Stochastic Maximum Likelihood Optimization via Hypernetworks

Abdul-Saboor Sheikh, Kashif Rasul, Andreas Merentitis & Urs Bergmann
`{saboor.sheikh, kashif.rasul, urs.bergmann}@zalando.de`
`andreas.merentitis@ieee.org`
Zalando Research, Zalando SE
Mühlenstraße 25, 10243 Berlin, Germany

Abstract

This work explores maximum likelihood optimization of neural networks through hypernetworks. A hypernetwork initializes the weights of another network, which in turn can be employed for typical functional tasks such as regression and classification. We optimize hypernetworks to directly maximize the conditional likelihood of target variables given input. Using this approach we obtain competitive empirical results on regression and classification benchmarks.

1 Introduction

Implicit distribution (ID) representation has lately been a subject of much interest. IDs govern variables that result from linear or non-linear transformations applied to another set of (standard) random variates. Through successive transformations, IDs can be equipped with enough capacity to represent functions of arbitrary complexity (i.e., high-dimensionality, multimodality, etc.). This has seen IDs become mainstream in approximate probabilistic inference in deep learning, where they have been extensively applied to approximate intractable posterior distributions over latent variables and model parameters conditioned on data [9, 19, 12, 25, 18, 16, 23, 6, among many others]. Although the probability density function of an ID may not be accessible (e.g., due to non-invertible mappings), the reparameterization trick [9] still allows for an efficient and unbiased means for sampling it. This makes IDs amenable to (gradient-based) stochastic optimization regimes.

In this work we employ IDs as hypernetworks. The idea behind a *hypernetwork* (Figure 1) is to have a neural network that outputs the weights of another *main* neural network. The main network is then applied for standard tasks such as regression or classification. We optimize the parameters of the hypernetwork using stochastic gradient descent (SGD e.g. [7]) to maximize the (marginalized) conditional likelihood of targets given input. In empirical analysis on various regression and classification benchmarks, we find our approach to be in general better than standard maximum likelihood learning, while it also performs competitively with respect to a number of approximate Bayesian deep learning methodologies.

2 Related Work

Our optimization framework is not fully Bayesian; however instead of maintaining point parameter estimates as in standard maximum likelihood optimization of neural networks, our approach optimizes an ID that governs the parameters of a neural network. In contrast to Bayesian Neural Networks (BNNs) [17, 15, 26, 3, 5, 11, among others], where the target is to approximate the posterior distribution of neural network parameters (e.g., using variational inference [3]), we deploy and tune the parameter distribution while directly optimizing for the objective of the main neural network. Hence our approach is much akin to gradient-based evolutionary optimization [22].

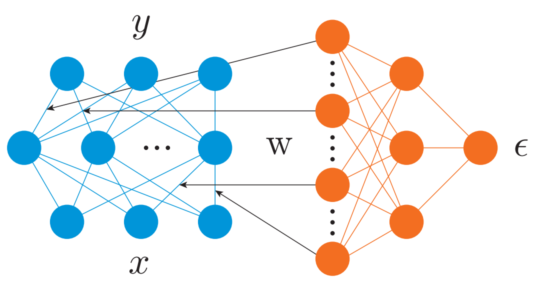

Figure 1: Schematic of a hypernetwork that takes a noise variable ϵ and outputs the weights of the main network.

2.1 Hypernetworks

This work uses the hypernetworks approach, which has its roots in this paper [20]. The term itself was introduced in the paper [4] where the complete model was trained via backpropagation, like in this paper, and used efficiently compressed weights to reduce the total size of the network.

2.2 Bayesian Neural Networks

Studied since the mid 90's [17] [15] BNNs can be classified into two main methods, either using MCMC [26] or learning an approximate posterior using stochastic variational inference (VI) [3], expectation propagation [5] or α-divergence [11]. In the VI setting one can interpret dropout [2] as a VI method allowing cheap samples from $q(\theta)$ but resulting in a unimodal approximate posterior. Bayes by Backprop [1] can also be viewed as a simple Bayesian hypernet, where the hypernetwork only performs element wise shift and scale of the noise. To solve the issue of scaling [13] propose a hypernet that generates scaling factors on the means of a factorial Gaussian distribution allowing for a highly flexible approximate posterior, but requiring an auxiliary inference network for the entropy term of the variational lower bound. Finally, [10] is closely related to this work, where a hypernetwork learns to transform a noise distribution to a distribution of weights of a target network, and is trained via VI.

3 Method

Given training data $\mathcal{D}$ containing $(X, Y) = \{x^{(n)}, y^{(n)}\}_{n=1}^{N}$, we consider the task of maximizing the marginal conditional likelihood of targets Y given X:

$$p(Y|X) = \prod_n \int p(y^{(n)}|x^{(n)}, \theta)\, p(\theta)\, \mathrm{d}\theta \;\approx\; \frac{1}{S}\prod_n \sum_s p(y^{(n)}|x^{(n)}, \theta_s); \quad \theta_s \sim p(\theta), \tag{1}$$

where the approximation allows us to estimate the intractable integral by Monte Carlo sampling. While the logarithm of $p(y|x, \theta)$ in (1) can be taken to be proportional to a loss function $l(y, f(x; \mathrm{w})) :=\| y - f(x; \mathrm{w}) \|$ defined between the ground-truth y and the output of a deep neural network $f(x; \mathrm{w})$, $p(\theta)$ (with $\theta = \mathrm{w}$) can be chosen to have any parametric form, such as fully factorized Gaussian distributions across each dimension of the parameter vector w. Such non-structured distributions would however be undesired for not allowing dependencies among dimensions of w. Also from a sampling point of view it would be inefficient, in particular in high-dimensional cases. We therefore define an implicit distribution $q(\mathrm{w}; \Theta)$ such that it has a capacity to model arbitrary dependency structures among w, while by means of the reparameterization trick [9], it also allows for cheap sampling $\mathrm{w} \sim q(\mathrm{w}; \Theta)$ by constructing w as:

$$\mathrm{w} = g(\epsilon; \Theta); \quad \epsilon \sim p(\epsilon), \tag{2}$$

where $g(\epsilon; \Theta)$ is a hypernetwork that gets activated by a latent noise variable ϵ drawn from a standard distribution $p(\epsilon)$, such as a Gaussian or uniform distribution. Given (2) and the main network $f(x; \mathrm{w})$, we can write down the following function as an approximation to the logarithm of (1):

$$\mathcal{F}(\mathcal{D}, \Theta) = \sum_n l(y^{(n)}, f(x^{(n)}; \mathrm{w}_s)); \quad \mathrm{w}_s \sim q(\mathrm{w}; \Theta). \tag{3}$$

Since we draw one parameter sample per data point, we have dropped the summation over S in (3). Given $\mathcal{D}$ we can optimize (3) w.r.t. Θ by applying gradient-based optimization schemes such as SGD. To predict y_* given an input x_* we have:

$$y_* = \int f(x_*; \mathrm{w})\, q(\mathrm{w}; \Theta)\, \mathrm{dw} \;\approx\; \frac{1}{S} \sum_s f(x_*; \mathrm{w}_s); \quad \mathrm{w}_s \sim q(\mathrm{w}; \Theta). \tag{4}$$

The objective function (3) is a stochastic relaxation of standard maximum likelihood optimization. It encourages the hypernetwork to mainly reproduce outputs for which the conditional likelihood was found to be maximal. This implies that $q(\mathrm{w}; \Theta)$ may eventually either converge to a delta peak at a (local) optimum or have its mass distributed across equally (sub)optimal regions, which in principle can coincide with the solution found by plain maximum likelihood (i.e., standard gradient descent techniques). In practice we however observe that our approach finds solutions that are generally better than standard gradient descent and are on par with more sophisticated Bayesian deep learning methodologies (see Sec. 4 for more details). This may be due to the fact that we start with a large population of solutions that shrinks down gradually as $q(\mathrm{w}; \Theta)$ continues to concentrate its mass to the region(s) of most promising solutions seen so far. This can be seen as akin to gradient-based efficient evolution strategies [22].

4 Experiments

We validate our approach in accordance with relevant literature on standard regression tasks on UCI datasets as well as MNIST classification. We use ReLU non-linearity in all our experiments for hidden layers in both hyper and main networks. We use dropout probability of 0.5 for the hidden units of hypernetworks during training. We use a softplus layer to scale (initially 1.0) the input to the hypernetwork in (2). For both our approach and standard SGD results reported below, we add Gaussian noise to input x in (3), with its scale (initially 0.5) tuned during end to end training as the output of a softplus unit.

For the regression experiments we follow the process as in [5]: we randomly keep 10% of the data for testing and the rest for training. Following the earlier works [5, 14], the main network architecture consists of a single hidden layer with 50 units (or 100 for the Protein and Year). For datasets with more than 5,000 data points (Kin8nm, Naval, Power, Protein and Year) we switch from mini-batches of size 1 to 512 (not fine-tuned). We run a total of 2000 epochs per repetition. As this number is not fine-tuned, it could cause overfitting in smaller datasets like Boston. Early stopping in such cases might improve the results.

We compare the results of our approach with the results of other state of the art methods, as well as plain SGD trained on the same target network $f(x; w)$ without employing a hypernetwork. For each task we output the mean RMSE over the last 20 epochs and then repeat the task 20 times to report the mean of these RMSEs and their standard errors in Table 1 for our method and SGD. It can be seen that in most cases the results are competitive, and often exceed the state of the art. In Figure 2 we further plot how the distribution over weights as represented by the hypernetwork evolves over training epochs for three (out of 20) independent repetitions of three of the regression benchmarks. While we observe a consistent diffusion behavior across multiple repetitions on a given dataset, we see that for different datasets the diffusion trajectories leading to convergence (to a delta peak) can vary notably. In the plots we also overlay the evolution of the average test RMSE for a comparison.

For the classification task, we used MNIST to train with varying numbers of layers and hidden units per layer, as in Table 2 for 2,000 epochs. We use a mini-batch size of 256. The hypernetwork for the $2{\times}400$ and $2{\times}800$ cases consist of 2 hidden layers of size 100 and 50 and for the $3{\times}150$ we have 2 hidden layers of 100 and 20. The error rate reported is calculated by taking the means of test errors from the last 20 epochs for our method and plain SGD.

5 Conclusion

We have presented a simple stochastic optimization approach for deep neural networks. The method employs implicit distributions as hypernetworks to model arbitrary dependencies among parameters of the main network. Despite being not fully Bayesian, our approach aims to model a distribution

Dataset	N	d	VI	PBP	Dropout	VMG	SGD	Our
Boston	506	13	4.32±0.29	3.01±0.18	2.97±0.85	**2.70±0.13**	3.73 ± 0.67	3.72 ± 1.05
Concrete	1,030	8	7.19±0.12	5.67±0.09	5.23± 0.53	4.89±0.12	5.29 ± 0.87	**4.74 ± 0.64**
Energy	768	8	2.65±0.08	1.80±0.05	1.66±0.19	**0.54±0.02**	0.95 ± 0.13	0.87 ± 0.10
Kin8nm	8,192	8	0.10±0.00	0.10±0.00	0.10±0.00	**0.08±0.00**	**0.08 ± 0.00**	**0.08 ± 0.00**
Naval	11,934	16	0.01±0.00	0.01±0.00	0.01±0.00	**0.00±0.00**	**0.00 ± 0.00**	**0.00 ± 0.00**
Power Plant	9,568	4	4.33±0.04	4.12±0.03	**4.02±0.18**	4.04±0.04	4.06 ± 0.25	**4.02 ± 0.18**
Protein	45,730	9	4.84±0.03	4.73±0.01	4.36±0.04	**4.13±0.02**	4.37 ± 0.03	4.65 ± 0.19
Wine	1,599	11	0.65±0.01	0.64±0.01	0.62±0.04	0.63±0.01	0.80 ± 0.05	**0.62 ± 0.04**
Yacht	308	6	6.89±0.67	1.02±0.05	1.11±0.38	0.71±0.05	0.77 ± 0.25	**0.57 ± 0.21**
Year	515,345	90	9.03±NA	8.87±NA	8.84±NA	8.78±NA	**8.74 ± 0.03**	**8.74 ± 0.03**

Table 1: Average test set RMSE and standard errors for the regression datasets for VI method of [3], PBP method of [5], Dropout uncertainty of [2], VMG of [14], standard SGD and our method.

Arch.	Max. Likel.	DropConnect	Bayes B. SM	Var. Dropout	VMG	SGD	Our
2×400	–	–	1.36	–	**1.15**	1.43	1.26
2×800	1.60	1.20	1.34	–	–	1.64	1.37
3×150	–	–	–	≈ 1.42	1.18	1.71	1.49
3×750	–	–	–	≈ 1.09	**1.05**	–	–

Table 2: Test errors for the MNIST dataset for Max. Likel. [21], DropConnect [24], Bayes by B. with scale mixture [1], Var. Dropout [8], VMG [14], mean for plain SGD and mean for our method.

over the parameters of a neural network, as opposed to maintaining point weight estimates as in standard maximum likelihood optimization of neural networks. In general we empirically outperform standard gradient descent optimization and demonstrate on par performance in a broader comparison with state of the art Bayesian methodologies on regression and classification tasks.

In the future we would like to focus on the scalability of our approach (e.g., through layer coupling [10]) as well as on a fully Bayesian extension of our optimization procedure.

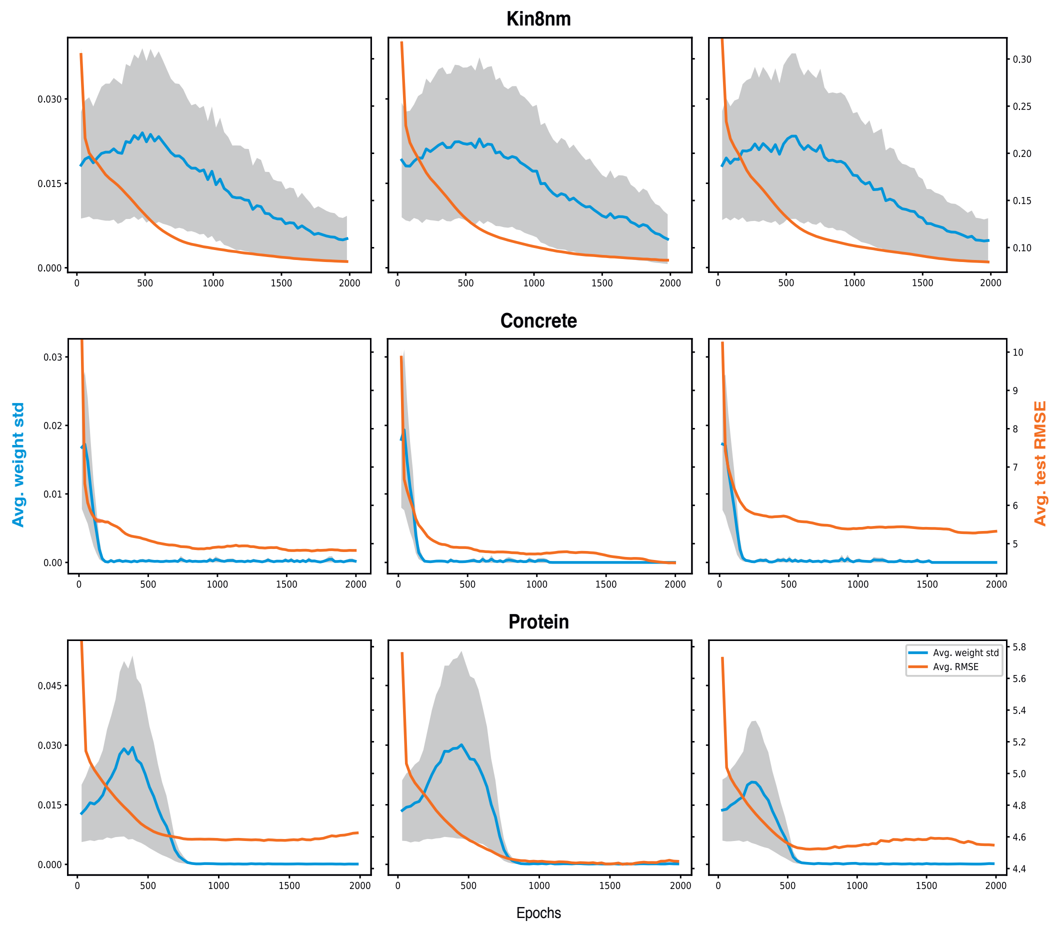

Figure 2: Each row plots three random repetitions of the regression task over a benchmark. The blue curves are the means of the standard deviations computed across each dimension of the hypernetwork output. The values are computed at every epoch from 10 samples of the hypernetwork output (i.e., 10 initializations of the main network). The gray areas cover 3 standard deviations around the mean values. The red curves are the average test set RMSE computed per epoch as for our results in Table 1 .

References

[1] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622. JMLR.org, 2015.

[2] Y. Gal and Z. Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *ArXiv e-prints*, June 2015.

[3] Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 2348–2356. Curran Associates, Inc., 2011.

[4] D. Ha, A. Dai, and Q. V. Le. HyperNetworks. *ArXiv e-prints*, September 2016.

[5] Jose Miguel Hernandez-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1861–1869, Lille, France, 07–09 Jul 2015. PMLR.

[6] Ferenc Huszár. Variational inference using implicit distributions. *CoRR*, abs/1702.08235, 2017.

[7] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[8] Diederik P Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2575–2583. Curran Associates, Inc., 2015.

[9] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

[10] D. Krueger, C.-W. Huang, R. Islam, R. Turner, A. Lacoste, and A. Courville. Bayesian Hypernetworks. *ArXiv e-prints*, October 2017.

[11] Yingzhen Li and Yarin Gal. Dropout inference in Bayesian neural networks with alpha-divergences. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2052–2061, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[12] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 2370–2378, 2016.

[13] C. Louizos and M. Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. *ArXiv e-prints*, March 2017.

[14] Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix gaussian posteriors. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, pages 1708–1716. JMLR.org, 2016.

[15] David J.C MacKay. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73 – 80, 1995. Proceedings of the Third Workshop on Neutron Scattering Data Analysis.

[16] L. Mescheder, S. Nowozin, and A. Geiger. Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. In *International Conference on Machine Learning (ICML) 2017*, August 2017.

[17] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.

[18] Rajesh Ranganath, Dustin Tran, Jaan Altosaar, and David M. Blei. Operator variational inference. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 496–504, 2016.

[19] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1530–1538, 2015.

[20] Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Comput.*, 4(1):131–139, January 1992.

[21] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, pages 958–, Washington, DC, USA, 2003. IEEE Computer Society.

[22] Yi Sun, Daan Wierstra, Tom Schaul, and Jürgen Schmidhuber. Efficient natural evolution strategies. *CoRR*, abs/1209.5853, 2012.

[23] Dustin Tran, Rajesh Ranganath, and David M. Blei. Deep and hierarchical implicit models. *CoRR*, abs/1702.08896, 2017.

[24] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L. Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In Sanjoy Dasgupta and David Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1058–1066. JMLR Workshop and Conference Proceedings, May 2013.

[25] Dilin Wang and Qiang Liu. Learning to draw samples: With application to amortized MLE for generative adversarial learning. *CoRR*, abs/1611.01722, 2016.

[26] Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pages 681–688. Omnipress, 2011.