Scalable Logit Gaussian Process Classification

Florian Wenzel[1,3] **Théo Galy-Fajou**[2], **Christian Donner**[2], **Marius Kloft**[3] **and Manfred Opper**[2]
[1]Humboldt University of Berlin, [2]Technical University of Berlin, [3]Technical University of Kaiserslautern
Contact: wenzelfl@hu-berlin.de

Abstract

We propose an efficient stochastic variational approach to Gaussian Process (GP) classification building on Pólya-Gamma data augmentation and inducing points, which is based on closed-form updates of natural gradients. We evaluate the algorithm on real-world datasets containing up to 11 million data points and demonstrate that it is up to two orders of magnitude faster than the state-of-the-art while being competitive in terms of prediction performance.

1 Introduction

In GP classification, naive inference typically scales cubic in the number of data points, and exact computation of posterior and marginal likelihood is intractable. Nevertheless, the combination of so-called sparse Gaussian process techniques with approximate inference methods, such as expectation propagation (EP) or the variational approach, have enabled GP classification for datasets containing millions of data points [1, 2, 3].

While these results are already impressive, we will show in this paper that a speedup of up to two orders magnitudes can be achieved. Our approach is based on replacing the ordinary (stochastic) gradients for optimizing the variational objective function by more efficient *natural gradients*, which recently have been successfully used in a variety of variational inference problems [e.g., 4, 5].

Our main contributions are as follows:

- We present a fast Gaussian process classification model using a logit link function. Our approach relies on Pólya-Gamma data augmentation and inducing points for Gaussian process inference.
- We derive an efficient inference algorithm based on stochastic variational inference and natural gradients [6]. All natural gradient updates are given in closed-form and do not rely on numerical quadrature methods or sampling approaches. Natural gradients have the advantage that they provide effective second-order optimization updates [6].
- In our experiments, we demonstrate that our approach drastically improves speed up to two orders of magnitude while being competitive in terms of prediction performance. We apply our method to massive real-world datasets up to 11 million points and demonstrate superior scalability.

2 Model

The logit GP Classification model is defined as follows. Let $X = (\boldsymbol{x}_1, \dots, \boldsymbol{x}_n) \in \mathbb{R}^{d \times n}$ be the d-dimensional training points with labels $\boldsymbol{y} = (y_1, \dots, y_n) \in \{-1, 1\}^n$. The likelihood of the labels is

$$p(\boldsymbol{y}|\boldsymbol{f}, X) = \prod_{i=1}^{n} \sigma(y_i f(\boldsymbol{x}_i)), \tag{1}$$

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

where $\sigma(z) = (1 + \exp(-z))^{-1}$ is the logit link function and f is the latent decision function. We place a GP prior over f and obtain the joint distribution of the labels and the latent GP

$$p(\boldsymbol{y}, \boldsymbol{f}|X) = p(\boldsymbol{y}|\boldsymbol{f}, X)p(\boldsymbol{f}|X), \tag{2}$$

where $p(\boldsymbol{f}|X) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, K_{nn})$ and K_{nn} denotes the kernel matrix evaluated at the training points X. For the sake of clarity we omit the conditioning on X in the following.

Pólya-Gamma data augmentation We augment the logit GP classification model by Pólya-Gamma random variabels which are defined as follows. The random variable $\omega \sim \mathrm{PG}(b, 0)$, $b > 0$ is defined by the moment generating function $\mathbb{E}_{\mathrm{PG}(\omega|\, b,0)}[\exp(-\omega t)] = (\cosh^b(\sqrt{t/2}))^{-1}$.The general $\mathrm{PG}(b, c)$ class which is derived by an exponential tilting of the $\mathrm{PG}(b, 0)$ density is given by $\mathrm{PG}(\omega|\, b, c) \propto \exp(-\frac{c^2}{2}\omega)\mathrm{PG}(\omega|\, b, 0)$.

We write the non-conjugate logistic likelihood function (1) in terms of Pólya-Gamma variables

$$\sigma(z_i) = (1 + \exp(-z_i))^{-1} = \frac{\exp(\frac{1}{2}z_i)}{2\cosh(\frac{z_i}{2})} = \frac{1}{2}\int \exp\left(\frac{z_i}{2} - \frac{z_i^2}{2}\omega_i\right) p(\omega_i)\mathrm{d}\omega_i, \tag{3}$$

where $p(\omega_i) = \mathrm{PG}(\omega_i|1, 0)$. For more details consult [7]. Using this identity and substituting $z_i = y_i f(x_i)$ we augment the joint density (2) with Pólya-Gamma variables

$$p(\boldsymbol{y}, \boldsymbol{\omega}, \boldsymbol{f}) = p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\omega})p(\boldsymbol{f})p(\boldsymbol{\omega}) \propto \exp\left(\frac{1}{2}\boldsymbol{y}^\top \boldsymbol{f} - \frac{1}{2}\boldsymbol{f}^\top \Omega \boldsymbol{f}\right) p(\boldsymbol{f})p(\boldsymbol{\omega}), \tag{4}$$

where $\Omega = \mathrm{diag}(\boldsymbol{\omega})$ is the diagonal matrix of the Pólya-Gamma variables $\{\omega_i\}$.

Sparse Gaussian process Inference in GP models typically has the computational complexity $\mathcal{O}(n^3)$. We aim to obtain a scalable approximation of our model and focus on inducing point methods [8]. We follow a similar approach as in [1] and reduce the complexity to $\mathcal{O}(m^3)$, where m is number of inducing points.

We augment the latent GP f with m additional input-output pairs $(Z_1, u_1), \ldots, (Z_m, u_m)$, termed as *inducing inputs* and *inducing variables*. The function values of the GP $\boldsymbol{f}$ and the inducing variables $\boldsymbol{u} = (u_1, \ldots, u_m)$ are connected via

$$p(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}\left(\boldsymbol{f}|K_{nm}K_{mm}^{-1}\boldsymbol{u}, \widetilde{K}\right), \quad p(\boldsymbol{u}) = \mathcal{N}\left(\boldsymbol{u}|0, K_{mm}\right), \tag{5}$$

where K_{mm} is the kernel matrix resulting from evaluating the kernel function between all inducing inputs, K_{nm} is the cross-kernel matrix between inducing inputs and training points and $\widetilde{K} = K_{nn} - K_{nm}K_{mm}^{-1}K_{mn}$. Including the inducing points in our model gives the augmented joint distribution

$$p(\boldsymbol{y}, \boldsymbol{\omega}, \boldsymbol{f}, \boldsymbol{u}) = p(\boldsymbol{y}|\boldsymbol{\omega}, \boldsymbol{f})p(\boldsymbol{\omega})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u}). \tag{6}$$

3 Inference

We aim to approximate the posterior of the inducing points $p(\boldsymbol{u}|\boldsymbol{y})$ and apply the methodology of variational inference to the marginal joint distribution $p(y, \omega, u) = p(\boldsymbol{y}|\boldsymbol{\omega}, \boldsymbol{u})p(\boldsymbol{\omega})p(\boldsymbol{u})$. We construct a variational lower bound on the evidence

$$\begin{aligned}
\log p(\boldsymbol{y}) &\leq \mathbb{E}_{q(\boldsymbol{u},\boldsymbol{\omega})}[\log p(\boldsymbol{y}|\boldsymbol{u}, \boldsymbol{\omega})] - \mathrm{KL}\left(q(\boldsymbol{u}, \boldsymbol{\omega})||p(\boldsymbol{u}, \boldsymbol{\omega})\right) \\
&\leq \mathbb{E}_{p(\boldsymbol{f}|\boldsymbol{u})q(\boldsymbol{u})q(\boldsymbol{\omega})}[\log p(\boldsymbol{y}|\boldsymbol{\omega}, \boldsymbol{f})] - \mathrm{KL}\left(q(\boldsymbol{u}, \boldsymbol{\omega})||p(\boldsymbol{u}, \boldsymbol{\omega})\right) \\
&=: \mathcal{L},
\end{aligned} \tag{7}$$

where the first inequality is the usual evidence lower bound (ELBO) in variational inference and in the second line we apply Jensen's inequality.

We follow a structured mean-field approach [9] and consider a variational distribution of the form $q(u, \omega) = q(u)q(\omega)$ with $q(\omega_i) = \mathrm{PG}(\omega_i|1, c_i)$ and $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{\mu}, \Sigma)$ and employ stochastic variational inference (SVI) [6] to optimize the variational bound (7) using stochastic optimization.

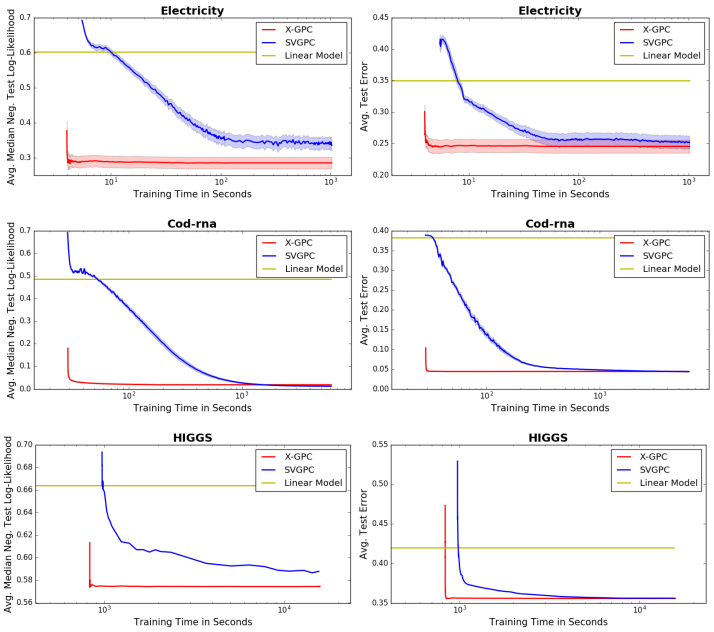

Figure 1: Average median of the negative test log-likelihood and average test prediction error as a function of training time (seconds in a $\log_{10}$ scale) on the datasets Electricity (45,312 points), Cod RNA (343,564 points) and Higgs (11 million points).

Since we have the variational objective in closed-form we are able to compute the natural gradients in closed-form as well. Using the natural gradient over the standard Euclidean gradient is favorable since natural gradients are invariant to reparameterization of the variational family [10, 11] and provide effective second-order optimization updates [12, 6].

This is in contrast to the model of [1] where the global updates cannot be computed in a closed-form and one relies on less efficient Euclidean gradient updates that are computed using numerical quadrature methods.

Closed-form updates Our algorithm alternates between updates of the local variational parameters $\boldsymbol{c} = (c_1, \ldots, c_n)$ and global parameters $\boldsymbol{\mu}$ and Σ. In each iteration we update the parameters based on a mini-batch of the data $\mathcal{S} \subset \{1, ..., n\}$ of size $s = |\mathcal{S}|$.

We update the *local parameters* $\boldsymbol{c}_\mathcal{S}$ in the mini-batch $\mathcal{S}$ by employing coordinate ascent

$$c_i = \sqrt{\widetilde{K}_{ii} + \boldsymbol{\kappa}_i \Sigma \boldsymbol{\kappa}_i^\top + \boldsymbol{\mu}^\top \boldsymbol{\kappa}_i^\top \boldsymbol{\kappa}_i \boldsymbol{\mu}}, \tag{8}$$

where $\boldsymbol{\kappa}_i = K_{im} K_{mm}^{-1}$ and $i \in \mathcal{S}$.

We update the *global parameters* based on stochastic estimates of the natural gradients of the global parameters. We use the natural parameterization of the variational Gaussian distribution, i.e the parameters $\boldsymbol{\eta}_1 := \Sigma^{-1}\boldsymbol{\mu}$ and $\eta_2 = -\frac{1}{2}\Sigma^{-1}$. The natural gradients w.r.t. natural parameters of the variational Gaussian distribution based on the mini-batch $\mathcal{S}$ are given by

$$\begin{aligned} \widetilde{\nabla}_{\boldsymbol{\eta}_1} \mathcal{L}_\mathcal{S} &= \frac{n}{2s} \boldsymbol{\kappa}_\mathcal{S}^\top \boldsymbol{y}_\mathcal{S} - \boldsymbol{\eta}_1 \\ \widetilde{\nabla}_{\eta_2} \mathcal{L}_\mathcal{S} &= -\frac{1}{2}\left(K_{mm}^{-1} + \frac{n}{s} \boldsymbol{\kappa}_\mathcal{S}^\top \Theta_\mathcal{S} \boldsymbol{\kappa}_\mathcal{S} \right) - \eta_2, \end{aligned} \tag{9}$$

where $\Theta = \mathrm{diag}(\boldsymbol{\theta})$ and $\theta_i = \frac{1}{4c_i} \tanh\left(\frac{c_i}{2}\right)$.

4 Experiments

To compare our method **X-GPC** (extremely fast Gaussian process classification) against the state-of-the-art **SVGPC** [1], we use the highly optimized implementation of SVGPC provided in the package

GPflow[1] [13], which builds on TensorFlow [14]. Both methods are applied to real-world datasets containing up to 11 million datapoints.

We train both methods using a mini-batch size of 100 points and 100 inducing points. The initial inducing points are chosen using the k-means clustering algorithm as done in [15] and are the same for both methods. In all experiments a squared exponential covariance function with a common length scale parameter for each dimension, an amplitude parameter and an additive noise parameter is used. The kernel hyper-parameters are initialized to the same values. All algorithms are run on a single CPU.

We experiment on 11 datasets from the OpenML website and the UCI repository ranging from 768 to 11 million points and report the average prediction error, the negative test log-likelihood (NLL) and the run time along with one standard deviation.

Dataset	n / d		**X-GPC**	SVGPC
aXa	36974 123	Error NLL Time	0.17 ± 0.07 0.16 ± 0.10 8.7 ± 0.9	0.17 ± 0.07 0.18 ± 0.12 571 ± 2.2
Bank Market.	45211 43	Error NLL Time	0.11 ± 0.09 0.10 ± 0.10 7.4 ± 1.8	0.11 ± 0.09 0.10 ± 0.09 609 ± 2.7
Click Predict.	399482 12	Error NLL Time	0.17 ± 0.00 0.17 ± 0.01 35 ± 2.7	0.17 ± 0.00 0.24 ± 0.01 1256 ± 191
Cod RNA	343564 8	Error NLL Time	0.04 ± 0.00 0.02 ± 0.00 134 ± 15	0.05 ± 0.00 0.01 ± 0.00 3002 ± 122
Cov Type	581012 54	Error NLL Time	0.32 ± 0.06 0.53 ± 0.06 29 ± 9.66	0.32 ± 0.05 0.54 ± 0.07 1004 ± 51
Diabetis	768 8	Error NLL Time	0.23 ± 0.07 0.31 ± 0.12 0.8 ± 0.1	0.23 ± 0.07 0.33 ± 0.11 405 ± 59
Electricity	45312 8	Error NLL Time	0.25 ± 0.07 0.29 ± 0.05 4.6 ± 1.2	0.25 ± 0.06 0.34 ± 0.05 888 ± 1.9
German	1000 20	Error NLL Time	0.25 ± 0.13 0.40 ± 0.19 1.03 ± 0.2	0.25 ± 0.13 0.39 ± 0.18 319 ± 15
Higgs	11M 22	Error NLL Time	0.36 ± 0.00 0.05 ± 0.01 14.1 ± 5.7	0.35 ± 0.00 0.03 ± 0.02 1019 ± 5.9
Shuttle	58000 9	Error NLL Time	0.02 ± 0.00 0.01 ± 0.00 139 ± 6.6	0.03 ± 0.00 0.00 ± 0.00 1501 ± 92
SUSY	5M 18	Error NLL Time	0.20 ± 0.01 0.21 ± 0.01 523 ± 25	0.21 ± 0.01 0.27 ± 0.01 10366 ± 360

Table 1: Average test prediction error, negative test log-likelihood (NLL) and time in seconds along with one standard deviation.

Numerical comparison X-GPC slightly improves prediction performance while being one to two orders of magnitude faster than SVPGC. More, precisely we obtain speed ups ranging from a factor 10.8 on Shuttle to a factor 506 on Diabetes.

Performance as a function of time Since X-GPC and SVGPC are based on an optimization scheme there is a trade-off between the run time of the algorithm and the prediction performance. We profile each method and monitor the negative test log-likelihood and prediction error on a hold-out test set as function of time. As a benchmark, we fit a linear model and use the logistic regression implementation in scikit-learn [16], which is based on LIBLINEAR [17].

The results are displayed in figure 1. X-GPC is already very close to the optimum after a few iterations due to its efficient natural gradient updates. The test log-likelihood and the prediction error of X-GPC converge around one to two orders of magnitude faster than SVGPC.

5 Conclusions

We proposed an efficient Gaussian process classification method that builds on Pólya-Gamma data augmentation and inducing points. The experimental evaluations shows that our method is up to two orders of magnitude faster than the state-of-the-art approach while being competitive in terms of prediction performance. Speed improvements are due to the data augmentation approach that enables efficient second order optimization.

[1] We use GPflow version 0.4.0.

Acknowledgments.

We thank Stephan Mandt, Patrick Jähnichen, Matthäus Deutsch and Robert Vandermeulen for fruitful discussions. This work was partly funded by the German Research Foundation (DFG) award KL 2698/2-1.

References

[1] J. Hensman and A. Matthews, "Scalable Variational Gaussian Process Classification," in *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics,*, 2015.

[2] A. Dezfouli and E. V. Bonilla, "Scalable inference for gaussian process models with black-box likelihoods," in *Advances in Neural Information Processing Systems 28*, pp. 1414–1422, 2015.

[3] D. Hernández-Lobato and J. M. Hernández-Lobato, "Scalable gaussian process classification via expectation propagation," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pp. 168–176, 2016.

[4] A. Honkela, T. Raiko, M. Kuusela, M. Tornio, and J. Karhunen, "Approximate riemannian conjugate gradient learning for fixed-form variational bayes," *Journal of Machine Learning Research*, vol. 11, pp. 3235–3268, 2010.

[5] P. Jähnichen, F. Wenzel, and M. Kloft, "Scalable inference in dynamic mixture models," *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2016.

[6] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, "Stochastic Variational Inference," *Journal of Machine Learning Research*, 2013.

[7] N. G. Polson, J. G. Scott, and J. Windle, "Bayesian inference for logistic models using pólya–gamma latent variables," *Journal of the American Statistical Association*, vol. 108, no. 504, pp. 1339–1349, 2013.

[8] E. Snelson and Z. Ghahramani, "Sparse GPs using Pseudo-inputs," *NIPS*, 2006.

[9] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference," *Found. Trends Mach. Learn.*, pp. 1–305, Jan. 2008.

[10] S. Amari and H. Nagaoka, *Methods of Information Geometry*. American Mathematical Society, 2007.

[11] J. Martens, "New insights and perspectives on the natural gradient method," *Arxiv Preprint*, 2017.

[12] S. Amari, "Natural grad. works efficiently in learning," *Neural Computation*, 1998.

[13] A. G. d. G. Matthews, M. van der Wilk, T. Nickson, K. Fujii, A. Boukouvalas, P. León-Villagrá, Z. Ghahramani, and J. Hensman, "GPflow: A Gaussian process library using TensorFlow," *Journal of Machine Learning Research*, vol. 18, pp. 1–6, apr 2017.

[14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[15] F. Wenzel, T. Galy-Fajou, M. Deutsch, and M. Kloft, "Bayesian nonlinear support vector machines for big data," in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2017.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[17] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.