Deep learning generalizes because the parameter-function map is biased towards simple functions

Guillermo Valle-Pérez, Chico Q. Camargo, Ard A. Louis

Departments of Physics, University of Oxford, UK

Why does deep learning generalize?

Supervised learning theory: Because it has an inductive bias

Ok, but why does it have an inductive bias?

VC theory: Limited expressivity maybe?

Zhang et al. (2017a): No, neural networks (NNs) can fit randomly labelled data

D Soudry et al., Zhang et al. (2017b), Zhang et al. (2018): Maybe SGD is what's biasing towards certain solutions

But many very different optimization algorithms ~~generalize well~~

Wu et al.: ~~Yeah!~~

Hmm, maybe it's an intrinsic property of the NN, like its parameter-function map?

What's that?

Let the space of functions that the model can express be $\mathcal{F}$. If the model has p real valued parameters, taking values within a set $\Theta \subseteq \mathbb{R}^p$,
the parameter-function map, $\mathcal{M}$, is defined as:

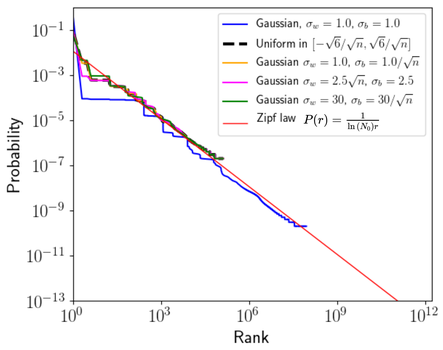
$$\mathcal{M} : \Theta \to \mathcal{F}$$
$$\theta \mapsto f_\theta$$

where f_θ is the function implemented by the model with choice of parameter vector θ.

Result 1: The parameter-function map is hugely biased

For all the neural network architectures we tried:

The volumes of regions of parameter space producing particular funcions span a huge range of orders of magnitude.



Oh, and how did you find that out?

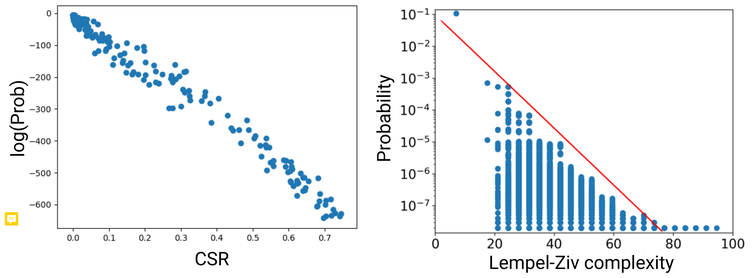
~~For a family of fully connected feedforward neural networks with 7 Boolean inputs and one Boolean output of varying depths and widths, we sampled parameters with several distributions. In Figure, we show the empirical frequencies by which different functions are obtained~~

~~For some larger neural networks with higher dimensional input spaces, we used a Gaussian process approximation to calculate the probability of different functions. This can be seen in Figures~~

Ok, but do we have any way to characterize the bias? What kinds of functions are the networks biased towards?

Result 2: The bias is towards simple functions

We found that in all cases, the probability of a function inversely correlated with its complexity (using a variety of measures of complexity)



Why are the networks biased?

~~No deeper explanation yet about why the parameter function map is biased. However, we do have some deeper reason, based on **algorithmic information theory** for why it is biased towards simple functions, given that it is biased.~~

Dingle et al.:

~~The probability $P(x)$ to obtain output x of a *simple* map f, upon sampling its inputs uniformly at random, depends only on the Kolmogorov complexity of the output $K(x)$:~~

$$~~P(x) \leq 2^{-K(x)}~~$$

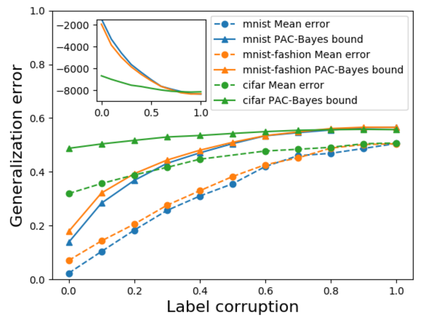
~~The main condition on the map is that its Kolmogorov complexity is negligible relative to that of the output $K(f) \ll K(x)$~~

~~Kolmogorov complexity is uncomputable, so we use computable approximations to it, like Lempel-Ziv complexity~~

~~The parameter function map satisfies $K(f) \ll K(x)$, and indeed we found that the bound works (red line in Figure)~~

Is this bias enough to explain the observed generalization?

Result 3: The bias is enough to explain "the bulk" of the generalization in our experiments



Interesting, and how did you determine that?

To explore this question:

- We use the **PAC-Bayesian framework** to translate probabilistic biases into generalization guarantees
- We make the assumption that the algorithm optimizing the parameters is unbiased, to isolate the effect of the parameter-function map. More precisely, we assume that the optimization algorithm samples the zero-error region close to uniformly (***Assumption 1***).

Can you provide more details on your method to obtain PAC-Bayes bounds?

Corollary 1 *(of ~~Langford's version~~ of the **PAC-Bayesian theorem** ~~(Langford et al.))~~ For any distribution P on any function space and realizable distribution $\mathcal{D}$ on a space of instances we have, for $0 < \delta \leq 1$, that with probability at least $1 - \delta$ over the choice of sample S of m instances*

$$-\ln(1 - \epsilon(Q^*)) \leq \frac{\ln \frac{1}{P(U)} + \ln\left(\frac{2m}{\delta}\right)}{m - 1}$$

where $\epsilon(Q^*)$ is the expected generalization error under distribution over functions $Q^*(c) = \frac{P(c)}{\sum_{c \in U} P(c)}$, U is the set of functions in $\mathcal{H}$ consistent with the sample S, and ~~where~~ $P(U) = \sum_{c \in U} P(c)$

Ah I see. So the bound depends on the data via $P(U)$, which is nothing but the marginal likelihood of the labels on the data, given by the prior $P(f)$. But how do you calculate $P(U)$ for neural networks, isn't that intractable?

J Lee et al.: Yes. However, $P(f)$ for deep fully connected neural networks approaches a Gaussian process as the width of the layers approaches infinity.

A Garriga-Alonso et al., R Novak et al.: also for convolutional networks, as the number of filters goes to infinity!

AGG Mathews et al.: and it seems the networks don't need to be that wide for the approximation to be good (we independently checked this too)

Thanks everyone! The **Gaussian process approximation** is what allows us to compute $P(U)$ for realistically-sized NNs. However, the marginal likelihood for a Gaussian process with Bernoulli likelihood (for binary classification, the setting of PAC-Bayes) is still intractable, and so we explored some approximation techniques: Variational, Laplace, expectation-propagation (EP), and found EP to work best for our purposes.

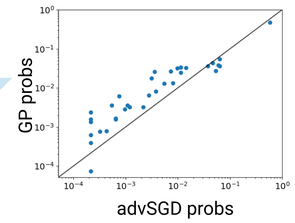
What's the effect of the optimization algorithm?

After all, different optimization algorithms do show differences in generalization in practice

Yes, but differences in generalization are typically of only a few percent.
However, you raise an important point. Although we have shown that the bias is *enough* to explain the bulk of the generalization, whether it is the *actual* origin of the generalization in DNNs depends on the behaviour of the optimization algorithm.

A sufficient (though not necessary) condition for the parameter-function map to be main origin of the generalization is that the optimization algorithm isn't too biased, namely ***Assumption 1*** is approximately valid.

We conjecture that it is for many common DNN optimization algorithms (note that for exact Bayesian sampling it is true, by definition), and show some empirical evidence supporting this .



Future work?

There are problems regarding the validity of Assumption 1, the EP or other approximations to $P(U)$, as well as the tightness of PAC-Bayes itself.

Furthermore, one can dig deeper to try to better understand the origin of the bias, and characterize it. In particular, there is the very important question of **why is the bias helpful for *real-world* tasks?**

Starring:
Zhang et al (2017a). Understanding deep learning requires rethinking generalization. Published in ICLR 2017
Wu et al. Towards understanding generalization of deep learning: Perspective of loss landscapes. arXiv preprint arXiv:1706.10239, 2017.
J Lee et al. Deep neural networks as gaussian processes. arXiv preprint arXiv:1711.00165, 2017.
A Garriga-Alonso et al. Deep convolutional networks as shallow Gaussian processes. Published in ICLR 2019
R Novak et al. Bayesian Deep Convolutional Networks with Many Channels are Gaussian Processes. Published in ICLR 2019
AGG Mathews et al. Gaussian Process Behaviour in Wide Deep Neural Networks. Published in ICLR 2018
Dingle et al. Input–output maps are strongly biased towards simple outputs. Nature communications, 9(1):761, 2018.
D Soudry et al. The implicit bias of gradient descent on separable data. arXiv preprint arXiv:1710.10345, 2017
Zhang et al. (2017b) Musings on deep learning: Properties of sgd. CBMM Memos 04/2017.
Zhang et al. (2018) Energy–entropy competition and the effectiveness of stochastic gradient descent in machine learning. Molecular Physics, pp. 1–10, 2018.
Langford et al. Bounds for averaging classifiers. 2001.