[Paper review 11]

Deep Neural Networks as Gaussian Processes

(Jaehoon Lee, et.al, 2018)

[Contents]

0. Abstract

when $H \rightarrow \infty$: single layer NN with a prior = GP (Neal, 1994)

contribution: "show infinitely wide deep networks = GP "

- 1) trained NN accuracy approaches that of the corresponding GP
- 2) GP uncertainty is strongly correlated with trained network prediction error

1. Introduction

DNN & GP

- DNN = flexible parametric models nowadays
- GP = traditional non-parametric tool
- (limit of ∞ width) the CLT implies that the function computed by NN = function drawn from GP (Neal, 1994)
- this substitution enables exact Bayesian inference for regression using NN (Williams, 1997)

1.1 Related Work

GP context

- infinite network = GP (Neal, 1994)
- GP prior for exact Bayesian Inference in Regression (Williams, 1997)
- building deep GP & observe degenerate form of kernels (Duvenaud et al, 2014)
- constructing kernels equivalent to infinitely wide DNN (Hazan & Jaakola, 2015)

outside GP context

- derives compositional kernels for polynomial rectified nonlinearities (Cho & Saul, 2009)
- extends the construnction of compositional kernels to NN (Daniely et al, 2016)

1.2 Summary of Contributions

begin by specifying the form of GP (which corresponds to deep, infinitely wide NN) = NNGP in terms of recursive, deterministic computation of the kernel function

Then, develop computationally efficient method to compute covariance function

2. Deep, Infinitely Wide NN are drawn from GPs

2.1 Notation

L : # of hidden layer

N_L : width of layer L

ϕ : pointwise non-linearity

$x \in \mathbb{R}^{d_{\text{in}}}$: input

x_i^l : ith component of the activations in lth layer, post-nonlinearity (= post activation)

z_i^l : ith component of the activations in lth layer, post-affine transformation (= pre activation)

$z^L \in \mathbb{R}^{d_{\text{out}}}$: output (= post-affine transformation)

W_{ij}^l, b_i^l : weight and bias (zero mean, and covariance with σ_w^2 / N_l and σ_b^2 each)

$\mathcal{GP}(\mu, K)$: GP with mean, covariance $\mu(\cdot), K(\cdot, \cdot)$, respectively.

2.2 Review of GP and 1-layer NN

The i th component of the network output, z_i^1, is computed as,

$$z_i^1(x) = b_i^1 + \sum_{j=1}^{N_1} W_{ij}^1 x_j^1(x), \quad x_j^1(x) = \phi\left(b_j^0 + \sum_{k=1}^{d_{in}} W_{jk}^0 x_k\right)$$

- x_k : pre-activation
- $x_i^l(x)$: post-activation
- $z_i^l(x)$: pre-activation

by CLT, as $N_1 \rightarrow \infty$

- $z_i^1(x)$ is Gaussian distributied
- any finite collection of $\left\{z_i^1\left(x^{\alpha=1}\right), \ldots, z_i^1\left(x^{\alpha=k}\right)\right\}$ will have a joint MVN (= GP)

$\therefore z_i^1 \sim \mathcal{GP}\left(\mu^1, K^1\right)$

- mean : $\mu^1(x) = \mathbb{E}\left[z_i^1(x)\right] = 0$
- covariance : $K^1\left(x, x'\right) \equiv \mathbb{E}\left[z_i^1(x) z_i^1\left(x'\right)\right] = \sigma_b^2 + \sigma_w^2 \mathbb{E}\left[x_i^1(x) x_i^1\left(x'\right)\right] \equiv \sigma_b^2 + \sigma_w^2 C\left(x, x'\right)$

2.3 GP and DNN

previous sections(works) can be extended to DEEPER layers

($N_1 \rightarrow \infty$, $N_2 \rightarrow \infty$, $N_3 \rightarrow \infty$ .....)

Suppose that z_j^{l-1} is GP. After $l-1$ steps..

$z_i^l(x) = b_i^l + \sum_{j=1}^{N_l} W_{ij}^l x_j^l(x), \quad x_j^l(x) = \phi\left(z_j^{l-1}(x)\right)$

- $z_i^l(x)$ is a sum of i.i.d random terms
- Thus, CLT works! $\left\{z_i^1\left(x^{\alpha=1}\right), \ldots, z_i^1\left(x^{\alpha=k}\right)\right\}$ follows MVN
- Therefore, $z_i^l \sim \mathcal{GP}\left(0, K^l\right)$

$z_i^l \sim \mathcal{GP}\left(0, K^l\right)$

- mean : 0
- covariance :

  $$\begin{aligned} K^l\left(x, x^{\prime}\right) & \equiv \mathbb{E}\left[z_i^l(x) z_i^l\left(x^{\prime}\right)\right] \\ & =\sigma_b^2+\sigma_w^2 \mathbb{E}_{z_i^{l-1} \sim \mathcal{GP}\left(0, K^{l-1}\right)}\left[\phi\left(z_i^{l-1}(x)\right) \phi\left(z_i^{l-1}\left(x^{\prime}\right)\right)\right] \\ & =\sigma_b^2+\sigma_w^2 F_\phi\left(K^{l-1}\left(x, x^{\prime}\right), K^{l-1}(x, x), K^{l-1}\left(x^{\prime}, x^{\prime}\right)\right) \end{aligned}$$

  (RECURSIVE relationship between K^l and K^{l-1} via deterministic function F, whose form depends only on the non-linearity ϕ $\rightarrow$ iterative series!)

For the base case K^0,

- weight: $W_{ij}^0 \sim \mathcal{N}\left(0, \sigma_w^2 / d_{\text{in}}\right)$ & bias : $b_j^0 \sim \mathcal{N}\left(0, \sigma_b^2\right)$
- $K^0\left(x, x^{\prime}\right)=\mathbb{E}\left[z_j^0(x) z_j^0\left(x^{\prime}\right)\right]=\sigma_b^2+\sigma_w^2\left(\frac{x \cdot x^{\prime}}{d_{\text{in}}}\right)$

2.4 Bayesian Training for NN, using GP priors

How GP prior over functions can be used to do Bayesian Inference (Rasmussen & Williams, 2006)

- data : $\mathcal{D}=\left\{\left(x^1, t^1\right), \ldots,\left(x^n, t^n\right)\right\}$
- distribution over functions : $z(x)$

  ($z \equiv\left(z^1, \ldots, z^n\right)$ on the training inputs $x \equiv\left(x^1, \ldots, x^n\right)$)
- targets on training set : $\mathbf{t}$
- goal : make prediction at test point x^*, using a distribution over functions $z(x)$

  $$\begin{aligned} P\left(z^* \mid \mathcal{D}, x^*\right) & =\int P\left(z^* \mid z, x, x^*\right) P(z \mid \mathcal{D}) d z \\ & =\frac{1}{P(\mathbf{t})} \int P\left(z^*, z \mid x^*, x\right) P(\mathbf{t} \mid z) d z \end{aligned}$$

$z^*, z \mid x^*, x \sim \mathcal{N}(0, \mathbf{K})$, where $\mathbf{K} = \begin{bmatrix} K_{\mathcal{D},\mathcal{D}} & K_{x^*,\mathcal{D}}^T \\ K_{x^*,\mathcal{D}} & K_{x^*,x^*} \end{bmatrix}$

- $K_{\mathcal{D},\mathcal{D}}$ is an $n \times n$ matrix whose (i, j) th element is $K\left(x^i, x^j\right)$ with $x^i, x^j \in \mathcal{D}$
- the i th element of $K_{x^*,\mathcal{D}}$ is $K\left(x^*, x^i\right), x^i \in \mathcal{D}$.

$P\left(z^* \mid \mathcal{D}, x^*\right) = z^* \mid \mathcal{D}, x^* \sim \mathcal{N}(\bar{\mu}, \bar{K})$

- mean : $\bar{\mu} = K_{x^*,\mathcal{D}}\left(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n\right)^{-1} \boldsymbol{t}$
- covariance : $\bar{K} = K_{x^*,x^*} - K_{x^*,\mathcal{D}}\left(K_{\mathcal{D},\mathcal{D}} + \sigma_\epsilon^2 \mathbb{I}_n\right)^{-1} K_{x^*,\mathcal{D}}^T$

form of the covariance function used is determined by the choice of GP prior

(NN : depth, nonlinearity, and weight and bias variances)

2.5 Efficient Implementation of the GP Kernel

constructing covariance matrix K^L

= computing Gaussian integral $\sigma_b^2 + \sigma_w^2 \mathbb{E}_{z_i^{l-1} \sim \mathcal{GP}(0, K^{l-1})}\left[\phi\left(z_i^{l-1}(x)\right) \phi\left(z_i^{l-1}\left(x'\right)\right)\right]$ for all train &test pairs

(recursively for all layers)

for some nonlinearities..

- RELU : integration can be done "analytically"
- kernel corresponding to arbitrary nonlinearities : must be done "numerically"

Simple way : compute integrals independently for each pair of data points & each layer

$\rightarrow \mathcal{O}\left(n_g^2 L\left(n_{\text{train}}^2 + n_{\text{train}} n_{\text{test}}\right)\right)$

Pre-process all the inputs to have identical norm

$\rightarrow \mathcal{O}\left(n_g^2 n_v n_c + L\left(n_{\text{train}}^2 + n_{\text{train}} n_{\text{test}}\right)\right)$

STEP

[step 1] Generate

- pre-activations $u = [-u_{\max}, \cdots, u_{\max}]$ ...... n_g elements
- variances $s = [0, \cdots, s_{\max}]$ ...... n_v elements
- correlations $c = (-1, \cdots, 1)$ ...... n_c elements

[step 2] Populate a matrix F

- involves numerically approximating Gaussian integral

  (in terms of marginal variances s and c)

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

3. For every pair of datapoints x and x' in layer l, compute $K^l(x, x')$ using Equation 5. Approximate the function $F_\phi\left(K^{l-1}(x, x'); K^{l-1}(x, x); K^{l-1}(x', x')\right)$ by bilinear interpolation into the matrix F from Step 2, where we interpolate into s using the value of $K^{l-1}(x, x)$, and interpolate into c using $\left(K^{l-1}(x, x')/K^{l-1}(x, x)\right)$. Remember that $K^{l-1}(x, x) = K^{l-1}(x', x')$, due to data preprocessing to guarantee constant norm.
4. Repeat the previous step recursively for all layers. Bilinear interpolation has constant cost, so this has cost $\mathcal{O}\left(L(n_{\text{train}}^2 + n_{\text{train}} n_{\text{test}})\right)$.

This computational recipe allows us to compute the covariance matrix for the NNGP corresponding
to any well-behaved nonlinearity ϕ