Summary

Abstract (copied)

At initialization, artificial neural networks (ANNs) are equivalent to Gaussian processes in the infinite-width limit, thus connecting them to kernel methods. We prove that the evolution of an ANN during training can also be described by a kernel: during gradient descent on the parameters of an ANN, the network function fθ (which maps input vectors to output vectors) follows the kernel gradient of the functional cost (which is convex, in contrast to the parameter cost) w.r.t. a new kernel: the Neural Tangent Kernel (NTK). This kernel is central to describe the generalization features of ANNs. While the NTK is random at initialization and varies during training, in the infinite-width limit it converges to an explicit limiting kernel and it stays constant during training. This makes it possible to study the training of ANNs in function space instead of parameter space. Convergence of the training can then be related to the positive-definiteness of the limiting NTK. We prove the positive-definiteness of the limiting NTK when the data is supported on the sphere and the non-linearity is non-polynomial. We then focus on the setting of least-squares regression and show that in the infinitewidth limit, the network function fθ follows a linear differential equation during training. The convergence is fastest along the largest kernel principal components of the input data with respect to the NTK, hence suggesting a theoretical motivation for early stopping. Finally we study the NTK numerically, observe its behavior for wide networks, and compare it to the infinite-width limit.

Introduction

- It's known ANN's can approximate any function with enough hidden units, but its unknown what they converge to.
    - Due to highly non convex loss surface
    - Results suggest if the network is wide enough there are few "bad" local minima
- Mystery: Why do these overparameterised models generalise well?
    - "seems paradoxical that a reasonably large neural network can fit random labels, while still obtaining good test accuracy when trained on real data"
    - Kernel methods have the same properties!
- In the infinite limit ANN's become Gaussian described by kernel, the kernel can be used in Bayesian inference or SVM with similar results to gradient descent trained ANNs
    - In this paper they show the behavior of the ANN's is described by a related kernel, the NTK.

Contributions (copied)

We study the network function f_θ of an ANN, which maps an input vector to an output vector, where θ is the vector of the parameters of the ANN. In the limit as the widths of the hidden layers tend to infinity, the network function at initialization, f_θ converges to a Gaussian distribution. In this paper, we investigate fully connected networks in this infinite-width limit, and describe the dynamics of the network function f_θ during training:

- During gradient descent, we show that the dynamics of f_θ follows that of the so-called kernel gradient descent in function space with respect to a limiting kernel, which only depends on the depth of the network, the choice of nonlinearity and the initialization variance.
- The convergence properties of ANNs during training can then be related to the positive-definiteness of the infinite-width limit NTK. In the case when the dataset is supported on a sphere, we prove this positive-definiteness using recent results on dual activation functions (4). The values of the network function f_θ outside the training set is described by the NTK, which is crucial to understand how ANN generalize.
- For a least-squares regression loss, the network function f_θ follows a linear differential equation in the infinite-width limit, and the eigenfunctions of the Jacobian are the kernel principal components of the input data. This shows a direct connection to kernel methods and motivates the use of early stopping to reduce overfitting in the training of ANNs.
- Finally we investigate these theoretical results numerically for an artificial dataset (of points on the unit circle) and for the MNIST dataset. In particular we observe that the behavior of wide ANNs is close to the theoretical limit.

Neural Networks

- In this paper, fully connected ANNs with $0, \dots, L$ layers each with $n_1, \dots, n_L$ neurons and with a Lipschitz, twice differentiable nonlinearity function σ with bounded second derivative (these assumptions simplify the proofs but "do not seem strictly needed")

- This paper focusses on the realization function, which maps parameters to functions.

- There is a seminorm on the function space defined for a fixed distribution p^{in} (in this paper assumed to be a empirical distribution on a finite dataset):

$$\langle f, g \rangle_{p^{in}} = \mathbb{E}_{x \sim p^{in}}[f(x)^\intercal g(x)]$$

- In remark 1 the importance of the width of the neural network is emphasised, in particular having the factor $\frac{1}{\sqrt{n_l}}$ in the initialisations are needed to get the right

asymptotic behavior as $n_1, \dots, n_L \to \infty$. However using these factors reduce the influence of the connection weights during training when n_l is large.

Kernel Gradient

- Goal of training an ANN → optimise f_θ in the function space wrt some cost function.
  - Even for convex cost function the composite cost (cost composed with realisation function) can be highly non-convex
- They show the network function follows kernel gradient descent wrt the Neural Tangent Kernel
- A multidimensional kernel is a map $K : \mathbb{R}^{n_0} \times \mathbb{R}^{n_0} \to \mathbb{R}^{n_L \times n_L}$ which maps pairs of points to symmetric matrices. It defines a bilinear map on the function space

$$\langle f, g \rangle_K = \mathbb{E}_{x,x' \sim p^{in}}[f(x)^\intercal K(x, x') g(x)]$$

  This kernel is positive definite wrt the seminorm above if $\|f\|_{p^{in}} > 0 \implies \|f\|_K > 0$
- They denote by $\mathcal{F}^*$ the dual of the function space with respect to the distribution p^{in} , the set of linear forms $\mu : \mathcal{F} \to \mathbb{R}$ of the form $\mu = \langle d, \cdot \rangle_{p^{in}}$ for some $d \in \mathcal{F}$. Which d is chosen doesn't matter in this paper.
  - Partial application of the kernel is a function in $\mathcal{F}$ so we can map (Φ_K) from a dual element μ to a function in $f\mu \in \mathcal{F}$ with values:

$$f_{\mu.i}(x) = \mu K_{i,\cdot}(x, \cdot) = \langle d, K_{i,\cdot}(x, \cdot) \rangle_{p^{in}}$$

- The cost functional only depends on the values of f at the data points (finite in this setting), so the cost functional is in $\mathcal{F}$ and its (functional) derivative at a point f_0 can be viewed as an element of the dual.
  - It is an expectation wrt p^{in}, but im not sure how it's in the dual, because how is it a linear form if its being evaluated at f_0
  - Then there exists a dual element $d|_{f_0} \in \mathcal{F}$ s.t $\partial_f^{in} C|_{f_0} = \langle d|_{f_0}, \cdot \rangle_{p^{in}}$
- The kernel gradient is the application of Φ_K to the functional derivative of the cost function.
  - Becauase of the kernel this can be generalised (as opposed to the functional derivative) to values outside of the dataset.
- A function "follows" the 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 function evolves according to a differential equation depending on the dual element defined above.
  - If the kernel K is positive definite wrt $\|\cdot\|_{p^{in}}$ then covergence to a critical point is guaranteed.

Random Functions approximation

- A kernel can be approximated by a choice of P random functions $f^{(p)}$ sampled independently from any distribution on $\mathcal{F}$ whose (non-centred) covariance is given by K
- Using these random functions the above kernel gradient descent can be studied
- The random functions defined the following random linear parametrisation:

  $$F^{lin} : \theta \to f_\theta^{lin} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \theta_p f^{(p)}$$

  With partial derivatives:

  $$\partial_{\theta_p} F^{lin}(\theta) = \frac{1}{\sqrt{P}} f^{(p)}$$

- Question: Why does the right-hand side equal the kernel gradient wrt $\tilde{K}$

Neural Tangent Kernel

- For ANN trained using gradient descent on the composition of the cost and realisation function, the network function follows the negative kernel gradient