NTK

A Preprint

Aritificial neural networks $\sim$ Gaussian process in the infinite width which can be described by kernel. Authors consider FC ANN with layers numbered from 0 to L. Each contain $n_0 \dots n_L$ neurons, Lipschitz(differentiable everywhere and $d(f(x_1); f(x_2)) \leq K \cdot d(x_1; x_2)$), twice differentiable nonlinearity function with bounded second derivative.

$$F^{(L)} : R^P \to \mathcal{F} \tag{1}$$

Dimension of parameter space is defined as $P = \sum_{l=0}^{L-1}(n_l + 1)n_{l+1}$. Parameters are init as $\sim \mathcal{N}(0; 1)$. Seminorm $|| \odot ||_{p_{in}}$ defined in terms of bilinear form $< f, g >_{p_{in}} = \mathbb{E}_{x\sim p_{in}}[f(x)^T g(x)]$. It is important to note that these assumptions simplify the proof but doesnt affect it. The input distribution is finite. Define the network function as $f_\theta(x) := \tilde{\alpha}^{(L)}(x; \theta)$, where the functions $\tilde{\alpha}^{(\ell)}(\cdot; \theta) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_\ell}$ (called *preactivations*) and $\alpha^{(\ell)}(\cdot; \theta) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_\ell}$ (called *activations*) are defined from the 0-th to the L-th layer by:

$$\begin{aligned}
\alpha^{(0)}(x; \theta) &= x, \\
\tilde{\alpha}^{(\ell+1)}(x; \theta) &= \frac{1}{\sqrt{n_\ell}} W^{(\ell)} \alpha^{(\ell)}(x; \theta) + \beta b^{(\ell)}, \\
\alpha^{(\ell)}(x; \theta) &= \sigma\left(\tilde{\alpha}^{(\ell)}(x; \theta)\right).
\end{aligned}$$

Usually β and $\frac{1}{\sqrt{n_\ell}}$ are absent so LeCun initialization: $W_{ij}^\ell \sim \mathcal{N}(0; \frac{1}{n_\ell})$ and $b_j^\ell \sim \mathcal{N}(0; 1)$. Parameter $\frac{1}{\sqrt{n_\ell}}$ is key parameter as weight will grow to infinity without it. he variance of its output remains constant. The network's activations stay in a stable, well-behaved range. This is what they mean by a "consistent asymptotic behavior". It's the key to making the infinite-width analysis possible. But there is also a side effect: if layer width is something like 10000 then the weight gradients are scaled by about 1/100. The authors scale gradient by setting parameter $\beta = 0.1$ so now gradient is small from beginning and we do not have this tendency to bias contributin most part in gradient.

Training of ANN consists in optimiziing f_θ with respect to functional cost: $C : \mathcal{F} \to \mathbb{R}$ such as regression or cross-entropy etc. Even for a convex cost func the composite of $C \odot F^{(L)}$ is generally higly non-convex. A multi-dimensional kernel K is a function

$$K : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}^{n_L \times n_L},$$

which maps any pair (x, x') to an $n_L \times n_L$ matrix such that

$$K(x, x') = K(x', x)^\top$$

(i.e., K is a symmetric tensor in $\mathcal{F} \otimes \mathcal{F}$).

Such a kernel defines a bilinear map on $\mathcal{F}$ by taking the expectation over independent samples $x, x' \sim p_{\text{in}}$:

$$\langle f, g \rangle_K := \mathbb{E}_{x,x'\sim p_{\text{in}}}\left[f(x)^\top K(x, x') g(x')\right].$$

$$f_{\mu,i}(x) = \mu K_{i,*}(x,*) = \langle d, K_{i,*}(x,*)\rangle$$

The authors prove that as the width of your network goes to infinity, the training dynamics simplify dramatically. The complex process of updating millions of individual parameters θ becomes equivalent to a much simpler process. In this infinite limit, the training dynamics are governed by a fixed kernel, the NTK. This kernel is determined only by the network's architecture (depth, activation function) and initialization scheme. Crucially, in this limit, the kernel does not change during training. As we cannot calculate kernel gradient directly we come to previously defined step. The kernel gradient $\nabla_K C|_{f_0} \in \mathcal{F}$ is defined as

$$\nabla_K C|_{f_0} := \Phi_K \left(\partial_{\text{in}} f\, C|_{f_0}\right).$$

In contrast to $\partial_{\text{in}} f\, C$, which is only defined on the dataset, the kernel gradient generalizes to values x outside the dataset thanks to the kernel K:

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

A time-dependent function $f(t)$ follows kernel gradient descent with respect to K if it satisfies the differential equation:

$$\partial_t f(t) = -\nabla_K C|_{f(t)}.$$

During kernel gradient descent, the cost $C(f(t))$ evolves as:

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

Convergence to a critical point of C is hence guaranteed if the kernel K is positive definite with respect to the $\|\cdot\|_{p_{\text{in}}}$ norm: the cost is then strictly decreasing except at points where $\|d|_{f(t)}\|_{p_{\text{in}}} = 0$.

If the cost C is convex and bounded from below, then the function $f(t)$ converges to a global minimum as $t \to \infty$.

In the 3.1 section authors show on toy data that when you perform gradient descent on the parameters down exactly how the function itself f_θ changes. he "kernel"that governs this function-space gradient descent is created on the fly from the random functions themselves. It's called the "tangent kernel"for this simple model. So as number increases, tangent kernel converges to single fixed deterministic kernel.