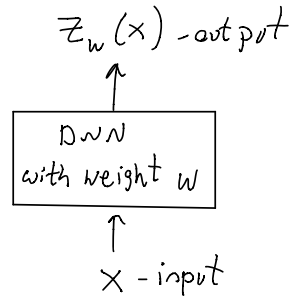
# Gaussian Processes and Deep Nueral Networks

## 1. GPs and randomly initialize DNNs:

Consider a three layer network

$$Z_j(x) = \underbrace{w^{(3)}_j \cdot \underbrace{\phi\Big(w^{(2)}_{jk} \underbrace{\phi(w^{(1)}_{ki} x_i)}_{h^{(1)}_k}\Big)}_{h^{(2)}_j}}_{\text{Einstein summation}}$$




Consider functions generated by random weights drawn from

$$P_0(w) \propto e^{-\frac{w w^T}{2\sigma_w^2}}$$

$$P[f] = \underbrace{\int dw}_{\text{many integrals}} P_0(w) \underbrace{\delta[f - z_w]}_{\text{delta function in function space}}$$

$\rightarrow \int Df\, \delta[f - f_0]\, f(x) = f_0(x)$

- What is $P[f]$? From its definition it gives high probability to functions realized by the DNN using many different likely weights.

- We next show that: I. If the first layer is fully-connected $P[f(x)] = P[f(Ox)]$ where $O$ is any orthogonal rotation in input space

  II. For a wide DNN $P[f]$ is a GP

III. For ReLU and Erf activations we can calculate $K(x,x')$ of that GP

I. $$P[f(Ox)] = \int dw\, P_0(w)\, \delta[f(Ox) - z_w(x)] = \int dw\, P_0(\underbrace{w^{(1)}}_{\text{weights of 1st layer}}, \underbrace{w^{(2)}}_{\text{weights of 2nd layer}}, \ldots)\, \delta[f(Ox) - z_w(x)]$$

$$= \int dw\, \frac{e^{-\mathrm{Tr}(w^{(1)T} w)}}{Z^{(1)}} \frac{e^{-\mathrm{Tr}(w^{(2)T} w^{(2)})}}{Z^{(2)}} \cdots \delta[f(Ox) - z_w(x)]$$

$$= \int dw\, \frac{e^{-\mathrm{Tr}(w^{(1)T} w)}}{Z^{(1)}} \frac{e^{-\mathrm{Tr}(w^{(2)T} w^{(2)})}}{Z^{(2)}} \cdots \delta[f(Ox) - z_{w^{(1)}O^T}(x)]$$

$$= \int dw\, P_0(w)\, \delta[f(Ox) - z_w(Ox)]$$

$$z_{\ldots w^{(2)}, w^{(1)} O^T}(x) = z_{\ldots w^{(2)}, w^{(1)}}(Ox)$$

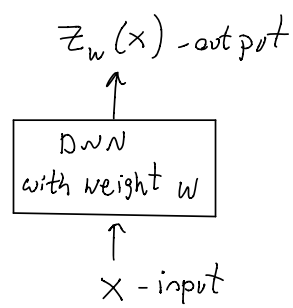
Recall $h = \phi(w^{(1)} \cdot x + b^{(1)})$

$$= \int dw\, P_0(w)\, \delta[f(x) - z_w(x)] = P[f(x)]$$

II. Recall:

$$z_j(x) = \underbrace{w^{(3)}_j}_{\text{Einstein summation}} \cdot \underbrace{\phi\Big(w^{(2)}_{jk} \underbrace{\phi(\overbrace{w^{(1)}_{ki} x_i}^{h^{(1)}_k})}_{h^{(1)}_k}\Big)}_{h^{(2)}_j}$$

$h^{(2)}_j$, $h^{(1)}_k$




- We are interested in $K(x,x') \equiv \int D[f]\, P[f]\, f(x) f(x')$

$$= \int D[f] \int dw\, P_0(w)\, \delta[f - z_w]\, f(x) f(x') = \int dw\, P_0(w)\, \underbrace{z_w(x) z_w(x')}_{\text{microscopically accessible}}$$

- <u>We unravel this bottom-up:</u>

- Consider first $\int dw^{(1)} P_0(w^{(1)})\, h^{\prime(1)}_{k,w^{(1)}}(x)\; h^{\prime(1)}_{k',w^{(1)}}(x')$

$$= \int dw^{(1)} P_0(w^{(1)})\, w^{(1)}_{ki}\, x_i\; w^{(1)}_{k'j}\, x'_j = \sigma_w^2 (x\cdot x')\, \delta_{kk'} = K^{(1)}(x,x')\,\delta_{kk'}$$

- We understand correlations of $h^{\prime(1)}(x)$, what about higher moments?

They are all determined, $h^{\prime(1)}(x)$ is exactly a GP:

$$h^{\prime(1)}_k(x) = \sum_{i=1}^{width} \underbrace{w^{(1)}_{ki}}_{\text{Gaussian variable}} x_i \iff \int dw^{(1)}_{k1} \dots dw_{k,width}\; \underbrace{e^{-\sum \frac{w_{ki}^2}{2\sigma_w^2}}}_{\text{A linear constraint on a multi-variate Gaussian is still a gaussian}} \delta\left[f - \sum w_{ki} x_i\right] = (\dots)$$

- Consider the second layer:

$$h^{\prime(2)}_j(x) = \sum_{k=1}^{width} \phi\left(h^{\prime(1)}_k(x)\right)\cdot w^{(2)}_{kj}$$

it looks like a sum of many uncorrelated random variable $\to$ Gaussian for width $\to \infty$

- Since its a gaussian, to characterize it all we need is

$$h^{\prime(2)}_j(x)\, h^{\prime(2)}_{j'}(x') = \int dw^{(2)} \int dh^{\prime(1)}_k\, P_0(w^{(2)}_{kj})\, P(h^{\prime(1)}_k(x), h^{\prime(1)}_k(x'))$$
$$(*) \sum_{kk'} \phi(h^{\prime(1)}_k(x))\, \phi(h^{\prime(1)}_{k'}(x'))\, w^{(2)}_{kj}\, w^{(2)}_{k'j'}$$

(*) We see here that $\sigma_w^2$ should scale as $\frac{1}{width}$ to keep $h^{\prime}_i$ finite

$$= \delta_{jj'}\, \overbrace{\sigma^2_{w^{(2)}} \cdot width}^{(*)} \int dh^{\prime(1)}_k\, P(h^{\prime(1)}_1(x), h^{\prime(1)}_1(x'))\, \phi(h^{\prime(1)}_1(x))\, \phi(h^{\prime(1)}_1(x'))$$

$$= \delta_{jj'}\, \sigma^2_{w^{(2)}} \cdot width \int dh^{\prime}_1(x)\, dh^{\prime}_1(x')\, e^{-(h_1(x)\, h_1(x'))}\; M \begin{pmatrix} h_1(x) \\ h_1(x') \end{pmatrix}$$

$$\underbrace{\sqrt{\det(M)}\;\phi(x')\phi(x')}_{I_\phi}$$

where $M^{-1} = \begin{pmatrix} K^{(1)}(x,x) & K^{(1)}(x,x') \\ K^{(1)}(x',x) & K^{(1)}(x',x') \end{pmatrix}$

- We are left with the two dimensional integral $I_\phi$ as a function of $K^{(1)}(x,x)$, $K^{(1)}(x,x')$, and $K^{(1)}(x',x')$

III. For $\phi(\ldots) = \mathrm{erf}(\ldots)$ and $\phi(\ldots) = \mathrm{ReLU}(\ldots)$ it can be solved (also with biases) [Cho & Saul 2009]

- $$I_{\mathrm{ReLU}} = \sqrt{K(x,x)}\sqrt{K(x',x')}\left[\sin(\theta) + (\pi-\theta)\cos(\theta)\right]$$

$$\theta = \mathrm{acos}\left(\frac{K(x,x')}{\sqrt{K(x,x)K(x',x')}}\right)$$

- This is already the answer for a depth two network namely $\int dw^{(1)} dw^{(2)}\, \underbrace{w^{(2)}_{jk}\cdot\phi(w^{(1)}_{ki}\cdot x_i)}_{z_{jv}(x) \text{ for a depth 2 network with } j \text{ different outputs}}\, \underbrace{w^{(2)}_{jk}\cdot\phi(w^{(1)}_{ki}\cdot x'_i)}_{z_{jw}(x')} =$

$$\delta_{jj'} K^{(2),\mathrm{ReLU}}(x,x') = \sigma_w^2 \cdot \text{width}\sqrt{\sigma_w^2 K^{(1)}(x,x)}\sqrt{\sigma_w^2 K^{(1)}(x',x')}\left[\sin(\theta) + (\pi-\theta)\cos(\theta)\right]$$

$$= \sigma_w^4 \cdot \text{width}\; \|x\|\|x'\|\left[\sin(\theta) + (\pi-\theta)\cos(\theta)\right]$$

$\theta$ = Angle between $x$ and $x'$

2. Wide DNNs trained with MSE loss, vanishing learning rate, weight decay, and noisy gradients perform Bayesian Inference on a GP

Consider the following Stochastic Gradient Descent (SGD) updates:

$$w_\alpha^{new} = w_\alpha^{old} - (lr)\partial_{w_\alpha}\left(L(z_{w^{old}}, \text{Dataset}) + \frac{\sigma^2}{\sigma_w^2}\sum w^2\right) + \xi_\alpha \qquad \langle \xi_\alpha^2 \rangle = 2\sigma^2\, lr$$

Denoting lr (learning rate) as $dt$ we get

$$\frac{\Delta w_\alpha}{dt} = -\partial_{w_\alpha}\left(L(z_{w^{old}}, \text{Dataset}) + \frac{\sigma^2}{\sigma_w^2}\sum_\alpha w_\alpha^2\right) + \frac{\xi_\alpha}{dt}$$

$$\Downarrow\ dt \to 0$$

$$\dot{w}_\alpha = -\partial_{w_\alpha}\left(L(z_w, \text{Dataset}) + \frac{\sigma^2}{2\sigma_w^2}\sum_\alpha w_\alpha^2\right) + \eta_\alpha \qquad \langle \eta_\alpha(time_1)\, \eta_\alpha(time_2)\rangle = \delta(time_1 - time_2)\,\sigma^2$$

units of $\frac{\sigma}{\sqrt{dt}}$

white noise

$\Downarrow$ At long times we have ergodicity

$$\cdot\ P(w) \propto e^{-\left(L(z_w, \text{Dataset}) + \frac{\sigma^2}{\sigma_w^2}\sum_\alpha w_\alpha^2\right)/2\sigma^2}$$

$$\underset{\text{MSE loss}}{=} e^{-\frac{\sum (z_w(x_n) - y_n)^2}{2\sigma^2}}\, e^{-\frac{\sum_\alpha w_\alpha^2}{2\sigma_w^2}}$$

$$= e^{-\frac{\sum (z_w(x_n) - y_n)^2}{2\sigma^2}}\, P_0(w)$$

- $P[f] = \int dw\, \delta[f - z_w]\, P(w) \propto \int dw\, e^{-\frac{\sum w_\alpha^2}{2\sigma_w^2}}\, \delta[f - z_w]\, e^{-\frac{\sum_{n=1}^{N} (z_w(x_n) - s_n)^2}{2\sigma^2}}$

$$= \int dw\, e^{-\frac{\sum w^2}{2\sigma_w^2}}\, \delta[f - z_w]\, e^{-\sum_{n=1}^{N} (f(x_n) - s_n)^2 / 2\sigma^2}$$

$$= P_0[f]\, e^{-\sum_{n=1}^{N} (f(x_n) - s_n)^2 / 2\sigma^2} \propto P_0[f]\, P[\text{Dataset} \mid f] \propto P[f \mid \text{Dataset}]$$

The probability after train on function space is exactly that of Bayesian Posterior!

(Oded Ben-David & Zohar Ringel 2019)

- In particular the DNN prediction on a new datapoint $x_*$ average over many trained weights is given by formula 1 from lesson 1

3. The NTK result: Wide DNN trained with vanishing learning rate, no weight decay, no-noise in gradients, and averaged over initializations give Bayesian Inference on a GP, different from the previous one, known as the NTK GP.

- Main Observation: For wide DNNs and full-batch GD, $\Delta W_{\text{trainy}} = W^{\text{trained}} - W^{\text{init}}$ changes very little namely $\Delta W = O\left(\frac{1}{\text{width}}\right)$ while

$$w_{init} = O\left(\frac{1}{\sqrt{width}}\right)$$

- Consequence: The DNN can be linearized around every datapoint

$$z_w(x_n) = z_{w_{init}}(x_n) + \Delta w \cdot \nabla_w z_{w_{init}}(x_n)$$

- This is already tractable analytically
  [See "Wide Neural Network of any depth behave as linear models..." 2019]

4. Numerical verifications:

- "Deep Neural Networks as Gaussian Processes" Lee et. al. 2018 → Show Fully-Connected GPs work as well as Fully-Connected DNN's on CIFAR-10 and MNIST
- S. Arora 2019 → Shows that CNN GPs reach 80% Accuracy on CIFAR10 while the corresponding CNN reaches 85%
- "Wide Neural Networks of any depth behave as linear models..." 2019 → Shows that the NTK results hold for CNN's as long as $N_{\substack{data \\ points}} << width$