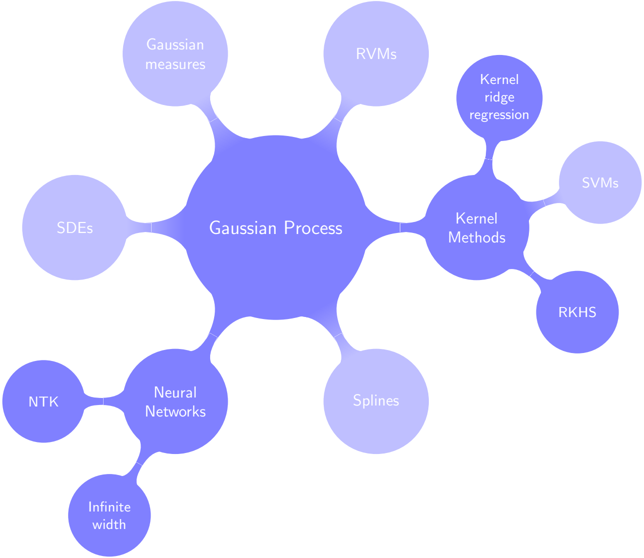
CS-E4895 Gaussian Processes
Lecture 8: Theory & Advanced Topics

Martin Trapp

Aalto University

Tuesday 21.03.2023

Connections



Hilbert Space

A vector space $\mathcal{V}$ is a set of vectors that is closed under addition and scalar multiplication.

If $\mathcal{V}$ is equipped with a norm $\|\cdot\|_{\mathcal{V}} \in \mathbb{R}$, it is a *normed (vector) space*.

A Hilbert space $\mathcal{H}$ is a complete[1] inner product space, with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and induced norm $\|x\|_{\mathcal{H}} = \sqrt{\langle x, x \rangle_{\mathcal{H}}}$.

Recall: Operations on inner products (scalar products):

- $\langle x, x \rangle \geq 0$ and $\langle x, \mathbf{0} \rangle = 0$
- $\langle x, y \rangle = \langle y, x \rangle$
- $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$
- $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle$

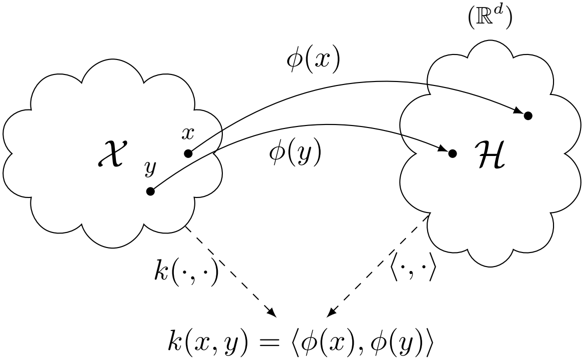
[1] All Cauchy sequences are convergent.

Recap: Kernel function

Recap from the previous lectures.

A function $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a *kernel* function if and only if there exists a Hilbert space $\mathcal{H}$ and a map $\phi\colon \mathcal{X} \to \mathcal{H}$ such that:

$$\boxed{k(x, y) = \langle \phi(x), \phi(y) \rangle} \tag{1}$$



Reproducing Kernel Hilbert Space

We said: Given a space $\mathcal{X}$ and a kernel k on $\mathcal{X}$, *there exists* a Hilbert space $\mathcal{H}$ and a map ϕ, such that:

$$k(x, y) = \langle \phi(x), \phi(y) \rangle \tag{2}$$

for all $x, y \in \mathcal{X}$.

First: Let $\phi \colon \mathcal{X} \to \mathbb{R}^{\mathcal{X}}$ and let us define:

$$k_x \coloneqq \phi(x) = k(x, \cdot) \tag{3}$$

Therefore, we have that: $k_x(y) = k(x, y)$.

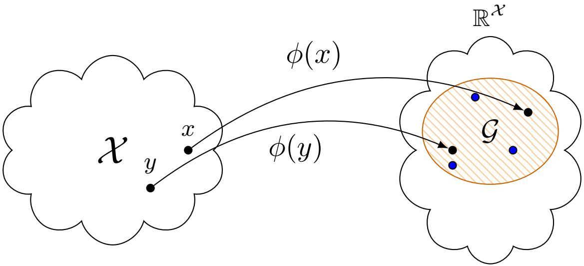
Second: Let $\mathcal{G}$ denote a vector space with span based on the images $\{k_x \mid x \in \mathcal{X}\}$, *i.e.*,

$$\mathcal{G} \coloneqq \{\sum_{i=1}^{m} \alpha_i \, k_{x_i} \mid \alpha_i \in \mathbb{R}, m \in \mathbb{N}, x_i \in \mathcal{X}\} \, . \tag{4}$$

Reproducing Kernel Hilbert Space

Let $\mathcal{G}$ denote a vector space with span based on the images $\{k_x \mid x \in \mathcal{X}\}$, *i.e.*,

$$\mathcal{G} := \{\sum_{i=1}^{m} \alpha_i \, k_{x_i} \mid \alpha_i \in \mathbb{R}, m \in \mathbb{N}, x_i \in \mathcal{X}\}\,. \tag{5}$$



Reproducing Kernel Hilbert Space

Now: Let's define an inner product on $\mathcal{G}$.

Note: By definition of a kernel function, we can only choose:

$$\langle k_x, k_y \rangle := k(x, y) \tag{6}$$

(Recall $k_x = k(x, \cdot)$, hence, $\langle k_x, k_y \rangle = \langle k(x, \cdot), k(y, \cdot) \rangle$.)

Therefore, for any $f, g \in \mathcal{G}$, with $f = \sum_i \alpha_i \, k_{x_i}$ and $g = \sum_j \beta_j \, k_{y_j}$, we have:

$$\langle f, g \rangle = \langle \sum_i \alpha_i \, k(x_i, \cdot), \sum_j \beta_j \, k(y_j, \cdot) \rangle \tag{7}$$

$$= \sum_{i,j} \alpha_i \, \beta_j \, \underbrace{\langle k_{x_i}, k_{y_j} \rangle}_{=k(x_i, y_j)} \tag{8}$$

Side note: To make $\mathcal{G}$ a Hilbert space, we need to make it complete, *i.e.*, ensure all Cauchy sequences converge = "ensure that no points are missing".

Reproducing Kernel Hilbert Space

Definition (Reproducing kernel Hilbert space (RKHS))

Let $\mathcal{H}$ be a Hilbert space of real functions f defined on an index set $\mathcal{X}$. Then $\mathcal{H}$ is called a reproducing kernel Hilbert space endowed with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ if there exists a kernel function $k \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the following properties:

- for every $x \in \mathcal{X}$, $k_x(y) = k(x, y)$ as a function of $y \in \mathcal{X}$ belongs to $\mathcal{H}$, and
- k has the reproducing property.

Reproducing property:

$$\langle k_x, f \rangle = \langle k_x, \sum_i \alpha_i k_{x_i} \rangle \tag{9}$$

$$= \sum_i \alpha_i \langle k_x, k_{x_i} \rangle = \sum_i \alpha_i k(x, x_i) = f(x) \tag{10}$$

Note: Given a kernel, there is a unique RKHS. Given an RKHS, there is a unique kernel. (Moore-Aronszajn theorem)

Representer Theorem

Setting:

- We are given a kernel k and denote the corresponding RKHS as $\mathcal{H}$.
- We want to learn a linear function $f(\mathbf{x})$ from a finite data set $\{\mathbf{x}_i, y_i\}_{i=1}^n$.

Theorem (Representer theorem)

Consider the risk minimization problem of the form:

$$\min_{f \in \mathcal{H}} \underbrace{R_n(\mathbf{y}, \mathbf{f})}_{\text{empirical risk}} + \lambda \underbrace{\Omega(\|f\|_{\mathcal{H}})}_{\text{regularizer}} \tag{11}$$

where $\mathbf{f} = \{f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)\}$, $\mathbf{y} = \{y_1, \ldots, y_n\}$, *and* λ *is a scaling parameter.*

Then eq. (11) *always has an optimal solution of the form:*

$$\boxed{f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \, k(\mathbf{x}_i, \mathbf{x})} \tag{12}$$

Representer Theorem: Example

Let's consider the following risk minimization problem:

$$\min_{f \in \mathcal{H}} \underbrace{\sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2}_{=R_n(\mathbf{y},\mathbf{f})} + \lambda \underbrace{\|f\|_{\mathcal{H}}^2}_{=\Omega(\|f\|_{\mathcal{H}})} . \tag{13}$$

This minimization problem is known as *kernel ridge regression*.

Let's plug in the solution according to the *representer theorem* ($f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \, k(\mathbf{x}_i, \mathbf{x})$):

$$\min_{f \in \mathcal{H}} \sum_{i=1}^{n} (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{\mathcal{H}}^2 \tag{14}$$

$$= \min_{\boldsymbol{\alpha}} \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \alpha_i \, k(\mathbf{x}_i, \mathbf{x}_j))^2 + \lambda \| \sum_{i=1}^{n} \alpha_i \, k(\mathbf{x}_i, \cdot) \|_{\mathcal{H}}^2 \tag{15}$$

Representer Theorem: Example

Let $\mathbf{K}$ denote the kernel matrix $\mathbf{K}_{\mathcal{X},\mathcal{X}}$, $\mathbf{y} = (y_1, \ldots, y_n)^\intercal$, then we can write the objective as:

$$J(\boldsymbol{\alpha}) = \sum_{i=1}^{n} (y_i - \sum_{j=1}^{n} \alpha_i \, k(\mathbf{x}_i, \mathbf{x}_j))^2 + \lambda \| \sum_{i=1}^{n} \alpha_i \, k(\mathbf{x}_i, \cdot) \|_{\mathcal{H}}^2 \tag{16}$$

$$= \boldsymbol{\alpha}^\intercal (\mathbf{K}\mathbf{K}^\intercal + \lambda \mathbf{K}) \boldsymbol{\alpha} - 2\mathbf{y}^\intercal \mathbf{K} \boldsymbol{\alpha} \tag{17}$$

Note that $J(\boldsymbol{\alpha})$ is convex and $\mathbf{K}$ is assumed positive-definite (hence, invertible).

Then, by taking $\frac{\partial J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = 0$ we obtain:

$$\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda I)^{-1} \mathbf{y} \tag{18}$$

and the prediction for test point $\mathbf{x}^*$ is, therefore,

$$\hat{f}(\mathbf{x}^*) = \sum_{i}^{n} \hat{\alpha}_i k(\mathbf{x}_i, \mathbf{x}^*) = \underbrace{\mathbf{k}^\intercal (\mathbf{K} + \lambda I)^{-1} \mathbf{y}}_{\substack{\text{predictive mean} \\ \text{of GP regression}}} \tag{19}$$

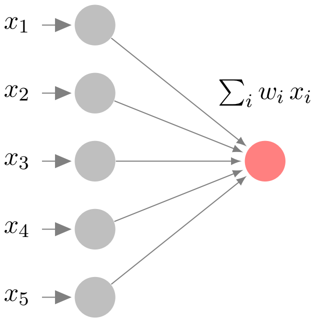
$\implies$ Minimizer to kernel ridge regression = predictive mean of GP regression.

Further readings

- C. E. Rasmussen & C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT press 2006, Chapter 6.
- A. Gretton, *Introduction to RKHS, and some simple kernel algorithms*, Lecture at UCL 2013.
- C. Heil, *Banach and hilbert space review.*, tech. rep., Georgia Tech, 2006.

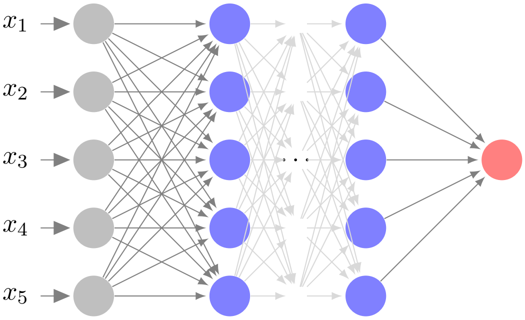
Neural Networks in 5 min

Perceptron



Multilayer Perceptron / NN

$$\boxed{z_j^{[l]} = \sum_i w_{i,j}^{[l]}\, \sigma(a_i^{[l-1]})}$$



- $\sigma(\cdot)$ is commonly chosen to be non-linear, *e.g.*, rectified linear unit (ReLU) defined as $\sigma(x) = \max\{x, 0\}$.
- The deep learning community extends this concept by a battery of modules & techniques which we will not discuss here.

Neural Network Classification: Example

Deterministic Neural Network | Bayesian Neural Network (BNN)

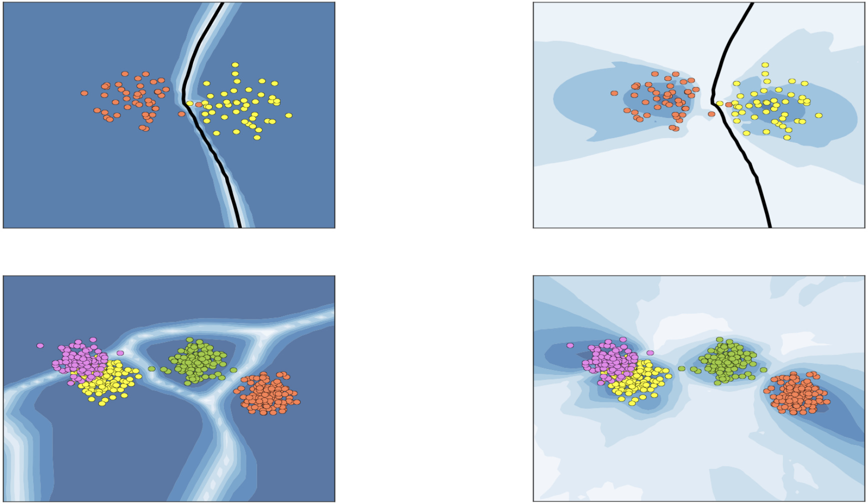
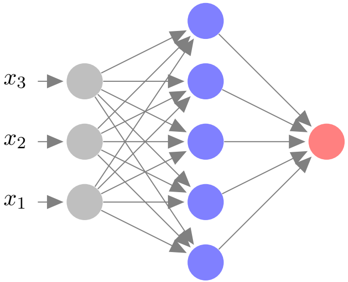


Illustration by: A. Kristiadi, M. Hein, and P. Hennig. *Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks*, ICML 2020.

Infinitely-wide single layer BNNs

First consider a single-hidden layer neural network of the form:

$$z_i^{[1]}(\mathbf{x}) = b_i^{[1]} + \sum_{j=1}^{K_1} w_{i,j}^{[1]} \, a_j^{[1]}(\mathbf{x}), \quad a_j^{[1]}(\mathbf{x}) = \sigma\left(b_j^{[0]} + \sum_{d=1}^{D} w_{j,d}^{[0]} \, x_d \right) \tag{20}$$



Assume the following generative process for the parameters:

$$w_{i,j}^{[1]} \sim \mathtt{N}(0, \frac{\sigma_w^2}{K_1}), \qquad b_i^{[1]} \sim \mathtt{N}(0, \sigma_b^2), \qquad w_{j,d}^{[0]} \sim \mathtt{N}(0, \frac{\sigma_w^2}{D}), \qquad b_j^{[0]} \sim \mathtt{N}(0, \sigma_b^2) \tag{21}$$

Infinitely-wide single layer BNNs

Since $z_i^{[1]}(\mathbf{x})$ is a sum of RVs with finite moments, then as $K_1 \to \infty$ the network output $z_i^{[1]}(\mathbf{x})$ converges in distribution to:

$$b_i^{[1]} + w_{i,1}^{[1]} a_1^{[1]}(\mathbf{x}) + w_{i,2}^{[1]} a_2^{[1]}(\mathbf{x}) + \ldots \xrightarrow{\mathcal{D}} \sqrt{1 + K_1}(Z_i(\mathbf{x}) - \mu) \quad (22)$$

with $Z_i(\mathbf{x}) \sim \mathtt{N}(0, \sigma^2)$. (CLT)

Following the multivariate CLT, we have that $\mathbf{Z}_i = (Z_i(\mathbf{x}_1), Z_i(\mathbf{x}_2), \ldots, Z_i(\mathbf{x}_n))^\intercal$ is joint multivariate Gaussian distributed.

Recall: A Gaussian process (GP) is a collection of RVs $\mathbf{F}$ indexed by $\mathcal{X}$, where any finite subset of $\mathbf{F}$ is joint multivariate Gaussian distributed, and of which any two overlapping finite sets are marginally consistent.

$\Longrightarrow$ A single-hidden layer BNN (prior) converges to a GP when $K_1 \to \infty$. (Neal 1994)

Infinitely-wide single layer BNNs

First question: What is the mean of this process?

$$\mathbb{E}[Z_i(\mathbf{x})] = \mathbb{E}[b_i^{[1]} + \sum_{j=1}^{K_1} w_{i,j}^{[1]}\, a_j^{[1]}(\mathbf{x})] \tag{23}$$

$$= \underbrace{\mathbb{E}[b_i^{[1]}]}_{=0} + \underbrace{\mathbb{E}[\sum_{j=1}^{K_1} w_{i,j}^{[1]}\, a_j^{[1]}(\mathbf{x})]}_{=\mathbf{0}^\intercal \mathbf{a}^{[1]}(\mathbf{x})=0} \tag{24}$$

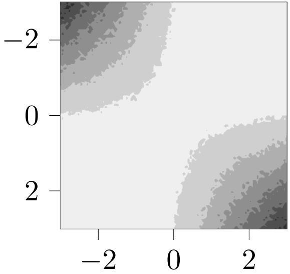
Second question: What is the induced kernel?
For this, let's examine the covariance:

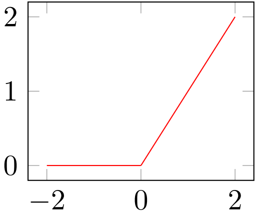
$$\mathbf{cov}(Z_i(\mathbf{x}_1), Z_i(\mathbf{x}_2)) = \mathbb{E}[Z_i(\mathbf{x}_1)Z_i(\mathbf{x}_2)] - \underbrace{\mathbb{E}[Z_i(\mathbf{x}_1)]\mathbb{E}[Z_i(\mathbf{x}_2)]}_{=0} \tag{25}$$

$$= \sigma_b^2 + \sigma_w^2\, \underbrace{\mathbb{E}[a_i^1(\mathbf{x}_1)a_i^1(\mathbf{x}_2)]}_{=k(\mathbf{x}_1,\mathbf{x}_2)} \tag{26}$$

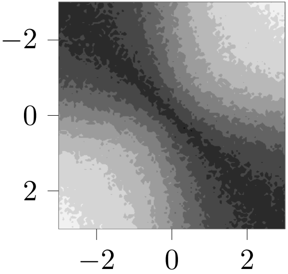
Infinitely-wide single layer BNNs

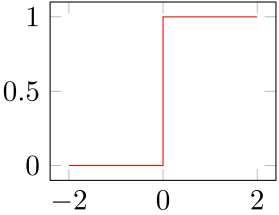
ReLU $\sim$ ArcCos-1 kernel



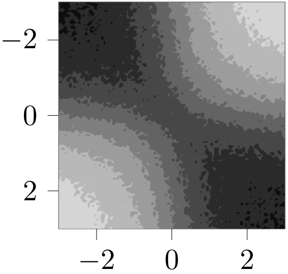


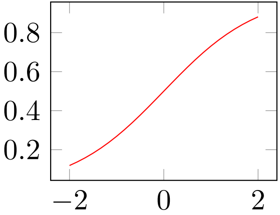
Step $\sim$ ArcCos-0 kernel





Sigmoid $\sim$ NN kernel





Infinitely-wide deep BNNs

Extensions to multiple hidden layers (aka deep neural networks).

$$z_i^{[1]}(\mathbf{x}) = b_i^{[1]} + \sum_{j=1}^{K_1} w_{i,j}^{[1]}\, a_j^{[1]}(\mathbf{x}), \quad a_j^{[1]}(\mathbf{x}) = \sigma\left(b_j^{[0]} + \sum_{d=1}^{D} w_{j,d}^{[0]}\, x_d\right) \tag{27}$$

Consider L layers with *i.i.d.* parameters, then $z_j^{[l-1]}(\mathbf{x})$ are *i.i.d.* draws from a GP.

Therefore, $z_j^{[l]}(\mathbf{x})$ is a sum of *i.i.d.* terms and

$$b_i^{[l]} + w_{i,1}^{[l]} a_1^{[l]}(\mathbf{x}) + w_{i,2}^{[l]} a_2^{[l]}(\mathbf{x}) + \ldots \xrightarrow{\mathcal{D}} \sqrt{1 + K_l}\,(Z_i^{[l]}(\mathbf{x}) - \mu) \tag{28}$$

with $Z_i^{[l]}(\mathbf{x}) \sim \mathtt{N}(0, \sigma^2)$ (CLT) and $\mathbf{Z}_i^{[l]} = (Z_i^{[l]}(\mathbf{x}_1), Z_i^{[l]}(\mathbf{x}_2), \ldots, Z_i^{[l]}(\mathbf{x}_n))^\intercal$ is joint multivariate Gaussian distributed (multivariate CLT).

$\implies$ Covariance structure now recursively defined.

Infinitely-wide deep BNNs

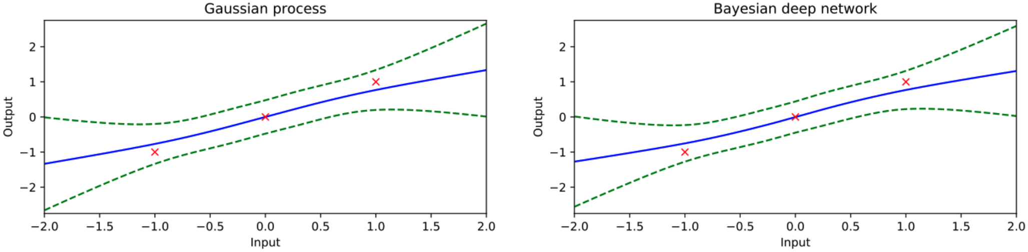


Illustration by: A. G. Matthews et al. *Gaussian process behaviour in wide deep neural networks*, ICLR 2018.

Note: Kernels obtained from infinite-width BNNs are fixed (not trainable), and do not accurately reflect the representation learning of NNs.

Further reading

- R. Neal *Bayesian Learning for Neural Networks*, PhD thesis, 1994.
- J. Lee et al. *Deep neural networks as Gaussian processes*, ICLR 2018.
- A. G. Matthews et al. *Gaussian process behaviour in wide deep neural networks*, ICLR 2018.

Neural Tangent Kernel

The Neural Tangent Kernel (NTK) establishes a link between gradient descent training in NNs and kernel methods.

Setting:

$$f(\mathbf{x};\theta) = \frac{1}{\sqrt{K}} \sum_{j=1}^{K} w_j^{[1]} \, \sigma\left(\sum_{d=1}^{D} w_{j,d}^{[0]} \, x_d \right) \tag{29}$$

where θ denotes all parameters (weights and biases).

Objective:

$$R_n = \frac{1}{2} \sum_{i=1}^{n} (f(\mathbf{x}_i;\theta) - y_i)^2 \tag{30}$$

Neural Tangent Kernel

$$R_n = \frac{1}{2}\sum_{i=1}^{n}(f(\mathbf{x}_i;\theta) - y_i)^2 \tag{31}$$

Gradient Descent (GD):

$$\theta_{t+1} = \theta_t - \eta\sum_{i=1}^{n}(f(\mathbf{x}_i;\theta) - y_i)\nabla_\theta f(\mathbf{x}_i;\theta_t) \tag{32}$$

where η is a learning rate that is usually small.

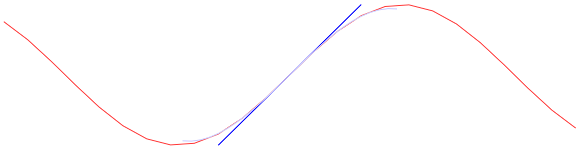
- If $f(\mathbf{x}_i;\theta_t)$ is linear: $\Rightarrow \nabla_\theta f(\mathbf{x}_i;\theta_t)$ is constant (only depends on $\mathbf{x}_i$)
- If $f(\mathbf{x}_i;\theta_t)$ is non-linear: $\Rightarrow \nabla_\theta f(\mathbf{x}_i;\theta_t)$ is changing over time

Empirical observation: If K is large then the parameters θ of a NN learned with GD do not change much over time. (lazy training)

Neural Tangent Kernel

Lazy training motivates: First-order Taylor approximation of $f(\mathbf{x}, \theta)$ around θ_0 (initialization):

$$f(\mathbf{x}; \theta) \approx f(\mathbf{x}; \theta_0) + \nabla_\theta f(\mathbf{x}; \theta_0)^\intercal (\theta - \theta_0) + \dots \tag{33}$$



Note: $\nabla_\theta f(\mathbf{x}; \theta_0)$ does not depend on θ and is nonlinear in $\mathbf{x}$.

Trick: We define $\phi(\mathbf{x}) \coloneqq \nabla_\theta f(\mathbf{x}; \theta_0)$.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \langle \nabla_\theta f(\mathbf{x}_i; \theta_0), \nabla_\theta f(\mathbf{x}_j; \theta_0) \rangle \tag{34}$$

Neural Tangent Kernel

The NTK allows us to derive the induced kernel of a infinite-width NN (often in closed-form).

We can also analyse the training dynamics:

$$\frac{\theta_{t-1} - \theta_t}{\eta} = -\nabla_\theta R_n(\theta_t), \quad R_n(\theta) = \frac{1}{2} \sum_{i=1}^{n} (f(\mathbf{x}_i; \theta) - y_i)^2 \tag{35}$$

As we have $\eta \to \infty$: (gradient flow)

$$\frac{d\theta(t)}{dt} = -\nabla_\theta R_n(\theta(t)) = -\nabla_\theta \hat{\mathbf{y}}(\theta(t)) \, (\hat{\mathbf{y}}(\theta(t)) - \mathbf{y}) \tag{36}$$

$$\frac{d\hat{\mathbf{y}}(\theta(t))}{dt} = -\underbrace{\nabla_\theta \hat{\mathbf{y}}(\theta(t))^\intercal \nabla_\theta \hat{\mathbf{y}}(\theta(t))}_{=\mathbf{K}_{\text{NTK}}} (\hat{\mathbf{y}}(\theta(t)) - \mathbf{y}) \tag{37}$$

Further readings

- A. Jacot, F. Gabriel, and C. Hongler, *Neural tangent kernel: Convergence and generalization in neural networks*, NeurIPS 2018.
- L. Chizat, F. Bach, *A Note on Lazy Training in Supervised Differentiable Programming*, tech. rep., INRIA 2019.
- `https://github.com/kwignb/NeuralTangentKernel-Papers`

Summary

Recap:

- Theory on RKHS allows us to better understand kernel functions.
- The representer theorem $\Rightarrow$ Optimal solution: $\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}_i, \mathbf{x})$
- Kernel ridge regression $=$ GP predictive mean
- BNN (at initialization) converges to a GP at infinite-width limit
- NTK establishes a link between NN training with GD and kernel methods

Advances in Probabilistic Machine Learning
`https://aaltoml.github.io/apml/`

CS-E407516: Special Course in ML, DS & AI: **Tractable Probabilistic Modelling**
`https://mycourses.aalto.fi/course/view.php?id=38446`