Batch Normalized Deep Kernel Learning for Weight Uncertainty

Alex Lewandowski
University of Alberta
`lewandowski@ualberta.ca`

1 Introduction

There has been recent interest in Gaussian processes with neural network parameterized kernels, a framework referred to as deep kernel learning [1, 2, 3, 4, 5]. While the idea of 'deep-ifying' kernels is not new [6, 7], only recently have they been trained end-to-end. This is largely due to advances in inducing point [8] and variational [9] methods which allow for scalable inference. However, current methods fail to propagate the uncertainty in neural network weights during training. While the Gaussian process layer in deep kernel learning can provide uncertainty estimates at testing, the estimates also fail to account for uncertainty in the neural network weights.

In the Bayesian deep learning literature, uncertainty in neural network weights is approached in one of two ways. One approach develops new training procedures that facilitate approximate variational inference [10, 11, 12, 13, 14]. The other reinterprets successful methods in the deep learning literature as approximate variational inference [15, 16, 17].

More recently, batch normalization [18] has been reinterpreted as approximate variational inference [19]. In this work, we propose batch normalization as a way of propagating uncertainty in deep kernel learning. We show that this formulation naturally corresponds to variational inference on the neural network weights and we derive variance estimates that account for uncertainty in the neural network weights.

2 Background

Gaussian process classification imposes a Bernoulli likelihood assumption as well as a probit inverse link function $\phi(x) = P(x < Z)$ where $Z \sim N(0, 1)$. Then, if we draw f from a Gaussian process prior $p(f) = \mathcal{N}(0, K)$, we have the following joint distribution

$$P(y, f) = \mathcal{N}(f|0, \mathbf{K}) \prod_{n=1}^{N} \phi(f_i(x))^{y_i} [1 - \phi(f_i(x))]^{1-y_i}$$

Where K is the kernel matrix, given by the kernel function evaluted at pairs of features. In particular, it specifies the covariance structure, i.e. $[K]_{ij} = k(x_i, x_j)$.

Variational Approximation The marginal likelihood implicit in the joint distribution specified above is not tractable due to the non-conjugate nature of the likelihood. Instead, we default to variational techniques [20, 21] to approximate the posterior. We follow the scalable variational framework of [9], writing the evidence lower bound as

$$\log p(y) \geq E_{q(u)}[\log p(y|u)] - KL[q(u)||p(u)] \tag{1}$$

The variational bound derived in [9] uses the fact that $\log p(y|u) \geq E_{p(f|u)} \log(p|f))$ and substituting into (1) we get

$$\log p(y) \geq E_{q(f)}[\log p(y|f)] - KL[q(u)||p(u)]$$

where $q(u) = \mathcal{N}(m, S)$, $p(u) = \mathcal{N}(0, K_{mm})$ and $KL(q||p) = \int q(x) \log(q(x)/p(x))dx$.

Batch normalization [18] refers to the normalization of inputs at each layer l, $\hat{x}^l = \frac{x^l - \mathbb{E}[x^l]}{\sqrt{\text{Var}[x^l]}}$. When training is done over a minibatch $\{x_i^0\}_{i=1}^B$, then $\mathbb{E}[x^l] \approx \mu^l = \frac{1}{N}\sum_i^B x_i^l$ and $\sqrt{\text{Var}[x^l]} \approx \sigma^l = \sqrt{\frac{1}{B-1}\sum_{i=1}^B (x_i^l - \mu^l)^2}$. The result is a reduction in internal covariance shift and an improvement in testing performance. [19] shows that batch normalization can be interpreted as approximate variational inference on the weights of a deep neural network. In this paper, we show that this correspondence is exact when the implicit variational family $q(w)$ is known.

3 Deep kernel learning with weight uncertainty

Deep kernel learning [1, 2, 3, 4, 5] is the end-to-end learning of Gaussian process kernels composed with deep neural networks $h_w : \mathbb{R}^d \to \mathbb{R}^k$ where the neural network is parameterized by its weights w. It is trivial to show that, for any valid kernel function k, $k \circ h_w$ is also a kernel. However, if h_w is stochastic, then $k(h_w(x), h_w(x))$ requires that $h_w(x)$ is evaluated once to ensure symmetry.

Batch normalized deep kernel learning In this paper, we consider h_w to be a batch normalized neural network with isoptropic Gaussian priors on the weights. Then, the Gaussian process classification variational objective [9] can still be written as:

$$\log p(y) \geq E_{q(u,w)}[\log p(y|u,w)] - KL[q(u,w)||p(u,w)]$$

Using the inequality $\log p(y|u,w) \geq E_{p(f|u,w)} \log(p|f))$, we get that

$$\log p(y) \geq E_{q(f)}[\log p(y|f)] - KL[q(u,w)||p(u,w)]$$

Where we wrote $q(f) = \int p(f|u,w)q(u,w)dudw = \int p(f|u,w)q(u|w)du\, q(w)dw$. We use the fact that $q(f|w) = \int p(f|u,w)q(u|w)du = \mathcal{N}(Am, K_{nn} + A(S - K_{mm})A^T)$ where $A = K_{nm}K_{mm}^{-1}$ and K_{nn} denotes the covariance matrix given by the kernel function evaluated at the data points, $[K_{nn}]_{ij} = k(h_\theta(x_i), h_\theta(x_j))$. K_{nm} is the covariance matrix given by the kernel function evaluate at the data points with the inducing points, $[K_{nm}]_{ij} = k(h_\theta(x_i), h_\theta(u_j))$. Lastly, K_{mm} is the covariance matrix given by the kernel function evaluated at the inducing points, $[K_{mm}]_{ij} = k(h_\theta(u_i), h_\theta(u_j))$. Then we are left with $\int q(f|w)q(w)dw$. This is intractable in general, due to the non-linearities used in deep neural networks. However, under the stochastic sampling of minibatches, we are implicitly sampling from the weights w. Hence, batch normalization facilitates an easy way of sampling $\int q(f|w)q(w)dw$, by sampling minibatches and evaluating the kernel function given a minibatch.

Turning our attention to the KL term, notice that $KL[q(u,w)||p(u,w)] = E_{q(w)}[KL[q(u|w)||p(u|w)]] + KL[q(w)||p(w)]$. The first term is easy to evaluate, since it is implicitly evaluted in the sampling of minibatches. The second term, $KL[q(w)||p(w)]$, poses an issue since we do not explicitly specify a variational family over w. If we had this term, we would have a correspondence between variational inference of the neural network weights, and batch normalized training. Since we do not have this term, we ignore it and continue with an approximate variational inference scheme on weights.

Prediction The predictive distribution with weight uncertainty closely follows that of [9]. If we want to predict a test point x^*, then we need the posterior of the latent gaussian process value f^*.

$$\begin{aligned} p(f^*|y) &= \int p(f^*|f,u,w)p(f,u,w|y)dfdudw \\ &\approx \int p(f^*|f,u,w)p(f|u,w)q(u|w)q(w)dfdudw \\ &= \mathbb{E}_{q(w)}\left[\int p(f^*|u,w)q(u|w)du\right] \end{aligned}$$

The mean and variance of $f|w$, denoted by μ_w^2 and σ_w, is tractable with respect to the distribution $p(f^*|w) = \int p(f^*|u,w)q(u|w)du$ for given w. Then by sampling M minibatches, we have

$$\begin{aligned} Var(f^*) &= \mathbb{E}_{p(f^*|y)}[(f^*)^2] - \mathbb{E}_{p(f^*|y)}[f^*]^2 \\ &= \frac{1}{M}\sum_{i=1}^{M}\left[\mathbb{E}_{p(f^*|\hat{w}_i)}[(f^*)^2)\right] - \left[\frac{1}{M}\sum_{i=1}^{M}\mathbb{E}_{p(f^*|\hat{w}_i)}[f^*]\right]^2 \\ &= \frac{1}{M}\sum_{i=1}^{M}\left[\sigma_{\hat{w}_i}^2 + \mu_{\hat{w}_i}^2\right] - \left[\frac{1}{M}\sum_{i=1}^{M}\mu_{\hat{w}_i}\right]^2 \end{aligned}$$

Batch normalization versus dropout as a proxy for uncertainty It has been noted in the literature [22] that dropout, interpreted as a form of approximate Bayesian inference in neural networks, provides measures of risk, but not uncertainty.

To see how batch normalization provides a proxy for uncertainty, note that the effective neural network weights depend on the data. While a specific minibatch corresponds to a fixed set of weights, the stochasticity inherent in sampling a minibatch corresponds to a sampling a set of weights. Essentially, batch normalized neural networks use the first and second moments of each layer as a proxy for the distribution over weights of a Bayesian neural network without batch normalization. Hence, the batch normalizion provides measures of uncertainty, but not risk, since there is confusion over which model parameters (neural network weights) apply over a number of minibatches.

Lastly, we note that dropout in deep kernel learning drives the necessary cholesky decomposition to a singularity. The reasons for this are unclear, however we have not seen the use of dropout in other neural network and Gaussian process hybrid models, indicating similar issues.

4 Experiments

We consider a binary classification task on the MNIST dataset between even and odd digits, and benchmark deep kernel learning against standalone deep neural networks with and without batch normalization. The architecture is the same for all models. However, deep kernel learning models have a final gaussian process layer, while deep neural networks have an extra feed forward layer.

Referring to Figure 1 (left), we see that DKL-BN is the best performer when $n = 10$ and again when $n = 50000$. In Figure 1 (right), we see that the size of the minibatch improves performance, partially due to the increased capacity of the inducing variables as well as the reduction in covariate shift by batch normalization. However, diminishing returns begin to occur when the minibatch size is large relative to the sample size. Lastly, we look at the most ambiguous predictions, which we define as $\underset{x}{\operatorname{argmax}} f(x)(1-(f(x)))$. We can see that both models have difficulty telling whether Figure 2 (left) is a 5 or a 6, and hence have difficulty determining if it is odd or even. They both lean towards an odd number since the mean prediction is greater than 0.5. Similarly for Figure 2 (right), DKL cannot tell whether the digit is a 1 or a 2. We see that DKL-BN is able to improve greatly over DKL's worst case, especially in terms of variance. On the other hand, DKL improvement over DKL-BN in its worst case is marginal in comparison. It is still not certain in its decision, evidenced by the relatively high variance.

	DKL-BN		DKL	
	Mean	Var	Mean	Var
Fig 2 (left)	0.656	0.225	0.732	0.195
Fig 2 (right)	0.917	0.075	0.494	0.499

Table 1: Mean prediction and associated variance estimated by both models under the points with maximal ambiguity. Note, a mean of 1.0 and 0 variance indicates that the MNIST digit is certainly odd

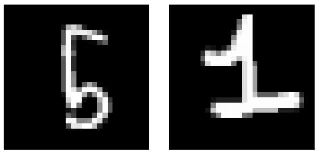
Figure 2: MNIST digits that maximize ambiguity for DKL-BN (left) and DKL (right)

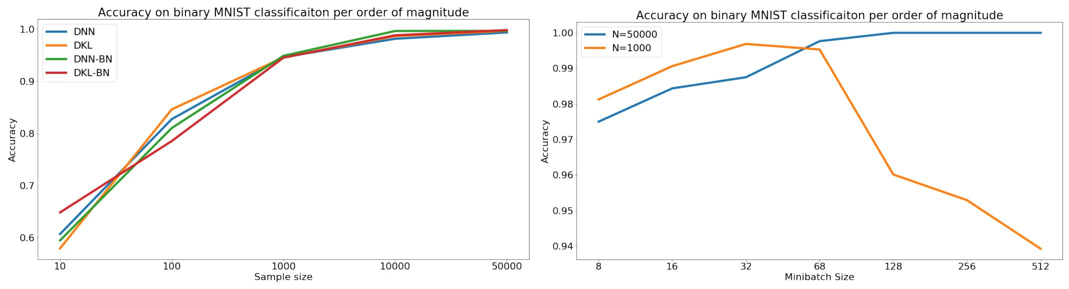

Figure 1: Left: Accuracy for the Deep Kernel Learning (DKL) and Deep Neural Network (DNN) models, with and without Batch Normalization (BN), as sample size increases. Right: Accuracy for DKL-BN for n = 1000 and n = 50000 as minibatch size increases

5 Discussion

In this paper, we proposed batch normalized deep kernel learning as a way of incorporating weight uncertainty. Similar to standalone deep neural networks, the inclusion of batch normalization improves testing performance. Additionally, batch normalized deep kernel learning greatly improves the estimated variance, which demonstrates that the model knows what it does not know. For future research, a proper approximation of the KL divergence between the weights' variational and prior distribution would be of interest. Another potential extension is to recurrent structure, such as GP-LSTM [4]. From this, it is clear that a Bayesian treatment of deep learning is illuminating for both Bayesian modelling and deep learning.

References

[1] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial Intelligence and Statistics*, pages 370–378, 2016.

[2] Andrew G Wilson, Zhiting Hu, Ruslan R Salakhutdinov, and Eric P Xing. Stochastic variational deep kernel learning. In *Advances in Neural Information Processing Systems*, pages 2586–2594, 2016.

[3] Roberto Calandra, Jan Peters, Carl Edward Rasmussen, and Marc Peter Deisenroth. Manifold gaussian processes for regression. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 3338–3345. IEEE, 2016.

[4] Maruan Al-Shedivat, Andrew Gordon Wilson, Yunus Saatchi, Zhiting Hu, and Eric P Xing. Learning scalable deep kernels with recurrent structure. *arXiv preprint arXiv:1610.08936*, 2016.

[5] John Bradshaw, Alexander G de G Matthews, and Zoubin Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in gaussian process hybrid deep networks. *arXiv preprint arXiv:1707.02476*, 2017.

[6] Cijo Jose, Prasoon Goyal, Parv Aggrwal, and Manik Varma. Local deep kernel learning for efficient non-linear svm prediction. In *International Conference on Machine Learning*, pages 486–494, 2013.

[7] Özlem Aslan, Xinhua Zhang, and Dale Schuurmans. Convex deep learning via normalized kernels. In *Advances in Neural Information Processing Systems*, pages 3275–3283, 2014.

[8] Andrew Wilson and Hannes Nickisch. Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *International Conference on Machine Learning*, pages 1775–1784, 2015.

[9] James Hensman, Alexander G de G Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. 2015.

[10] Alex Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, pages 2348–2356, 2011.

[11] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[12] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pages 1861–1869, 2015.

[13] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[14] Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*, 2017.

[15] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[16] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems*, pages 2575–2583, 2015.

[17] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[19] Anonymous. Bayesian uncertainty estimation for batch normalized deep networks. In *ICLR submission preprint*, 2018.

[20] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, (just-accepted), 2017.

[21] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[22] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, pages 4026–4034, 2016.

[23] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[24] Alexander G. de G. Matthews, Mark van der Wilk, Tom Nickson, Keisuke. Fujii, Alexis Boukouvalas, Pablo León-Villagrá, Zoubin Ghahramani, and James Hensman. GPflow: A Gaussian process library using TensorFlow. *Journal of Machine Learning Research*, 18(40):1–6, apr 2017.

[25] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A Implementation Details

Tensorflow [23] and GPflow [24] were used for implementation. Tensorflow is a standard neural network and machine learning library that implements automatic differentiation for optimization. GPflow is a Gaussian process framework built on top of Tensorflow. Specifically, we our base neural network shared between deep kernel learning and standalone neural networks are as follows: Convolutional (filter 32, kernel size 3) - Convolutional (filter 64, kernel size 3) - Feedforward(100 hidden units). ReLU activations are used between every layer and, when the model uses it, batch normalization. The standalone neral network then has: Feedforward(100 hidden units) - Feedforward(2 output units). While, the Gaussian process layer has: Feedforward(2 hidden units) - Gaussian process layer. The optimization was done via Adam optimizer [25] with $\alpha = 0.001$.