Laplace approximation for Deep NNs

May 23, 2023

Abstract

We introduce the Laplace approximation to the posterior of the parameters given data, then discuss how to apply it to deep neural networks. In particular we discuss computational issues and approximation techniques making the method possible in actual applications.

One of the few cases where $p(\omega|D)$ can be computed exactly is in linear regression problems, as both, $p(D|\omega)$ and $p(\omega)$ are Gaussian in ω and so $p(D)$ is just a Gaussian normalization factor.

1 Our setting

Let $D:=\{(x_i,y_i): x_i\in\mathbb{R}^m, y_i\in\mathbb{R}^d, i=1,\ldots,N\}$ be some data set sampled from an unknown distribution (X,Y), and $\hat{Y}=f_\omega(X)=(f_\omega^1,f_\omega^2,..,f_\omega^d)(X)$, with $\omega=(\omega_1,\omega_2,\ldots,\omega_M)\sim\mathcal{N}(0,\gamma)$, our regression or classification model, e.g. a deep neural network with M layers. We want to compute the *predictive distribution*:

$$p(\hat{y}|\,D,x)=\int p(\hat{y}|\,\omega,x)\,p(\omega|\,D)\,\mathrm{d}\omega,$$

for a new observation x and prediction $\hat{y}$.[1] As always, *the difficult quantity to be computed is the* posterior:

$$p(\omega|\,D)=\frac{p(D|\,\omega)\,p(\omega)}{p(D)},$$

because the *evidence* $p(D)=\int p(D|\,\omega)\,p(\omega)\,\mathrm{d}\omega$ is, in general, intractable. So far we have approximated the posterior by SVI but here we will present a different method, the *Laplace approximation.*

2 Why Laplace?

- In practice, Laplace can be much faster than SVI as we do not need to train the model from scratch as we do with ELBO maximization. As we will see, if you have your MLE model, then Laplace is an easy and cheap approximaton to get a Bayesian

1. For classification, $p(\hat{y}|\,\omega,x)=\frac{\mathrm{e}^{f_\omega^i(x)}}{\sum_{j=1}^d \mathrm{e}^{f_\omega^j(x)}}$ and for regression $p(\hat{y}|\,\omega,x)\propto\exp\left(-\frac{\sum_{i=1}^N (f_\omega(x)_i-\hat{y}_i)^2}{2\,\sigma^2}\right)$, with σ the standar deviation.

model almost for free.

- For SVI one needs to provide (i.e. guess) a class of functions for the variational approximation (the “guide” in PYRO) but this is not always easy. With Laplace one has a (generally) reasonable unimodal Gaussian approximation for the posterior around its maximum.

3 The Laplace approximation

In general, $\log p(\omega|\, D)$ is related to the cost function in MLE.[2] For instance, in our regression setting, we have:

$$\begin{aligned}\log p(\omega|\, D) &= \log p(D|\, \omega) + \log p(\omega) - \log p(D)\\ &= -\sum_{i=1}^{N} (f_\omega(X_i) - y_i)^2 - \frac{1}{\gamma^2}\sum_{l=1}^{M} \omega_l^2 + C(\sigma, D)\\ &= -L(\omega, D) + C(\sigma, D). \end{aligned} \tag{1}$$

From (1) we can see that the log posterior probability is basically the negative mean-squared error used in MLE for regression $L(\omega, D)$ (with an L_2 regularization term), up to a normalization constant, $C(\sigma, D)$, that is a function of the evidence $p(D)$ and the variance, σ, coming from the likelihood, $p(D\,|\,\omega)$.

A second-order Taylor expansion of $L(\omega, D)$:

$$\begin{aligned}L(\omega, D) \simeq\; & L(\omega^{\text{MLE}}, D)\\ & + \frac{1}{2}(\omega - \omega^{\text{MLE}})^T \nabla_\omega^2 L(\omega, D)|_{\omega^{\text{MLE}}}(\omega - \omega^{\text{MLE}}),\end{aligned}$$

yields to: $p(\omega|\, D) \simeq \mathrm{e}^{-L(\omega^{\text{MLE}}, D) + C(\sigma, D)}\, \mathrm{e}^{-\frac{1}{2}(\omega - \omega^{\text{MLE}})^\top \Sigma^{-1}(\omega - \omega^{\text{MLE}})}\,.$ (2)

Observe that (2) is a multivariate normal distribution with covariance Σ . This means that its normalization constant is $1/\sqrt{\det(2\,\pi\,\Sigma)}$. Finally we find the *Laplace approximation*:

$$p(\omega|\, D) \simeq \frac{1}{\sqrt{\det(2\,\pi\,\Sigma)}}\, \mathrm{e}^{-\frac{1}{2}(\omega - \omega^{\text{MLE}})^\top \Sigma^{-1}(\omega - \omega^{\text{MLE}})}, \tag{3}$$

where

$$\Sigma = \nabla_\omega^2 L(\omega, D)|_{\omega^{\text{MLE}}}.$$

In summary, we have approximated the posterior by a Gaussian distribution.

2. In general the log of the posterior is related to the MAP (Maximum A Posteriori) estimate but as here the prior is Gaussian, the MLE and the MAP are basically the same.

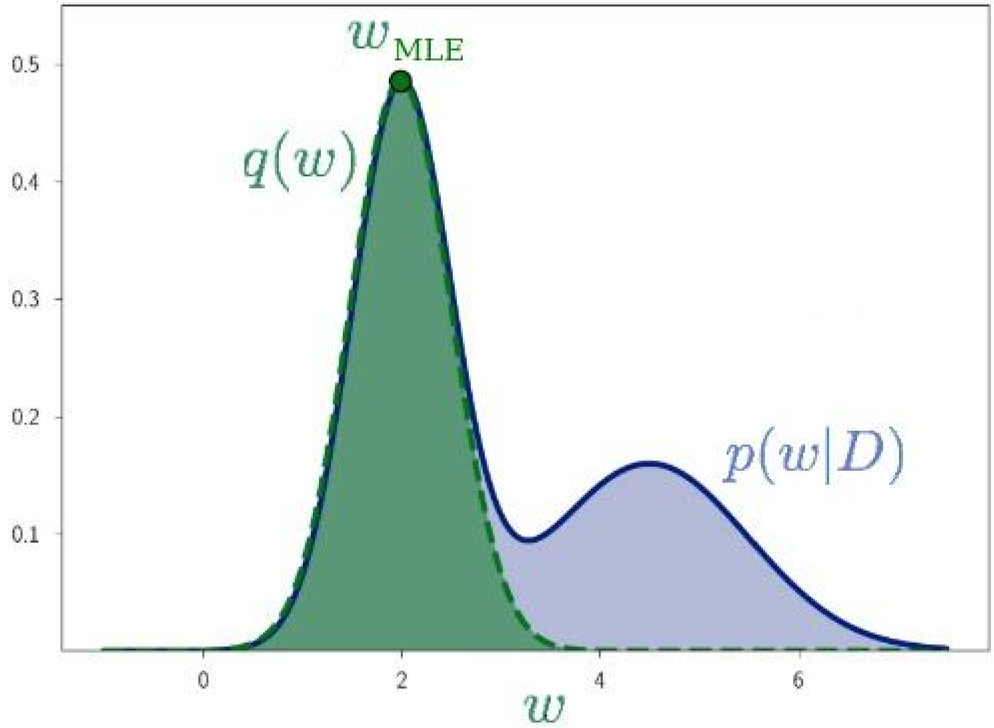

Figure 1. Laplace approximation $q(\omega)$ (in green) of the posterior $p(\omega|\,D)$ (in blue).

In practice, to reach the ω_{MLE} we will use a first-order optimization method like (stochastic) gradient descent or similar and the "only" thing we will need to compute for the posterior will be the covariance matrix Σ and its inverse. This means that we will be able to use transfer learning in order to perform Bayes inference on MLE models trained on large clusters!

4 Laplace approximation and DNNs

Remember that Laplace assumes an expansion around ω^{MLE} for the posterior and so (4) is a reasonable assumption within this regime.

4.1 Main issues

Issue 1. The covariance matrix $\Sigma = \nabla_\omega^2 L(\omega, D)|_{\omega^{\mathrm{MLE}}}$ in (3) is in general **not positive definite** (not a good covariance matrix).

It can be shown that by linearizing our model around ω^{MLE}, i.e. doing

$$f_\omega(x) \simeq f_\omega^{\mathrm{MLE}}(x) + \sum_i \left.\frac{\partial f_\omega(x)}{\partial \omega_i}\right|_{\omega^{\mathrm{MLE}}} (\omega_i - \omega_i^{\mathrm{MLE}}) \tag{4}$$

the covariance matrix can be written in terms of a *positive-definite matrix*, $G(\omega, D)$, known as the *Generalized Gauss-Newton (GGN) matrix* (see Appendix A):

$$\Sigma = \nabla_\omega^2 L(\omega, D)|_{\omega^{\mathrm{MLE}}} \simeq G(\omega^{\mathrm{MLE}}, D) \geq 0, \tag{5}$$

Issue 2. For a large DNN, computing $G^{-1}(\omega, D)$ is $\mathcal{O}(|G|^3)$.

It can be shown that $G(\omega, D)$ has a useful representation for almost all cost functions $L(\omega, D)$ used in practical applications (see Appendix B):

$$G(\omega, D) = \begin{pmatrix} \sum_n A_{1,1}(x_n, y_n) & \cdots & \sum_n A_{1,M}(x_n, y_n) \\ \sum_n A_{2,1}(x_n, y_n) & \cdots & \sum_n A_{2,M}(x_n, y_n) \\ \vdots & \ddots & \vdots \\ \sum_n A_{M,1}(x_n, y_n) & \cdots & \sum_n A_{M,M}(x_n, y_n) \end{pmatrix} \tag{6}$$

with $A_{i,j}(x_n, y_n)$ a matrix of dimension $\mathcal{O}(\dim(\omega_i) \dim(\omega_j))$, coupling the i-th and j-th layers of the DNN. As the matrices $A_{i,j}(x_n, y_n)$ are large it is hard to compute the sum over n in (6). However, we can use another approximation, the *Kronecker Factored Approximate Curvature (KFAC)*. It can be shown that $A_{i,j}(x_n, y_n) = B_{i,j}(x_n, y_n) \otimes C_{i,j}(x_n, y_n)$ with $\otimes$ the Kronecker product (see appendix B) and so we define the KFAC as:

$$\sum_n A_{i,j}(x_n, y_n) \simeq \sum_n B_{i,j}(x_n, y_n) \otimes \sum_n C_{i,j}(x_n, y_n). \tag{7}$$

A large sum on the large matrices $A_{i,j}$ is replaced by many sums of smaller matrices $B_{i,j}$ and $C_{i,j}$ of order much smaller than $\mathcal{O}(\dim(\omega_i) \dim(\omega_j))$.

In addition to KFAC other approximations are used to make the matrix A sparser and therefore easier to invert, like the *diagonal KFAC* ($A_{i,j} = 0$ for $i \neq j$) or the *tri-diagonal KFAC* ($A_{i,j} = 0$ for $i > j + 1$ and $i < j - 1$).

Issue 3. The predictive probability distribution,

$$p(\hat{y}|D, x) \simeq \int p(\hat{y}|\omega, x)\, p^{\text{Laplace}}(\omega|D)\, \mathrm{d}\omega$$

is hard to compute for DNNs.

Approximation: The idea is to use the linear regime (4) to approximate the predictive distribution.

Regression: With the usual assumption that the likelihood, $p(\hat{y}|\omega, x)$, is normally distributed with variance σ^2 the posterior can be computed analytically and is given by $p(\hat{y}|D, x) = N(\hat{y}; f_{\omega_{\text{MAP}}}(x), J(x)^T G\, J(x) + \sigma^2 \mathbb{I})$ with $J(x) = \nabla_\omega f_\omega(x)|_{\omega_{\text{MAP}}}$ and $f_\omega(x)$ the DNN (see [DKI+21]).

Classification: After probit-type approximation on the softmax $p(\hat{y}|\omega, x)$ the predictive becomes a categorical distribution (see [DKI+21]).

Exercise 1. (14-laplace-dnn-exercise.ipynb) In this exercise, we compute the Laplace approximation for a classification problem with a DNN, using the libraries PYRO [BCJ+19, PPJ22] and LAPLACE [DKI+21].

4.2 Subnetwork approximation

To reduce the computational burden, one can replace a fully Bayesian network by a Bayesian subnetwork ω_S:

$$\begin{aligned} p_S^{\text{Laplace}}(\omega|D) &\simeq p^{\text{Laplace}}(\omega_S|D) \prod_r \delta(\omega_r - \omega_r^{\text{MAP}}) \\ &= N(\omega_S, \omega_S^{\text{MAP}}, G_S^{-1}) \prod_r \delta(\omega_r - \omega_r^{\text{MAP}}). \end{aligned}$$

In practical applications we will choose a small subnetwork $|G_S| < |G|$. The main idea is to minimize the distance between p^{Laplace} and p_S^{Laplace}. For that, G_S is fully computed but a diagonal approximation to G is used in order to invert it easily.

This turns out to be a good approximation as explained in [DNA+21].

Appendix A The *Generalized Gauss-Newton (GGN) matrix*

Using the chain rule in $\nabla^2_\omega L(\omega, D)|_{\omega^{\text{MLE}}}$:

$$\begin{aligned}\sum_i \frac{\partial}{\partial\omega_l}\frac{\partial}{\partial\omega_m} L_i(\vec{f}_\omega)\bigg|_{\omega^{\text{MLE}}} &= \sum_{i,\alpha,\beta} \frac{\partial f^\alpha_\omega}{\partial\omega_l}\frac{\partial^2 L_i(f_\omega)}{\partial f^\alpha_\omega \partial f^\beta_\omega}\frac{\partial f^\beta_\omega}{\partial\omega_m}\bigg|_{\omega^{\text{MLE}}} \\ &+ \sum_{i,\alpha} \frac{\partial L_i(f_\omega)}{\partial f^\alpha_\omega}\frac{\partial^2 f^\alpha_\omega}{\partial\omega_l\partial\omega_m}\bigg|_{\omega^{\text{MLE}}}\end{aligned} \tag{8}$$

for $i = 1, .., N; \alpha, \beta = 1, .., d$, and where we used the notation

$$L(\omega, D) = \sum_{i=1}^N L(x_i, y_i, f_\omega) = \sum_{i=1}^N L_i(f_\omega).$$

A.1 Approximation

It is easy to see that the blue term in (8) is zero for a perfect regressor (as it is proportional to $f_\omega(X_i) - y_i$) and similarly for a perfect classifier, so we will approximate $\nabla^2_\omega L(\omega, D)$ by:

$$\begin{aligned}(\nabla^2_\omega L(f_\omega, D))_{l,m} &= \sum_i \frac{\partial}{\partial\omega_l}\frac{\partial}{\partial\omega_m} L_i(f_\omega) \\ &\simeq \sum_{i,\alpha,\beta} \frac{\partial f^\alpha_\omega}{\partial\omega_l}\frac{\partial^2 L_i(f_\omega)}{\partial f^\alpha_\omega \partial f^\beta_\omega}\frac{\partial f^\beta_\omega}{\partial\omega_m} \\ &:= G(\omega, D)_{l,m} \\ &\geq 0,\end{aligned} \tag{9}$$

where the positive definite matrix $G(\omega, D)$ is called *Generalized Gauss-Newton (GGN) matrix*.

Observe that the argument we are using here for the approximation (9) is not very strong, as in real applications we never have a perfect regresor or classifier and this means that for large data sets the blue term in (8) could be quite large. However, there is a better way to see that (9) is a good approximation by linearizing the DNN $f_\omega(x)$ around ω^{MAP}:

$$f_\omega(x) \simeq f^{\text{MAP}}_\omega(x) + \sum_i \frac{\partial f_\omega(x)}{\partial\omega_i}\bigg|_{\omega^{\text{MAP}}} (\omega_i - \omega^{\text{MAP}}_i) \tag{10}$$

By using (10) it is easy to see that the blue term in (8) is zero (as it involves second order derivatives in ω), i.e. the Laplace approximation in the linear regime becomes:

$$\begin{aligned}L^{\text{Laplace}}(\omega, D) &= L(\omega^{\text{MAP}}, D) \\ &+ \frac{1}{2}(\omega - \omega^{\text{MAP}})^T \nabla^2_\omega L(\omega, D)|_{\omega^{\text{MAP}}}(\omega - \omega^{\text{MAP}}) \\ &= L(\omega^{\text{MAP}}, D) \\ &+ \frac{1}{2}(\omega - \omega^{\text{MAP}})^T G(\omega^{\text{MAP}}, D)\,(\omega - \omega^{\text{MAP}})\end{aligned}$$

Appendix B Inverting the GGN

For a large DNN, computing $G^{-1}(\omega, D)$ is of $\mathcal{O}(|G|^3)$. It can be shown that the GGN matrix is equal to the empirical Fisher matrix for many cost functions of interest, including the usual regression and classification cases. The *empirical Fisher matrix* is given by

$$F(\vec{\omega}, D) := \sum_{n=1}^{N} \nabla_\omega \log p(y_n|x_n)\, \nabla_\omega \log p(y_n|x_n)^\top. \tag{11}$$

In order to write the matrix elements of (11) in a simple way we will use the notation $\log p(y_n|x_n, \vec{\omega}) = l(x_n, y_n, \vec{\omega}) = l_n$, with $\sum_n l_n$ the MLE cost function for regression without the regularization term. Finally, linearizing the weights ω, i.e. $(\omega_i)_{\gamma,\mu} \to \omega_{i_\alpha}$ we find:

$$F_{ij;\alpha\beta}(\omega, D) = \sum_{n=1}^{N} (\nabla_{\omega_{i,\alpha}} l_n)\, (\nabla_{\omega_{j,\beta}} l_n)^\top, \tag{12}$$

where i, j are indexes on the DNN layers and α, β on the weight space. Observe that the elements $F_{i,j}$ in (12) are in general big matrices and so the sum for large N is hard to compute.

B.1 Approximation

i. Diagonal approximation:

$$F_{ij;\alpha\beta}(\omega, D) \to F_{ij;\alpha\beta}(\omega, D)\, \delta_{i,j}\, \delta_{\alpha\beta}$$

ii. Kronecker Factored Approximate Curvature (KFAC):

$$\begin{aligned} F_{ij} &= \sum_n \nabla_{\omega_i} L_n \nabla_{\omega_j} L_n^T \\ &= \mathbb{E}[a_{i-1}\, a_{j-1}^T \otimes g_i\, g_j^T] \\ &\simeq \mathbb{E}[a_{i-1}\, a_{j-1}^T] \otimes E\,[g_i\, g_j^T] \\ &= \sum_n B_{i,j}(x_n, y_n) \otimes \sum_n C_{i,j}(x_n, y_n) \end{aligned}$$

with $B_{i,j}$ and $C_{i,j}$ matrices of smaller order than $\nabla_{\omega_i} L_n \nabla_{\omega_j} L_n^T$ (See [MG15, Rod22]).

Bibliography

[BCJ+19] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep Universal Probabilistic Programming. *Journal of Machine Learning Research*, 20(28):1–6, 2019.

[DKI+21] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace Redux - Effortless Bayesian Deep Learning. In *Advances in Neural Information Processing Systems 34*. 2021.

[DNA+21] Erik Daxberger, Eric Nalisnick, James U. Allingham, Javier Antoran, and Jose Miguel Hernandez-Lobato. Bayesian Deep Learning via Subnetwork Inference. In *Proceedings of the 38th International Conference on Machine Learning*, pages 2510–2521. PMLR, jul 2021.

[EDHK21] Runa Eschenhagen, Erik Daxberger, Philipp Hennig, and Agustinus Kristiadi. Mixtures of Laplace Approximations for Improved Post-Hoc Uncertainty in Deep Learning. In *Bayesian Deep Learning Workshop, NeurIPS 2021*. ArXiv, nov 2021.

[IKB21] Alexander Immer, Maciej Korzepa, and Matthias Bauer. Improving predictions of Bayesian neural nets via local linearization. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 703–711. PMLR, mar 2021.

[KHB19] Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems 32*, volume 32. Curran Associates, Inc., 2019.

[MG15] James Martens and Roger Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2408–2417. PMLR, jun 2015.

[PPJ22] Du Phan, Neeraj Pradhan, and Martin Jankowiak. Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro. In *Program Transformations for significado karate doML Workshop at NeurIPS 2019*. Jul 2022.

[RBB18] Hippolyt Ritter, Aleksandar Botev, and David Barber. A Scalable Laplace Approximation for Neural Networks. In *Sixth International Conference on Learning Representations*. Vancouver, Canada, 2018.

[Rod22] Ivan Rodriguez. TfL Seminar: Going Bayesian through Laplace approximation. June 2022.