Neural Tangents: Fast and Easy Infinite Neural Networks in Python

Jevgenija Rudzusika

KTH Royal Institute of Technology

March 24, 2020

Outline

- Gaussian Processes.
- “Deep Neural Networks as Gaussian Processes”, ICLR 2018, [Lee et al., 2018].
- “Deep Convolutional Networks as shallow Gaussian Processes”, ICLR 2019, [Garriga-Alonso et al., 2019].

Outline

- Gaussian Processes.
- "Deep Neural Networks as Gaussian Processes", ICLR 2018, [Lee et al., 2018].
- "Deep Convolutional Networks as shallow Gaussian Processes", ICLR 2019, [Garriga-Alonso et al., 2019].
- "Neural Tangent Kernel: Convergence and Generalization in Neural Networks", NIPS 2018, [Jacot et al., 2018].
- "Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent", NIPS 2019, [Lee et al., 2019].

Outline

- Gaussian Processes.
- "Deep Neural Networks as Gaussian Processes", ICLR 2018, [Lee et al., 2018].
- "Deep Convolutional Networks as shallow Gaussian Processes", ICLR 2019, [Garriga-Alonso et al., 2019].
- "Neural Tangent Kernel: Convergence and Generalization in Neural Networks", NIPS 2018, [Jacot et al., 2018].
- "Wide Neural Networks of Any Depth Evolve as Linear Models Under Gradient Descent", NIPS 2019, [Lee et al., 2019].
- "Neural Tangents: Fast and Easy Infinite Neural Networks in Python", ICLR 2020, [Novak et al., 2020].

Gaussian processes.

- **Definition:** A Gaussian process is a stochastic process $Y(x)$, such that for every finite collection of $\{x_i\}_{i=1}^n$ random variables $\{Y(x_i)\}_{i=1}^n$ have a multivariate normal distribution.

Gaussian processes.

- **Definition:** A Gaussian process is a stochastic process $Y(x)$, such that for every finite collection of $\{x_i\}_{i=1}^n$ random variables $\{Y(x_i)\}_{i=1}^n$ have a multivariate normal distribution.
- **Example:** Regression

$$Y(x) = \cos(x) + \varepsilon \tag{1}$$

Gaussian processes.

- **Definition:** A Gaussian process is a stochastic process $Y(x)$, such that for every finite collection of $\{x_i\}_{i=1}^n$ random variables $\{Y(x_i)\}_{i=1}^n$ have a multivariate normal distribution.
- **Example:** Regression

$$Y(x) = \cos(x) + \varepsilon \tag{1}$$

- **Inference**: Given samples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, a posterior distribution of $y = Y(x)$ at a new point x.

Gaussian processes.

- **Definition:** A Gaussian process is a stochastic process $Y(x)$, such that for every finite collection of $\{x_i\}_{i=1}^n$ random variables $\{Y(x_i)\}_{i=1}^n$ have a multivariate normal distribution.
- **Example:** Regression

$$Y(x) = \cos(x) + \varepsilon \tag{1}$$

- **Inference**: Given samples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, a posterior distribution of $y = Y(x)$ at a new point x.
- **Prior:** Kernel function $K(x, x')$ describes a co-variance between $Y(x)$ and $Y(x')$, thus $K(x, x) = Var(\varepsilon)$.

Gaussian processes.

Bayesian inference.

- $P(y|x, \mathcal{D}) \sim \mathcal{N}(\hat{\mu}, \hat{K})$

$$\begin{aligned} \hat{\mu} &= K_{x,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\varepsilon^2 I_n)^{-1}\mathbf{y} \\ \hat{K} &= K_{x,x} - K_{x,\mathcal{D}}(K_{\mathcal{D},\mathcal{D}} + \sigma_\varepsilon^2 I_n)^{-1} K_{x,\mathcal{D}}^T \end{aligned} \tag{2}$$

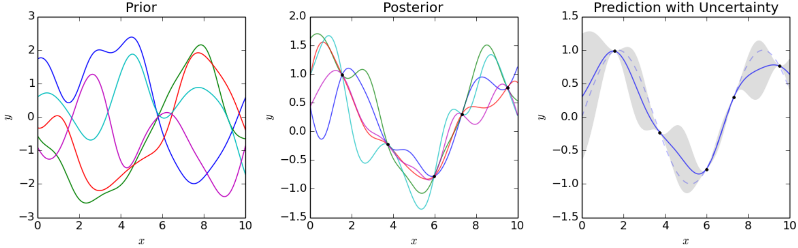

Figure: $\sigma_\varepsilon^2 = 0$, [Wikipedia contributors, 2020]

Neural Network GP

Fully-connected layers

- $z^{l-1}(x)$, $x^{l}(x)$ are pre/post-activation features for a hidden layer l

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

- $b_j^l \sim \mathcal{N}(0, \sigma_b^2)$, $W_{i,j}^l \sim \mathcal{N}(0, \frac{\sigma_w^2}{N_l})$, thus $x_j^l(x)$ and $z_j^l(x)$ - i.i.d.

Neural Network GP

Fully-connected layers

- $z^{l-1}(x)$, $x^{l}(x)$ are pre/post-activation features for a hidden layer l

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

- $b_j^l \sim \mathcal{N}(0, \sigma_b^2)$, $W_{i,j}^l \sim \mathcal{N}(0, \frac{\sigma_w^2}{N_l})$, thus $x_j^l(x)$ and $z_j^l(x)$ - i.i.d.
- By the CLT, $\{z_i^l\} \sim \mathcal{GP}(0, K^l)$ as $N_l \to \infty$.

Neural Network GP

Fully-connected layers

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

$$
K^0\left(x, x'\right) = \mathbb{E}\left[z_j^0(x) z_j^0\left(x'\right)\right] = \sigma_b^2 + \sigma_w^2\left(\frac{x \cdot x'}{d_{\text{in}}}\right)
$$

Neural Network GP

Fully-connected layers

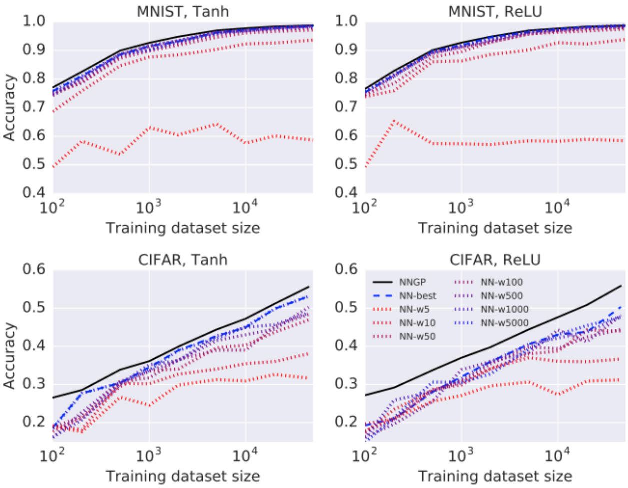

Figure: Experimental results from [Lee et al., 2018].

Neural Network GP

Convolutional layers

- Consider feature maps as multidimensional i.i.d. random variables.

Neural Network GP

Convolutional layers

- Consider feature maps as multidimensional i.i.d. random variables.
- Use multidimensional CLT as number of channels $C_l \to \infty$.

Neural Network GP

Convolutional layers

- Consider feature maps as multidimensional i.i.d. random variables.
- Use multidimensional CLT as number of channels $C_l \to \infty$.
- If you are interested only in variance at the final layer, you only need variance from the previous layers, because weights are independent and zero centered:

$$K_{\mu}^{(\ell+1)}\left(\mathbf{X},\mathbf{X}'\right) = \mathrm{C}\left[A_{i,\mu}^{(\ell+1)}(\mathbf{X}), A_{i,\mu}^{(\ell+1)}\left(\mathbf{X}'\right)\right] = \sigma_{\mathrm{b}}^2 + \sum_{j=1}^{C^{(n)}} \sum_{\nu=1}^{H^{(\ell)}D^{(\ell)}} \sigma_w^2 \mathbb{E}\left[\phi\left(A_{j,\nu}^{(l)}(\mathbf{X})\right)\phi\left(A_{j,\nu}^{(\ell)}\left(\mathbf{X}'\right)\right)\right] \tag{3}$$

Neural Tangent Kernel

Kernel Gradient Descent

Sample-then-optimize approach:

Neural Tangent Kernel

Kernel Gradient Descent

Sample-then-optimize approach:

- Associate a choice of parameters with a function $f \in \mathcal{F}$.

Neural Tangent Kernel

Kernel Gradient Descent

Sample-then-optimize approach:

- Associate a choice of parameters with a function $f \in \mathcal{F}$.
- Inner product

$$\langle f, g \rangle_{p^{in}} = \mathbb{E}_{x \sim p^{in}} \left[f(x)^T g(x) \right] \tag{4}$$

Neural Tangent Kernel

Kernel Gradient Descent

Sample-then-optimize approach:

- Associate a choice of parameters with a function $f \in \mathcal{F}$.
- Inner product

$$\langle f, g \rangle_{p^{in}} = \mathbb{E}_{x \sim p^{in}} \left[f(x)^T g(x) \right] \tag{4}$$

- Kernel product

$$\langle f, g \rangle_K := \mathbb{E}_{x, x' \sim p^{in}} \left[f(x)^T K\left(x, x'\right) g\left(x'\right) \right] \tag{5}$$

Neural Tangent Kernel

Kernel Gradient Descent

- Derivative of a cost function with respect to f

$$\partial_f^{in} C\Big|_{f_0} = \Big\langle d|_{f_0}, \cdot \Big\rangle_{p^{in}} \tag{6}$$

Neural Tangent Kernel

Kernel Gradient Descent

- Derivative of a cost function with respect to f

$$\partial_f^{in} C\Big|_{f_0} = \left\langle d|_{f_0}, \cdot \right\rangle_{p^{in}} \tag{6}$$

- The Kernel Gradient is defined as

$$\nabla_K C|_{f_0}(x) = \frac{1}{N} \sum_{j=1}^{N} K\left(x, x_j\right) d|_{f_0}\left(x_j\right) \tag{7}$$

Neural Tangent Kernel

Kernel Gradient Descent

- Derivative of a cost function with respect to f

$$\partial_f^{in} C\Big|_{f_0} = \left\langle d|_{f_0}, \cdot \right\rangle_{p^{in}} \tag{6}$$

- The Kernel Gradient is defined as

$$\nabla_K C|_{f_0}(x) = \frac{1}{N} \sum_{j=1}^{N} K\left(x, x_j\right) d|_{f_0}\left(x_j\right) \tag{7}$$

- It leads to the steepest descent

$$\partial_t C|_{f(t)} = -\left\langle d|_{f(t)}, \nabla_K C|_{f(t)} \right\rangle_{p^{in}} = -\left\| d|_{f(t)} \right\|_K^2 \tag{8}$$

Neural Tangent Kernel

Optimization with respect to parameters

- Approximate kernel by sampling functions $f^{(p)}$:

$$\mathbb{E}\left[f_k^{(p)}(x)f_{k'}^{(p)}\left(x'\right)\right] = K_{kk'}\left(x, x'\right) \tag{9}$$

Neural Tangent Kernel

Optimization with respect to parameters

- Approximate kernel by sampling functions $f^{(p)}$:

$$\mathbb{E}\left[f_k^{(p)}(x)f_{k'}^{(p)}\left(x'\right)\right] = K_{kk'}\left(x, x'\right) \tag{9}$$

- Consider linear parametrization:

$$f_{\theta(t)} = \frac{1}{\sqrt{P}}\sum_{p=1}^{P}\theta_p(t)f^{(p)} \tag{10}$$

Neural Tangent Kernel

Optimization with respect to parameters

- Approximate kernel by sampling functions $f^{(p)}$:

$$\mathbb{E}\left[f_k^{(p)}(x)f_{k'}^{(p)}(x')\right] = K_{kk'}(x, x') \tag{9}$$

- Consider linear parametrization:

$$f_{\theta(t)} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \theta_p(t) f^{(p)} \tag{10}$$

- Optimizing the cost with respect to θ is equivalent to kernel gradient descent with the Neural Tangent Kernel

$$\Theta = \sum_{p=1}^{P} \partial_{\theta_p} f_{\theta(t)} \otimes \partial_{\theta_p} f_{\theta(t)} = \frac{1}{P} \sum_{p=1}^{P} f^{(p)} \otimes f^{(p)} \xrightarrow{P \to \infty} K \tag{11}$$

Neural Tangent Kernel

Theoretical results

- In the general case parametrization is not linear, therefore $\partial_{\theta_p} f_{\theta(t)}$ and Θ_t depend on $\theta(t)$.

Neural Tangent Kernel

Theoretical results

- In the general case parametrization is not linear, therefore $\partial_{\theta_p} f_{\theta(t)}$ and Θ_t depend on $\theta(t)$.
- **Theorem, [Jacot et al., 2018]:** In the infinite-width limit NTK converges in probability to a determenistic kernel, thus stays almost constant during training.

Neural Tangent Kernel

Theoretical results

- In the general case parametrization is not linear, therefore $\partial_{\theta_p} f_{\theta(t)}$ and Θ_t depend on $\theta(t)$.
- **Theorem, [Jacot et al., 2018]:** In the infinite-width limit NTK converges in probability to a determenistic kernel, thus stays almost constant during training.
- **Theorem, [Lee et al., 2019]:** For a sufficiently wide neural network and sufficiently small learning rate the training trajectory is close the the trajectory of a linearized network:

$$f_t^{\text{lin}}(x) \equiv f_0(x) + \nabla_\theta f_0(x)|_{\theta=\theta_0} \omega_t \tag{12}$$

Neural Tangent Kernel

Theoretical results

- **Theorem, [Lee et al., 2019]:** As the width goes to infinity distribution $f_t^{\text{lin}}(x)$ converges to a normal distribution with mean

$$\mu\left(\mathcal{X}_T\right) = \Theta\left(\mathcal{X}_T, \mathcal{X}\right)\Theta^{-1}\left(I - e^{-\eta\Theta t}\right)\mathcal{Y} \tag{13}$$

which in turn converges to $\Theta\left(\mathcal{X}_T, \mathcal{X}\right)\Theta^{-1}\mathcal{Y}$ as $t \to \infty$.

Neural Tangent Kernel

Theoretical results

- **Theorem, [Lee et al., 2019]:** As the width goes to infinity distribution $f_t^{\text{lin}}(x)$ converges to a normal distribution with mean

$$\mu\left(\mathcal{X}_T\right) = \Theta\left(\mathcal{X}_T, \mathcal{X}\right) \Theta^{-1}\left(I - e^{-\eta\Theta t}\right) \mathcal{Y} \tag{13}$$

  which in turn converges to $\Theta\left(\mathcal{X}_T, \mathcal{X}\right) \Theta^{-1} \mathcal{Y}$ as $t \to \infty$.
- **Remark:** *"The distribution resulting from GD training does not generally correspond to a Bayesian posterior"*, [Lee et al., 2019].

Neural Tangents

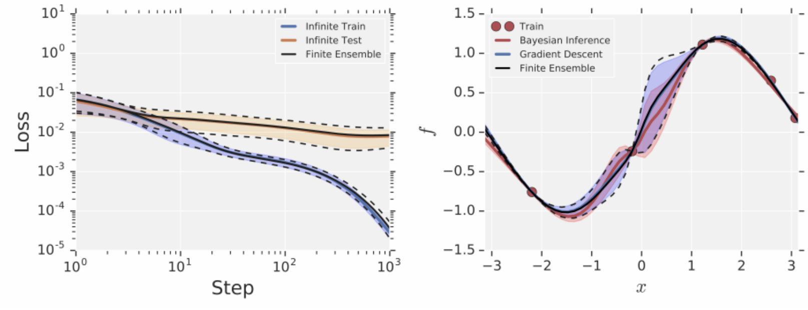

Figure: Training dynamics for an ensemble of finite-width networks compared with an infinite network, [Novak et al., 2020].

Neural Tangents

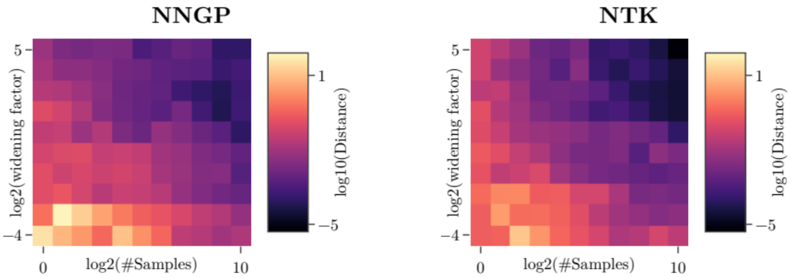

Figure: Convergence of the Monte Carlo (MC) estimates, [Novak et al., 2020].

Neural Tangents

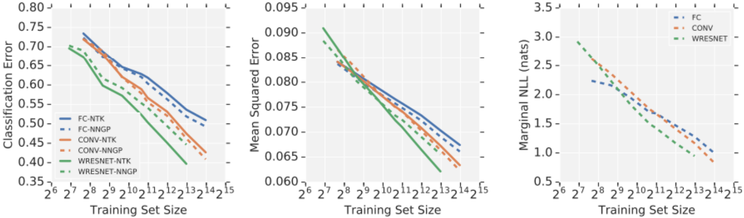

Figure: CIFAR-10 classification with varying neural network architectures, [Novak et al., 2020].

Thank you!

References I

Garriga-Alonso, A., Rasmussen, C. E., and Aitchison, L. (2019).
Deep convolutional networks as shallow gaussian processes.
In *International Conference on Learning Representations*.

Jacot, A., Gabriel, F., and Hongler, C. (2018).
Neural tangent kernel: Convergence and generalization in neural networks.
In *Advances in neural information processing systems*, pages 8571–8580.

Lee, J., Sohl-dickstein, J., Pennington, J., Novak, R., Schoenholz, S., and Bahri, Y. (2018).
Deep neural networks as gaussian processes.
In *International Conference on Learning Representations*.

References II

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. (2019).
Wide neural networks of any depth evolve as linear models under gradient descent.
In *Advances in neural information processing systems*, pages 8570–8581.

Novak, R., Xiao, L., Hron, J., Lee, J., Alemi, A. A., Sohl-Dickstein, J., and Schoenholz, S. S. (2020).
Neural tangents: Fast and easy infinite neural networks in python.
In *International Conference on Learning Representations*.

References III

Wikipedia contributors (2020).
Gaussian process — Wikipedia, the free encyclopedia.
https://en.wikipedia.org/w/index.php?title=Gaussian_process&oldid=944998270.
[Online; accessed 23-March-2020].