Theoretical Principles of Deep Learning

Class 5: Neural Tangent Kernel

Hédi Hadiji

Université Paris-Saclay - CentraleSupelec

hedi.hadiji@l2s.centralesupelec.fr

January 15th, 2023

Table of Contents

Plan

Last time

- Generalization bounds
- Rademacher complexity

Today: Study the linearized model

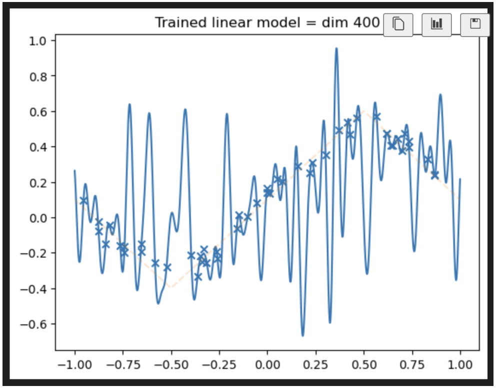
Reading Material:

- Telgarsky's notes

Rethinking generalization

Classical theory of generalization is:

- more complex models have larger complexity
- interpolating the data is bad for generalization

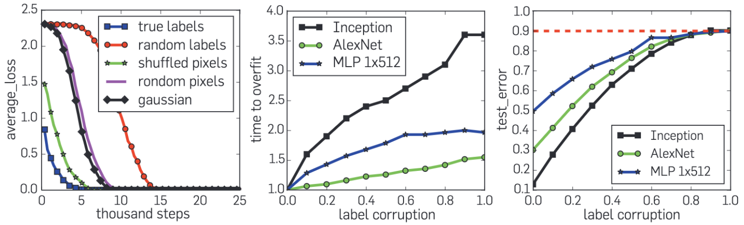


Some empirical observations

On **real data**, deep neural networks can

- interpolate and generalize
- interpolate random labels easily

Figure 1. Fitting random labels and random pixels on CIFAR10. (a) The training loss of various experiment settings decaying with the training steps. (b) The relative convergence time with different label corruption ratio. (c) The test error (also the generalization error since training error is 0) under different label corruptions.



(a) Learning curves (b) Convergence slowdown (c) Generalization error growth

[Zhang et al. '17] *Understanding deep learning requires rethinking generalization*

Consequences of these observations

- Not clear what overfitting means in deep learning
- Explicit regularization methods are not necessary (although they often improve performance)
- Simply measuring the complexity of the class of neural networks will not explain the success of deep learning.

Successful bounds will need to incorporate the labels or the optimization algorithm.

e.g. the Rademacher complexity of the class of deep neural nets reachable by GD on typical image sets is likely large.

Table of Contents

Generalization in the Lazy Regime

Overfitting and generalization are difficult to analyze for arbitrary neural nets. However, sometimes the training of neural nets is close to that of a linear models: in the **Lazy Regime**.

In class 3, we discussed a regime in which the function represented by the neural net stays close to its linear approximation during training:

$$h(w) \approx \bar{h}(w) := h(w_0) + \langle \nabla h(w_0), w - w_0 \rangle$$

In particular, very wide nets stay in the lazy regime.

Program for today

Today: assume we are in the lazy regime, and study the generalization properties of the approximating linear model.

Focus on the **overparameterized** regime, where the number of trainable parameters p is larger than number of data points n.

Notation: $h(w) = h(w, \cdot)$

Reminder: overparameterized least-squares and the kernel trick

Consider an arbitrary feature map: $\phi : \mathbb{R} \to \mathbb{R}^p$, and the hypotheses

$$h(w) = h(w_0) + \langle w - w_0, \phi(x) \rangle$$

assume wlog that $h(w_0) = 0$.

Least-squares training objective

$$\arg\min_{w \in \mathbb{R}^p} \frac{1}{2n} \left\| \mathbf{\Phi}^\top (w - w_0) - \mathbf{Y} \right\|^2$$

where

$$\mathbf{\Phi} = \begin{pmatrix} \vdots & & \vdots \\ \vdots & & \vdots \\ \phi(x_1) & \dots & \phi(x_n) \\ \vdots & & \vdots \\ \vdots & & \vdots \end{pmatrix} \in \mathbb{R}^{p \times n} \quad \text{and} \quad \mathbf{Y} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \in \mathbb{R}^n .$$

GD and least-squares regression

Theorem (Limit of GD)

If Rank$(\mathbf{\Phi}) = n$*, then the GD updates converge to*

$$w_\infty - w_0 = \mathbf{\Phi}\left(\mathbf{\Phi}^\top \mathbf{\Phi}\right)^{-1}\mathbf{Y} = \sum_{i=1}^{n} \phi(x_i)(H^{-1}\mathbf{Y})_i \,,$$

where H is the Gram matrix of the samples

$$H = \mathbf{\Phi}^\top \mathbf{\Phi} = \left(\langle \phi(x_i), \phi(x_j)\rangle\right)_{i,j\in[n]} \,.$$

Proof. Board.

Consequences:

- Interpolation: For all $i \in [n]$

$$h(w_\infty, x_i) = y_i$$

- More generally: for any $x \in \mathcal{X}$

$$h(w_\infty, x) = \sum_{i=1}^{n} \langle \phi(x), \phi(x_i) \rangle (H^{-1}\mathbf{Y})_i$$

Crucial observation: the feature map only appears in expressions as $\langle \phi(x), \phi(x') \rangle$. Therefore, we can compute the function learned without computing explicitly the feature map, **even if** $p = \infty$.

Writing an algorithm in this form is called the ‘kernel trick’

Kernelized least-squares

Given an abstract feature map $\phi : x \to \mathbb{R}^p$, the associated **kernel**:

$$k : (x, x') \mapsto \langle \phi(x), \phi(x') \rangle$$

If we can compute $k(x, x')$, then we can compute the iterations of gradient descent without ever computing Φ explicitly.
In the case of linear least-squares, if

$$H = \big(k(x_i, x_j)\big)_{i,j}$$

is invertible, then the output of SGD can be written using the kernel trick for any $x \in \mathcal{X}$ as

$$h(w_\infty, x) = \sum_{i=1}^{n} k(x, x_i)(H^{-1}\mathbf{Y})_i$$

Many algorithms can be kernelized this way.

Table of Contents

NTK

The NTK is the kernel function from the linear approximation of a neural network around its initialization.

Nice because:

- can be computed explicitly in the ∞-width limit
- purely convex method
- amenable to analysis

An explicit example

Example: Consider a one-dimensional example of a network without biases

$$h(w_0, x) = \sum_{i=1}^{m} a_i \sigma(b_i x)$$

a_i, b_i are initialized as random variables with variance proportional to number of inputs, i.e.,

$$\mathrm{Var}(a_i) = \frac{v_a}{n} \quad \text{and} \quad \mathrm{Var}(b_i) = v_b.$$

Or, equivalently, consider the parametrization

$$h(w_0, x) = \frac{1}{\sqrt{m}} \sum_{i=1}^{m} a_i \sigma(b_i x) \quad \text{with} \quad \mathrm{Var}(a_i) = 1 \quad \text{and} \quad \mathrm{Var}(b_i) = 1.$$

What is the corresponding kernel?

One dimensional input, two-layer net

Let us see the corresponding kernel. Differentiate with respect to the weights

$$\nabla_w h(w_0, x) = \begin{pmatrix} \vdots \\ \sigma(b_i x) \\ \vdots \\ a_i x\, \sigma'(b_i x) \\ \vdots \end{pmatrix}$$

Then the kernel value is

$$\langle \nabla_w h(w_0, x), \nabla_w h(w_0, x') \rangle = \frac{1}{m} \sum_{i=1}^{m} \sigma(b_i x)\sigma(b_i x') + \frac{1}{m} \sum_{i=1}^{m} a_i^2 x x' \sigma'(b_i x)\sigma'(b_i x')$$

What happens as $m \to \infty$?

Two-layer neural net

The Neural Tangent Kernel converges pointwise almost surely to

$$k(x,x') = \mathbb{E}[\sigma(Bx)\sigma(Bx')] + \mathbb{E}[A^2 xx'\sigma'(Bx)\sigma'(Bx')]$$

More generally if the inputs x is multidimensional and all the weights are initialized to $\mathcal{N}(0,1)$, then

$$k(x,x') = \langle x,x'\rangle \mathbb{E}_{B\sim\mathcal{N}(0,I_d)}[\sigma'(\langle B,x\rangle)\sigma'(\langle B,x'\rangle)] + \mathbb{E}_{B\sim\mathcal{N}(0,I_d)}[\sigma(\langle B,x\rangle)\sigma(\langle B,x'\rangle)]\,.$$

This can sometimes be computed explicitly e.g.

One hidden layer ReLU NTK

For ReLU activations with Gaussian initialization,

$$k(x,x') = \|x\|\|x'\|\kappa\left(\frac{\langle x,x'\rangle}{\|x\|\|x'\|}\right)$$

where

$$\kappa(\xi) = \frac{2\xi}{\pi}(\pi - \arccos(\xi)) + \frac{\sqrt{1-\xi^2}}{\pi}\,.$$

Table of Contents

Generalization analysis

Theorem (Rademacher complexity of linear regression)

If $\mathcal{H} = \{x \mapsto \langle \phi(x), w - w_0 \rangle \mid \|w - w_0\| \leqslant W\}$ *then*

$$\mathcal{R}_S(\mathcal{H}) = \mathbb{E}\left[\sup_{h \in \mathcal{H}} \sum_{i=1}^{n} \sigma_i h(w, x_i)\right] \leqslant \frac{W \max \|\phi(x_i)\|}{\sqrt{n}}.$$

In our case, we know that

$$\|w_\infty - w_0\| = \|\mathbf{\Phi} H^{-1}\mathbf{Y}\| = \sqrt{\mathbf{Y}^\top H^{-1}\mathbf{Y}}$$

so we should be able to restrict the Rademacher complexity to models with distance $\sqrt{\mathbf{Y}^\top H^{-1}\mathbf{Y}}$ from initialization, to obtain generalization bounds of the form: with prob $\geqslant 1 - \delta$

$$R^{\text{0-1}}(h(w_\infty)) \leqslant c\sqrt{\frac{\mathbf{Y}^\top H^{-1}\mathbf{Y}}{n}} + \sqrt{\frac{\log(1/\delta)}{n}}$$

where $R^{\text{0-1}}(h(w_\infty))$ is the probability of error of $h(w_\infty)$.

Careful the previous reasoning is not valid because the bound on W depends on the data!! However it is possible to turn it into a sound argument.

Theorem (Generalization for overparameterized kernel regression)

If $\lambda_{\min}(H) > \lambda$ with probability at least $1-\delta$, then with probability at least $1-\delta$, the output of GD converges to a hypothesis such that

$$R^{0\text{-}1}(h(w_\infty)) \leqslant c\sqrt{\frac{\mathbf{Y}^\top H^{-1}\mathbf{Y}}{n}} + c'\sqrt{\frac{\log(n/(\lambda\delta))}{n}}$$

simplified from Thm. 5.1 in 'Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks' *Arora et al. 2019*

Gives a criterion to understand learnability by an NTK. It suffices that

- H be invertible (with some uniformity on its eigenvalues)
- $\mathbf{Y}^\top H^{-1}\mathbf{Y}$ grows slower than n

Prook sketch: Adaptive bound

Under the event that $\lambda_{\text{min}} > \lambda$, we have

$$\mathbf{Y}^\top H^{-1}\mathbf{Y} \leqslant \frac{n}{\lambda}$$

Then discretize for W in $[0, Y^\top H^{-1} Y]$ and union bound over the sets. Actually can sacrifice a factor 2 in the bound to replace $\log(n/\lambda)$ by $\log\log(n/\lambda)$.

Extending this analysis

To conclude analysis

- Must check that the network function stays close its linear approximation for finite width.

To analyze more practical nets:

- Exact formulae for the infinite width NTK can be derived for large families of networks: arbitrary depth (limit might depend on the way you make the width go to infinity), convolutions, residual networks, etc.
- Most of this can be extended to other losses: the NTK does not the depend on the loss, but the limit point of (S)GD does, so careful analysis is necessary.

Some limits of the NTK

From a theory point of view

- NTK analysis essentially **freezes the features at initialization**
- This removes a fundamental property of neural networks in the wild.

From afar one might wonder why we do not use NTKs instead of training neural nets. The computational complexity of computing the GD iterates grows (at least) quadratically with the number of points.

Table of Contents

Conclusion and Next time

Today

- Defined the Neural Tangent Kernel associated to a neural network
- Discussed the optimization and generalization

In lab: will implement an NTK

Next time **discuss other theories of generalization**