Neural Networks As Gaussian Processes

Gaussian Process

- Most task can be represented in a form of a search of posterior distibution
- Multivariate normal distribution - usual assumption

$$P(\boldsymbol{Y}|\boldsymbol{X}) \sim \mathcal{GP}(\mu, \Sigma)$$

- Data can be easily centered, so the goal is to find Σ

NN equivalence

If we describe the behaviour of a NN, then for each layer:

$$x_i^l(\boldsymbol{x}) = \phi\left(\sum_{j=0}^{N} \boldsymbol{W}_{ji}\boldsymbol{x}_j^{l-1}\right)$$

$$K^l(x, x') = \sigma_b^2 + \sigma_w^2 \mathbb{E}[\phi(x^{l-1}), \phi(x'^{l-1})]$$

$$K^0(x, x') = \sigma_b^2 + \sigma_w^2 \left(\frac{x \cdot x'}{d_{in}}\right)$$

Recurent kernel calculation

$$K^l(x,x') = \sigma_b^2 + \sigma_w^2\, F_\phi\Big(K^{l-1}(x,x'),\, K^{l-1}(x,x),\, K^{l-1}(x',x')\Big)$$

ReLU

Analytical kernel computation:

$$K^l(x,x') = \sigma_b^2 + \frac{\sigma_w^2}{2\pi}\sqrt{K^{l-1}(x,x)K^{l-1}(x',x')}\left(\sin\theta_{x,x'}^{l-1} + (\pi - \theta_{x,x'}^{l-1})\cos\theta_{x,x'}^{l-1}\right)$$

$$\theta_{x,x'}^{l-1} = cos^{-1}\left(\frac{K^l(x,x')}{\sqrt{K^l(x,x)K^l(x',x')}}\right)$$

ReLU

$$K^l(x,x') = \sigma_b^2 + \sigma_w^2\, F_\phi\Big(K^{l-1}(x,x'),\, K^{l-1}(x,x),\, K^{l-1}(x',x')\Big)$$

$$F_{ij} = \frac{\sum_{ab} \phi(u_a)\phi(u_b) \exp\left(-\frac{1}{2}\begin{bmatrix} u_a \\ u_b \end{bmatrix}^T \begin{bmatrix} s_i & s_i c_j \\ s_i c_j & s_i \end{bmatrix}^{-1} \begin{bmatrix} u_a \\ u_b \end{bmatrix}\right)}{\sum_{ab} \exp\left(-\frac{1}{2}\begin{bmatrix} u_a \\ u_b \end{bmatrix}^T \begin{bmatrix} s_i & s_i c_j \\ s_i c_j & s_i \end{bmatrix}^{-1} \begin{bmatrix} u_a \\ u_b \end{bmatrix}\right)}$$

Objective

- Implement the kernel that corresponds to an infinite-width NN with arbitrary depth and Gaussian distributed weights
- Compare the classification performance of NNGPs and feed-forward NNs
- Examine the correlation of predictive error and output variance
- Study the time performance between analytical kernel and approximated kernel

Experiment

- Training GPs
    - Framework: GPyTorch
    - The model outputs predictive score for each 10 classes
    - Classes with largest score are chosen
- Training NNs
    - Follow the original paper’s configuration
    - On PyTorch

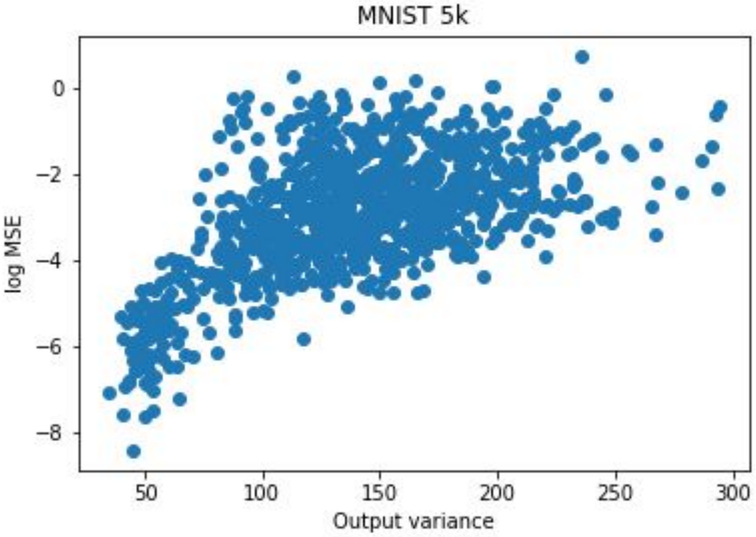
Accuracy

Table 1. The NNGP often outperforms finite width networks. Test accuracy on MNIST dataset. Best models are specified by (depth-width-σ_w^2-σ_b^2 for NNs and (depth-σ_w^2-σ_b^2 for GPs.

Num training	Model (ReLU)	Test accuracy
MNIST:100	NN-2-5000-0.10-0.11	69.33
	GP-12-1.52-1.2	69.45
MNIST:1k	NN-2-5000-3.19-0	89.26
	GP-5-1.0-0.28	92.14
MNIST:5k	NN-3-2000-2.92-0.22	94.71
	GP-2-0.61-0.07	96.03
MNIST:10k	NN-2-2000-0.42-0.16	96.51
	GP-3-0.8-0.07	97.37

Variance as uncertainty

- Output variance is correlated to empirical error



Time performance

- 600-800s to train a GP model on MNIST of 10k size
- 400s to evaluate on 10k testset
- Training and inference time grows with model's depth

Efficient implementation of GP kernel

- Results on MNIST:1k

Model's depth	Analytical kernel (s)	Approximated kernel (s)
1 (no hidden layer)	27	27
5	106	80
10	208	140

Next Steps

- Neural Tangent Kernel
- Deep Gaussian Process
- Bayesian deep learning

Questions?