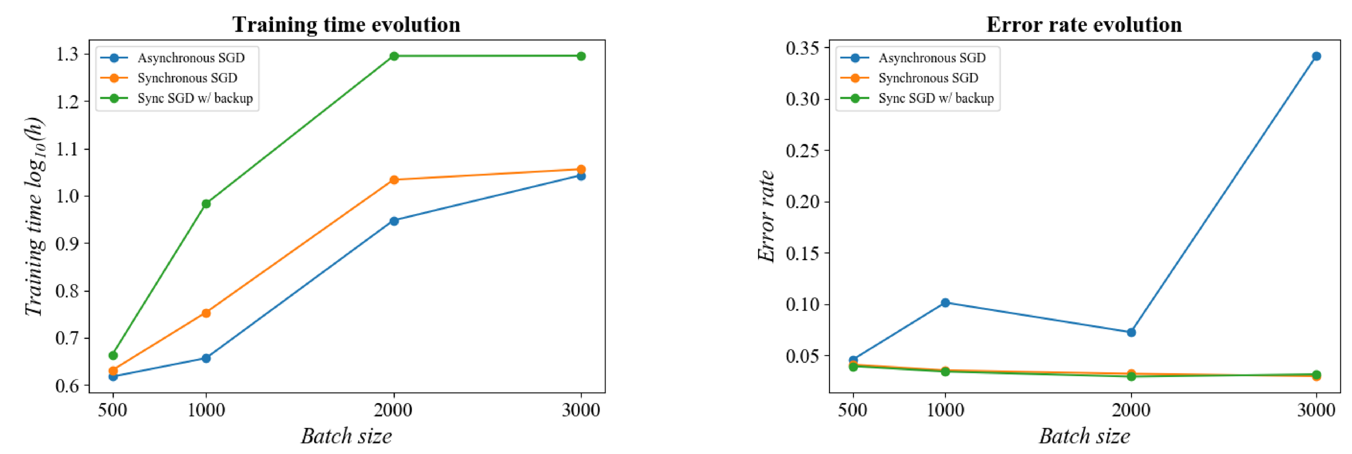
Experimental methodology

- Use mini-batch Stochastic gradient descent (SGD) algorithm
- Study the impact of design parameters:
  - Batch size
  - Learning rate
  - Number of Workers
  - Number of Parameter Servers
- Evaluate the performance through standard metrics:
  - Training time
  - Error Rate

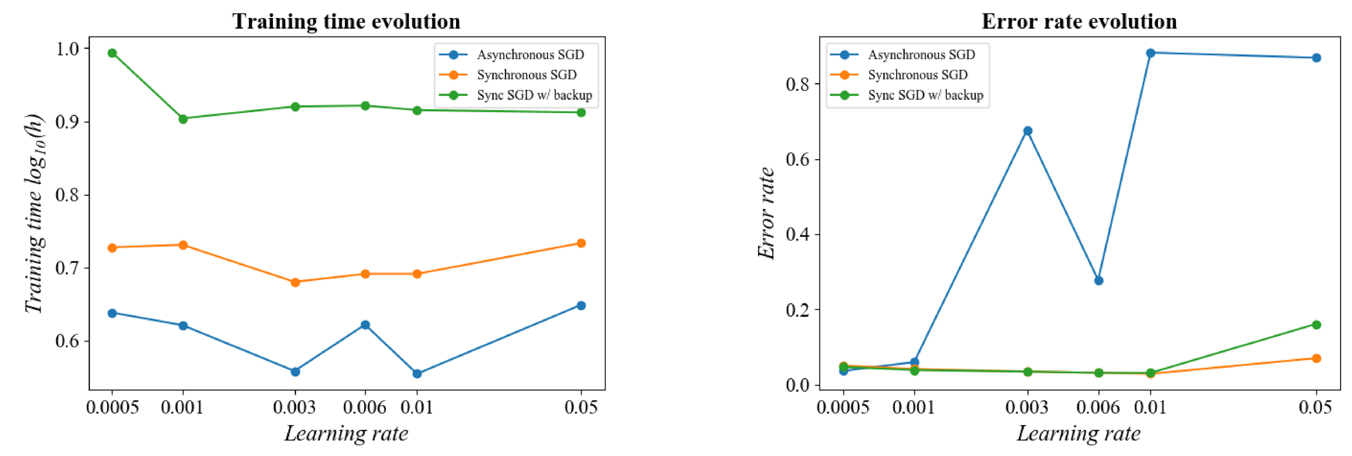
Experimental Setup and Results

- Classification task;
- MNIST dataset with 60000 samples;
- DGP model configuration: 2 hidden layers, 500 random features, 50 GPs in the hidden layers.

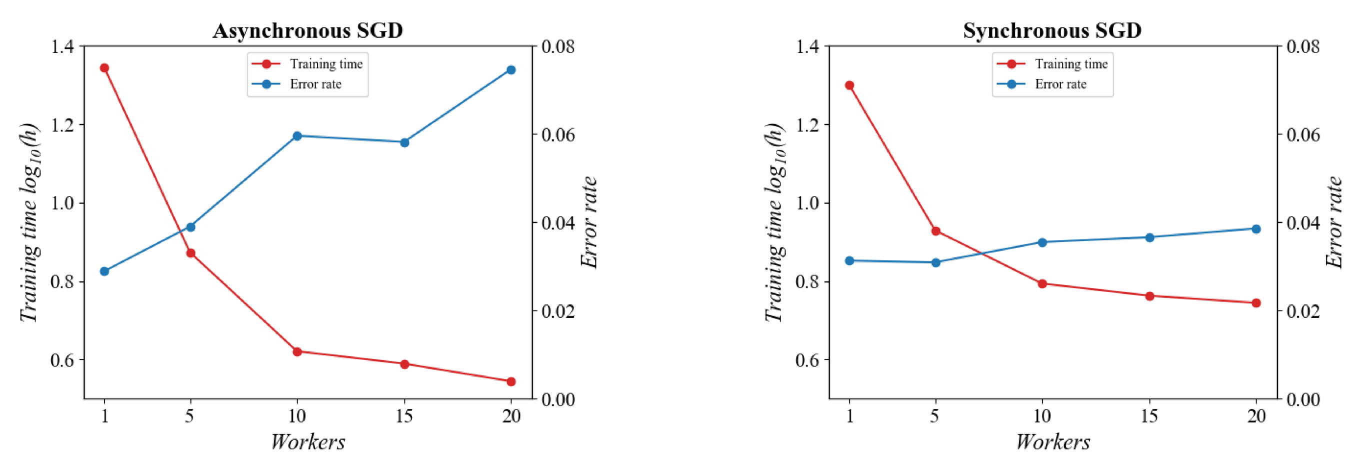
Batch size



Learning rate



Number of workers



Number of Parameter Servers

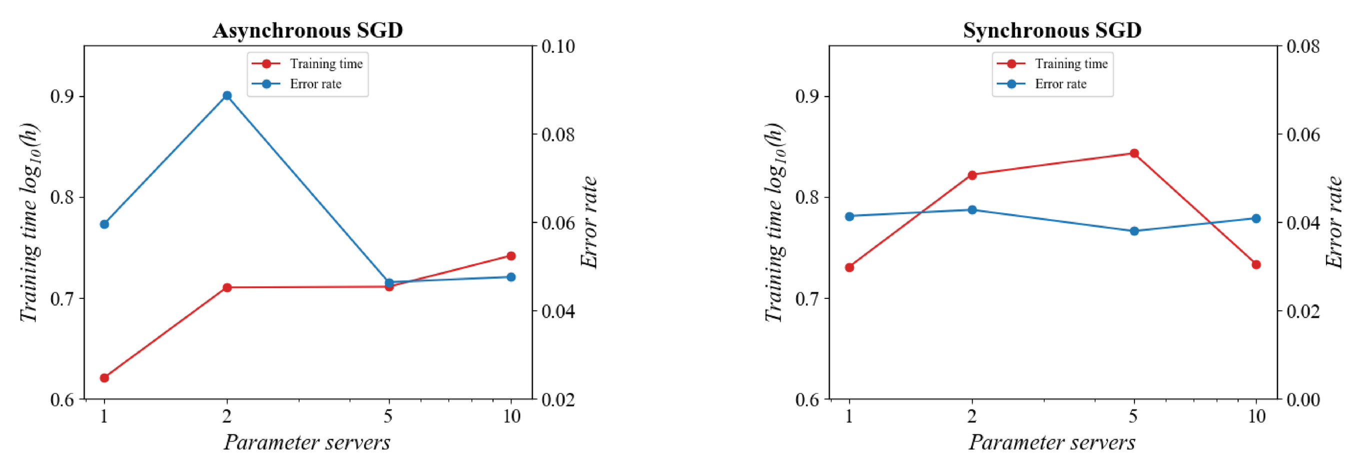


Fig. 5: Training time and error rate as function of different parameters.

Conclusions

Results of the experimental study:

- Large batch size and high learning rate:
  - ✓ improve Synchronous SGD convergence speed;
  - ✗ increase model quality deterioration with Asynchronous SGD;
- Scaling up the number of workers:
  - Asynchronous SGD reduces the training time, at expense of high error rate;
  - Synchronous SGD shows a smaller speed-up, but maintains low error rate;
- Scaling up the number of Parameter Servers:
  - can help to avoid bottlenecks at network level;
  - produces negligible effects if the computation work prevails on the communication task;

Future works:

- Repeat the experiments with a different distributed setting;
- Improve SGD distributed implementation, modifying the updates collection phase.

References

[1] K. Cutajar, E. Bonilla, P. Michiardi, M. Filippone. *Random Feature Expansions for Deep Gaussian Processes*, ICML 2017.

[2] M. Abadi, P. Barham, J. Chen et al. *TensorFlow: A system for large-scale machine learning*, OSDI 2016.