UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION

Matthew C. Scicluna [1]

March 26 2018

[1]Montréal Institute of Learning Algorithms
Université de Montréal

Table of contents

1. Introduction
2. Background
3. Results
4. Theoretical Results
5. Discussion

Introduction

Main Question

Main Question

What distinguishes Neural Networks that generalize well from those that don't?

- Capacity ?
- Regularization ?
- How we train the model?

Traditional View

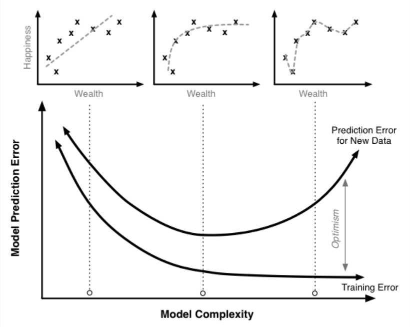


Figure 1: Traditional view of generalization. Image taken from [1]

- Traditional reasoning: We want models with enough capacity to fit the underlying signal ...
- But not so large that they can also fit the idiosyncrasies of the particular dataset!

Newer View

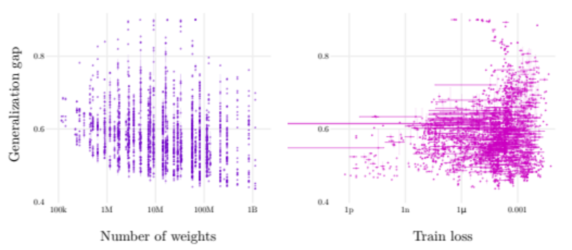


Figure 2: New view of generalization. Image taken from [2]

- Large over-parameterized neural networks generalize better then their smaller counterparts
- We see this in practice: models get bigger and bigger but performance also increases!

Motivation

Why do we care about the problem?

- Make neural networks more interpretable
- May lead to more principled and reliable model architecture design

Background

Previous Approaches

We can bound the Generalization Error using measures of complexity such as:

- VC Dimension
- Rademacher Complexity
- Uniform Stability

Additionally, regularization can help (including Early Stopping)

Related Work

In 2016 Hardt et al. gives an Upper bound on Generalization error on models using SGD[1] using uniform stability [3]

BUT

Uniform stability is a property of a learning algorithm and is not affected by the labelling of the training data.

[1] with a small, fixed number of epochs

Related Work

It is well established that Neural Networks are universal function approximators [4]:

- feed forward network with single hidden layer with finitely many neurons can approximate continuous functions on a compact subset of $\mathbb{R}^n$.
- This paper shows that neural networks don't even need to be that large to memorize a fixed amount of data (more on this later).

Limitations

Main Message

Classic results (e.g. PAC bounds) are insufficient in that they cannot distinguish between neural networks with dramatically different generalization performance.

This is demonstrated in the paper [5]. The central finding:

Deep neural networks easily fit random labels

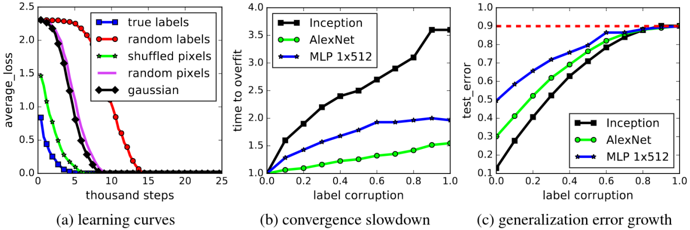
Results

Experiment

Setup: trained several standard architectures on the data with various modifications:

1. True labels → No modifications
2. Random labels → randomly changed some labels
3. shuffled pixels → apply some fixed permutation of pixels to all images
4. Random pixels → apply some random permutation of pixels to all images
5. Gaussian → Generate pixels for all images from a Gaussian

Main Results



(a) learning curves

(b) convergence slowdown

(c) generalization error growth

Figure 3: Fitting random labels and random pixels on CIFAR10.

Results

In most cases, the training error went to zero while test error was high

Notice:

the model capacity, hyperparameters, and the optimizer remained the same!

Results

Highlights:

- training takes longer for the randomized labels
- random labels are the hardest to train, even when compared to totally corrupted inputs – due to less correlated inputs?
- The neural networks handles partial corruptions well by capturing whatever signal is left while fitting the noisy part using brute-force

Results

Explicit regularization may improve generalization performance, but is neither necessary nor by itself sufficient for controlling generalization error

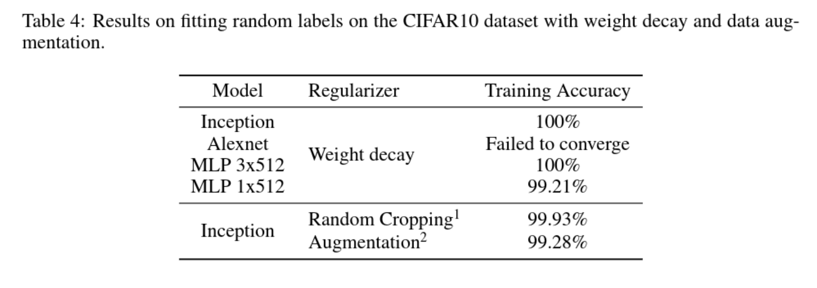


Table 4: Results on fitting random labels on the CIFAR10 dataset with weight decay and data augmentation.

Model	Regularizer	Training Accuracy
Inception	Weight decay	100%
Alexnet		Failed to converge
MLP 3x512		100%
MLP 1x512		99.21%
Inception	Random Cropping[1]	99.93%
	Augmentation[2]	99.28%

Theoretical Results

Some Definitions

First some definitions:

- **Representational Capacity**: what functions neural networks can fit over the entire domain (e.g. Universal Representation Theorem)
- **Finite-sample expressivity**: the expressive power of a learning algorithm on a finite sample of size n.

Finite-sample expressivity

Results on representational capacity can be extended to the finite sample domain using uniform convergence theorems. However, the bounds are much larger. Paper directly analyzes finite-sample expressivity.

Finite-sample expressivity

Generically large neural networks can express any labelling of the training data.

Theorem

There exists[2] a two-layer neural network with ReLU activations and $2n + d$ weights that can represent any function on a sample of size n in d dimensions.

[2] NOT all networks satisfy this

Proof

Lemma 1

For any two interleaving sequences of n real numbers $b_1 < x_1 < b_2 < x_2 \cdots < b_n < x_n$, the $n \times n$ matrix $A = [max\{x_i - b_j, 0\}]_{ij}$ has full rank. Its smallest eigenvalue is $min_i\{x_i - b_i\}$[3]

[3] Slides for this proof from Aldo Lamarre

Proof

For weight vectors $w, b \in \mathbb{R}^n$ and $a \in \mathbb{R}^d$, consider the function $c : \mathbb{R}^n \to \mathbb{R}$,

$$c(x) = \sum_{j=1} w_j \, max\{a^T x - b_j, 0\}$$

Proof

For weight vectors $w, b \in \mathbb{R}^n$ and $a \in \mathbb{R}^d$, consider the function $c : \mathbb{R}^n \to \mathbb{R}$,

$$c(x) = \sum_{j=1} w_j \, max\{a^T x - b_j, 0\}$$

- This can be done trivially with a depth 2 neural network with relu.

Proof

For weight vectors $w, b \in \mathbb{R}^n$ and $a \in \mathbb{R}^d$, consider the function $c : \mathbb{R}^n \to \mathbb{R}$,

$$c(x) = \sum_{j=1} w_j \, max\{a^T x - b_j, 0\}$$

- Now, fixing a sample $S = z_1, \dots, z_n$ of size n and a target vector $y \in R_n$. We need to find weights a, b, w so that $y_i = c(z_i)$ for all $i \in \{1, \dots, n\}$

Proof

For weight vectors $w, b \in \mathbb{R}^n$ and $a \in \mathbb{R}^d$, consider the function $c : \mathbb{R}^n \to \mathbb{R}$,

$$c(x) = \sum_{j=1} w_j \, max\{a^T x - b_j, 0\}$$

- First, choose a and b such that with $\xi_i = a^T z_i$ we have the interleaving property $b_1 < x_1 < b_2 < \cdots < b_n < x_n$ Next, consider the set of n equations in the n unknowns w,

  $$y_i = c(z_i), i \in \{1, \ldots, n\}$$

  We have $c(z_i) = Aw$, where $A = [max\{\xi_i - b_i, 0\}]_{ij}$ is the matrix of Lemma 1.

Proof

For weight vectors $w, b \in \mathbb{R}^n$ and $a \in \mathbb{R}^d$, consider the function $c : \mathbb{R}^n \to \mathbb{R}$,

$$c(x) = \sum_{j=1} w_j \, max\{a^T x - b_j, 0\}$$

- Now, fixing a sample $S = z_1, \ldots, z_n$ of size n and a target vector $y \in R_n$. We need to find weights a, b, w so that $y_i = c(z_i)$ for all $i \in \{1, \ldots, n\}$
- We chose a and b so that the lemma applies and hence A has full rank. We can now solve the linear system $y = Aw$ to find suitable weights w.

Discussion

Putting it together...

To recap:

1. We just showed that generically large neural networks can express any labelling of the training data
2. And so it is not surprising to see that networks learn the training data perfectly...
3. but it is surprising that we can't explain well why they don't just memorize all the time!

Any Questions?

References

- D. Sowinski, "What is generalization in machine learning?." Post.

- R. Novak, Y. Bahri, D. A. Abolafia, J. Pennington, and J. Sohl-Dickstein, "Sensitivity and Generalization in Neural Networks: an Empirical Study," *ArXiv e-prints*, Feb. 2018.

- M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," *CoRR*, vol. abs/1509.01240, 2015.

- D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien, "A Closer Look at Memorization in Deep Networks," *ArXiv e-prints*, June 2017.

- C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *CoRR*, vol. abs/1611.03530, 2016.