AutoGP: Exploring the Capabilities and Limitations of Gaussian Process Models

Karl Krauth   **Edwin V. Bonilla**
The University of New South Wales
karl.krauth@gmail.com   e.bonilla@unsw.edu.au

Kurt Cutajar   **Maurizio Filippone**
EURECOM
{kurt.cutajar, maurizio.filippone}@eurecom.fr

Abstract

We investigate the capabilities and limitations of Gaussian process (GP) models by jointly exploring three complementary directions: (i) scalable and statistically efficient inference; (ii) flexible kernels; and (iii) objective functions for hyperparameter learning alternative to the marginal likelihood. Our approach outperforms all previous GP methods on the MNIST dataset; performs comparatively to kernel-based methods using the RECTANGLES-IMAGE dataset; and breaks the 1% error-rate barrier in GP models on the MNIST8M dataset, while showing unprecedented scalability (8 million observations) in GP classification. Overall, our approach represents a significant breakthrough in kernel methods and GP models, bridging the gap between deep learning and kernel machines.

1 INTRODUCTION

Recent advances in deep learning (DL) have revolutionized the application of machine learning in areas such as computer vision speech recognition and natural language processing (LeCun et al., 2015). Although certain kernel-based methods have also been successful in such domains (Cho and Saul, 2009; Mairal et al., 2014), it is still unclear whether these methods can indeed catch up with the recent DL breakthroughs.

Aside from the benefits obtained from using compositional representations, we believe that the main components contributing to the success of DL techniques are: (i) their scalability to large datasets and efficient computation via GPUs; (ii) their large representational power; and (iii) the use of well-targeted objective functions for the problem at hand.

In the kernel world, Gaussian process (GP; Rasmussen and Williams, 2006) models are attractive because they are elegant Bayesian nonparametric approaches to learning from data. Nevertheless, besides the limitations intrinsic to local kernel machines (Bengio et al., 2005), it is clear that GP-based methods have not fully explored the desiderata mentioned above.

Firstly, with regards to (i) scalability, despite recent advances in inducing-variable approaches and variational inference in GP models (Titsias, 2009; Hensman et al., 2013, 2015a; Dezfouli and Bonilla, 2015), the study of truly large datasets in problems other than regression and the investigation of GPU-based acceleration in GP models are still under-explored areas. We note that these issues are shared by non-probabilistic kernel methods such as support vector machines (SVMs).

Furthermore, concerning (ii) representational power, kernel methods have been plagued by the overuse of very limited kernels such as the squared exponential kernel, also known as the radial-basis-function (RBF) kernel. Even worse, it is common in methods such as SVMs to constrain the kernel to having a single length-scale (or bandwidth in SVM parlance) shared across all inputs instead of using a different length-scale for each dimension. The latter approach is commonly known as automatic relevance determination (ARD).

Finally, and perhaps most importantly, GP methods have largely ignored defining (iii) well-targeted objective functions, and instead have focused heavily on optimizing the marginal likelihood (Rasmussen and Williams, 2006, Ch. 5). This includes variational approaches that optimize a lower bound of the marginal likelihood (Titsias, 2009). Although different approaches for hyperparameter learning in GP models are discussed by Rasmussen and Williams (2006, §5.4.2), and indeed carried out by Sundararajan and Keerthi (2001); Sundararajan

et al. (2007); Sundararajan and Keerthi (2008) and more recently by Vehtari et al. (2016), their performance results are somewhat limited by their disregard for other directions of improvement (scalability and greater representational power) mentioned above.

In this work we push the capabilities of GP-models, while also investigating their limitations, by addressing the above issues jointly. In particular,

1. we develop scalable and statistically efficient inference methods for GP models using GPU computation and stochastic variational inference along with the reparameterization trick (Kingma and Welling, 2014);

2. we investigate the flexibility of models using ARD kernels for high-dimensional problems, as well as "deep" ARC-COSINE kernels (Cho and Saul, 2009); and

3. we study the impact of employing a leave-one-out-based objective function on hyperparameter learning.

As we rely on automatic differentiation (Baydin et al., 2015) for the implementation of our model in Tensor-Flow (Abadi et al., 2015), we refer to our approach as AutoGP. Our most significant results show that AutoGP: (i) has superior performance to previous GP-based methods on the MNIST dataset; (ii) achieves comparable performance to previous kernel-based methods using the RECTANGLES-IMAGE dataset; and (iii) breaks the 1% error-rate barrier in GP models using the MNIST8M dataset, while showing unprecedented scalability (8 million observations) in multiclass GP classification.

Overall, AutoGP represents a significant breakthrough in kernel methods in general and in GP models in particular, bridging the gap between deep learning approaches and kernel machines.

2 RELATED WORK

The majority of works related to scalable kernel machines primarily target issues pertaining to the storage and computational requirements of algebraic operations involving kernel matrices. Low-rank approximations are ubiquitous among these approaches, with the Nyström approximation being one of the most popular methods in this category (Williams and Seeger, 2001). Nevertheless, most of these approximations can be understood within the unifying probabilistic framework of Quiñonero-Candela and Rasmussen (2005).

The optimization of inducing inputs using Nyström approximations for GPs made significant progress with the work on sparse-GP variational inference in Titsias (2009). This exposed the possibility to develop stochastic gradient optimization for GP models (Hensman et al., 2013) and handle non-Gaussian likelihoods (Nguyen and Bonilla, 2014; Dezfouli and Bonilla, 2015; Hensman et al., 2015a; Sheth et al., 2015), which has sparked interest in the implementation of scalable inference frameworks such as those in Hensman et al. (2016) and Deisenroth and Ng (2015). These developments greatly improved the generality and applicability of GP models to a variety of applications, and our work follows from this line of ideas.

Other independent works have focused on kernel design (Wilson and Adams, 2013; Cho and Saul, 2009; Mairal et al., 2014; Wilson et al., 2016). This has sparked some debate as to whether kernel machines can actually compete with deep nets. Evidence suggests that this is possible; notable examples include the work by Huang et al. (2014); Lu et al. (2014). These works also provide insights into the aspects that make kernel methods less competitive to deep neural networks, namely lack of application specific kernels, and scalability at the price of poor approximations. These observations are corroborated by our experiments, which report how the combination of these factors impacts performance of GPs.

Because GPs are probabilistic kernel machines, it is natural to target the optimization of the marginal likelihood. However, alternative objective functions have also been considered, in particular the leave-one-out cross-validation (LOO-CV) error. Originally this was done by Sundararajan and Keerthi (2001) for regression; later extended by Sundararajan et al. (2007) to deal with sparse GP formulations; and broadened by Sundararajan and Keerthi (2008); Vehtari et al. (2016) to handle non-Gaussian likelihoods. While the results in these papers seem inconclusive as to whether this can generally improve performance, LOO-CV learning may allow to better accommodate for incorrect model specifications (Wahba, 1990, §4.8). Motivated by this observation, we explore this direction by extending the variational formulation to optimize a LOO-CV error. We also note that none of these previous approaches can actually handle large datasets.

3 GP MODELS

We are interested in supervised learning problems where we are given a training dataset of input-output pairs $\mathcal{D} = \{\mathbf{x}_n, \mathbf{y}_n\}_{n=1}^n$, with $\mathbf{x}_n$ being a D-dimensional input vector and $\mathbf{y}_n$ being a P-dimensional output. Our

main goal is to learn a probabilistic mapping from inputs to outputs so that for a new input $\mathbf{x}_\star$, we can estimate the probability of its associated label $p(\mathbf{y}_\star|\mathbf{x}_\star)$.

To this end, we follow a Gaussian process modeling approach (GP; Rasmussen and Williams, 2006), where latent functions are assumed to be distributed according to a GP, and observations are modeled via a suitable conditional likelihood given the latent functions. A function f_j is said to be distributed according to a Gaussian process with mean function $\mu_j(\mathbf{x})$ and covariance function $\kappa_j(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}_j)$, which we denote with $f_j \sim \mathcal{GP}(\mu_j(\mathbf{x}), \kappa_j(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}_j))$, if any subset of function values follow a Gaussian distribution.

Since we are dealing with multiple outputs $\{\mathbf{y}_n\}$, we follow the standard practice of considering Q underlying latent functions $\{f_j\}_{j=1}^{Q}$ which are drawn independently from zero-mean GP priors along with i.i.d. conditional likelihood models. Such a modeling approach encompasses a large class of machine learning problems, including GP-regression, GP-classification and modeling of count data (Rasmussen and Williams, 2006).

Consequently, when realized at the observed data, our probabilistic model is given by:

$$p(\mathbf{f}|\mathbf{X}, \boldsymbol{\theta}) = \prod_{j=1}^{Q} p(\mathbf{f}_{\cdot j}|\mathbf{X}, \boldsymbol{\theta}_j) = \prod_{j=1}^{Q} \mathcal{N}(\mathbf{f}_{\cdot j}; \mathbf{0}, \mathbf{K}_j), \quad (1)$$

$$p(\mathbf{y}|\mathbf{f}) = \prod_{n=1}^{N} p(\mathbf{y}_n|\mathbf{f}_{n\cdot}), \quad (2)$$

where $\mathbf{f}$ is the $(N \times Q)$-dimensional vector of latent function values; $\mathbf{f}_{\cdot j} = \{f_j(\mathbf{x}_n)\}_{n=1}^{N}$ denotes the latent values corresponding to the jth GP; $\mathbf{K}_j$ is the covariance matrix obtained by evaluating the covariance function $\kappa_j(\cdot, \cdot; \boldsymbol{\theta})$ at all pairs of training inputs; $\mathbf{y}$ is the vector of all $(N \times P)$ output observations, with $\mathbf{y}_n$ being nth output observation, and $\mathbf{f}_{n\cdot} = \{f_j(\mathbf{x}_n)\}_{j=1}^{Q}$ being the corresponding vector of latent function values. We refer to the covariance parameters $\boldsymbol{\theta}$ as the hyperparameters.

One commonly used covariance function is the so-called squared exponential (or RBF kernel):

$$\kappa(\mathbf{x}, \mathbf{x}'; \boldsymbol{\theta}) = \sigma^2 \exp\left(-\sum_{i=1}^{D} \frac{(x_i - x_i')^2}{\ell_i^2}\right), \quad (3)$$

where $\boldsymbol{\theta} = \{\sigma^2, \ell_1^2, \ldots, \ell_D^2\}$ and D is the input dimensionality. When all the length-scales, ℓ_i, are constrained to be the same we refer to the above kernel as *isotropic*; otherwise, we refer to it as the squared exponential covariance with automatic relevance determination (ARD).

3.1 INFERENCE PROBLEMS

The main inference problems in GP models are (i) estimation of the posterior over the latent functions $p(\mathbf{f}|\mathbf{X}, \mathbf{y}, \boldsymbol{\theta})$ and subsequent estimation of the predictive distribution $p(\mathbf{y}_\star|\mathbf{x}_\star, \mathbf{X}, \mathbf{y}, \boldsymbol{\theta})$; and (ii) inference over the hyperparameters $\boldsymbol{\theta}$.

For general likelihood models, both inference problems are analytically intractable, as they require the computation of non-trivial high-dimensional integrals. Furthermore, even for the simplest case when the conditional likelihood is Gaussian, evaluating the posterior and hyperparameter estimation is computationally prohibitive, as they entail time and memory requirements of $\mathcal{O}(N^3)$ and $\mathcal{O}(N^2)$ respectively.

In addition to dealing with general likelihoods and scalability issues, the performance of GP models (and kernel machines), is highly dependent on the covariance function (or kernel) used. Most previous work on GP methods and SVMs in the machine learning literature, use the squared exponential kernel in Equation (3). Furthermore, especially in high dimensions, this kernel is constrained to its isotropic version. As pointed out by Bengio et al. (2005), such an approach is severely limited by the curse of dimensionality when learning complex functions.

Finally, the estimation of hyperparameters based solely on marginal likelihood optimization can be very sensitive to model misspecification, with predictive approaches such as those in Sundararajan and Keerthi (2001) being seemingly more appealing for this task, especially in problems such as classification, where we are ultimately interested in having lower error rates and better calibrated predictive probabilities.

In the following sections, we show how to deal with the above issues, namely non-Gaussian likelihood models and scalability; more flexible kernels; and better objective functions for hyperparameter learning, with the endmost goal of improving performance significantly over current GP approaches.

4 VARIATIONAL INFERENCE

In this section we detail our method for scalable and statistically efficient inference in Gaussian process models with general likelihoods as specified by Equations (1) and (2). We adopt the variational framework of Dezfouli and Bonilla (2015), which we prefer over the MCMC method of Hensman et al. (2015b) as we avoid the sampling overhead which is especially significant in very large datasets.

4.1 AUGMENTED MODEL

In order to have a scalable inference framework, we augment our prior with M inducing variables $\{\mathbf{u}_{\cdot j}\}$ per latent process and corresponding inducing inputs $\{\mathbf{Z}_j\}$ such that,

$$p(\mathbf{u}) = \prod_{j=1}^{Q} \mathcal{N}(\mathbf{u}_{\cdot j}; \mathbf{0}, \mathbf{K}_{\mathbf{zz}}^{j}), \tag{4}$$

$$p(\mathbf{f}|\mathbf{u}) = \prod_{j=1}^{Q} \mathcal{N}(\mathbf{f}_{\cdot j}; \tilde{\boldsymbol{\mu}}_j, \widetilde{\mathbf{K}}_j), \text{ where} \tag{5}$$

$$\tilde{\boldsymbol{\mu}}_j = \mathbf{K}_{\mathbf{xz}}^{j}(\mathbf{K}_{\mathbf{zz}}^{j})^{-1}\mathbf{u}_{\cdot j}, \text{ and} \tag{6}$$

$$\widetilde{\mathbf{K}}_j = \mathbf{K}_{\mathbf{xx}}^{j} - \mathbf{A}_j\mathbf{K}_{\mathbf{zx}}^{j} \text{ with } \mathbf{A}_j = \mathbf{K}_{\mathbf{xz}}^{j}(\mathbf{K}_{\mathbf{zz}}^{j})^{-1}, \tag{7}$$

where $\mathbf{K}_{\mathbf{uv}}^{j}$ is the covariance matrix obtained by evaluating the covariance function $\kappa_j(\cdot, \cdot; \boldsymbol{\theta})$ at all pairwise columns of matrices $\mathbf{U}$ and $\mathbf{V}$. We note that this prior is equivalent to that defined in Equation (1), which is obtained by integrating out $\mathbf{u}$ from the joint $p(\mathbf{f}, \mathbf{u})$. However, as originally proposed by Titsias (2009), having an explicit representation of $\mathbf{u}$ will allow us to derive a scalable variational framework without additional assumptions on the train or test conditional distributions (Quiñonero-Candela and Rasmussen, 2005).

We now define our approximate joint posterior as:

$$q(\mathbf{f}, \mathbf{u}|\boldsymbol{\lambda}) \overset{\text{def}}{=} p(\mathbf{f}|\mathbf{u})q(\mathbf{u}|\boldsymbol{\lambda}), \text{ with} \tag{8}$$

$$q(\mathbf{u}|\boldsymbol{\lambda}) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{Q} \mathcal{N}(\mathbf{u}_{\cdot j}; \mathbf{m}_{kj}, \mathbf{S}_{kj}), \tag{9}$$

where $p(\mathbf{f}|\mathbf{u})$ is the conditional prior given in Equation (5) and $q(\mathbf{u}|\boldsymbol{\lambda})$ is our variational posterior with $\boldsymbol{\lambda} = \{\pi_k, \mathbf{m}_{kj}, \mathbf{S}_{kj}\}$, which we will refer to as variational parameters. Note that we assume a variational posterior in the form of a mixture-of-Gaussians for added flexibility compared to using a single distribution.

4.2 EVIDENCE LOWER BOUND

Such a definition of the variational posterior allows for simplification of the log-evidence lower bound ($\mathcal{L}_{\text{elbo}}$) such that no $\mathcal{O}(N^3)$ operations are required,

$$\mathcal{L}_{\text{elbo}}(\boldsymbol{\lambda}, \boldsymbol{\theta}) = -\text{KL}(q(\mathbf{u}|\boldsymbol{\lambda})\|p(\mathbf{u})) +$$
$$\sum_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathbb{E}_{q_{k(n)}(\mathbf{f}_{n\cdot}|\boldsymbol{\lambda}_k)}[\log p(\mathbf{y}_n|\mathbf{f}_{n\cdot})], \tag{10}$$

where $\text{KL}(q\|p)$ denotes the KL divergence between distributions q and p; $\mathbb{E}_{p(X)}[g(X)]$ denotes the expectation of $g(X)$ over distribution $p(X)$; and $q_{k(n)}(\mathbf{f}_{n\cdot}|\boldsymbol{\lambda}_k)$ is a Q-dimensional diagonal Gaussian with means and variances given by

$$b_{knj} = \mathbf{a}_{jn}^{T}\mathbf{m}_{kj}, \tag{11}$$

$$\sigma_{knj}^{2} = [\widetilde{\mathbf{K}}_j]_{n,n} + \mathbf{a}_{jn}^{T}\mathbf{S}_{kj}\mathbf{a}_{jn}, \tag{12}$$

where $\mathbf{a}_{jn} \overset{\text{def}}{=} [\mathbf{A}_j]_{:,n}$ denotes the M-dimensional vector corresponding to the nth column of matrix $\mathbf{A}_j$; $\widetilde{\mathbf{K}}_j$ and $\mathbf{A}_j$ are given in Equation (7); and, $[\widetilde{\mathbf{K}}_j]_{n,n}$ denotes the (n, n)th entry of matrix $\widetilde{\mathbf{K}}_j$.

In order to compute the log-evidence lower bound in Equation (10) and its gradients, we use Jensen's inequality to bound the KL term (as in Dezfouli and Bonilla, 2015) and estimate the expected likelihood term using Monte Carlo (MC).

4.3 THE REPARAMETERIZATION TRICK

One of the key properties of the log-evidence lower bound in Equation (10) is its decomposition as a sum of expected likelihood terms on the individual observations. This allows for the applicability of stochastic optimization methods and, therefore, scalability to very large datasets.

Due to the expectation term in the expression of $\mathcal{L}_{\text{elbo}}$, gradients of $\mathcal{L}_{\text{elbo}}$ need to be estimated. Dezfouli and Bonilla (2015) follow a similar approach to that in general black-box variational methods (Ranganath et al., 2014), and use the property $\nabla_{\boldsymbol{\lambda}}\mathbb{E}_{q(\mathbf{f}|\boldsymbol{\lambda})}[\log p(\mathbf{y}|\mathbf{f})] = \mathbb{E}_{q(\boldsymbol{\lambda})}[\nabla_{\boldsymbol{\lambda}}\log q(\mathbf{f}|\boldsymbol{\lambda})\log p(\mathbf{y}|\mathbf{f})]$ and Monte Carlo (MC) sampling to produce unbiased estimates of these gradients.

While such an approach is truly black-box in that it does not require detailed knowledge of the conditional likelihood and its gradients, the estimated gradients may have significantly large variance, which could hinder the optimization process. In fact, Dezfouli and Bonilla (2015) and Bonilla et al. (2016) use variance-reduction techniques (Ross, 2006, §8.2) to ameliorate this effect. Nevertheless, the experiments in Bonilla et al. (2016) on complex likelihood functions such as those in Gaussian process regression networks (GPRNs; Wilson et al., 2012) indicate that such techniques may be insufficient to control the stability of the optimization process, and a large number of MC samples may be required.

In contrast, if one is willing to relax the constraints of a truly black-box approach by having access to the imple-

mentation of the conditional likelihood and its gradients, then a simple trick proposed by Kingma and Welling (2014) can significantly reduce the variance in the gradients and make the stochastic optimization of $\mathcal{L}_{\text{elbo}}$ much more practical. This has come to be known as the *reparameterization trick*.

We now present explicit expressions of our estimated $\mathcal{L}_{\text{elbo}}$, focusing on an individual expected likelihood term in Equation (10). Thus, the individual expectations can be estimated as:

$$\epsilon_{knj} \sim \mathcal{N}(0,1), \tag{13}$$

$$f_{nj}^{(k,i)} = b_{knj} + \sigma_{knj}\epsilon_{knj}, \quad j = 1, \ldots, Q, \tag{14}$$

$$\widehat{\mathcal{L}}_{\text{ell}}^{(n)} = \frac{1}{S} \sum_{k=1}^{K} \pi_k \sum_{i=1}^{S} \log p(\mathbf{y}_n | \mathbf{f}_n^{(k,i)}), \tag{15}$$

where S is the number of MC samples and $f_{nj}^{(k,i)}$ is an element in the vector of latent functions $\mathbf{f}_{n\cdot}^{(k,i)}$.

4.4 MINI-BATCH OPTIMIZATION

We can now obtain unbiased estimates of the gradients of $\mathcal{L}_{\text{elbo}}$ to use in mini-batch stochastic optimization. In particular, for a mini-batch Ω of size B we have that the estimated gradient is given by:

$$\nabla_{\boldsymbol{\eta}} \widehat{\mathcal{L}}_{\text{elbo}} = -\nabla_{\boldsymbol{\eta}} \text{KL}(q(\mathbf{u}|\boldsymbol{\lambda}) \| p(\mathbf{u})) + \frac{N}{B} \sum_{n \in \Omega} \nabla_{\boldsymbol{\eta}} \widehat{\mathcal{L}}_{\text{ell}}^{(n)},$$

where $\boldsymbol{\eta} \in \{\boldsymbol{\lambda}, \boldsymbol{\theta}\}$, and $\boldsymbol{\theta}$ now includes, in addition to the hyperparameters, the inducing inputs across all latent processes $\mathbf{Z}_j$. The corresponding gradients are obtained through automatic differentiation using TensorFlow.

4.5 FULL APPROXIMATE POSTERIOR

An important characteristic of our approach is that, unlike most previous work that uses stochastic optimization in variational inference along with the reparameterization trick (see e.g. Kingma and Welling, 2014; Dai et al., 2016), we do not require a fully factorized approximate posterior to estimate our gradients using only simple scalar transformations of uncorrelated Gaussian samples, as given in Equation (14). Indeed, the posterior over the latent functions is fully correlated and is given by

$$q(\mathbf{f}|\boldsymbol{\lambda}) = \sum_{k=1}^{K} \pi_k \prod_{j=1}^{Q} \mathcal{N}(\mathbf{f}_{\cdot j}; \mathbf{b}_{kj}, \boldsymbol{\Sigma}_{kj}), \text{ where} \tag{16}$$

$$\mathbf{b}_{kj} = \mathbf{A}_j \mathbf{m}_{kj}, \text{ and} \tag{17}$$

$$\boldsymbol{\Sigma}_{kj} = \widetilde{\mathbf{K}}_j + \mathbf{A}_j \mathbf{S}_{kj} \mathbf{A}_j^T. \tag{18}$$

This result is a property inherited from the original framework of Nguyen and Bonilla (2014) and Dezfouli and Bonilla (2015), who showed that, even when having a full approximate posterior, only expectations over univariate Gaussians are required in order to estimate the expected log likelihood term in $\mathcal{L}_{\text{elbo}}$. A less general result, for the case of one latent function and a single Gaussian posterior, has been shown by Nickisch and Rasmussen (2008); Opper and Archambeau (2009).

5 FLEXIBLE KERNELS

We now turn our attention to increase the flexibility of GP models through the investigation of kernels beyond the isotropic RBF kernel. This is a complementary direction for exploring the capabilities of GP models, but it benefits from the results of the previous section in terms of large-scale inference and computation of gradients via (automatic) back-propagation.

For the RBF kernel, we limit our experiments to the automatic relevance determination (ARD) setting in Equation (3), including problems of large input dimensionality such as MNIST and CIFAR10.

Furthermore, given the interesting results reported by Cho and Saul (2009), we investigate their ARC-COSINE kernel, which was proposed as a kernel that mimics the computation of neural networks. In particular, they define an ARC-COSINE kernel of degree d and depth l using the following recursion:

$$\kappa_d^{(l+1)}(\mathbf{x}, \mathbf{x}') = \frac{1}{\pi} \left(\kappa_d^{(l)}(\mathbf{x}, \mathbf{x}) \kappa_d^{(l)}(\mathbf{x}', \mathbf{x}') \right)^{d/2} J_d(\phi_d^{(l)}),$$

where the kernel at depth $l = 1$ is $\kappa_d^{(1)}(\mathbf{x}, \mathbf{x}') = \frac{1}{\pi} \|\mathbf{x}\|^d \|\mathbf{x}'\|^d J_d(\phi)$ and the angle corresponding to degree d and depth l is given by

$$\phi_d^{(l)} = \cos^{-1} \left(\kappa_d^{(l)}(\mathbf{x}, \mathbf{x}') \left(\kappa_d^{(l)}(\mathbf{x}, \mathbf{x}) \kappa_d^{(l)}(\mathbf{x}', \mathbf{x}') \right)^{-1/2} \right).$$

All the angular dependences are then modeled with the function

$$J_d(\phi) = (-1)^d (\sin \phi)^{2d+1} \left(\frac{1}{\sin \phi} \frac{\partial}{\partial \phi} \right)^d \left(\frac{\pi - \phi}{\sin \phi} \right),$$

with the base angle defined as $\phi = \cos^{-1} \left(\frac{\mathbf{x} \cdot \mathbf{x}'}{\|\mathbf{x}\| \|\mathbf{x}'\|} \right)$.

6 LEAVE-ONE-OUT LEARNING

Here we focus on the average leave-one-out log predictive probability for hyperparameter learning, as an alternative to optimization of the marginal likelihood. This

Algorithm 1 $\mathcal{L}_{\text{oo}}$-based hyperparameter learning.

> **repeat**
>> **repeat**
>>> $(\boldsymbol{\theta}, \boldsymbol{\lambda}) \leftarrow (\boldsymbol{\theta}, \boldsymbol{\lambda}) + \alpha \nabla_{\boldsymbol{\theta}, \boldsymbol{\lambda}} \widehat{\mathcal{L}}_{\text{elbo}}(\boldsymbol{\theta}, \boldsymbol{\lambda})$
>> **until** satisfied
>> **repeat**
>>> $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \nabla_{\boldsymbol{\theta}} \widehat{\mathcal{L}}_{\text{oo}}(\boldsymbol{\theta})$
>> **until** satisfied
> **until** satisfied

can be particularly useful in problems such as classification where one is mainly interested in having lower error rates and better calibrated predictive probabilities.

This objective function is obtained by leaving one datapoint out of the training set and computing its log predictive probability when training on the other points; the resulting values are then averaged across all the datapoints. Interestingly, in our GP model this can be computed without the need for training N different models, as the resulting expression is given by:

$$\mathcal{L}_{\text{oo}}(\boldsymbol{\theta}) \approx -\frac{1}{N} \sum_{n=1}^{N} \log \int \frac{q(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\lambda}, \boldsymbol{\theta})}{p(\mathbf{y}_n|\mathbf{f}_{n\cdot})} \, d\mathbf{f}_{n\cdot}, \quad (19)$$

where the approximation stems from using the variational marginal posterior $q(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})$ instead of the true marginal posterior $p(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})$ for datapoint n. We also note that we have made explicit the dependency of the posterior on all the data. The derivation of this expression is given in the supplement. Since $\mathcal{L}_{\text{oo}}(\boldsymbol{\theta})$ contains an expectation with respect to the marginal posterior we can also estimate it via MC sampling.

Equation (19) immediately suggests an alternating optimization scheme where we estimate the approximate posterior $q(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})$ through optimization of $\mathcal{L}_{\text{elbo}}$, as described in §4, and then learn the hyperparameters via optimization of $\mathcal{L}_{\text{oo}}$. Algorithm 1 illustrates such an alternating scheme for hyperparameter learning in our model using the leave-one-out objective and vanilla stochastic gradient descent (SGD), where $\widehat{\mathcal{L}}_{\text{elbo}}$ and $\widehat{\mathcal{L}}_{\text{oo}}$ denote estimates of the corresponding objectives when using mini-batches.

7 EXPERIMENTS

We evaluate AutoGP across various datasets with number of observations ranging from $12,000$ to 8.1M and input dimensionality between 31 and 3072. The aim of this section is (i) to demonstrate the effectiveness of the reparameterization trick in reducing the number of samples needed to estimate the expected log likelihood; (ii)

to evaluate non-isotropic kernels across a large number of dimensions; (iii) to assess the performance obtained by LOO-CV hyperparameter learning; and (iv) to analyze the performance of AutoGP as a function of time. Details of the experimental set-up can be found in the supplementary material.

7.1 STATISTICAL EFFICIENCY

Bonilla et al. (2016) has shown that sparse variational approaches to GP models can be as accurate as MCMC (which are only applicable to small-scale problems) but orders of magnitude faster and as good as approaches that have been hard-coded for specific likelihood models (e.g. EP and quadratic lower bounds). Here we show that we can improve on the method of Bonilla et al. (2016) by enhancing its statistical efficiency.

Therefore, For our first experiment we look at the behavior of our model as we vary the number of samples used to estimate the expected log likelihood. We trained our model on the SARCOS dataset (Vijayakumar and Schaal, 2000), an inverse dynamics problem for a seven degrees-of-freedom anthropomorphic robot arm.

We used the Gaussian process regression network (GPRN) model of Wilson et al. (2012) as our likelihood function. Bonilla et al. (2016) found that GPRNs require a large number of samples ($10,000$) to yield stable optimization of the $\mathcal{L}_{\text{elbo}}$ in their method. This is most likely due to the increased variance induced by multiplying latent samples together. The high variance of this likelihood model makes it an ideal candidate for us to evaluate how the performance of AutoGP varies with the number of samples.

As shown in Figure 1, increasing the number of samples decreased the number of epochs it took for our model to converge; however, our model always converged to the same optimal value. This is in stark contrast to SAVIGP (Bonilla et al., 2016) which was able to converge to the optimal value of 0.0195 with $10,000$ samples, but converged to sub-optimal values (> 0.05) as the number of samples was lowered. This shows that our MC estimator is significantly more stable than SAVIGP's black-box estimator, without requiring any variance reduction methods. In practice, reducing the number of samples leads to an improved runtime. A gradient update using $10,000$ samples takes around 0.17 seconds, whereas an update using 10 samples only takes around 0.03 seconds, which makes up for the extra epochs needed to converge.
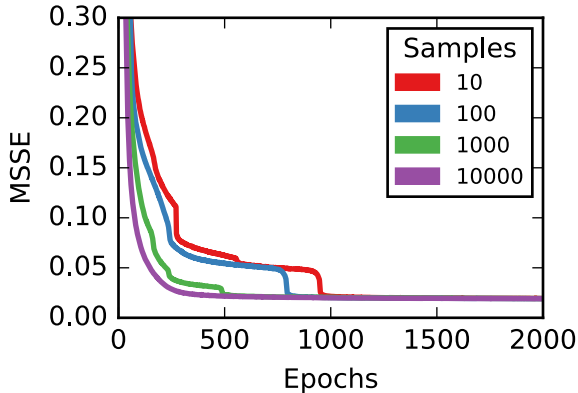
Figure 1: The mean standardized square error (MSSE) of AutoGP for different number of MC samples on the SARCOS dataset. The MSSE was averaged across all 7 joints and 50 inducing inputs were used to train the model. An epoch represents a single pass over the entire training set.

7.2 KERNEL PERFORMANCE

In this section we compare the performance of AutoGP across two different kernels on a high-dimensional dataset.

We trained our model on the RECTANGLES-IMAGE dataset (Larochelle et al., 2007) which is a binary classification task that was created to compare shallow models (e.g. SVMs), and deep learning architectures. The task involves recognizing whether a rectangle has a larger width or height. Randomized images were overlaid behind and over the rectangles, which makes this task particularly challenging.

We compare the multilayer arc-cosine kernel (ARC-COSINE) described by Cho and Saul (2009) with an RBF kernel with automatic relevance determination (RBF-ARD). ARC-COSINE was devised to mimic the architecture of deep neural networks through successive kernel compositions, and has shown good results when combined with non-Bayesian kernel machines (Cho and Saul, 2009). Unlike ARC-COSINE, RBF-ARD is commonly used with Gaussian processes (Hensman et al., 2013). However, RBF-ARD has not been applied to large-scale datasets with a large input dimensionality due to limitations in scalability. Given that our implementation uses automatic differentiation, and is thus able to efficiently compute gradients, we can use RBF-ARD with no noticeable performance overhead.

We trained our model using 10, 200, and 1000 inducing points across both kernels. We experimented with various different depths and degrees for ARC-COSINE and
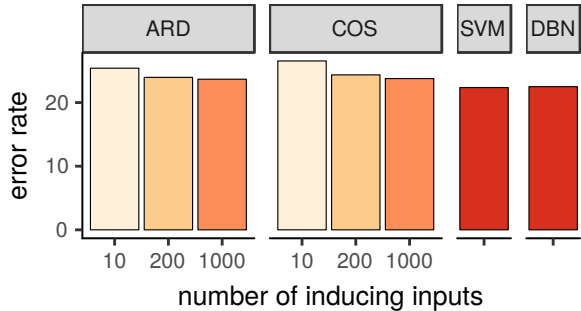

Figure 2: Error rates for binary classification using a logistic likelihood on RECTANGLES-IMAGE. ARD and COS refer to AutoGP with a RBF-ARD kernel, and a 3-layer ARC-COSINE kernel of degree 1 respectively. DBN refers to a 3-layer deep belief network and SVM denotes a support vector machine with a ARC-COSINE kernel of depth 6 and degree 0.

found that 3 layers and a degree of 1 worked best with AutoGP. As such, we ran all ARC-COSINE experiments with these settings.

For reference, Cho and Saul (2009) report an error rate of 22.36% using SVMs with ARC-COSINE. Larochelle et al. (2007) report error rates of 24.04% on SVMs with a simple RBF kernel and 22.50% with a deep belief network with 3 hidden layers (DBN).

Figure 2 shows that AutoGP is competitive with previous approaches, except when using 10 inducing points. Our results also show that RBF-ARD performs comparatively (and sometimes better) than ARC-COSINE. This is likely because ARC-COSINE is better suited to deep architectures, such as the multilayer kernel machine of Cho and Saul (2009).

7.3 LOO-CV HYPERPARAMETER LEARNING

In this section we analyze our approach when learning hyperparameters via optimization of the leave-one-out objective ($\widehat{\mathcal{L}}_{\text{oo}}$), as described in §6, and compare it with the standard variational method that learns hyperparameters via sole optimization of the variational objective ($\widehat{\mathcal{L}}_{\text{elbo}}$). To this end, we use the MNIST dataset using 10, 200, and 1000 inducing points and the RBF-ARD kernel across all settings.

Our results indicate that optimizing the leave-one-out objective leads to a significant performance gain. While Figure 3 shows the corresponding error rates, we refer the reader to the supplementary material for similar re-
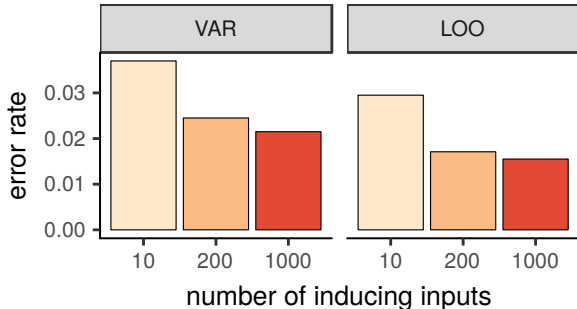
Figure 3: Error rates of AutoGP for multiclass classification using a softmax likelihood on MNIST. VAR refers to learning using only the variational objective $\widehat{\mathcal{L}}_{\text{elbo}}$, while LOO refers to learning hyperparameters using the leave-one-out objective $\widehat{\mathcal{L}}_{\text{oo}}$.
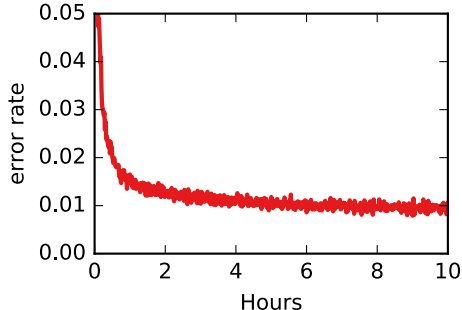

Figure 4: Error rate of AutoGP over time for multiclass classification using a softmax likelihood on MNIST8M. Training was done with 200 inducing points and $\mathcal{L}_{\text{oo}}$ for hyperparameter learning.

sults on negative log probabilities (NLPs). In summary, at 1000 inducing points our LOO-CV approach attains an error rate of 1.55% and a mean NLP of 0.061; At 200 inducing points it obtains an error rate of 1.71% and a mean NLP of 0.063. This represents a 40%–45% decrease in mean NLP and error rate from the equivalent model that only optimized the variational objective. Furthermore, our results outperform the work of Hensman et al. (2015b), Gal et al. (2014), and Bonilla et al. (2016) who reported error rates of 1.96%, 5.95% and 2.74% respectively, that are the best results reported on this data set in the GP literature.

We have shown that our approach reduces the error rate on the test set down to 1.55% without artificially extending the MNIST dataset through transformations, or doing any kind of pre-processing of the data. These results, while not on par with state-of-the-art vision results, show that Gaussian process models can perform competitively with non-Bayesian methods, while solving the harder problem of full posterior estimation. For example, standard convolutional neural networks achieve an error rate of 1.19% without any pre-processing (Benenson, 2013). We will show in the following section how we can further improve our results by artificially extending the training data on MNIST through distortions and affine transformations.

7.4 AUTOGP ALL TOGETHER

We finish by evaluating the performance of AutoGP using all its main components simultaneously, namely statistical efficiency, flexible kernels and LOO-CV hyperparameter learning, using three datasets: RECTANGLES-IMAGE, CIFAR10, and MNIST8M. Our goal is to gain

further insights as to how our model compares against other works in the machine learning community. As before, we optimized our model with 200 inducing points and used the RBF-ARD kernel across all experiments.

We begin by looking at the RECTANGLES-IMAGE dataset which we have already investigated in Section 7.2. Optimizing $\widehat{\mathcal{L}}_{\text{oo}}$ slightly improves performance in this case, with a reduction in error rate from 24.06% to 24.04% and a reduction in mean NLP from 0.495 to 0.485, when using 200 inducing points. The error rate can be further reduced to 23.6% when using 1,000 inducing points (with corresponding NLP of 0.478). As a reference, performance of recent deep architectures is: deep trans-layer autoencoder networks (ER = 13.01%, Zhu et al., 2015), PCANet (ER = 13.94%, Chan et al., 2014), and two-layer stacked contractive auto-encoders (ER = 21.54%, Rifai et al., 2011). Bruna and Mallat (2013) have reported results using invariant scattering convolutional networks with an error rate of 8.01%.

We now turn our attention to the MNIST8M (Loosli et al., 2007) dataset, which artificially extends the MNIST dataset to 8.1 million training points by pseudo-randomly transforming existing MNIST images. To the best of our knowledge, no other GP models have been shown to scale to such a large dataset. We were able to achieve an error rate of 0.89% and a mean NLP of 0.033, breaking the 1% error-rate barrier in GP models. This shows that our model is able to perform on par with deep architectures with some pre-processing of the MNIST dataset. We also note that we were able to rapidly converge given the dataset size. As shown in Figure 4 AutoGP achieves near optimal results within the first hour, and converges to the optimal value within 10 hours. As a reference, Henao and Winther (2012) achieved similar performance (0.86% error rate) on MNIST when con-

sidering an augmented active set (which is analogous to the concept of inducing inputs/variables) and a 9th-degree polynomial covariance function.

Finally, we look at CIFAR10, a multiclass image classification task and we find that results are less conclusive. We achieve an error rate of 44.95% and a mean NLP of 1.33, which is well below the state-of-the-art, with some models achieving less than 10% error rate. However, we note that we still compare favorably to SVMs which have been shown to achieve error rates of 57.7% (Le et al., 2013). We also perform on par with older deep architectures (Rifai et al., 2011). We believe these results provide a strong motivation for investigating application-specific kernels (such as those studied by Mairal et al., 2014) in GP models.

Table 1: Test error rate and mean NLP of AutoGP trained with 200 inducing points on various datasets. The $\widehat{\mathcal{L}}_{oo}$ objective was used for hyperparameter learning.

Dataset	Error rate	mean NLP
RECTANGLES-IMAGE	24.06%	0.485
MNIST8M	0.89%	0.033
CIFAR10	44.95%	1.333

8 CONCLUSION & DISCUSSION

We have developed AutoGP, an inference framework for GP models that pushes the performance of current GP methods by exploring three complementary directions: (i) scalable and statistically efficient variational inference; (ii) flexible kernels; and (iii) objective functions for hyperparameter learning alternative to the marginal likelihood.

We have shown that our framework outperforms all previous GP approaches on MNIST; achieves comparable performance to previous kernel approaches on the RECTANGLES-IMAGE dataset; and can break the 1% error barrier using MNIST8M. Overall, this represents a significant breakthrough in Gaussian process methods and kernel machines. While our results on CIFAR10 are well below the state-of-the-art achieved with deep learning, we believe we can further reduce the gap between kernel machines and deep architectures by using application-specific kernels such as those recently proposed by Mairal et al. (2014).

One of the latest works that investigates LOO-CV and its connections with other information criteria for model selection can be found in Vehtari et al. (2017). We believe

these connections are under-explored in the GP literature, where the marginal likelihood is the preferred choice.

References

Martín Abadi, Ashish Agarwal, Paul Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767*, 2015.

Rodrigo Benenson. What is the class of this image? discover the current state of the art in objects classification. http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html, 2013. [Online; accessed 13-October-2016].

Yoshua Bengio, Olivier Delalleau, and Nicolas L Roux. The curse of highly variable functions for local kernel machines. In *NIPS*, 2005.

Edwin V Bonilla, Karl Krauth, and Amir Dezfouli. Generic inference in latent Gaussian process models. *arXiv preprint arXiv:1609.00577*, 2016.

Joan Bruna and Stephane Mallat. Invariant scattering convolution networks. *TPAMI*, pages 1872–1886, 2013.

Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. PCANet: A simple deep learning baseline for image classification? *arXiv preprint arXiv:1404.3606*, 2014.

Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *NIPS*. 2009.

Zhenwen Dai, Andreas Damianou, Javier González, and Neil Lawrence. Variational Auto-encoded Deep Gaussian Processes. In *ICLR*, 2016.

Marc Peter Deisenroth and Jun Wei Ng. Distributed Gaussian processes. In *ICML*, 2015.

Amir Dezfouli and Edwin V. Bonilla. Scalable inference for Gaussian process models with black-box likelihoods. In *NIPS*. 2015.

Yarin Gal, Mark van der Wilk, and Carl Rasmussen. Distributed variational inference in sparse Gaussian process regression and latent variable models. In *NIPS*. 2014.

Ricardo Henao and Ole Winther. Predictive active set selection methods for Gaussian processes. *Neurocomputing*, 80: 10–18, 2012.

James Hensman, Nicolo Fusi, and Neil D Lawrence. Gaussian processes for big data. In *UAI*, 2013.

James Hensman, Alexander Matthews, and Zoubin Ghahramani. Scalable variational Gaussian process classification. In *AISTATS*, 2015a.

James Hensman, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani. MCMC for variationally sparse Gaussian processes. In *NIPS*. 2015b.

James Hensman, Alexander G. de G. Matthews, Alexis Boukouvalas, Keisuke Fujii, Pablo Leon, Valentine Svensson, and Mark van der Wilk. GPflow. https://github.com/GPflow/GPflow, 2016.

Po-Sen Huang, Haim Avron, Tara N. Sainath, Vikas Sindhwani, and Bhuvana Ramabhadran. Kernel methods match deep neural networks on TIMIT. In *ICASSP*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *ICML*, 2007.

Quoc V. Le, Tamás Sarlós, and Alexander J. Smola. Fastfood - computing Hilbert space expansions in loglinear time. In *ICML*, 2013.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Gaëlle Loosli, Stéphane Canu, and Léon Bottou. Training invariant support vector machines using selective sampling. In *Large Scale Kernel Machines*, pages 301–320. MIT Press, Cambridge, MA., 2007.

Zhiyun Lu, Avner May, Kuan Liu, Alireza Bagheri Garakani, Dong Guo, Aurélien Bellet, Linxi Fan, Michael Collins, Brian Kingsbury, Michael Picheny, and Fei Sha. How to scale up kernel methods to be as good as deep neural nets. *CoRR*, abs/1411.4000, 2014.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *NIPS*, pages 2627–2635, 2014.

Trung V. Nguyen and Edwin V. Bonilla. Automated variational inference for Gaussian process models. In *NIPS*. 2014.

Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary Gaussian process classification. *JMLR*, 9(10), 2008.

Manfred Opper and Cédric Archambeau. The variational Gaussian approximation revisited. *Neural Computation*, 21 (3):786–792, 2009.

Joaquin Quiñonero-Candela and Carl Edward Rasmussen. A unifying view of sparse approximate Gaussian process regression. *JMLR*, 6:1939–1959, 2005.

Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black box variational inference. In *AISTATS*, 2014.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. The MIT Press, 2006.

Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *ICML*, 2011.

Sheldon M Ross. *Simulation*. Burlington, MA: Elsevier, 2006.

Rishit Sheth, Yuyang Wang, and Roni Khardon. Sparse variational inference for generalized GP models. In *ICML*, 2015.

S Sundararajan and S Sathiya Keerthi. Predictive approaches for choosing hyperparameters in Gaussian processes. *Neural Computation*, 13(5):1103–1118, 2001.

S. Sundararajan and S. Sathiya Keerthi. Predictive approaches for Gaussian process classifier model selection. Technical report, 2008.

S. Sundararajan, S. Sathiya Keerthi, and Shirish Shevade. Predictive approaches for sparse Gaussian process regression. Technical report, 2007.

Michalis Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *AISTATS*, 2009.

Aki Vehtari, Tommi Mononen, Ville Tolvanen, Tuomas Sivula, and Ole Winther. Bayesian leave-one-out cross-validation approximations for Gaussian latent variable models. *Journal of Machine Learning Research*, 17(103):1–38, 2016.

Aki Vehtari, Andrew Gelman, and Jonah Gabry. Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, 27(5):1413–1432, Sep 2017.

Sethu Vijayakumar and Stefan Schaal. Locally weighted projection regression: An O(n) algorithm for incremental real time learning in high dimensional space. In *ICML*, 2000.

Grace Wahba. *Spline models for observational data*. CBMS-NSF Regional Conference series in applied mathematics. Society for Industrial and Applied Mathematics, 1990.

Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *NIPS*, 2001.

Andrew G. Wilson, David A. Knowles, and Zoubin Ghahramani. Gaussian process regression networks. In *ICML*, 2012.

Andrew Gordon Wilson and Ryan Prescott Adams. Gaussian process kernels for pattern discovery and extrapolation. In *ICML*, pages 1067–1075, 2013.

Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P. Xing. Deep kernel learning. In *AISTATS*, 2016.

Wentao Zhu, Jun Miao, Laiyun Qing, and Xilin Chen. Deep trans-layer unsupervised networks for representation learning. *arXiv preprint arXiv:1509.08038*, 2015.

AutoGP: Exploring the Capabilities and Limitations of Gaussian Process Models — Supplementary Material

Karl Krauth   **Edwin V. Bonilla**   **Kurt Cutajar**   **Maurizio Filippone**

The University of New South Wales   EURECOM

karl.krauth@gmail.com   e.bonilla@unsw.edu.au   {kurt.cutajar, maurizio.filippone}@eurecom.fr

1   DERIVATION OF LEAVE-ONE-OUT OBJECTIVE

In this section we derive an expression for the leave-one-out objective and show that this does not require training of N models. A similar derivation can be found in Vehtari et al. (2016). Let $\mathcal{D}_{\neg n} = \{\mathbf{X}_{\neg n}, \mathbf{y}_{\neg n}\}$ be the dataset resulting from removing observation n. Then our leave-one-out objective is given by:

$$\mathcal{L}_{\text{oo}}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{n=1}^{N} \log p(\mathbf{y}_n|\mathbf{x}_n, \mathcal{D}_{\neg n}, \boldsymbol{\theta}). \tag{1}$$

We now that the marginal posterior can be computed as:

$$p(\mathbf{f}_{n\cdot}|\mathcal{D}) = p(\mathbf{f}_{n\cdot}|\mathbf{X}_{\neg n}, \mathbf{y}_{\neg n}, \mathbf{x}_n, \mathbf{y}_n) = \frac{p(\mathbf{y}_n|\mathbf{f}_{n\cdot})p(\mathbf{f}_{n\cdot}|\mathbf{x}_n, \mathcal{D}_{\neg n})}{p(\mathbf{y}_n|\mathbf{x}_n, \mathcal{D}_{\neg n}, \boldsymbol{\theta})} \tag{2}$$

and re-arranging terms

$$\int p(\mathbf{f}_{n\cdot}|\mathbf{x}_n, \mathcal{D}_{\neg n}, \boldsymbol{\theta})d\mathbf{f}_{n\cdot} = \int \frac{p(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})p(\mathbf{y}_n|\mathbf{x}_n, \mathcal{D}_{\neg n}, \boldsymbol{\theta})}{p(\mathbf{y}_n|\mathbf{f}_{n\cdot})}d\mathbf{f}_{n\cdot} \tag{3}$$

$$p(\mathbf{y}_n|\mathbf{x}_n, \mathcal{D}_{\neg n}, \boldsymbol{\theta}) = 1 / \int \frac{p(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})}{p(\mathbf{y}_n|\mathbf{f}_{n\cdot})}d\mathbf{f}_{n\cdot} \tag{4}$$

$$\log p(\mathbf{y}_n|\mathbf{x}_n, \mathcal{D}_{\neg n}; \boldsymbol{\theta}) = -\log \int \frac{p(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})}{p(\mathbf{y}_n|\mathbf{f}_{n\cdot})}d\mathbf{f}_{n\cdot}, \tag{5}$$

and substituting this expression in Equation (1) we have

$$\mathcal{L}_{\text{oo}}(\boldsymbol{\theta}) = -\frac{1}{N}\sum_{n=1}^{N} \log \int p(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})\frac{1}{p(\mathbf{y}_n|\mathbf{f}_{n\cdot})}\, d\mathbf{f}_{n\cdot}. \tag{6}$$

We see that the objective only requires estimation of the marginal posterior $p(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})$, which we can approximate using variational inference, hence:

$$\mathcal{L}_{\text{oo}}(\boldsymbol{\theta}) \approx -\frac{1}{N}\sum_{n=1}^{N} \log \int q(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})\frac{1}{p(\mathbf{y}_n|\mathbf{f}_{n\cdot})}\, d\mathbf{f}_{n\cdot}, \tag{7}$$

where $q(\mathbf{f}_{n\cdot}|\mathcal{D}, \boldsymbol{\theta})$ is our approximate variational posterior.

Table 1: The datasets used in the experiments and the corresponding models used. N_{train}, N_{test}, D are the number of training points, test points and input dimensions respectively.

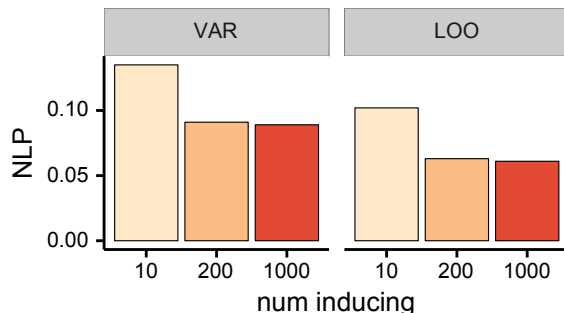| Dataset	N_{train}	N_{test}	D	Model
SARCOS	$44,484$	$4,449$	21	GPRN
RECTANGLES-IMAGE	$12,000$	$50,000$	784	Binary classification
MNIST	$60,000$	$10,000$	784	Multi-class classification
CIFAR10	$50,000$	$10,000$	3072	Multi-class classification
MNIST8M	8.1M	$10,000$	784	Multi-class classification

Figure 1: NLP for multiclass classification using a softmax likelihood model on the MNIST dataset. VAR shows the performance of AutoGP where all parameters are learned using only the variational objective $\widehat{\mathcal{L}}_{\text{elbo}}$, while LOO represents the performance of AutoGP when hyperparameters are learned using the leave-one-out objective $\widehat{\mathcal{L}}_{\text{oo}}$.

2 ADDITIONAL DETAILS OF EXPERIMENTS

2.1 EXPERIMENTAL SET-UP

The datasets used are described in Table 1. We trained our model stochastically using the RMSProp optimizer provided by TensorFlow (Abadi et al., 2015) with a learning rate of 0.003 and mini-batches of size 1000. We initialized inducing point locations by using the k-means clustering algorithm, and initialized the posterior mean to a zero vector, and the posterior covariances to identity matrices. When jointly optimizing $\widehat{\mathcal{L}}_{\text{oo}}$ and $\widehat{\mathcal{L}}_{\text{elbo}}$, we alternated between optimizing each objective for 100 epochs. Unless otherwise specified we used 100 Monte-Carlo samples to estimate the expected log likelihood term.

All timed experiments were performed on a machine with an Intel(R) Core(TM) i5-4460 CPU, 24GB of DDR3 RAM, and a GeForce GTX1070 GPU with TensorFlow 0.10rc.

2.2 ADDITIONAL RESULTS

Figure 1 shows the NLP for our evaluation of the LOO-CV-based hyperparameter learning. As with the error rates described in the main text, the NLP obtained with LOO-CV are significantly better than those obtained with a purely variational approach.

References

Martín Abadi, Ashish Agarwal, Paul Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

Aki Vehtari, Tommi Mononen, Ville Tolvanen, Tuomas Sivula, and Ole Winther. Bayesian leave-one-out cross-validation approximations for Gaussian latent variable models. *Journal of Machine Learning Research*, 17(103): 1–38, 2016.