Bayesian Neural Networks

Ilya Zharikov, Roman Isachenko, Artem Bochkarev

Abstract

The recent advances in deep learning allow us to solve really hard real word problems with high accuracy. What these methods lack is interpretability of the model and uncertainty guarantees for the predictions. Bayesian framework allows to solve both these two issues, and with recent development in variational inference technique it is possible to take the best of two worlds and implement bayesian neural network. Our project is dedicated to testing different priors for neurons weights and exploring advantages and disadvantages of bayesian approach in deep learning.

Introduction

Deep learning is one of the most promising and quickly advancing areas of machine learning research [1]. Deep models proved to be superior to other algorithms in many fields such as image classification [2], speech recognition [3], unsupervised learning [4] and reinforcement learning [5]. Still, these models are mostly used as some black box in order to solve complicated problems. We often want to know not only the prediction of the network on given input, but we would like to understand the uncertainty [6] of the model on this object. It is also desirable to be able to do transfer learning [7], i.e. be able to obtain correct results on new types of data, using previous model as prior knowledge.

On the other side of the spectrum there is bayesian approach [8]. The main advantage of this framework is that we operate with parameters distribution, instead of their point estimation. This allows us to calculate uncertainties in predictions and representations. Another advantage of the bayesian approach is that we are free to select prior distribution for the parameters of our model [9], which may include real knowledge or some non-informative assumption. Finally, we may want to build hierarchical models because data is too complex and good priors are hard to guess.

With recent advances in computational technology and GPU it is very compelling to try mixing deep learning with bayesian framework. This would allow us to achieve the same high performance of neural network, as well give us more freedom with different prior selection and interpretability of the results using uncertainty prediction. One of the obstacles for such approach is that full bayesian inference is very memory expensive, given huge parameters number (we need to store whole distribution instead of one point). What is more important, calculation of the model evidence is intractable for high-dimensional data. MCMC sampling methods [10] like Metropolis-Hastings or Gibbs are too computationally expensive to perform on big models.

In our project we decided to implement alternative way for approximation of posterior probability, namely ADVI [11]. The main idea of the method is that we constitute true posterior with some distribution from known parametric family and then we use stochastic optimization procedure to minimize their difference.

We conducted computational experiments on several datasets. We used PyMC3 [12] and Lasagne [13] for probabilistic and deep learning frameworks respectively. We used synthetic datasets as well as MNIST hand-written digits dataset. For these datasets we constructed bayesian neural network models, optimized them with ADVI, played with different priors and drew conclusions from model uncertainty.

Problem Statement

Bayesian approach

Let suppose that $\mathbf{X}$ is the observed data which comes from the unknown distribution $p(\mathbf{X})$. This distribution defines our model and is called model evidence. We assume that our model also contains latent variables $\boldsymbol{\theta}$. The likelihood function is given by conditional probability distribution $p(\mathbf{X}|\boldsymbol{\theta})$. If we know the prior distribution $p(\boldsymbol{\theta})$ over hidden variables, we could use Bayes' theorem to derive the posterior distribution $p(\boldsymbol{\theta}|\mathbf{X})$

$$p(\boldsymbol{\theta}|\mathbf{X}) = \frac{p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{X})}, \tag{1}$$

where the evidence function $p(\mathbf{X})$ could be obtained by integration over hidden variables

$$p(\mathbf{X}) = \int p(\mathbf{X}|\boldsymbol{\theta})p(\boldsymbol{\theta})p(\boldsymbol{\theta}).$$

The posterior distribution shows the transformed initial prior knowledge of $\boldsymbol{\theta}$ given the observed data $\mathbf{X}$.

The main goal of bayesian inference to compute the posterior distribution. In some cases we could derive the posterior in the closed form formula. However, it is impossible for complex models where the latent variables lies in high-dimensional space. The problem is the denominator of (1), since it is obtained by integrating over all possible values of $\boldsymbol{\theta}$.

Suppose that we have the posterior distribution $p(\boldsymbol{\theta}|\mathbf{X})$, then we could use the usual Maximum A Posteriori (MAP) approach to obtain point estimation for the parameters

$$\boldsymbol{\theta}_{\text{MAP}} = \arg\max_{\boldsymbol{\theta}} \left[\ln p(\boldsymbol{\theta}|\mathbf{X})\right] = \arg\max_{\boldsymbol{\theta}} \left[\ln p(\mathbf{X}|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta})\right].$$

In the case of flat prior distribution $p(\boldsymbol{\theta}) = \text{const}$, this estimation coincides with the Maximum Likelihood Estimation (MLE) approach

$$\boldsymbol{\theta}_{\text{MLE}} = \arg\max_{\boldsymbol{\theta}} \left[\ln p(\mathbf{X}|\boldsymbol{\theta})\right].$$

Variational inference

One of the widely used approach to get posterior distribution is sampling methods such as Metropolis-Hastings, Gibbs, NUTS algorithms. The general idea behind sampling methods is to obtain procedure for generating samples from the true posterior distribution. Most of these methods are based on Monte Carlo Markov Chains (MCMC) procedures. The main disadvantages of such methods is that they are very slow for high-dimensional data.

In order to deal with high-dimensional data we could use analytical deterministic approximation of the posterior. Variational inference approach determines the function which is the closest to the desired posterior distribution. The distance is measured by Kullback-Leibler divergence

$$\mathrm{KL}(q||p) = -\int q(y) \ln \left[\frac{p(y)}{q(y)}\right] dy.$$

This distance function is non-negative and equals to zero iff $q \equiv p$

$$\mathrm{KL}(q||p) \geqslant 0; \quad \mathrm{KL}(q||p) = 0 \Leftrightarrow q \equiv p.$$

But the problem is that we don't know the true posterior and can't minimize the Kullback-Leibler divergence explicitly. Variational inference approach solves the equivalent task which follows from the equation:

$$\begin{aligned}\ln p(\mathbf{X}) = \int q(\boldsymbol{\theta}) \ln p(\mathbf{X}) d\boldsymbol{\theta} = \int q(\boldsymbol{\theta}) \ln \left[\frac{p(\mathbf{X}, \boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathbf{X})}\right] d\boldsymbol{\theta} = \\ = \int q(\boldsymbol{\theta}) \ln \left[\frac{q(\boldsymbol{\theta}) p(\mathbf{X}, \boldsymbol{\theta})}{p(\boldsymbol{\theta}|\mathbf{X}) q(\boldsymbol{\theta})}\right] d\boldsymbol{\theta} = -\int q(\boldsymbol{\theta}) \ln \left[\frac{p(\boldsymbol{\theta}|\mathbf{X})}{q(\boldsymbol{\theta})}\right] d\boldsymbol{\theta} + \\ + \int q(\boldsymbol{\theta}) \ln \left[\frac{p(\mathbf{X}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right] d\boldsymbol{\theta} = \mathrm{KL}(q||p) + \mathrm{ELBO}(q).\end{aligned}$$

Here, the log model evidence function is decomposed into the Kullback-Leibler divergence and the Empirical Lower BOund (ELBO) term. Since the left hand side is independent of q function, the minimization of $\mathrm{KL}(q||p)$ is equivalent to the maximization of $\mathrm{ELBO}(q)$ which we can easily compute.

$$q^* = \arg\min_q \mathrm{KL}(q||p) = \arg\max_q \mathrm{ELBO}(q). \tag{2}$$

This lower bound can be expressed in the following way

$$\begin{aligned}\mathrm{ELBO}(q) &= \int q(\boldsymbol{\theta}) \ln \left[\frac{p(\mathbf{X}, \boldsymbol{\theta})}{q(\boldsymbol{\theta})}\right] d\boldsymbol{\theta} = \\ &= \int q(\boldsymbol{\theta}) \ln p(\mathbf{X}, \boldsymbol{\theta}) d\boldsymbol{\theta} - \int q(\boldsymbol{\theta}) \ln q(\boldsymbol{\theta}) d\boldsymbol{\theta} = \\ &= \mathbb{E}_{q(\boldsymbol{\theta})} \left[\ln p(\mathbf{X}, \boldsymbol{\theta})\right] - \mathbb{E}_{q(\boldsymbol{\theta})} \left[\ln q(\boldsymbol{\theta})\right].\end{aligned} \tag{3}$$

The joint probability distribution $p(\mathbf{X}, \boldsymbol{\theta})$ is given by product of likelihood function $p(\mathbf{X})$ and prior distribution $p(\boldsymbol{\theta})$.

Automatic differentiation variational inference

In [link to ADVI] the Automatic Differentiation Variational Inference (ADVI) method was proposed to solve the problem (2).

At the first stage ADVI method automatically transform the original constrained variables $\boldsymbol{\theta}$ to the unconstrained real-valued variables $\boldsymbol{\zeta} = T(\boldsymbol{\theta})$. ADVI then defines the corresponding variational problem on the transformed variables, that is, to minimize $\mathrm{KL}\left(q(\boldsymbol{\zeta})||p(\boldsymbol{\zeta}|\mathbf{X})\right)$. With this transformation, all latent variables are defined on the same space. E. g. if some component θ of $\boldsymbol{\theta}$ should be non-negative ($\theta \in \mathbb{R}_+$), then one could use the logarithm transformation $\zeta = T(\theta) = \log(\theta) \in \mathbb{R}$.

The joint probability function for transformed latent variables equals

$$p(\mathbf{X}, \boldsymbol{\zeta}) = p(\mathbf{X}, T^{-1}(\boldsymbol{\zeta}))|\det J_{T^{-1}}(\boldsymbol{\zeta})|,$$

where $J_{T^{-1}}(\boldsymbol{\zeta})$ is the Jacobian of the inverse T transformation.

The lower bound (3) in real coordinate space is given by

$$\mathrm{ELBO}(q(\boldsymbol{\zeta})) = \mathbb{E}_{q(\boldsymbol{\zeta})}\left[\ln p(\mathbf{X}, T^{-1}(\boldsymbol{\zeta})) + \ln|\det J_{T^{-1}}(\boldsymbol{\zeta})|\right] - \mathbb{E}_{q(\boldsymbol{\zeta})}\left[\ln q(\boldsymbol{\zeta})\right]. \quad (4)$$

The next stage of the ADVI algorithm includes stochastic optimization. Firstly, we assume that $q(\boldsymbol{\zeta})$ comes from fixed parametric family. It allows us to avoid variational optimization and replace it by parameter optimization. Usually ADVI uses $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ family with diagonal $\boldsymbol{\Sigma}$. The general positive-definite form of the covariance matrix $\boldsymbol{\Sigma}$ is also possible, but it leads to excessive computations. After this parametrization the problem (2) is equivalent to

$$\boldsymbol{\mu}^*, \boldsymbol{\Sigma}^* = \arg\max_{\boldsymbol{\mu}, \boldsymbol{\Sigma}} \mathrm{ELBO}(q).$$

The solution of this problem is obtained using stochastic optimization. However, we cannot directly use automatic differentiation on the ELBO. This is because the objective function is expectation over optimized parameters. To avoid this problem one could use elliptical standardization as reparametrization trick [link]. We convert the Gaussian variational approximation to the standard Gaussian $\boldsymbol{\eta} = S(\boldsymbol{\zeta})$

$$q(\boldsymbol{\eta}) = \mathcal{N}(\boldsymbol{\eta}|0, \mathbf{I}).$$

The standardization transforms the variational lower bound from(4) into

$$\mathrm{ELBO}(q(\boldsymbol{\eta})) = \mathbb{E}_{\mathcal{N}(\boldsymbol{\eta}|0,\mathbf{I})}\Big[\ln p\left(\mathbf{X}, T^{-1}(S^{-1}(\boldsymbol{\eta}))\right) + \\ + \ln|\det J_{T^{-1}}(S^{-1}(\boldsymbol{\eta}))|\Big] - \mathbb{E}_{\mathcal{N}(\boldsymbol{\eta}|0,\mathbf{I})}\left[\ln q(\boldsymbol{\eta})\right].$$

Now the expectation is independent of the optimized parameters and we can easily compute gradients over $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

In the original paper a new adaptive step-size gradient sequence was proposed. This sequence is implemented in used frameworks. But it is of no interest for our research.

Numerical experiments

Datasets

In our research we used synthetic data for simple binary classification problem that's not linearly separable and real data such as MNIST dataset for multi-classification problem. One can see the detailed description of used data below.

Synthetic data.

Synthetic data is consist of two-dimensional binary classification datasets (moons and circles) that are challenging to certain algorithms, because they are not linearly separable. Also considered datasets are very useful for visualisation.

1. Moons: two interleaving half circles (see fig.1 (a));
2. Circles: a large circle containing a smaller circle (see fig.1 (b)).

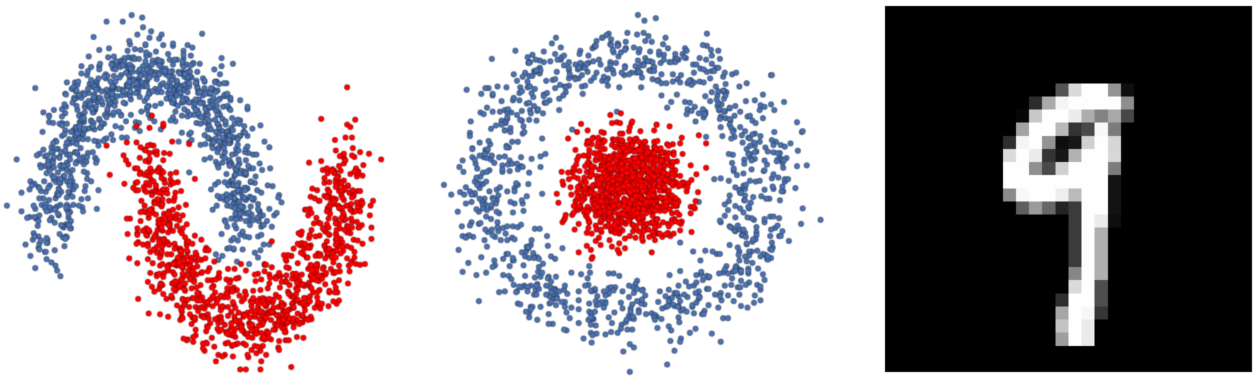
Real data.

As for the real data was selected MNIST[1] dataset. The data files contain gray-scale images of hand-drawn digits, from zero through nine.

Each image is 28 pixels in height and 28 pixels in width, for a total of 784

[1]MNIST dataset — https://www.kaggle.com/c/digit-recognizer/data

pixels in total (see fig.1 (c)). Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255, inclusive.

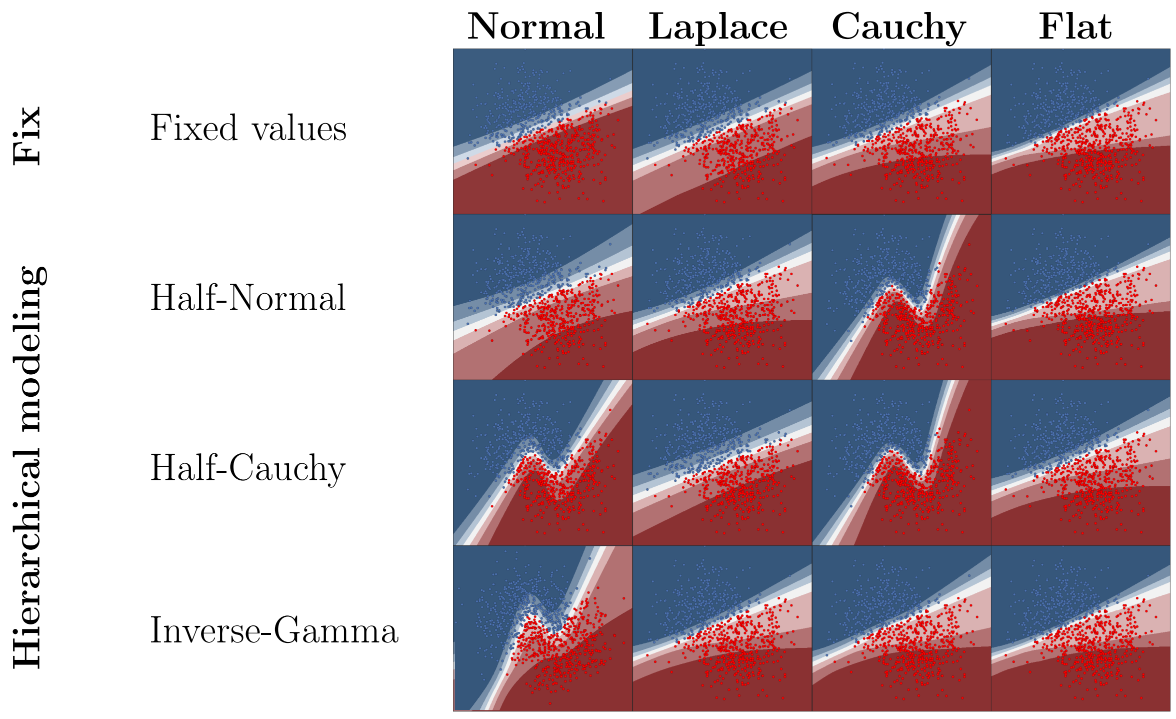


(a) Moons (b) Circles (c) MNIST

Figure 1: Synthetic and real data examples

Priors and hyper-priors influence

On the following one can see the design of the describing experiment.



So we chose different priors and combined it with different hyper-priors and using ADVI approach obtained the posterior distributions of the model parameters. In this experiment our model is a network with the following structure:

•

Contributions

Ilya Zharikov programmed main classes and routines, built project architecture. Also tried to implement experiment on CIFAR-10 and cats-vs-dogs dataset.

Roman Isachenko conducted experiments and drew conclusions from synthetic datasets.

Artem Bochkarev conducted experiments and drew conclusions from MNIST dataset.

Everyone were equally involved in creating presentation, writing this report, as well as general project development.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[3] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42, 2012.

[4] Quoc V Le. Building high-level features using large scale unsupervised learning. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8595–8598. IEEE, 2013.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

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

[7] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.

[8] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[9] William J Browne, David Draper, et al. A comparison of bayesian and likelihood-based methods for fitting multilevel models. *Bayesian analysis*, 1(3):473–514, 2006.

[10] Larry Wasserman. *All of statistics: a concise course in statistical inference*. Springer Science & Business Media, 2013.

[11] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *arXiv preprint arXiv:1603.00788*, 2016.

[12] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in python using pymc3. *PeerJ Computer Science*, 2:e55, 2016.

[13] Sander Dieleman, Jan Schluter, Colin Raffel, Eben Olson, et al. Lasagne: First release., August 2015.