Variational Inference in Neural Networks using an Approximate Closed-Form Objective

Wolfgang Roth and Franz Pernkopf
Signal Processing and Speech Communication Laboratory
Graz University of Technology
roth@tugraz.at, pernkopf@tugraz.at

Abstract

We propose a closed-form approximation of the intractable KL divergence objective for variational inference in neural networks. The approximation is based on a probabilistic forward pass where we successively propagate probabilities through the network. Unlike existing variational inferences schemes that typically rely on stochastic gradients that often suffer from high variance our method has a closed-form gradient. Furthermore, the probabilistic forward pass inherently computes expected predictions together with uncertainty estimates at the outputs. In experiments, we show that our model improves the performance of plain feed-forward neural networks. Moreover, we show that our closed-form approximation works well compared to model averaging and that our model is capable of producing reasonable uncertainties in regions where no data is observed.

1 Introduction

In recent years, a lot of work has been dedicated to a Bayesian treatment of different kinds of neural network (NN) architectures [4, 1, 7, 12, 2, 5]. The goal of Bayesian inference is to infer a posterior distribution over the NN parameters and compute predictions as expectations with respect to that posterior, rather than computing predictions based on a single point estimate. An analytical treatment of this approach is usually not tractable for NNs. Therefore, various approximations have been used. Especially the variational inference framework has shown promising results since it renders the inference task as an optimization problem for which a vast amount of literature is available [11]. Here, the goal is to find an approximate variational distribution $q(\boldsymbol{W})$ that is as close as possible to the true posterior $p(\boldsymbol{W}|\mathcal{D})$.[1] Typically the Kullback-Leibler divergence $\mathrm{KL}(q(\boldsymbol{W})||p(\boldsymbol{W}|\mathcal{D}))$ between approximate and true posterior is used as optimization criterion. Subsequently the expectation is computed with respect to the variational distribution $q(\boldsymbol{W})$.

However, applying variational inference to NNs is not straightforward since the variational objective and its gradient is typically not available in closed-form. A common way to overcome this issue is the so called reparameterization trick where the gradient of the expectation is replaced with the expectation of a gradient [7, 1]. This way, approximate gradients can be sampled from a simple distribution and used for stochastic gradient methods to optimize expectations for which no analytic solutions exist. Our work is closely related to [1] but follows a different spirit: We approximate the intractable expectation directly to obtain a closed-form approximation of the objective and the gradient thereof. Similar as in [12, 4], we make several approximations and propagate probabilities through the network. At the end of this probabilistic forward pass, we arrive at an approximation to the log-likelihood defined by the NN that can be computed in closed-form.

[1] $\mathcal{D}$ denotes the given dataset.

Our method has several advantages compared to the reparameterization trick: (i) Our method removes the source of stochasticity introduced by sampling the gradient. When using the reparameterization trick in stochastic optimization, it might be necessary to average several gradients in order to reduce the variance to a decent level. The resulting computational overhead is avoided by our method. Nevertheless, our method is compatible with stochastic minibatch optimization. (ii) Our approximate function can be used with more sophisticated optimization algorithms such as quasi-Newton methods that are still not suitable for stochastic gradients. This is especially interesting in the case of small datasets where it is not expensive to compute a gradient using all data samples. Furthermore, since our approximation results in a closed-form objective and a closed-form gradient, it is easier to decide if a local minimum has been reached by using standard optimality conditions. (iii) After training it is common to make predictions based on the mean weights of $q(\boldsymbol{W})$ or to average predictions of samples drawn from $q(\boldsymbol{W})$. Our approximated forward pass can also be used to compute expected predictions with respect to the variational distribution $q(\boldsymbol{W})$. This results in less computational overhead at test time while still taking the model uncertainty into account.

2 Neural Networks and Variational Inference

The structure of a fully connected feed-forward NN with L layers is determined by the amount of neurons $\{d_0, d_1, \dots, d_L\}$, where d_0 is the dimensionality of the inputs, d_L is the dimensionality of the output[2] and d_l for $1 \leq l < L$ denotes the number of neurons in layer l. The parameters of the NN are then given by a set of weight matrices $\boldsymbol{W} = \{\boldsymbol{W}^l\}_{l=1}^L$ where $\boldsymbol{W}^l = (w_{i,j}^l)_{i=1,\dots,d_l, j=1,\dots,d_{l-1}}$.[3] An NN defines a function $\boldsymbol{y} = f(\boldsymbol{x}^0)$ by iteratively applying a linear transformation $\boldsymbol{a}^l = \boldsymbol{W}^l\, \boldsymbol{x}^{l-1}$ followed by a non-linear activation function $\boldsymbol{x}^l = \boldsymbol{\phi}_l(\boldsymbol{a}^l)$. If $l < L$, we use the ReLU function $\boldsymbol{\phi}_l(\boldsymbol{a}) = \max(0, \boldsymbol{a})$ [10].[4] If $l = L$, we use the identity in case of regression and the softmax activation function in case of classification. The output of the NN is then $\boldsymbol{y} = \boldsymbol{x}^L$. For convenience, we define $\boldsymbol{W}^{>k} = \{\boldsymbol{W}^l\}_{l=k+1}^L$.

To adopt a Bayesian treatment of NNs we assume a prior distribution $p(\boldsymbol{W})$ over the weights. It is common to assume that the weights are a-priori independent, i.e. $p(\boldsymbol{W})$ factorizes into a product of factors for each individual weight. Typical choices for the prior distribution are zero-mean Gaussian or Laplace distributions but also scale-mixture priors have shown to be beneficial [1]. Moreover, the output of an NN can be interpreted as a likelihood function $p(\mathcal{D}|\boldsymbol{W})$. The prior together with the likelihood induce a posterior distribution $p(\boldsymbol{W}|\mathcal{D})$ over the weights.

However, $p(\boldsymbol{W}|\mathcal{D})$ is typically intractable. The aim of variational inference is therefore to approximate $p(\boldsymbol{W}|\mathcal{D})$ with a simpler variational distribution $q(\boldsymbol{W}|\boldsymbol{\nu})$ and then perform inference based on $q(\boldsymbol{W}|\boldsymbol{\nu})$. Here $\boldsymbol{\nu}$ denotes the set of variational parameters that are optimized so that $q(\boldsymbol{W}|\boldsymbol{\nu})$ is as close as possible to the posterior $p(\boldsymbol{W}|\mathcal{D})$ in some sense. In variational inference it is common to minimize $\mathrm{KL}(q(\boldsymbol{W}|\boldsymbol{\nu})||p(\boldsymbol{W}|\mathcal{D}))$. We adopt the common mean-field approximation with an approximate posterior $q(\boldsymbol{W}|\boldsymbol{\nu})$ that factorizes into a product of factors $q(w|\nu_w)$ for each weight $w \in \boldsymbol{W}$. We select the variational distribution $q(w|\nu_w)$ to be Gaussians with mean and variance parameters $\nu_w = (\mu_w, \sigma_w)$. This doubles the total amount of parameters to optimize while adding useful uncertainty estimates to the weights.

3 Approximating the Variational Objective

Using basic transformations, minimizing $\mathrm{KL}(q(\boldsymbol{W})|p(\boldsymbol{W}|\mathcal{D}))$ is equivalent to minimizing $\mathrm{KL}[q(\boldsymbol{W})||p(\boldsymbol{W})] - \mathbb{E}_{q(\boldsymbol{W})}[\log p(\mathcal{D}|\boldsymbol{W})]$.[5] The latter expression is much more convenient since it does not involve the intractable true posterior $p(\boldsymbol{W}|\mathcal{D})$. While the KL term allows for analytic solutions for common choices of prior and approximate posterior combinations, the expected likelihood is usually intractable for networks of any decent size. Therefore, we approximate the likelihood term by successively propagating Gaussian distributions through the network. Before explaining each approximation in detail, we start with a high-level description of the approximation. Then,

[2]We assume one-hot encoded vectors $\boldsymbol{t}$ in case of classification.

[3]For simplicity we included the bias weights in the weight matrices.

[4]The $\max$ function is applied element-wise.

[5]In the sequel we omit the dependence of the variational distribution on $\boldsymbol{\nu}$ and simply write $q(\boldsymbol{W})$.

we provide the variational objective including detailed analytical expressions for the KL term with Gaussian or Laplace priors $p(\boldsymbol{W})$ and Gaussian approximate posteriors $q(\boldsymbol{W})$.

3.1 Approximating the Expected Log-Likelihood

In particular, the expected log-likelihood is approximated by

$$\begin{aligned}
\mathbb{E}_{q(\boldsymbol{W})}\left[\log p(\boldsymbol{t}|\boldsymbol{x}^0, \boldsymbol{W})\right] &= \int q(\boldsymbol{W}) \log p(\boldsymbol{t}|\boldsymbol{x}^0, \boldsymbol{W}) d\boldsymbol{W} && (1)\\
&= \int q(\boldsymbol{W}) \log p(\boldsymbol{t}|\boldsymbol{a}^1(\boldsymbol{x}^0, \boldsymbol{W}^1), \boldsymbol{W}^{>1}) d\boldsymbol{W} && (2)\\
&\approx \int q(\boldsymbol{W}^{>1}) \mathcal{N}\left(\boldsymbol{a}^1|\boldsymbol{\mu}_{a^1}, \boldsymbol{\sigma}_{a^1}\right) \log p(\boldsymbol{t}|\boldsymbol{a}^1, \boldsymbol{W}^{>1}) d\boldsymbol{a}^1 d\boldsymbol{W}^{>1} && (3)\\
&\approx \int q(\boldsymbol{W}^{>1}) \mathcal{N}\left(\boldsymbol{x}^1|\boldsymbol{\mu}_{x^1}, \boldsymbol{\sigma}_{x^1}\right) \log p(\boldsymbol{t}|\boldsymbol{x}^1, \boldsymbol{W}^{>1}) d\boldsymbol{x}^1 d\boldsymbol{W}^{>1} && (4)\\
&\vdots\\
&\approx \int \mathcal{N}(\boldsymbol{a}^L|\boldsymbol{\mu}_{a^L}, \boldsymbol{\sigma}_{a^L}) \log p(\boldsymbol{t}|\boldsymbol{a}^L) d\boldsymbol{a}^L && (5)\\
&\approx \log p(\boldsymbol{t}|\boldsymbol{\mu}_{a^L}) + \frac{1}{2} \sum_{i=1}^{d_L} \sigma_{a_i^L} \left(\nabla^2 \log p(\boldsymbol{t}|\boldsymbol{\mu}_{a^L})\right)_{ii}. && (6)
\end{aligned}$$

Similar as in [12, 4], we successively apply a central limit argument and propagate probabilities through the network to obtain a tractable approximation. In (2), we emphasize that activations are given as a sum of random variables. Given a sufficiently large amount of input neurons, the distribution of neuron activations can therefore be well approximated using Gaussians as in (3). For computational convenience, we assume that the activations of each neuron are independent. Furthermore, in (4) we approximate the distributions after applying the activation functions again with Gaussians by moment matching. These two approximations are iterated up to layer L. A tractable approximation of the expectation of the log-likelihood at the outputs is achieved in (6) by using a second-order Taylor approximation of the log-likelihood with a diagonal Hessian approximation around the mean $\boldsymbol{\mu}_{a^L}$. The gradient of this function is rather cumbersome to calculate but is readily obtained using automatic differentiation frameworks.

3.1.1 Approximating the Activations

Assuming that the inputs $\boldsymbol{x}^{l-1}$ in layer l are independent and using the mean-field assumption of the variational distribution, the activations $\boldsymbol{a}^l$ can be well approximated by a diagonal Gaussian $\mathcal{N}(\boldsymbol{a}^l|\boldsymbol{\mu}_{a^l}, \boldsymbol{\sigma}_{a^l})$. Due to the independence assumption, the means and the variances of the activations are computed as

$$\begin{aligned}
\mu_{a_i^l} &= \mathbb{E}\left[\sum_{j=1}^{d_{l-1}} w_{ij}^l\, x_j^{l-1}\right] = \sum_{j=1}^{d_{l-1}} \mu_{w_{ij}^l}\, \mu_{x_j^{l-1}} && (7)\\
\sigma_{a_i^l} &= \sum_{j=1}^{d_{l-1}} \mathbb{E}\left[(w_{ij}^l)^2\right] \mathbb{E}\left[(x_j^{l-1})^2\right] - \mathbb{E}\left[w_{ij}^l\right]^2 \mathbb{E}\left[x_j^{l-1}\right]^2 && (8)\\
&= \sum_{j=1}^{d_{l-1}} \sigma_{w_{ij}^l}\, \mu_{(x_j^{l-1})^2} + \mu_{w_{ij}^l}^2 \left(\mu_{(x_j^{l-1})^2} - \mu_{x_j^{l-1}}^2\right), && (9)
\end{aligned}$$

where $\mu_{(x_j^{l-1})^2}$ denotes the raw second moment $\mathbb{E}[(x_j^{l-1})^2]$. In case of $l = 1$, we assume no variance at the inputs and thus the second term of (9) cancels.

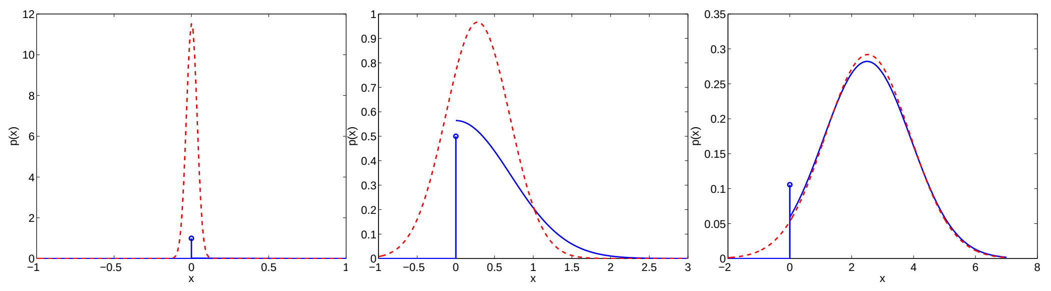
3.1.2 Approximating the ReLU by Moment Matching

In the next step, the Gaussian approximation of the activations is propagated through the activation function. We limited ourselves to the ReLU activation function although analytical solutions for other functions such as step functions can be calculated as well. In our case, the resulting distribution is a mixture of a point mass at zero and a truncated Gaussian that we again approximate by a Gaussian. It suffices to compute the quantities $\mathbb{E}_{\mathcal{N}(\mu_{a_i^l}, \sigma_{a_i^l})}[\max(0, a_i^l)]$ and $\mathbb{E}_{\mathcal{N}(\mu_{a_i^l}, \sigma_{a_i^l})}[\max(0, (a_i^l)^2)]$. The approximation is computed as

$$\mathbb{E}\left[x_i^l\right] = \mu_{x_i^l} = \frac{\mu_{a_i^l}}{2}\left(1 + \operatorname{erf}\left(\frac{\mu_{a_i^l}}{\sqrt{2\sigma_{a_i^l}}}\right)\right) + \sqrt{\frac{\sigma_{a_i^l}}{2\pi}} \exp\left(-\frac{\mu_{a_i^l}^2}{2\sigma_{a_i^l}}\right) \tag{10}$$

$$\mathbb{E}\left[(x_i^l)^2\right] = \mu_{(x_i^l)^2} = \frac{\sigma_{a_i^l} + \mu_{a_i^l}^2}{2}\left(1 + \operatorname{erf}\left(\frac{\mu_{a_i^l}}{\sqrt{2\sigma_{a_i^l}}}\right)\right) + \mu_{a_i^l}\sqrt{\frac{\sigma_{a_i^l}}{2\pi}} \exp\left(-\frac{\mu_{a_i^l}^2}{2\sigma_{a_i^l}}\right). \tag{11}$$

Rather than computing the variance of the Gaussian approximation, we compute the raw second moment $\mathbb{E}[(x_i^l)^2]$ which can readily be used in the expressions derived in Section 3.1.1. Figure 1 illustrates the Gaussian approximation of the ReLU for several values of μ_a and σ_a.



(a) $\mu_a = -2.5$, $\sigma_a = 1$ (b) $\mu_a = 0$, $\sigma_a = 0.5$ (c) $\mu_a = 2.5$, $\sigma_a = 2$

Figure 1: Examples of Gaussian distributions propagated through the ReLU activation function. The blue line shows the true distribution as a mixture of a point mass at zero and a truncated Gaussian. The red dashed line shows the Gaussian approximation of the blue line obtained by moment matching.

3.1.3 Approximating the NN output

For regression it is common to assume that the targets t are Gaussian distributed with mean a^L and some fixed variance β. In this case the integral in (5) can be solved analytically, i.e.

$$\mathbb{E}_{\mathcal{N}(\mu_{a^L}, \sigma_{a^L})}\left[\log \mathcal{N}(t|a^L, \beta)\right] = -\frac{\log 2\pi\beta}{2} - \frac{1}{2\beta}(t - \mu_{a^L})^2 - \frac{\sigma_{a^L}}{2\beta}. \tag{12}$$

For simplicity we have assumed that there is only a single output neuron but the extension to multiple outputs is straightforward. In case of classification, where the NN output is determined by the softmax function, the integral in (5) does not admit an analytical solution. Therefore, we approximate the log-softmax function in (5) by its second-order Taylor expansion around the mean $\boldsymbol{\mu}_{a^L}$. Using a diagonal Hessian approximation for the Taylor expansion, (6) is given by

$$\mathbb{E}_{\mathcal{N}(\mu_{a^L}, \sigma_{a^L})}\left[\log \operatorname{smax}_t(\boldsymbol{a}^L)\right] = \log \operatorname{smax}_t(\boldsymbol{\mu}_{a^L}) - \frac{1}{2}\sum_{t'} \sigma_{a^L_{t'}} \operatorname{smax}_{t'}(\boldsymbol{\mu}_{a^L})\left(1 - \operatorname{smax}_{t'}(\boldsymbol{\mu}_{a^L})\right), \tag{13}$$

where we used the notation $\operatorname{smax}_t(\boldsymbol{a}) = \exp(a_t)/\sum_{t'}\exp(a_{t'})$.

3.2 The KL term

The KL term admits analytical solutions for common choices of prior and approximate posterior combinations. In particular, we select a Gaussian approximate posterior distribution $q(\boldsymbol{W})$ and zero-mean Gaussian or Laplace prior distributions $p(\boldsymbol{W})$. The KL term can be separated into $\mathbb{E}_{q(\boldsymbol{W})}[\log q(\boldsymbol{W})] - \mathbb{E}_{q(\boldsymbol{W})}[\log p(\boldsymbol{W})]$. The first term is the negative differential entropy of a

Gaussian given by

$$\mathbb{E}_{q(\boldsymbol{W})}[\log q(\boldsymbol{W})] = -\sum_{w \in \boldsymbol{W}} \frac{\log(2\pi\sigma_w) + 1}{2}. \tag{14}$$

The expected log prior $\mathbb{E}_{q(\boldsymbol{W})}[\log p(\boldsymbol{W})]$ for a zero-mean Gaussian prior $p(\boldsymbol{W})$ with variance γ is given by

$$\mathbb{E}_{q(\boldsymbol{W})}[\log p(\boldsymbol{W})] = -|\boldsymbol{W}| \frac{\log 2\pi\gamma}{2} - \sum_{w \in \boldsymbol{W}} \frac{\sigma_w + \mu_w^2}{2\gamma}. \tag{15}$$

Here $|\boldsymbol{W}|$ denotes the number of weights in the NN. Using a zero-mean Laplace prior $p(\boldsymbol{W})$ with variance γ we obtain

$$\mathbb{E}_{q(\boldsymbol{W})}[\log p(\boldsymbol{W})] = -|\boldsymbol{W}| \log(2\gamma) - \frac{1}{\gamma} \sum_{w \in \boldsymbol{W}} \left(\sqrt{\frac{2\sigma_w}{\pi}} \exp\left(-\frac{\mu_w^2}{2\sigma_w}\right) + \mu_w \operatorname{erf}\left(\frac{\mu_w}{\sqrt{2\sigma_w}}\right) \right). \tag{16}$$

3.3 Likelihood Weighting

In contrast to plain MAP estimation, in our framework the variance γ of the prior $p(\boldsymbol{W})$ cannot be directly used to trade off between a data term $\log p(\mathcal{D}|\boldsymbol{W})$ and a complexity term $\log p(\boldsymbol{W})$. For instance, in MAP estimation increasing γ corresponds to increasing the influence of the data term. In our model we would also enforce an increase in variance σ_w that could be prohibitive for achieving good performance. Furthermore, we observed in experiments that the KL term for large networks and small datasets is often orders of magnitudes larger than the expected log-likelihood term. As a result, the optimization procedure mainly focuses on keeping the approximate posterior $q(\boldsymbol{W})$ close to the prior $q(\boldsymbol{W})$ whereas the influence of the data is too small. Hence, we propose to add an additional multiplicative weight λ to the expected log-likelihood term that can be interpreted as creating λ copies of the original dataset $\mathcal{D}$. It appears, that this issue is circumvented in [1] with their proposed KL-reweighting scheme. Due to the exponential weight decay only a few minibatches are influenced by the KL term whereas the vast majority is solely influenced by the expected log-likelihood.

4 Experiments

4.1 Classification Performance

We evaluated the performance of our model on the MNIST handwritten digit recognition dataset [9] and variants thereof [8]. Each dataset contains 28×28 pixel images showing one of the ten digits. MNIST consists of 60000 training samples and 10000 test samples where we split the training set into 50000 training samples and 10000 validation samples used for hyperparameter optimization. The MNIST variants consist of 10000 training samples, 2000 validation samples and 50000 test samples.

We compare the performance of several NN classifiers. We tested plain feed-forward NNs with ℓ^2 regularization (*NN*). We evaluated the regularization parameter $\lambda \in \{2^{-4}, \ldots, 2^{18}\}$. Furthermore, we tested Bayes by Backprop (*NN BBB*) using the setup described in [1]. For our model (*NN VI*) we used Gaussian and Laplace priors $p(\boldsymbol{W})$. We evaluated the prior variance $\gamma \in \{10^{-2}, \ldots, 10^{2}\}$ and the likelihood factor $\lambda \in \{10^{0}, \ldots, 10^{3}\}$.

For all models we used ReLU activations and two fully connected hidden layers each having either 400, 800 or 1200 neurons. The weights and weight means are initialized with a zero-mean uniform distribution having variance $2/d_{in}$ where d_{in} is the number of input neurons as suggested in [3]. Similar as in [1], we reparameterized the variance σ_w with unconstrained parameters ρ_w as $\sigma_w(\rho_w) = \log(1 + \exp(\rho_w))$. For *NN VI* the variance parameters ρ were initialized to -10 and for *NN BBB* they were initialized to -5.[6] For *NN* and *NN VI* optimization was performed using ADAM [6] with a step size of $\alpha = 10^{-3}$. For *NN BBB* optimization was performed using *rmsprop* and the KL-reweighting scheme described in [1]. For *NN VI* we computed the test error using the approximated expected predictions and for *NN BBB* we used the mean weights of $q(\boldsymbol{W})$.

[6]For *NN BBB* σ_w is actually the standard deviation and not the variance.

The best results of each model are summarized in Table 1. Our model outperforms plain NNs on all datasets whereas *NN BBB* is only outperformed on two datasets. *NN VI* achieved similar performances for both Gaussian and Laplace priors $p(\boldsymbol{W})$. As stated in [1] the performance of *NN BBB* heavily profits from the use of a scale mixture prior, i.e. a mixture of two zero-mean Gaussians with different variances. However, the scale mixture prior does not result in a closed-form solution of the KL term. This is unproblematic in their setup since they are sampling the gradients anyway.

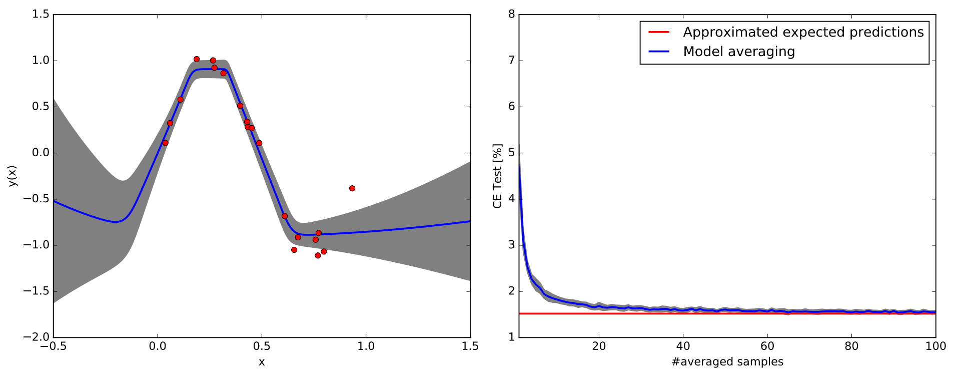
Table 1: Classification errors (%) on MNIST [9] and MNIST variants [8].

DATASET	NN	NN BBB	NN VI
MNIST	1.55	1.30	1.52
MNIST-BASIC	3.50	3.39	3.37
MNIST-BACK	20.79	21.49	20.77
MNIST-BACK-RAND	16.08	11.03	14.80
MNIST-ROT	10.93	9.46	10.39
MNIST-ROT-BACK	51.51	48.28	51.04

4.2 Uncertainty Estimates and Model Averaging

In the next experiment we evaluated the output uncertainty produced by our model. Therefore, we generated 20 points from $\sin(2\pi x)$ with some added Gaussian noise. Training was performed with the L-BFGS quasi-Newton algorithm. The experiment is illustrated in Figure 2a. Our model produces reasonable uncertainties in regions where many points are observed whereas the estimated uncertainty grows in regions where no data is available. Note that uncertainties are produced in a single forward pass rather than by evaluating the outputs of several NNs sampled from $q(\boldsymbol{W})$.

We also evaluated the quality of the approximated expected predictions. This is illustrated in Figure 2b. We averaged the outputs for different numbers of NNs sampled from $q(\boldsymbol{W})$ and compared the classification error with that obtained by the approximated expected predictions. Model averaging for each amount of samples was performed 25 times and we report the mean (blue line) and standard deviation (shaded region) of these experiments. Averaging only a few samples results in a poor performance. As more samples are averaged, the performance gets closer to the performance obtained by our approximated expectated predictions (red line). This indicates that our expected prediction method approximates the true expectation well.



(a) *sinus dataset* (b) *mnist*

Figure 2: (a) Simple dataset (red points) generated from $\sin(2\pi x) + \epsilon$. The blue line shows the mean output of the network and the gray region shows the standard deviation estimated by our model at the output. (b) Classification errors (CE) of expected predictions with respect to $q(\boldsymbol{W})$ based on our closed-form approximation (red line) and based on model averaging using different numbers of NN samples (blue line). The gray region shows the standard deviation of model averaging.

5 Conclusion

We presented a method to approximate the intractable KL objective of variational inference by a closed-form expression. The approximation is computed by iteratively propagating probabilities through the network to approximate the intractable expected log-likelihood. Unlike recently proposed methods that rely on the reparameterization trick to obtain gradient samples, the gradient of our closed form expression is readily available. Therefore, optimizing our objective does not suffer from high variance gradients. Furthermore, our objective can also be used in more sophisticated optimization algorithms such as quasi-Newton methods that are still not suited for stochastic gradients. The approximation scheme can also be used to compute expected predictions and to produce output uncertainties.

Acknowledgments

This work was supported by the Austrian Science Fund (FWF) under the project number P27803-N15. Furthermore, we acknowledge NVIDIA for providing GPU computing resources.

References

[1] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1613–1622, 2015.

[2] A. Graves. Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2348–2356, 2011.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *International Conference on Computer Vision (ICCV)*, pages 1026–1034, 2015.

[4] J. M. Hernandez-Lobato and R. Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1861–1869, 2015.

[5] G. E. Hinton and D. van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual ACM Conference on Computational Learning Theory (COLT)*, pages 5–13, 1993.

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015. arXiv: 1412.6980.

[7] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR)*, 2014. arXiv: 1312.6114.

[8] H. Larochelle, D. Erhan, A. C. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 473–480, 2007.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814, 2010.

[11] J. Nocedal and S. Wright. *Numerical Optimization*. Springer New York, 2 edition, 2006.

[12] D. Soudry, I. Hubara, and R. Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *Advances in Neural Information Processing Systems 27*, pages 963–971, 2014.