Stochastic Gradient Descent in Correlated Settings: A Study on Gaussian Processes

Hao Chen*[1], Lili Zheng*[1], Raed AL Kontar[2], Garvesh Raskutti[1]

[1] Department of Statistics, University of Wisconsin-Madison
[2] Department of Industrial and Operations Engineering, University of Michigan
* Equal contribution

Past success of Stochastic Gradient Descent

minimize empirical loss

$$R_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$$

Applied on deep learning:

zero training loss

good generalization power

How to Escape Saddle Points Efficiently

Chi Jin* Rong Ge† Praneeth Netrapalli‡ Sham M. Kakade§
Michael I. Jordan¶

On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima

Nitish Shirish Keskar*
Northwestern University
Evanston, IL 60208
keskar.nitish@u.northwestern.edu

Dheevatsa Mudigere
Intel Corporation
Bangalore, India
dheevatsa.mudigere@intel.com

Jorge Nocedal
Northwestern University
Evanston, IL 60208
j-nocedal@northwestern.edu

Mikhail Smelyanskiy
Intel Corporation
Santa Clara, CA 95054
mikhail.smelyanskiy@intel.com

Ping Tak Peter Tang
Intel Corporation
Santa Clara, CA 95054
peter.tang@intel.com

Learning Overparameterized Neural Networks via Stochastic Gradient Descent on Structured Data

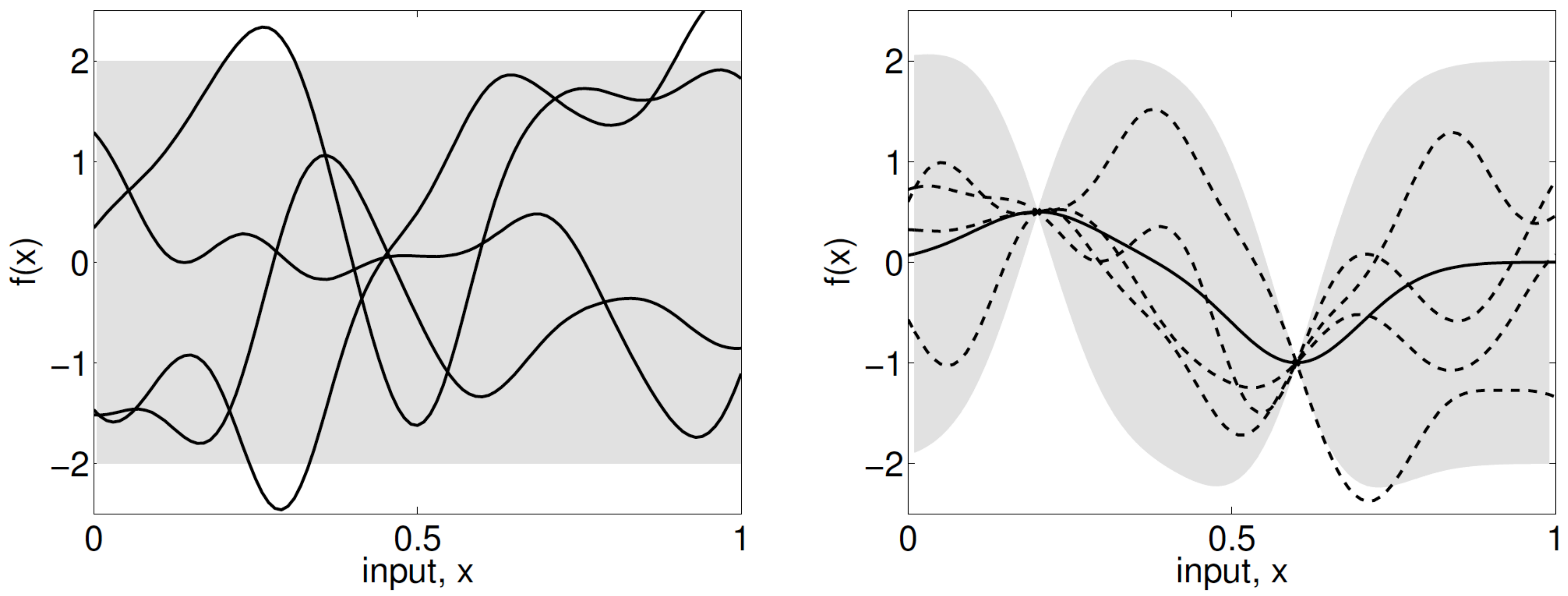
Yuanzhi Li
Computer Science Department
Stanford University
Stanford, CA 94305
yuanzhil@stanford.edu

Yingyu Liang
Department of Computer Sciences
University of Wisconsin-Madison
Madison, WI 53706
yliang@cs.wisc.edu

Abstract

Neural networks have many successful applications, while much less theoretical understanding has been gained. Towards bridging this gap, we study the problem of learning a two-layer overparameterized ReLU neural network for multi-class classification via stochastic gradient descent (SGD) from random initialization. In the overparameterized setting, when the data comes from mixtures of well-separated distributions, we prove that SGD learns a network with a small generalization error, albeit the network has enough capacity to fit arbitrary labels. Furthermore, the analysis provides interesting insights into several aspects of learning neural

SGD for Gaussian Processes (GPs)



(a), prior (b), posterior

Figure adopted from [Rusmassen and Williams, 2005]

Can we copy the success of SGD from deep learning to GPs?

$$f \sim \mathcal{GP}(0, \sigma_f^2 k(\cdot, \cdot)), \quad \mathbf{x}_1, \ldots, \mathbf{x}_n \overset{\text{i.i.d.}}{\sim} \mathbb{P}$$

$$y_i = f(\mathbf{x}_i) + \epsilon_i, \quad \epsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_\epsilon^2), \quad 1 \le i \le n\,.$$

Estimation for $\boldsymbol{\theta}^* = (\sigma_f^2, \sigma_\epsilon^2)^\top$

minimize marginal Gaussian log-likelihood $\ell(\boldsymbol{\theta}; \mathbf{X}_n, \mathbf{y}_n)$

Approximation method utilizing GPU

Exact Gaussian Processes on a Million Data Points

Ke Alexander Wang[1*] Geoff Pleiss[1*] Jacob R. Gardner[2]
Stephen Tyree[3] Kilian Q. Weinberger[1] Andrew Gordon Wilson[1,4]
[1]Cornell University, [2]Uber AI Labs, [3]NVIDIA, [4]New York University

Abstract

Gaussian processes (GPs) are flexible non-parametric models, with a capacity that grows with the available data. However, computational constraints with standard inference procedures have limited exact GPs to problems with fewer than about ten thousand training points, necessitating approximations for larger datasets. In this paper, we develop a scalable approach for exact GPs that leverages multi-GPU parallelization and methods like linear conjugate gradients, accessing the kernel matrix only through matrix multiplication. By partitioning and distributing kernel matrix multiplies, we demonstrate that an exact GP can be trained on over a million points, a task previously thought to be impossible with current computing hardware, in less than 2 hours. Moreover, our approach is generally applicable, without constraints to grid data or specific kernel classes. Enabled by this scalability, we perform the first-ever comparison of exact GPs against scalable GP approximations on datasets with $10^4 - 10^6$ data points, showing dramatic performance improvements.

Challenges

Minimize Gaussian log-likelihood:

$$\ell(\boldsymbol{\theta}; \mathbf{X}_n, \mathbf{y}_n) = \frac{1}{2n}[\mathbf{y}_n^\top \mathbf{K}_n^{-1}(\boldsymbol{\theta})\mathbf{y}_n + \log|\mathbf{K}_n(\boldsymbol{\theta})| + n\log(2\pi)]$$

$$\mathbf{K}_n(\boldsymbol{\theta}) = \theta_1 \begin{pmatrix} k(x_1, x_1) & \cdots & k(x_1, x_N) \\ & \vdots & \\ \cdots & k(x_i, x_j) & \cdots \\ & \vdots & \\ k(X_N, x_1) & \cdots & k(X_N, x_1) \end{pmatrix} + \theta_2 I_n$$

Strong correlations among samples

Highly non-linear w.r.t. data points

Stochastic gradients are **biased** for the full gradient

Non-convexity

Our findings

$$(\theta_1^{(K)} - \theta_1^*)^2 \leq \frac{8G^2}{\gamma^2(K+1)} + Cm^{-\frac{1}{2}+\varepsilon}.$$

$$(\theta_2^{(K)} - \theta_2^*)^2 \leq \frac{8G^2}{\gamma^2(K+1)} + C(\log m)^{-\frac{1}{2}+\varepsilon}.$$

K: number of iterations
optimization error rate
of strongly convex loss

m: mini batch size
statistical error,
vanishes as m
increases

Case studies

Dataset	Size	D	**RMSE**	**Training Time** (min)	**Memory Usage** (GB)
OTL Circuit	2,000,000	6	0.401 ± 0.000	33.43 ± 4.40	0.99 ± 0.00
Wing Weight	2,000,000	10	0.072 ± 0.004	78.78 ± 9.26	1.22 ± 0.00

Lower prediction error

Significant faster training

			RMSE				**Training Time** (min)			
Dataset	Size	D	**sgGP**	**EGP**	**SGPR**	**SVGP**	**sgGP**	**EGP**	**SGPR**	**SVGP**
Levy	10,000	4	**0.265** $\pm$ 0.003	0.312 $\pm$ 0.003	0.564 $\pm$ 0.010	0.582 $\pm$ 0.013	**0.51** $\pm$ 0.00	11.48 $\pm$ 1.28	4.04 $\pm$ 0.51	14.58 $\pm$ 0.07
Griewank	10,000	6	**0.071** $\pm$ 0.000	0.185 $\pm$ 0.073	0.132 $\pm$ 0.003	0.093 $\pm$ 0.005	**0.61** $\pm$ 0.01	15.25 $\pm$ 3.72	1.93 $\pm$ 0.31	13.18 $\pm$ 0.58
Bike	17,379	17	**0.221** $\pm$ 0.002	0.228 $\pm$ 0.002	0.276 $\pm$ 0.004	0.250 $\pm$ 0.010	**1.98** $\pm$ 0.03	31.48 $\pm$ 7.45	5.31 $\pm$ 2.05	25.26 $\pm$ 3.97
Energy	19,735	27	**0.786** $\pm$ 0.001	0.802 $\pm$ 0.007	0.843 $\pm$ 0.006	0.795 $\pm$ 0.005	**3.15** $\pm$ 0.04	54.39 $\pm$ 8.01	5.41 $\pm$ 0.73	25.09 $\pm$ 5.50
PM2.5	41,757	15	0.287 $\pm$ 0.002	**0.286** $\pm$ 0.003	0.638 $\pm$ 0.005	0.540 $\pm$ 0.010	**5.21** $\pm$ 0.04	385.51 $\pm$ 42.59	13.59 $\pm$ 2.30	52.46 $\pm$ 10.08
Protein	45,730	9	**0.663** $\pm$ 0.006	0.694 $\pm$ 0.004	0.715 $\pm$ 0.003	0.676 $\pm$ 0.004	**3.40** $\pm$ 0.03	500.33 $\pm$ 65.62	19.55 $\pm$ 1.66	55.27 $\pm$ 13.09
Query	100,000	4	**0.053** $\pm$ 0.000	—	0.058 $\pm$ 0.002	0.061 $\pm$ 0.000	**6.40** $\pm$ 0.10	—	20.73 $\pm$ 1.63	124.73 $\pm$ 22.25
Borehole	1,000,000	8	**0.172** $\pm$ 0.000	—	0.176 $\pm$ 0.000	0.173 $\pm$ 0.000	**67.29** $\pm$ 13.39	—	857.60 $\pm$ 76.02	1380.86 $\pm$ 11.32