A Comparison of Stochastic Gradient MCMC using Multi-Core and GPU Architectures

Sergio Hernández, José Valdés and Matias Valdenegro

[1]Laboratorio de Procesamiento de Información Geoespacial.
Universidad Católica del Maule. Chile
shernandez@ucm.cl

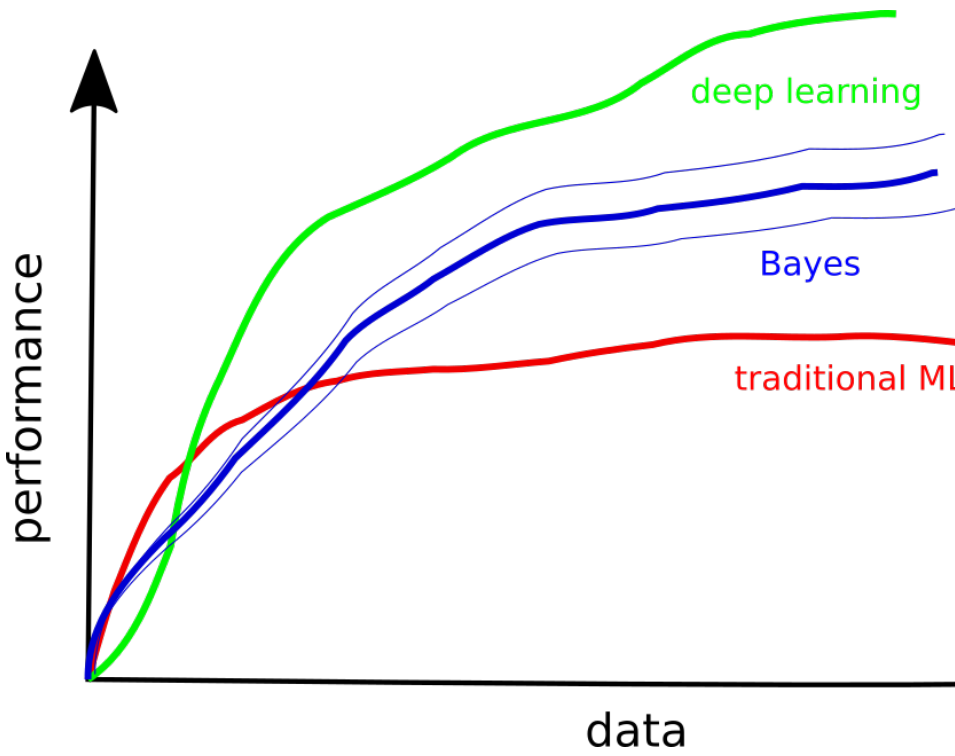
[2]Centro de Innovación en Ingeniería Aplicada.
Universidad Católica del Maule. Chile.

[3]German Research Center for Artificial Intelligence
Robotics Innovation Center. Germany.

INGELECTRA XXVIII

Introduction

- Deep learning models are traditionally used in big data scenarios.
- Stochastic Gradient Descent (SGD) is an optimization technique traditionally used for out-of-core training deep learning.
- Bayesian inference techniques can be used to capture the uncertainty of the model but it comes with a high computational cost.



Stochastic Gradient MCMC

- Let $\mathbf{D} = \{\mathbf{d_1}, \ldots, \mathbf{d_N}\}$ represent a typical dataset used for classification problems, as a set of tuples $\mathbf{d} = (\mathbf{x}, y)$ that contains features $\mathbf{x} \in \mathbb{R}^D$ and labels $y = \{1, \ldots, K\}$.

Stochastic Gradient MCMC

- Let $\mathbf{D} = \{\mathbf{d_1}, \ldots, \mathbf{d_N}\}$ represent a typical dataset used for classification problems, as a set of tuples $\mathbf{d} = (\mathbf{x}, y)$ that contains features $\mathbf{x} \in \mathbb{R}^D$ and labels $y = \{1, \ldots, K\}$.
- SGD iterates over mini-batches of size $B \ll N$ and computes a point estimate θ^*.

Stochastic Gradient MCMC

- Let $\mathbf{D} = \{\mathbf{d_1}, \ldots, \mathbf{d_N}\}$ represent a typical dataset used for classification problems, as a set of tuples $\mathbf{d} = (\mathbf{x}, y)$ that contains features $\mathbf{x} \in \mathbb{R}^D$ and labels $y = \{1, \ldots, K\}$.
- SGD iterates over mini-batches of size $B \ll N$ and computes a point estimate θ^*.
- In the Bayesian framework, the unknown parameter θ is considered as a random variable.

Stochastic Gradient MCMC

- Let $\mathbf{D} = \{\mathbf{d_1}, \ldots, \mathbf{d_N}\}$ represent a typical dataset used for classification problems, as a set of tuples $\mathbf{d} = (\mathbf{x}, y)$ that contains features $\mathbf{x} \in \mathbb{R}^D$ and labels $y = \{1, \ldots, K\}$.
- SGD iterates over mini-batches of size $B \ll N$ and computes a point estimate θ^*.
- In the Bayesian framework, the unknown parameter θ is considered as a random variable.

SGD (Robbins & Monro, 1951)

$$\theta^* = \text{argmax}\, p(\mathbf{D}|\theta)$$

SGLD (Welling & Teh, 2013)

$$p(\theta|\mathbf{D}) = \frac{p(\mathbf{D}|\theta)p(\theta)}{\int p(\mathbf{D}|\theta)p(\theta)d\theta}$$

SGD update

$$\theta_{t+1} = \theta_t + \left(\epsilon_t \sum_i^B \nabla \log p(\mathbf{d_i}|\theta_t)\right)$$

SGLD update

$$\theta_{t+1} = \theta_t + \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{B}\sum_i^B \nabla \log p(\mathbf{d_i}|\theta_t)\right) + \eta_t$$

CPU/GPU SG-MCMC

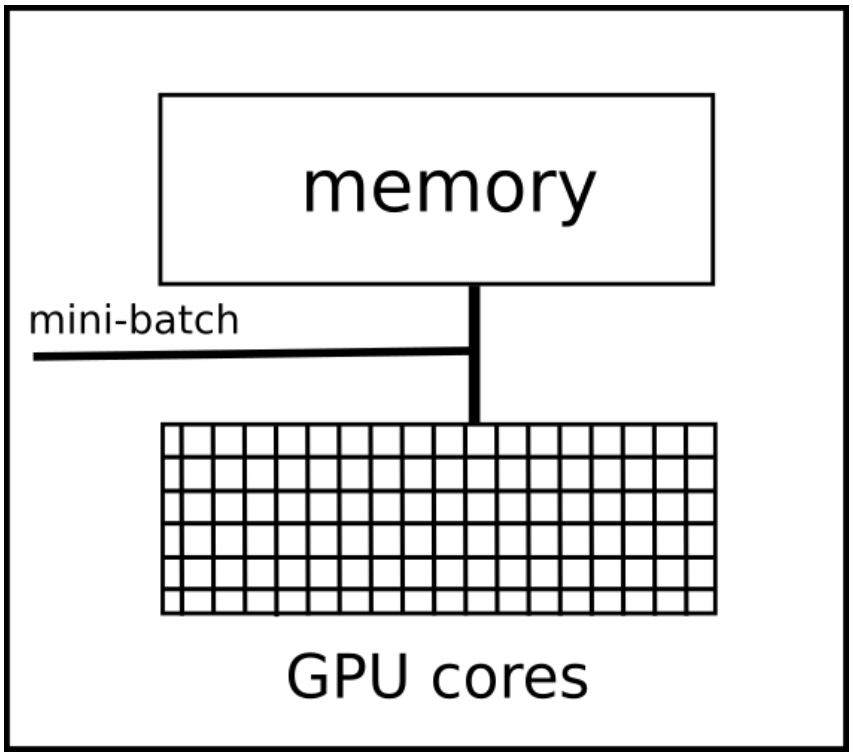


Figure: GPU implementation.

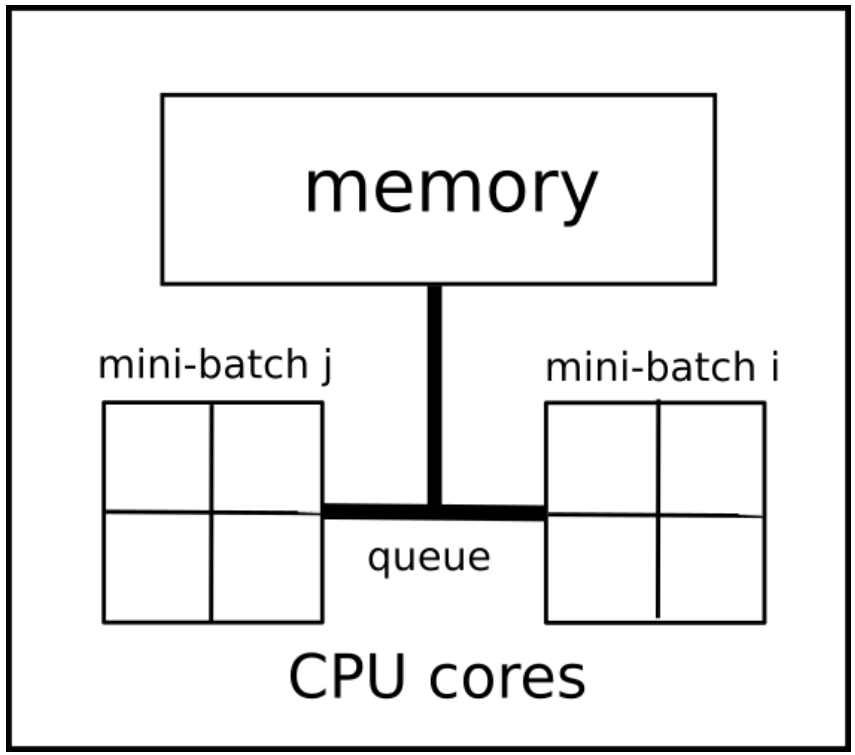


Figure: CPU implementation.

Experimental results

- Serial versions of the SGD and SGLD algorithms are tested on CPU and compared with parallel implementations on GPU.
- All experiments are run on a server with an Intel Xeon E5-2620 CPU with 12 cores using a single fully connected softmax layer (which is widely used in transfer learning).
- The server is also equipped with an NVIDIA TITAN GPU and both implementations were developed using Python 3.4.9.

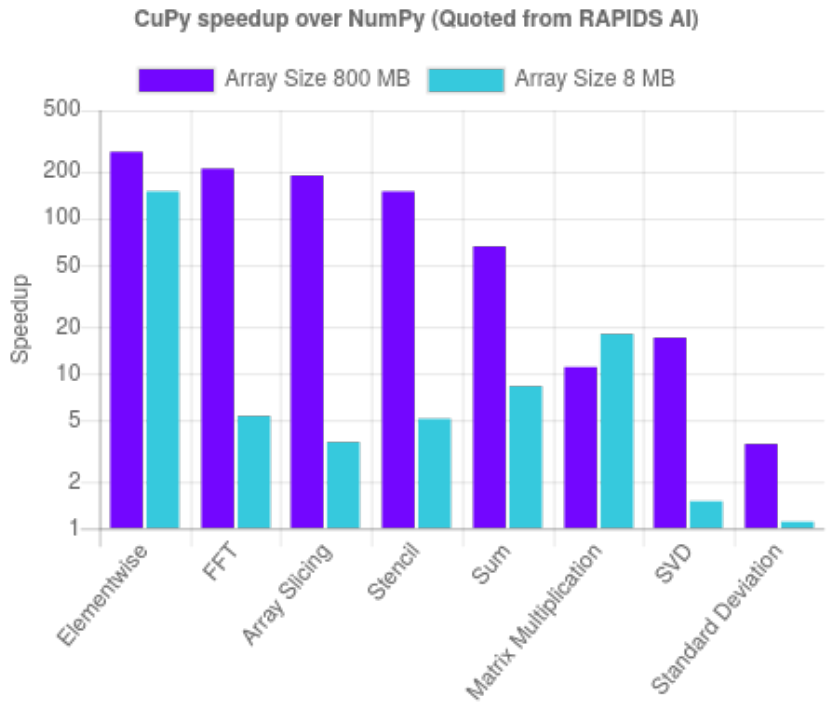


Figure: `https://medium.com/rapids-ai/single-gpu-cupy-speedups-ea99cbbb0cbb`

Synthetic Data (SGD)

The number of classes was set to $K = 3$, the number of epochs was set to $E = 2 \times 10^4$ and the size of the mini-batches was set to $B = 50$ for all examples.

D	N	CPU Time [s]	GPU Time [s]	Speed-Up
10	1000	11.98	124.85	10.42
10	10000	119.90	1245.98	10.39
10	50000	594.71	6186.05	10.40
10	100000	1195.62	12261.716	10.25
50	1000	13.67	123.13	9.00
50	10000	135.96	1230.19	9.04
50	50000	689.89	6137.79	8.89
50	100000	1487.26	12240.06	8.22
100	1000	15.71	123.53	7.86
100	10000	156.94	1231.121	7.84
100	50000	806.78	6176.91	7.65
100	100000	1770.83	12394.13	6.99

Table: Run time comparison of CPU and GPU implementations of SGD for a softmax regression problem

Synthetic Data (SGLD)

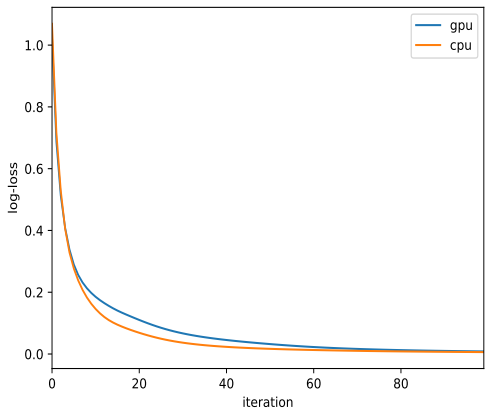
D	N	CPU Time [s]	GPU Time [s]	Speed-Up
10	1000	24.75	294.04	11.87
10	10000	245.22	2946.67	12.01
10	50000	1186.81	14518.66	12.23
10	100000	2380.18	29010.43	12.18
50	1000	25.70	288.02	11.48
50	10000	246.64	2885.16	11.69
50	50000	1230.02	14467.73	11.76
50	100000	2496.15	28930.81	11.59
100	1000	30.95	288.21	9.31
100	10000	279.28	2903.20	10.39
100	50000	1450.57	14531.93	10.01
100	100000	2796.20	29117.73	10.41

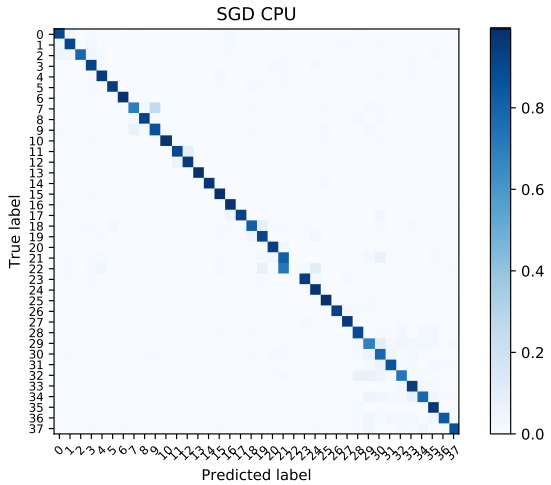
Table: Run time comparison of CPU ans GPU implementations of SGLD for a softmax regression problem

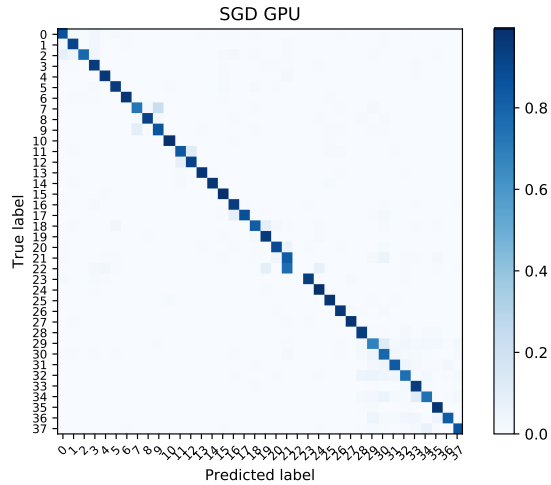
Bayesian Transfer Learning

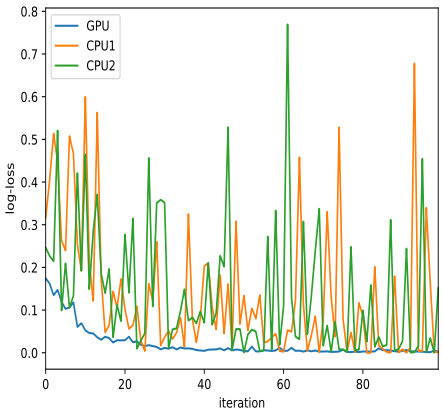
- The Plant Village dataset contains 54306 images with combinations of 14 crops and 26 diseases, leaving a total number of $K = 38$ classes.
- The VGG16 deep neural network with weights trained on the Imagenet dataset is used to extract features with dimensionality 25088 from the plants diseases dataset.
- The original class distribution is unbalanced, therefore random over-sampling is used to achieve a uniform class distribution, leaving a total number os $N = 203500$ training samples and 5700 test samples.

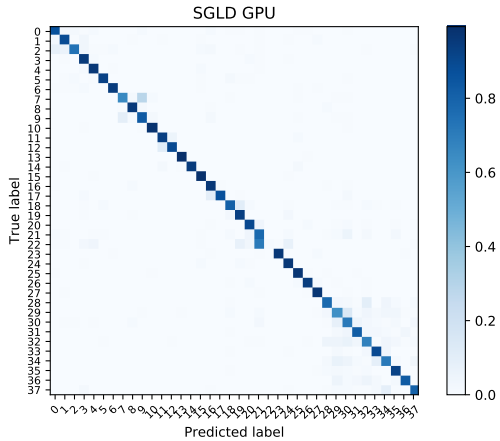
Bayesian Transfer Learning











Conclusions

- In this work, a novel comparison between multi-core CPU and GPU architectures for SGLD is performed.
- The speedups achieved with synthetic data are not compatible with other real data sets such as transfer learning data sets, which is mainly due to the size of the data batches and the model parameters.
- Future work will consider hybrid approaches based on multi-core CPU and multi-GPU that could also be beneficial in real life scenarios.