Gaussian Process in Neural Network

Yan Huang

June 2, 2022

1 Neural Network and Gaussian Process

1.1 What is neural network

Neural network is a model designed to simulate the biological neural network, which is consisting of many artificial neurons. It was proposed by psychologist W.S.McCulloch and Mathematician W.Pitts in 1943.

There are various neural networks, like feedforward neural network (FNN), convolution neural network (CNN) and recurrent neural network (RNN), etc. FNN is the earliest neural network architecture, the classical application is classification. CNN is widely used in image processing; RNN is popular in natural language processing, speech recognition and time series analysis. In this review, we will only focus on FNN, particularly, the correspondence between FNN and Gaussian process (GP). The major reference is [1].

1.2 Formal definition of feedforward neural network

We consider an L hidden-layer FNN with width N_l and an activation function g for the l-th hidden layer, $1 \leq l \leq L$. At the j-th neuron in the l-th hidden layer of the network, the pre-activation and post-activation are denoted by $z_j^{[l]}$ and $x_j^{[l]}$, $1 \leq i \leq N_l$. Let $\boldsymbol{x} = \boldsymbol{x}^{[0]} \in \mathbb{R}^{d_{in}}$ denote the inputs of the network and $\boldsymbol{y} = \boldsymbol{z}^{[L+1]} \in \mathbb{R}^{d_{out}}$ denote the outputs. We also denote that $N_0 = d_{in}$ and $N_{L+1} = d_{out}$. The weight and bias between the $l-1$-th and l-th hidden layers are denoted by $W_{ij}^{[l]}$ and $b_i^{[l]}$, $1 \leq l \leq L+1$, $1 \leq i \leq N_l$, $1 \leq j \leq N_{l-1}$, where N_l denotes the width of l-th hidden layer. Then for $1 \leq l \leq L$, $1 \leq i \leq N_l$,

$$z_i^{[l]}(\boldsymbol{x}) = \sum_{j=1}^{N_{l-1}} W_{ij}^{[l]} x_j^{[l-1]}(\boldsymbol{x}) + b_i^{[l]}$$

$$x_i^{[l]}(\boldsymbol{x}) = g(z_i^{[l]}(\boldsymbol{x}))$$

and for $1 \leq i \leq d_{out}$,

$$FNN_i(\boldsymbol{x}) = z_i^{[L+1]}(\boldsymbol{x}) = \sum_{j=1}^{N_L} W_{ij}^{[L+1]} x_j^{[L]}(\boldsymbol{x}) + b_i^{[L+1]}$$

Normally, we use mean square error (MSE) as the loss function, and train the FNN by backpropagation with stochastic gradient descent (SGD).

Generally, FNN with only 2 hidden layers is a universal approximator. More precisely, if activation function $g \in L^\infty_{loc}(\mathbb{R})$ such that the closure of the points of discontinuity of g is a Lebesgue null set, let K be a compact set, then FNN with 2 hidden layers are dense in $C(K)$ [2].

This theorem gives us a guarantee that FNN is capable of approximating every continuous function in theory, however, it doesn't tell us how to construct an FNN to approximate it and the convergence rate is what. Besides, neural network lacks interpretation due to the highly complicated nonlinear architecture,

and we often can't get the optimal solution on account of the non-convexity of the corresponding optimization problem, thus we call neural network is a black box. As a consequence, the researches on theories of neural network are flourishing. In the following contents, we will introduce a new perspective on understanding deep and wide neural network.

1.3 Infinitely wide neural network and Gaussian process

In this section, we claim that the infinitely wide L hidden-layer FNN can be viewed as a Gaussian process, hereafter referred to as the neural network Gaussian process (NNGP). The detail of proof is from [3].

Assumptions:

For the first hidden layer ($l = 1$), the weight and bias are i.i.d;

For the l-th hidden layer, $2 \leq l \leq L+1$, $W_{ij}^{[l]}$'s are i.i.d with mean $\frac{\mu_w^{[l]}}{N_{l-1}}$ and variance $\frac{\sigma_w^{2[l]}}{N_{l-1}}$, $1 \leq i \leq N_l$, $1 \leq j \leq N_{l-1}$;

For the l-th hidden layer, $2 \leq l \leq L+1$, $b_i^{[l]}$'s are i.i.d with mean $\mu_b^{[l]}$ and variance $\sigma_b^{2[l]}$, $1 \leq i \leq N_l$;

Let $N_l \to \infty$, $1 \leq l \leq L$.

Theorem 1 *$\{z_i^{[l]}(\boldsymbol{x}) \mid i \in \mathbb{N}^+\}$ are i.i.d for $1 \leq l \leq L+1$.*

proof 1 *We show it by induction. When $l = 1$, $z_i^{[1]}(\boldsymbol{x}) = w_i^{[1]\mathrm{T}}(\boldsymbol{x}) + b_i^{[1]}$, obviously the lemma is true. Assume it holds true for l, $1 \leq l \leq L$. Consider two $\boldsymbol{x}, \boldsymbol{x'}$, hence we have $\mathbb{E}[z_i^{[l+1]}(\boldsymbol{x})] = \mu_b^{[l]} + \mu_w^{[l+1]}\mathbb{E}_{z_\cdot^{[l]}}[g(z_\cdot^{[l]}(\boldsymbol{x}))]$ and*

$$Cov[z_i^{[l+1]}(\boldsymbol{x}), z_j^{[l+1]}(\boldsymbol{x'})] = \frac{(\mu_w^{[l+1]})^2}{N_l} Cov[x_\cdot^{[l]}(\boldsymbol{x}), x_\cdot^{[l]}(\boldsymbol{x'})] + \sigma_w^{2[l+1]}\delta_{ij}\mathbb{E}[x_\cdot^{[l]}(\boldsymbol{x}) \cdot x_\cdot^{[l]}(\boldsymbol{x'})]$$
$$\to \sigma_w^{2[l+1]}\delta_{ij}\mathbb{E}_{z_\cdot^{[l]}}[g(z_\cdot^{[l]}(\boldsymbol{x})) \cdot g(z_\cdot^{[l]}(\boldsymbol{x'}))], N_l \to \infty$$

Therefore, when $i \neq j$, we have $Cov[z_i^{[l+1]}(\boldsymbol{x}), z_j^{[l+1]}(\boldsymbol{x'})] = 0$, i.e. $\{z_i^{[l+1]}(\boldsymbol{x}) \mid i \in \mathbb{N}^+\}$ are uncorrelated. In fact, they are independent since they are independently defined as the combination of $\{x_i^{[l]}(\boldsymbol{x}) \mid i \in \mathbb{N}^+\}$.

Now we prove that the infinitely wide FNN can be viewed as a GP.

Theorem 2 *For $2 \leq l \leq L+1$, $z_\cdot^{[l]} \sim \mathcal{GP}(h^{[l]}, k^{[l]})$, where the mean and covariance functions (also called GP kernel) are*

$$h^{[l]}(\boldsymbol{x}) = \mu_w^{[l]}\mathbb{E}_{z_\cdot^{[l-1]}}[g(z_\cdot^{[l-1]}(\boldsymbol{x}))]$$
$$k^{[l]}(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_b^{2[l]} + \sigma_w^{2[l]}\mathbb{E}_{z_\cdot^{[l-1]}}[g(z_\cdot^{[l-1]}(\boldsymbol{x})) \cdot g(z_\cdot^{[l-1]}(\boldsymbol{x'}))]$$

proof 2 *Since $z_\cdot^{[l]}$ is a sum of N_{l-1} i.i.d random variables, by the central limit theorem, it will be Gaussian distributed. Similarly, by the multidimensional central limit theorem, any finite collection of $z_\cdot^{[l]}$ follows a joint multivariate Gaussian distribution, which is exactly the definition of a Gaussian process.*

Therefore, the outputs of infinitely wide FNN can be written as

$$y_\cdot \sim \mathcal{GP}(h^{[L+1]}, k^{[L+1]}) = \mathcal{GP}(h_{FNN}, k_{FNN})$$

Note that the form of the covariance function used is determined by the choice of the neural network model class, which depends on depth, activation function, and variances of weight and bias. We can also rewrite the expression of $k^{[l]}(\boldsymbol{x}, \boldsymbol{x'})$ to emphasize the recursive relationship between $k^{[l]}$ and $k^{[l-1]}$, which is

$$k^{[l]}(\boldsymbol{x}, \boldsymbol{x'}) = \sigma_b^{2[l]} + \sigma_w^{2[l]} F_g(k^{[l-1]}(\boldsymbol{x}, \boldsymbol{x'}), k^{[l-1]}(\boldsymbol{x}, \boldsymbol{x}), k^{[l-1]}(\boldsymbol{x'}, \boldsymbol{x'}))$$

where F_g is a function depends only on activation function g. This gives an iterative series of computations which can be performed to obtain $k^{[l]}$ for the Gaussian process describing the network's final output.

For certain activation functions, $k^{[l]}(\boldsymbol{x}, \boldsymbol{x'})$ can be computed analytically. In the case of ReLU function, it is the arccosine kernel [4].

1.4 Bayesian inference of infinitely wide neural network

As discussed above, each output of FNN follows a prior GP, now we use Bayesian non-parametric method to obtain a posterior distribution of each output of FNN. Giving a training dataset $\mathcal{D} = \{(\boldsymbol{x^1}, t^1), ..., (\boldsymbol{x^n}, t^n)\}$ consisting of input-observation pairs and another different input $\boldsymbol{x}$. Let $\boldsymbol{X} = (\boldsymbol{x^1}, \boldsymbol{x^2}..., \boldsymbol{x^n})$, $\boldsymbol{t} = (t^1, t^2, ..., t^n)$ be the observations of $\boldsymbol{X}$ and $\boldsymbol{y} = (y^1, y^2, ..., y^n)$ be the outputs of $\boldsymbol{X}$, we want to make a Bayesian prediction at $\boldsymbol{x}$.

$$P(y \mid D, \boldsymbol{x}) = \int P(y \mid \boldsymbol{y}, D, \boldsymbol{x}) P(\boldsymbol{y} \mid D) d\boldsymbol{y} = \frac{1}{P(\boldsymbol{t})} \int P(y, \boldsymbol{y} \mid \boldsymbol{X}, \boldsymbol{x}) P(\boldsymbol{t} \mid \boldsymbol{y}) d\boldsymbol{y}$$

y is the corresponding output of $\boldsymbol{x}$, $P(\boldsymbol{t} \mid \boldsymbol{y})$ corresponds to observation noise. Usually, we assume the noise is Gaussian with variance σ_ϵ^2 and centered at $\boldsymbol{y}$. Therefore, we can write FNN as the form

$$y_\cdot \sim \mathcal{GP}(h_{FNN}, k_{FNN}) + \mathcal{N}(0, \sigma_\epsilon^2)$$

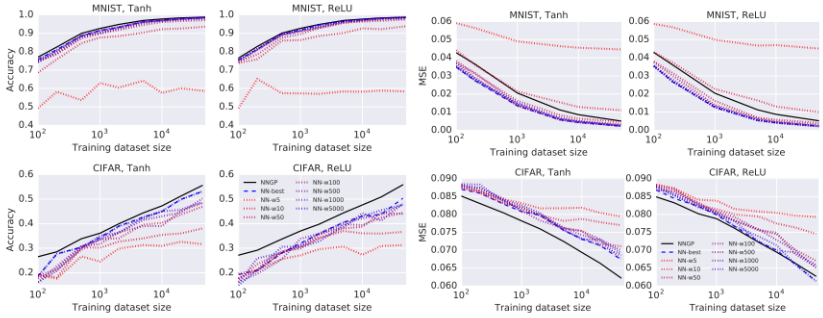
Theorem 3 *$y_\cdot \mid D, \boldsymbol{x} \sim \mathcal{GP}(h_{FNN}, k_{FNN})$ with*

$$h_{FNN}(\boldsymbol{x}) = h_{FNN}(\boldsymbol{x}) + k_{FNN}(\boldsymbol{X}, \boldsymbol{x})^{\mathrm{T}} [K + \sigma_\epsilon^2 I_n]^{-1} \boldsymbol{t}$$
$$k_{FNN}(\boldsymbol{x}, \boldsymbol{x'}) = k_{FNN}(\boldsymbol{x}, \boldsymbol{x'}) - k_{FNN}(\boldsymbol{X}, \boldsymbol{x})^{\mathrm{T}} [K + \sigma_\epsilon^2 I_n]^{-1} k_{FNN}(\boldsymbol{X}, \boldsymbol{x'})$$

where $k_{FNN}(\boldsymbol{X}, \boldsymbol{x}) := \{k_{FNN}(\boldsymbol{x^1}, \boldsymbol{x}), ..., k_{FNN}(\boldsymbol{x^n}, \boldsymbol{x})\}$ and $K := [k_{FNN}(\boldsymbol{x^i}, \boldsymbol{x^j})]_{i,j=1}^n \in \mathbb{R}^{n \times n}$.

proof 3 *See the theorem 3.1 of* [5].

Usually, we use maximal likelihood estimation to train the hyperparameters $\Theta = \{\mu_w^{[l]}, \sigma_w^{2[l]}, \mu_b^{[l]}, \sigma_b^{2[l]} \mid l = 1, ..., L+1\}$ and σ_ϵ^2.



(a) Accuracy (b) Mean squared error

Figure 1: The NNGP often outperforms finite width networks, and neural network performance more closely resembles NNGP performance with increasing width. Test accuracy and mean squared error on MNIST and CIFAR-10 dataset are shown for the best performing NNGP and best performing SGD trained neural networks for given width. 'NN-best' denotes the best performing (on the validation set) neural network across all widths and trials. Often this is the neural network with the largest width.

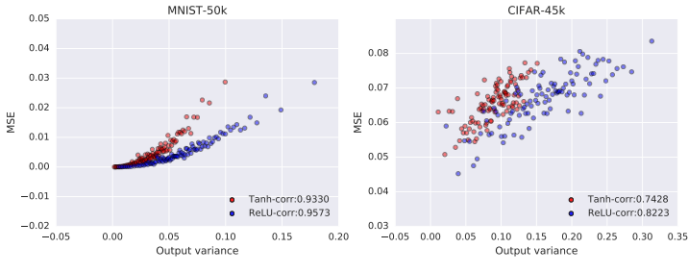


Figure 3: The Bayesian nature of NNGP allows it to assign a prediction uncertainty to each test point. This prediction uncertainty is highly correlated with the empirical error on test points. The $x-$axis shows the predicted MSE for test points, while the $y-$axis shows the realized MSE. To allow comparison of *mean* squared error, each plotted point is an average over 100 test points, binned by predicted MSE. The hyperparameters for the NNGP are depth$= 3$, $\sigma_w^2 = 2.0$, and $\sigma_b^2 = 0.2$. See Appendix Figure 8 for dependence on training set size.

2 Experiment

[1] compared NNGPs with Adam (Adaptive Momentum Estimation) trained neural networks on the permutation invariant MNIST and CIFAR-10 datasets. The baseline neural network is a FNN with identical width at each hidden layer. The loss function is mean squared error, chosen so as to allow direct comparison to GP predictions. The activation functions were chosen to be either ReLU or tanh. Class labels were encoded as a one-hot, zero-mean, regression target.

Performance: NNGPs are better than the finite width neural networks. Additionally, as the layer width increasing, the performance of best neural network approaches NNGPs. [Figure 1]

Uncertainty: All predictions have uncertainty estimates due to the Bayesian inference. In this experiment, they observed that the NNGP uncertainty estimate was highly correlated with prediction error. [Figure 3]

3 Conclusions and Connections

3.1 Conclusions

[1] constructed a relationship between infinitely wide neural networks and Gaussian process. Use of a GP prior on functions enables exact Bayesian inference for regression from matrix computations, and hence we are able to obtain predictions and uncertainty estimates from deep neural networks without stochastic gradient-based training. The performance of NNGPs is competitive with the best neural networks (within specified class of fully-connected models) trained on the same regression task with similar hyperparameters. In their experiments, they observed the performance of the best neural network approached to the NNGPs with increasing width. Additionally, the NNGPs provide explicit estimates of uncertainty. This may be useful in predicting model failure in critical applications of deep learning, or for active learning tasks where it can be used to identify the best datapoints to hand label.

However, there is a huge problem of NNGPs, the computational cost of GP kernel is expensive, we need to develop an efficient algorithm to compute it.

3.2 Connections between machine learning and stochastic theory

Besides the NNGPs we talk about, GPs are also used in regression and classification tasks at the earliest [6]. Hidden Markov Models are widely applied in reinforcement learning and natural language process. Other than building models by stochastic models, researchers also proposed using neural network to generate a stochastic process, which was called neural process [7].

SDE also appears in deep learning. Recently published a paper about regarding SDE as the infinite-dimensional GAN (generative adversarial networks), so we can fit the parameters of SDE by SDE-GAN which is based on Wasserstein distance or latent SDE which is based on KL divergence [8].

References

[1] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein, "Deep neural networks as gaussian processes," 2017. [Online]. Available: https://arxiv.org/abs/1711.00165

[2] J. Berner, P. Grohs, G. Kutyniok, and P. Petersen, "The modern mathematics of deep learning," 2021. [Online]. Available: https://arxiv.org/abs/2105.04026

[3] M. Guo, "A brief note on understanding neural networks as gaussian processes," 2021. [Online]. Available: https://arxiv.org/abs/2107.11892

[4] Y. Cho and L. Saul, "Kernel methods for deep learning," in *Advances in Neural Information Processing Systems*, Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, Eds., vol. 22. Curran Associates, Inc., 2009. [Online]. Available: https://proceedings.neurips.cc/paper/2009/file/5751ec3e9a4feab575962e78e006250d-Paper.pdf

[5] M. Kanagawa, P. Hennig, D. Sejdinovic, and B. K. Sriperumbudur, "Gaussian processes and kernel methods: A review on connections and equivalences," 2018. [Online]. Available: https://arxiv.org/abs/1807.02582

[6] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 11 2005. [Online]. Available: https://doi.org/10.7551/mitpress/3206.001.0001

[7] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. W. Teh, "Neural processes," 2018. [Online]. Available: https://arxiv.org/abs/1807.01622

[8] P. Kidger, "On neural differential equations," 2022. [Online]. Available: https://arxiv.org/abs/2202.02435