Projet Statistiques Bayesiennes

Yarin GAL, Zoubin GHAHRAMANI

Dropout as a Bayesian approximation

Élèves:
Albane Miron de l'Espinay
Erwan Bourceret
Matthieu Doutreligne

January 30, 2018

Abstract

This document reviews, comments and implements some results from [Gal and Ghahramani(2016)].

After stating the motivation of bayesian approximation for dropout in Neural Networks (NNs), we recall the classical dropout in NNs. Then, we explain the gaussian process modeling of this setup developped by [Gal and Ghahramani(2016)]. We deduce uncertainty estimation from this model and comment our implementation applied to real data.

Contents

1 Motivation

In the classification or regression problem, we look for a relevant function to predict a label or a value for an input data. This function must predict well for data points that are close to points that we have already seen and may have bad results for new points far from those already learned.

Neural Networks models with dropout are widely used in both regression and classification problems but they do not give any info on the precision of their output. The main goal of the paper is to show that the distribution of the output of these neural networks can be approximated using a Gaussian Process.

2 Dropout in Neural Network

For the simplicity of notation and without loss of generality (in the case of feed-forward Neural Networks) we study the case of a single hidden layer neural network. Let $\boldsymbol{x}$ be the input vector of dimension Q. Let K be the number of hidden units, and the activation function be σ (non-linear). We also introduce a weight matrices : $\boldsymbol{W}_1$ of dimension $K \times Q$ which makes the liaison from the input to the hidden layer and $\boldsymbol{W}_2$ of dimension $1 \times K$, and a biais vector $\boldsymbol{b}$ of dimension K. The standard Neural Network (NN) model would then outputs :

$$\hat{y} = \boldsymbol{W}_2 \cdot \sigma(\boldsymbol{W}_1 \cdot \boldsymbol{x} + \boldsymbol{b})$$

The loss function $\mathcal{L}$ is defined as follows :

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} E(y_i, \hat{y}_i) + \lambda_1 \|\boldsymbol{W}_1\|_2^2 + \lambda_2 \|\boldsymbol{W}_2\|_2^2 + \lambda_3 \|\boldsymbol{b}\|_2^2 \tag{1}$$

Where E is a classic error function. The training stage of the NN aims at minimizing this loss function.

In Dropout Models, the process randomly assigns the value $\mathbf{0}$ to some element of the vector $\boldsymbol{W}_2$ with a certain probability. It means that some of the rows of $\boldsymbol{W}_1$ are not taking into account when we compute the gradient. As each row represent an unit, it can be seen as a random deactivation of

units for each iteration and computes the gradient of the loss function $\mathcal{L}(y, \hat{y})$ with these new values of $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$, where y is the target value.

3 Gaussian process model of Neural Network

In this part, we derive quickly the derivation from [Gal and Ghahramani(2016)] establishing that Neural Networks with Dropouts are equivalent to an approximation of deep Gaussian Process first introduced in.

3.1 Gaussian processes

Gaussian Process (GP) are a popular method to model distributions over functions. They are robust to overfitting an allow the computation of uncertainty estimates.

In a classic regression task using GPs, we note $(x_i)_{i=1\cdots N} \in \mathbb{R}^Q$, the features and $(y_i)_{i=1\cdots N} \in \mathbb{R}^D$ the output variable. We express the **posterior** as :

$$p(f|\mathbf{X}, \mathbf{Y}) \propto \underbrace{p(\mathbf{Y}|\mathbf{X}, f)}_{likelihood} \underbrace{p(f)}_{prior}$$
$$\text{with } \mathbf{X} \in \mathbb{R}^{N \times Q} \text{ and } \mathbf{Y} \in R^{N \times d}$$

In the context of gaussian processes, we consider the following distributions :

$$\mathbf{F}|\mathbf{X} \sim \mathcal{N}(0, K(\mathbf{X}, \mathbf{X})) \text{ for the } \textbf{prior}$$
$$\mathbf{Y}|\mathbf{F} \sim \mathcal{N}(\mathbf{F}, \tau^{-1} \cdot I_N) \text{ for the } \textbf{likelihood}$$

with

$K(\mathbf{X}, \mathbf{X})$ the $N \times N$ covariance matrix

τ^{-1} a precision parameter.

Finally, we can derive the posterior probability at a new point (x, y) given the data $(\mathbf{X}, \mathbf{Y})$ and the parameteres w of f:

$$p(y|x, \mathbf{X}, \mathbf{Y}) = \int_w p(y|x, w) p(w|\mathbf{X}, \mathbf{Y}) \mathrm{d}w$$

The computation of the posterior in this expression is intactable because of the inversion of the covariance $K(\cdot, \cdot)$, so we have to make use of variationnal inference.

3.2 Variationnal Inference

Computing the posterior is delicate, so we use the variational distribution $q(w)$ instead of $p(w|\mathbf{X}, \mathbf{Y})$. This variational distribution is chosen close to $p(\cdot|\mathbf{X}, \mathbf{Y})$ in the sense that it minimizes the Kullback Leibler divergence between these two distributions:

$$q(w) = \underset{\tilde{q}}{\text{argmax}}\ \mathbf{KL}\left[\tilde{q}(w)\ ||\ p(w|X, Y)\right] = \underset{\tilde{q}}{\text{argmax}} \sum_{w} \tilde{q}(w) \log \frac{\tilde{q}(w)}{p(w|X, Y)}$$

Which is equivalent to maximize the *log evidence lower bound*:

$$\mathcal{L}_{VI} = \int q(w) \log p(Y|X, w) \mathrm{d}w - \mathbf{KL}[q(w)||p(w)]$$

3.3 Dropout as variational approximation

We will detail here some steps leading to the model of the Neural Networks with dropouts before each layer as an approximated Gaussian Process with Variational Inference. We focus on the regression framework given that the classification case can be easily recovered essentially by adding a softmax step.

verifier que c'est bien juste ça

3.3.1 Reparametrization of the NN with dropouts

We first have to precise the GP. Considering a non-linearity function $\sigma(\cdot)$, we can define the covariance function K:

$$K(x, y) = \int \sigma(w^T x + b)\sigma(w^T x + b)p(w)p(b)\mathrm{d}w\mathrm{d}b$$

Approximated by Monte Carlo Integration :

$$\hat{K}(x, y) = \frac{1}{K} \sum_{k=1}^{K} \sigma(w_k^T x + b_k)\sigma(w_k^T x + b_k)$$

where $w_k \sim p(w)$ and $b_k \sim p(b)$.

We note $\mathbf{W_1} = [w_k]_{k=1}^K$ the $Q \times K$ matrix parametrising the covariance function and obtain the predictive distribution:

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{F})p(F|\mathbf{W_1}, b)p(\mathbf{W_1})p(b)\mathrm{d}\mathbf{F}\mathrm{d}\mathbf{W_1}\mathrm{d}b$$

Introducing $\Phi(x, \mathbf{W_1}, b) = \frac{1}{\sqrt{K}}\sigma(\mathbf{W_1}^T x + b) \in \mathbb{R}^K$, and writing $\Phi = [\Phi(x_n, W_1, b)]_{n=1}^N$ we have $\hat{K}(X, X) = \Phi\Phi^T \in \mathbb{R}^{K\times K}$. We can then rewrite the predictive distribution:

$$p(\mathbf{Y}|\mathbf{X}) = \int \mathcal{N}(Y|\ 0, \Phi\Phi^T + \tau^{-1}\mathbb{I}_N)p(\mathbf{W_1})p(b)\mathrm{d}\mathbf{W_1}\mathrm{d}b$$

We look at the distribution over $\mathbf{Y}$ as a joint normal distribution over the D columns of $\mathbf{Y}$: $\mathcal{N}(Y|0, \Phi\Phi^T + \tau^{-1}\mathbb{I}_N) = \mathcal{N}(y_1..y_D|0, \Phi\Phi^T + \tau^{-1}\mathbb{I}_N)$ where we rewrite these distributions with a new auxiliary random variable $w_d \sim \mathcal{N}(0, \mathbb{I}_k)$:

$$\mathcal{N}(y_d|0, \Phi\Phi^T + \tau^{-1}\mathbb{I}_N) = \int \mathcal{N}(y_d|\Phi w_d, \tau^{-1}\mathbb{I}_N)\mathcal{N}(w_d|0, \mathbb{I}_K)dw_d$$

Writing $W_2 = [w_d]_{d=1}^D$, we finally obtain the reparametrization of the GP:

$$p(\mathbf{Y}|\mathbf{X}) = \int p(\mathbf{Y}|\mathbf{X}, \mathbf{W_1}, \mathbf{W_2}, b)p(\mathbf{W_1})p(\mathbf{W_2})p(b)dW_1dW_2d_b$$

3.3.2 Variational inference in the GP model

We have to define a variational distribution on W_1, W_2, b and to develop the objective of maximization on the log evidence lower bound. We recall that the goal is to link this new objective to the loss function of the classic NN with Dropout (1).

Let $q(\mathbf{W_1}, \mathbf{W_2}, b) = q(\mathbf{W_1})q(\mathbf{W_2})q(b)$. The variational distribution on W_1 and W_2 are both two components gaussian mixture factorized over the number of units in the layer. And q(b) is a simple Gaussian distribution:

$$\begin{aligned}
q(\mathbf{W_1}) &= \prod_{q=1}^{Q} q(w_q) \\
q(w_q) &= p_1\mathcal{N}(m_q, \boldsymbol{\sigma}^2\mathbb{I}_K) + (1-p_1)\mathcal{N}(0, \boldsymbol{\sigma}^2\mathbb{I}_K) \\
q(\mathbf{W_2}) &= \prod_{k=1}^{K} q(w_k) \\
q(w_k) &= p_2\mathcal{N}(m_k, \boldsymbol{\sigma}^2\mathbb{I}_D) + (1-p_2)\mathcal{N}(0, \boldsymbol{\sigma}^2\mathbb{I}_D) \\
q(b) &= \mathcal{N}(m, \boldsymbol{\sigma}^2\mathbb{I}_K)
\end{aligned}$$

The two components reflect the Bernoulli dropouts for each weights of the classic dropout NN.

We have now to maximize the log evidence lower bound over the variational distribution parameters, $M_1 = [m_q]_{q=1..Q}, M_2 = [m_k]_{k=1..K}$ and m:

$$\mathcal{L}_{GP-VI} = \int q(\mathbf{W_1}, \mathbf{W_2}, b) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W_1}, \mathbf{W_2}, b) - \mathbf{KL}[q(\mathbf{W_1}, \mathbf{W_2}, b)||p(\mathbf{W_1}, \mathbf{W_2}, b)]$$

From now on, we will only work on regression tasks

Interest to the first term The previous section let us rewrite $\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W_1}, \mathbf{W_2}, b)$ as a sum of gaussians over the columns of $\mathbf{Y}$ (on y_d):

$$\begin{aligned}\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W_1}, \mathbf{W_2}, b) &= \sum_{d=1}^{D} \log \mathcal{N}(y_d; \Phi w_d, \tau^{-1}\mathbb{I}_N) \\ &= -\frac{ND}{2} log(2\pi) + \frac{ND}{2} log(\tau) - \sum_{d=1}^{D} \frac{\tau}{2}||y_d - \Phi w_d||_2^2\end{aligned}$$

Instead of summing over the columns we can sum over the rows. Recalling that $\hat{y}_n = \Phi(x_n, \mathbf{W_1}, b)\mathbf{W_2}$ we have :

$$\begin{aligned}\log p(\mathbf{Y}|\mathbf{X}, \mathbf{W_1}, \mathbf{W_2}, b) &= \sum_{n=1}^{N} \log \mathcal{N}(y_n; \Phi(x_n, \mathbf{W_1}, b)\mathbf{W_2}, \tau^{-1}\mathbb{I}_D) \\ &= -\frac{ND}{2} \log(2\pi) + \frac{ND}{2} \log(\tau) - \sum_{n=1}^{N} \frac{\tau}{2}||y_n - \hat{y}_n||_2^2\end{aligned}$$

We reparametrize the distributions on W_1, W_2, b in order to integrate over the standard normal and the Bernoulli distributions.
We note :

- $q(z_{1,q}) = Bernoulli(p_1)$ for $q = 1..Q$
- $q(\epsilon_1) = [\mathcal{N}(0, \mathbb{I}_Q)]_{k=1..K}$

- $q(z_{2,k}) = Bernoulli(p_2)$ for $k = 1..K$
- $q(\epsilon_2) = [\mathcal{N}(0, \mathbb{I}_K)]_{q=1..Q}$
- $q(\epsilon) = \mathcal{N}(0, \mathbb{I}_K)$

we have:

$$W_1 = z_1(M_1 + \boldsymbol{\sigma}\epsilon_1) + (1 - z_1)\boldsymbol{\sigma}\epsilon_1 \tag{2}$$
$$W_2 = z_2(M_2 + \boldsymbol{\sigma}\epsilon_2) + (1 - z_2)\boldsymbol{\sigma}\epsilon_2 \tag{3}$$
$$b = m + \sigma\boldsymbol{\epsilon} \tag{4}$$

Using Monte-Carlo integration on this new parametrization, we get:

$$\int q(\mathbf{W_1}, \mathbf{W_2}, b) \log p(\mathbf{Y}|\mathbf{X}, \mathbf{W_1}, \mathbf{W_2}, b) \mathrm{d}W_1 \mathrm{d}W_2 \mathrm{d}b \approx \sum_{n=1}^{N} \log p(y_n|x_n, \hat{W}_1^n, \hat{W}_2^n, \hat{b}_n)$$

and where we note $\hat{W}_1^n, \hat{W}_2^n, \hat{b}_n$ the quantities defined in 2 sampled over the two Bernoullis and the standard gaussians distributions.

Interest to the Kullback-Leibler divergence Next, we have to approximate the second term of the log evidence lower bound, the Kullback-Leibler divergence between the variational distribution $q(W_1, W_2, b)$ and the prior over $\omega = (W_1, W_2, b)$ where $p(\omega)$ is a multivariate normal distribution. We will not derive this approximation here but only express the results:

$$KL[q(W_1)||p(W_1)] \approx QK(\boldsymbol{\sigma}^2 - log(\boldsymbol{\sigma}^2) - 1) + \frac{p_1}{2} \sum_{q=1}^{Q} m_q^T m_q + C$$

$$KL[q(W_2)||p(W_2)] \approx QK(\boldsymbol{\sigma}^2 - log(\boldsymbol{\sigma}^2) - 1) + \frac{p_2}{2} \sum_{k=1}^{K} m_k^T m_k + C'$$

$$KL[q(b)||p(b)] = \frac{1}{2}(m^T m + K(\boldsymbol{\sigma}^2 - log(\boldsymbol{\sigma}^2) - 1)) + C''$$

where C,C' and C" are constants wrt to M_1, M_2 and m.

Removing the constant terms, we finally obtain:

$$\mathcal{L}_{GP-MC} \propto \sum_{n=1}^{N} \log p(y_n|x_n, \hat{W}_1^n, \hat{W}_2^n, \hat{b}^n) - \frac{p_1}{2}||M_1||_2^2 - \frac{p_2}{2}||M_2||_2^2 - \frac{1}{2}||m||_2^2$$

$$\mathcal{L}_{GP-MC} \propto \sum_{n=1}^{N} \frac{\tau}{2}||y_n - \hat{y}_n||_2^2 - \frac{p_1}{2}||M_1||_2^2 - \frac{p_2}{2}||M_2||_2^2 - \frac{1}{2}||m||_2^2$$

Scaling this objective function (to be maximized) and multiplying it by -1, we recover the equation 1 (to be minimized):

$$\mathcal{L}_{GP-MC} \propto \frac{1}{2N}||y_n - \hat{y}_n||_2^2 + \frac{p_1}{2\tau N}||M_1||_2^2 + \frac{p_2}{2\tau N}||M_2||_2^2 + \frac{1}{2\tau N}||m||_2^2$$

Before disserting about the model uncertainty, we can can see that our choice of variational distribution led us to a loss function close to the one we have for the dropout model with small approximations. Allowing the identification we can have the weights decay of the NN model : $\lambda_i = \frac{p_i}{2\tau N}, i = 1, 2$.

4 Assessing uncertainty for regression tasks

In the previous paragraph we have shown that the loss function for Dropout Neural Networks looks like the loss function used to approximate a Gaussian Process. More specifically the identification of the objective functions allows us to identify the weights decay of the NN model : $\lambda_i = \frac{p_i}{2\tau N}, i = 1, 2$.

Plus, by taking a value close to zero for $\boldsymbol{\sigma}$ we can do the following approximation :

$$W_1^n \approx z_1^n M_1,\ W_2^n \approx z_2^n M_2,\ b \approx m$$

and have

$$\hat{y}_n = \sqrt{\frac{1}{K}}\sigma(x_n z_1^n M_1 + m) z_2^n M_2$$

where z_1^n and z_2^n are Bernoulli vectors of parameters p_1 and p_2.

Thus, we can having maximize the quantity above with the corresponding value for the parameter τ and get the optimal values for M_1, M_2 and m as they would have been computed with the NN. We can then empirically simulate predictive moments from the distribution of interest

$$q(\boldsymbol{y}^*|\boldsymbol{x}^*) = \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w})q(\boldsymbol{w})dw$$

by using Monte Carlo simulation : we simulate T Bernoulli vectors z_1 of size Q and T Bernoulli vectors z_2 of size K to get T predictive values for $\hat{y}^*$.

$$\hat{y}_t^* = \sqrt{\frac{1}{K}}\sigma(x^* z_1^t M_1 + m)z_2^t M_2$$

and write :

$$E_{q(y^*|x^*)}(y^*) \approx \frac{1}{T}\sum_{i=1}^{T} y_t^* \tag{5}$$

$$Var_{q(y^*|x^*)}(y^*) = \tau^{-1} I_D + \frac{1}{T}\sum_{i=1}^{T}(y_t^*)^T y_t^* - E_{q(y^*|x^*)}((y^*)^T)E_{q(y^*|x^*)}(y^*) \tag{6}$$

We stress that we can also obtain these y_t^* with the dropout NN, and that the main point of the paper is to show the identification of the τ parameter which gives access to the uncertainty of the prediction. In our experiments we will simulate y_t^*'s with the two methods and compare results (for regression tasks only).

5 Implementation and results

5.1 Classification

We implemented the method of classification uncertainty described in [Gal and Ghahramani(2016)] with one hidden layer (100 units) neural network with dropouts applied after each dense layers. We finally added a softmax dense layer to get back the number of classes to be predicted. We used keras implementation for the neural network architecture and the training. We used RMSprop optimizer

and the categorical crossentropy loss. After training, we sample T (taking T = 100) forward passes in the network to compute a distribution over the prediction by re-sampling T times the dropout inside the neural network.

Finally, we obtain for each predicted class a distribution of T samples of predicted softmax values (ranging from 0 to 1). Taking the mean of these distributions and then the argmax over the different classes leads to the prediction of the class. The idea behind uncertainty estimation is here to observe the overlap of these distributions. If there is a big overlap between the predicted class and the other one, one can trust the prediction made by the network. Inversely, if the predicted class distribution overlaps strongly with the distribution of another class, we would like to inspect in more detail this sample. We first applied this model to the Mnist dataset as in the paper (see `one_hnn.ipynb`). After that we try the model on the NotMNIST dataset similar to MNIST but slightly more complex on 10 characters (from A to J) that we split in 80000 train samples and 20000 test samples. We try to show the uncertainty changes by selecting a character and then making different predictions for rotated versions of the characters. As we see in figure 1 the network predicts well (understand with a satisfying confidence) the original, character, but once rotated the true class distribution gets more and more mixed up with the other classes distributions.

As some development of this work, it could be interesting to implement some measure of the overlap between classes distributions. Using some tool such as the discrete Bhattacharyya distance, we could hope for a quantitative measure of the uncertainty based on the overlap of the distributions. This distance which writes $D(p, q) = \sum_{i=1}^{n} \sqrt{p_i q_i}$ is a rudimentary measure of the overlap between distributions. We could tune a threshold for this distribution to detect automatically uncertain samples.

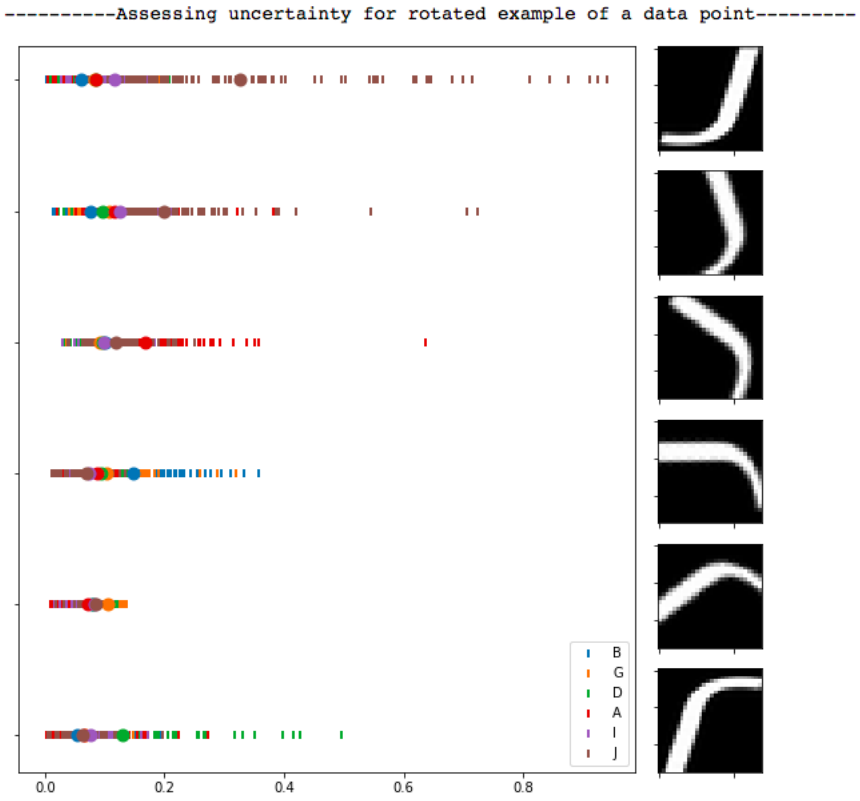

Figure 1: Rotation of one character

5.2 Regression

We try to mimic the regression estimates with bayesian dropouts. This model is the same as the classification one, without the softmax. Predictions and standard errors are obtained with equations 5 and 6 by computing T forward passes on the data. We want to compare it to the gaussian process with variational inference approximation. Thus we implemented (in the file `utils.py`) the lower bound optimization over W, the parameters of the Gaussian process.

We chose to apply this to a dataset of PM2.5 concentrations in Chinese cities with approximately 50000 data points and 12 features (year, hour of the day, precipitation...). The two models are implemented in `regression_NN.ipynb` for the neural network with dropouts and in `regression_gaussian.ipynb` for the Gaussian process. The results are comparable, which is reassuring :

the two methods that we have implemented produce results compatible with the theory developed in the article.

References

[Gal and Ghahramani(2016)] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. apr 2016. URL `https://arxiv.org/pdf/1506.02142.pdf`.