A Survey of the Methods for Deriving Bounds on the Generalization Error of Deep Neural Networks

Thomas Walker

Contents

Affiliation Imperial College London
Correspondence thomas.walker21@imperial.ac.uk
Date July 2023

Part I
Research on the Generalization of Neural Networks

1 Introduction

Generalization is the ability to effectively extrapolate concepts in a manner that is consistent with unseen data. For a deep neural network, this means that it can learn representations of a set of training data that not only captures the information in the training data but also enables it to make good inferences on unseen data. The capacity of deep neural networks to generalize to unseen data has been a major focus of enquiry. Over the last few years, investigations have been conducted to understand why it is that deep neural networks do not overfit training samples despite having a far greater number of parameters. Some investigations approach the problem from a theoretical perspective, whilst others take empirical approaches. From the theoretical perspective, assumptions are made about the underlying network that facilitates the application of mathematical arguments to explain this phenomenon. Although this approach provides enlightenment of the underlying processes, often the assumptions are rather strict and lead to results that are vacuous in practice. On the other hand, empirical work tries to make sense of this by performing controlled experiments. The results from these are largely correlational and are not grounded like the theoretical results, however, they provide insight into how this effect can be exploited and amplified to construct motivated networks with enhanced performance.

Generalization error for a deep neural network is the difference in the performance of the network on the training data and on unseen test data. In this survey, a rough chronological path is taken through the directions taken to yield bounds on the generalization error. Both theoretical and empirical investigations have led to quantitative and qualitative insights into the ability of deep neural networks to learn from training samples in a manner that generalizes well to unseen data. Throughout the survey bounds will be tested and explored in the simplified setting of fully connected ReLU neural networks, with the accompanying code being available here. The bounds will be applied for neural networks performing classification tasks. Networks will receive training data, from which it is to learn a classification function that accurately partitions the data into their respective classes.

Fully connected neural networks consist of stacked layers of hidden units, where each hidden unit is connected to every hidden unit in the previous layer and to every hidden unit in the subsequent layer. The parameters of the neural networks dictate how connected hidden units interact. Usually, each connection between hidden units has an associated weight and bias parameter. Inputs at a hidden unit are propagated to the next layer by multiplying the weight parameter and adding the bias parameter to the input value along each of the branching connections. The value at a hidden unit is the value of an activation function applied to the sum of the manipulated inputs of the previous

layer that are incident to that particular unit. Typically, the activation function is non-linear which increases the network's ability to learn complex functions.
The learning algorithm will be dictated by a loss function, which is a function designed to quantify the quality of the network. Gradient-based methods are the primary methodology used to train networks set up in this way. The loss function defines a hyper-surface in a space of dimension equal to that of the number of parameters of the network. Finding the global minimum of this surface is desirable as it corresponds to the set of parameters that maximizes a particular notion of quality. Through training the parameters of the network are altered to try and reach the lower regions of the loss landscape. Gradient-based methods do this by observing how the loss changes with respect to perturbations in the parameters at different training examples to determine the direction in which the parameters should be manipulated to move down the loss landscape. Computing the gradients at each individual training example is computationally expensive, and so instead gradients are calculated for batches of examples. These batches reduce computational costs but also introduce some noise into the algorithm that can be useful in helping the algorithm escape local minima of landscape and find more stable regions. This procedure is called Stochastic Gradient Descent (SGD) and has a central role in endowing neural networks with their generalization capacities.
Throughout the survey this setup will be formalized setup and investigated rigorously to get probabilistic bounds on how the learned function performs on test data, properties of it will be exploited to compress networks whilst still maintaining performance, and ideas such as Information theory can be applied to gain insight to the underlying processes of the learning algorithm. Empirically metrics will be devised to give a heuristic on how the network is generalizing through the training procedure and they will be used to provide insight into how the gradient-based learning algorithms lead to effective learned representations of the training data.

2 Stochastic Gradient Descent

The architecture of deep neural networks was proposed long before they manifested as a useful machine learning technique. This delay was due in part to the difficulty in training the large architectures in a stable and effective manner. Learning algorithms such as Stochastic Gradient Descent (SGD) have extracted remarkable properties from deep neural network architectures. Many of the properties are still mysterious to researchers, and these architectures seem to have greater potential than what was previously thought. To try and grapple with this it is important to understand the precise mechanisms of SGD as this has instantiated the networks with the majority of these properties. It has been observed that SGD is able to train these networks in the over-parameterized setting such that they converge to global minima of the loss landscape. In [12] an attempt is made to explain this using dynamic stability. This approach illustrates how the randomness induced by SGD is vital, and why regular gradient descent (GD) is not an effective learning algorithm for training neural networks.

As is the case with most machine learning scenarios, one is trying to minimize the training error

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x),$$

where each $f_i(x)$ can be thought of as the loss of the i^{th} example of a training set at the parameter value x. A general optimizer for this problem can be written as

$$x_{t+1} = x_t - G(x_t; \xi_t), \tag{1}$$

where ξ_t is a random variable independent of x_t and each ξ_t are i.i.d. Note that

- for GD, $G(x_t; \xi_t) = \eta \nabla f_{\xi_t}(x_t)$, and
- for SGD, $G(x_t; \xi_t) = \frac{\eta}{n}\sum_{i=1}^{n} \nabla f_i(x_t)$.

Definition 2.1 *Call x^* a fixed point (1) if for any ξ it follows that $G(x^*, \xi) = 0$.*

Definition 2.2 *Let x^* be a fixed point of (1). For the linearized dynamical system,*

$$\tilde{x}_{t+1} = \tilde{x}_t - A_{\xi_t}(\tilde{x}_t - x^*)$$

where $A_\xi = \nabla_x G(x^*, \xi_t)$. The fixed point x^* is linearly stable if there exists a constant C such that

$$\mathbb{E}\left(\|\tilde{x}_t\|^2\right) \leq C\,\|\tilde{x}_0\|^2$$

for all $t > 0$.

If it is assume that $f(x^*) = 0$, and the approximation

$$f(x) \approx \frac{1}{2n}\sum_{i=1}^{n} (x - x^*)^\top H_i (x - x^*)$$

with $H_i \nabla^2 f_i(x^*)$, is used then the linearized SGD is given by

$$x_{t+1} = x_t - \frac{\eta}{B}\sum_{j=1}^{B} H_{\xi_j}(x_t - x^*).$$

Where B is the batch size and $\xi = \{\xi_1, \ldots, \xi_B\}$ is a uniform, non-replaceable random sampling of size B on $\{1, \ldots, n\}$.

Definition 2.3 *Let $H = \frac{1}{n}\sum_{i=1}^{n} H_i$ and $\Sigma = \frac{1}{n}\sum_{i=1}^{n} H_i^2 - H^2$. Let $a = \lambda_{\max}(H)$ be the sharpness, and $s = \lambda_{\max}\left(\Sigma^{\frac{1}{2}}\right)$ be the non-uniformity.*

Theorem 2.4 *The global minimum x^* is linearly stable for SGD with learning rate η and batch size B if the following is satisfied*

$$\lambda_{\max}\left((1-\eta H)^2 + \frac{\eta^2(n-B)}{B(n-1)}\Sigma\right) \leq 1.$$

For $d = 1$ this is a necessary and sufficient condition.

Remark 2.5 *A simpler necessary condition is*

$$0 \leq a \leq \frac{2}{\eta}, \text{ and } 0 \leq s \leq \frac{1}{\eta}\sqrt{\frac{B(n-1)}{n-B}}. \tag{2}$$

For a fixed learning rate η,

1. GD can converge to minima satisfying $a \leq \frac{2}{\eta}$, and
2. SGD can converge to minima satisfying (2).

Hence, SGD can filter out minima with large non-uniformity. The difference between GD and SGD is that SGD must converge to solutions that fit the data uniformly well.

3 Bayesian Machine Learning

Here we will outline an introduction to Bayesian learning given by [14]. In Part *II* of this survey PAC-Bayes bounds are developed from the Bayesian framework of machine learning. Suppose that the data is of the form $\mathcal{D}_n = (X_i, Y_i)_{i=1}^n$ where each sample is a realisation of some random variables $(X, Y) \in \mathbb{R}^d \times \mathbb{R}$ with underlying distribution $\mathbb{P}$. Bayesian machine learning is used to determine a predictor $\hat{\phi}$ such that $\hat{\phi}(X') \approx Y'$. A learning algorithm is a map from data to predictors, we will denote the set of possible predictors $\mathcal{F}$. To assess the quality of predictors we define a loss function $l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ which quantifies the discrepancy between the ground truth label to the label given by the predictor. Define the risk of a predictor $\hat{\phi}$ as

$$R : \hat{\phi} \mapsto \mathbb{E}_{\mathbb{P}}\left(l\left(\hat{\phi}(X), Y\right)\right).$$

However, $\mathbb{P}$ is usually unknown, therefore in practice, we use the empirical risk which is computable,

$$r_n : \hat{\phi} = \frac{1}{n}\sum_{i=1}^{n} l\left(\hat{\phi}(X_i), Y_i\right).$$

Note that $\mathbb{E}\left(r_n\left(\hat{\phi}\right)\right) = R\left(\hat{\phi}\right)$. Under the Bayesian framework, we assign a prior distribution π on $\mathcal{F}$ for $\hat{\phi}$. The posterior distribution ρ then takes the form

$$\rho\left(\hat{\phi}|\mathcal{D}_n\right) \propto \mathcal{L}\left(\mathcal{D}_n|\hat{\phi}\right)\pi\left(\hat{\phi}\right),$$

where $\mathcal{L}$ is the likelihood function. From here there are multiple methodologies of choosing $\hat{\phi}$ from the posterior distribution ρ. For example, one may take $\hat{\phi}$ to be the mean, median or a random realisation of P.

4 Complexity Measures

It is generally accepted that having a more straightforward function that correctly classifies a dataset is more likely to generalize well to unseen data. The generalization bounds we will discuss will often reward simpler models, where the definition of simple may vary in different contexts. A lot of work in this field tends to be empirical as heuristics are derived for complexity that can be monitored during training. Sometimes theoretically motivated complexity measures are proposed that can be used to derive explicit generalization bounds. An important complexity measure is Rademacher complexity, which forms the basis of one of the main results in generalization theory. Suppose we have a hypothesis class $\mathcal{H}$ (i.e. the possible neural networks defined by a particular set of hyper-parameters), and a training set S. For a loss function l, let $l \circ \mathcal{H} := \{l \circ h : h \in \mathcal{H}\}$.

Definition 4.1 ([9]) *Let $\mathcal{G}$ be a family of functions mapping from $\mathcal{Z}$ to $[a,b]$ and $S = (z_1, \ldots, z_m)$ a fixed sample of size m with elements in $\mathcal{Z}$. Then, the empirical Rademacher complexity of $\mathcal{G}$ with respect to the sample S is defined as:*

$$\mathfrak{R}_S(\mathcal{G}) = \frac{1}{m}\mathbb{E}_{\xi\in\{\pm1\}^m}\left(\sup_{f\in\mathcal{G}}\sum_{i=1}^{m}\xi_i f(x_i)\right).$$

Definition 4.2 ([9]) *Let $\mathcal{D}$ be a distribution from which samples are drawn. For any integer $m \geq 1$, the Rademacher complexity of a family of functions $\mathcal{G}$ is the expectation of the empirical Rademacher complexity over all samples of size m drawn from $\mathcal{D}$. That is,*

$$\mathfrak{R}_m(\mathcal{G}) = \mathbb{E}_{S\sim\mathcal{D}^m}\left(\hat{\mathfrak{R}}_S(\mathcal{G})\right).$$

Theorem 4.3 ([9]) *Let $\mathcal{G}$ be a family of functions mapping from $\mathcal{Z}$ to $[0,1]$. Then for any $\delta > 0$ with probability $1-\delta$ over the draw of an* i.i.d *sample S of size m then,*

$$\mathbb{E}(g(z)) \leq \frac{1}{m}\sum_{i=1}^{m} g(z_i) + 2\mathfrak{R}_m(\mathcal{G}) + \sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{2m}}, \text{ and}$$

$$\mathbb{E}(g(z)) \leq \frac{1}{m}\sum_{i=1}^{m} g(z_i) + 2\hat{\mathfrak{R}}_m(\mathcal{G}) + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2m}}.$$

holds for all $g \in \mathcal{G}$.

Rademacher complexity is a complexity measure that yields theoretical bounds on the expected error of the function of our network. By understanding our hypothesis class can separate the data points of our training set. Subsequent work aims to develop a tractable heuristic for this measure. Refer to [23] to see how Rademacher complexity can be bounded by VC dimension. The setup is as follows, we are performing binary classification on input data $\mathcal{X} \subseteq \mathbb{R}^d$ and labels $\mathcal{Y} = \{\pm1\}$. With $\mathcal{A} \subseteq \mathcal{B} = \{a : \mathcal{X} \to \{\pm1\}\}$ let

$$\mathcal{A} \circ \{x_1, \ldots, x_n\} = \left\{(a(x_1), \ldots, a(x_n)) \in \{\pm1\}^n : a \in \mathcal{A}\right\}.$$

Note that $\mathcal{A} \circ x$ is finite even if $\mathcal{A}$ is infinite.

Definition 4.4 *The growth function of $\mathcal{A}$ is defined as*

$$\tau_{\mathcal{A}}(n) := \sup_{x \in \mathcal{X}} |\mathcal{A} \circ x|$$

for any integer $n \geq 1$.

That is, $\tau_{\mathcal{A}}$ is the maximal cardinality of the set of distinct labelling of n points in $\mathcal{X}$ obtained using classifiers from $\mathcal{A}$.

Proposition 4.5 *For any $x = \{x_1, \ldots, x_n\} \in \mathcal{X}^n$ we have*

$$\mathfrak{R}(\mathcal{A} \circ x) \leq \sqrt{\frac{2 \log(\tau_{\mathcal{A}}(n))}{n}}.$$

Note that $\tau_{\mathcal{A}}(n) \leq 2^n$.

Definition 4.6 *The Vapnik-Chervonenkis (VC) dimension of $\mathcal{A}$ is the largest integer n such that $\tau_{\mathcal{A}}(n) = 2^n$,*

$$\mathrm{VC}(\mathcal{A}) := \max \left\{ n \in \mathbb{N} : \tau_{\mathcal{A}}(n) = 2^n \right\}$$

The quantities $\tau_{\mathcal{A}}(1), \tau_{\mathcal{A}}(2), \ldots$ are known as shatter coefficients, and we say that $\mathcal{A}$ shatters $\{x_1, \ldots, x_n\}$ if $|\mathcal{A} \circ \{x_1, \ldots, x_n\}| = 2^n$. Hence, VC dimension is the maximum number of different elements that can be shattered by $\mathcal{A}$.

Proposition 4.7 *For any $x = \{x_1, \ldots, x_n\} \in \mathcal{X}^n$ we have*

$$\mathfrak{R}(\mathcal{A} \circ x) \leq \sqrt{\frac{2\mathrm{VC}(\mathcal{A}) \log\left(\frac{en}{\mathrm{VC}(\mathcal{A})}\right)}{n}}.$$

This is a data-independent bound that holds for any $x \in \mathcal{X}^n$. It can be improved through a technique known as chaining. Chaining will not be explored for this particular setting, however, we will see how it can be applied to improve our information-theoretic bounds.

Part II
PAC-Bayes Bounds

5 Introduction

Bayesian machine learning is a framework for managing randomness and uncertainty in learning tasks. It allows us to deduce Probably Approximately Correct (PAC) bounds, which bound the performance of a network with high probabilities [14]. The first bounds were given by [1] in a time when deep neural networks were not a mainstream technique in the field of machine learning. In this part of the survey, we investigate how this preliminary work has been contextualised into modern deep neural networks to generate tightened bounds that prove useful in practice. For example, [4] exploits the properties of stochastic gradient descent (SGD) to establish PAC-Bayes bounds that are non-vacuous.
Later on, we will understand how the over-parameterization of deep neural networks leads to compression frameworks [7] that can be used to generate tight PAC-Bayes that hold over compressed networks [15].

6 Optimizing PAC-Bayes Bounds via SGD

Suppose the training data is $S_m = \{(x_i, y_i)\}_{i=1}^m$ where $x_i \in \mathcal{X} \subseteq \mathbb{R}^k$ are features and $y_i \in \{\pm 1\}$ is the associated label. Consider a set $\mathcal{M}$ of probability measures on the space $\mathbb{R}^k \times \{\pm 1\}$, and assume the training data is an i.i.d sample from some $\mu \in \mathcal{M}$. The hypothesis space is the set of classifiers

$$\mathcal{H} = \left\{h_w = H(w, \cdot) : \mathbb{R}^k \to \{\pm 1\} : w \in \mathbb{R}^d\right\}.$$

The loss function $l : \mathbb{R} \times \{\pm 1\} \to \{0, 1\}$ is the 0-1 loss where $l(y, \hat{y}) = \mathbb{I}(\text{sgn}(\hat{y}) = y)$. For optimization purposes the convex surrogate loss function $\tilde{l} : \mathbb{R} \times \{\pm 1\} \to \mathbb{R}_+$ is used

$$\tilde{l}(y, \hat{y}) = \frac{\log\left(1 + \exp\left(-\hat{y}\, y\right)\right)}{\log(2)}.$$

The associated errors derived from these loss functions are:

- $\hat{e}(h, S_m) = \frac{1}{m}\sum_{i=1}^m l(h(x_i), y_i)$, the empirical classification error of hypothesis h on S_m.
- $\tilde{e}(h, S_m) = \frac{1}{m}\sum_{i=1}^m \tilde{l}(h(x_i), y_i)$, the empirical error of a hypothesis h on S_m,
- $e_\mu(h) = \mathbb{E}_{S_m \sim \mu^m}\left(\hat{e}(h, S_m)\right)$, the expected error for hypothesis h under μ,
- $\hat{e}(Q, S_m) = \mathbb{E}_{w \sim Q}\left(\hat{e}(h_w, S_m)\right)$, the expected empirical error under randomized classifier Q on $\mathcal{H}$, and
- $e(Q) = \mathbb{E}_{w \sim Q}(e_\mu(h_w))$, the expected error for Q on $\mathcal{H}$.

6.1 Kullback-Liebler Divergence

The Kullback-Liebler (KL) divergence is a measure of similarity between probability measures defined on the same measurable space.

Definition 6.1 *Let Q, P be probability measures on Euclidean space $\mathbb{R}^d$ with densities q, p respectively. The Kullback-Liebler (KL) divergence of P from Q is*

$$\mathrm{KL}(Q, P) := \int \log\left(\frac{q(x)}{p(x)}\right) q(x)dx.$$

Remark 6.2 *Note the asymmetry in the definition.*

For the multivariate normal distributions $N_q \sim \mathcal{N}(\mu_q, \Sigma_q)$ and $N_p \sim \mathcal{N}(\mu_p, \Sigma_p)$ defined on $\mathbb{R}^d$ it follows that,

$$\mathrm{KL}(N_q, N_p) = \frac{1}{2}\left(\mathrm{tr}\left(\Sigma_p^{-1}\Sigma_q\right) - d + (\mu_p - \mu_q)^\top \Sigma_p^{-1}(\mu_p - \mu_q) + \log\left(\frac{\det \Sigma_p}{\det \Sigma_q}\right)\right).$$

Similarly, for Bernoulli distributions $\mathcal{B}(q) \sim \mathrm{Bern}(q)$ and $\mathcal{B}(p) \sim \mathrm{Bern}(p)$ it follows that

$$\mathrm{KL}(q, p) := \mathrm{KL}(\mathcal{B}(q), \mathcal{B}(p)) = q\log\left(\frac{q}{p}\right) + (1-q)\log\left(\frac{1-q}{1-p}\right),$$

where in the first equality there is a slight abuse notation. For $p^* \in [0,1]$ bounds of the form $\mathrm{KL}(q, p^*) \leq c$ for some $q \in [0,1]$ and $c \geq 0$ are of interest. Hence, introduce the notation

$$\mathrm{KL}^{-1}(q, c) := \sup\{p \in [0,1] : \mathrm{KL}(q, p) \leq c\}.$$

6.2 Probabilistic Bounds

The following theorems give probabilistic bounds that will prove useful in generating probabilistic generalization bounds.

Theorem 6.3 *Let $E_1, E_2, \ldots$ be events. Then $\mathbb{P}\left(\bigcup_n E_n\right) \leq \sum_n \mathbb{P}(E_n)$.*

Theorem 6.4 *For every $p, \delta \in (0,1)$ and $n \in \mathbb{N}$, with probability at least $1-\delta$ over $\mathbf{x} \sim \mathcal{B}(p)^n$*

$$\mathrm{KL}\left(\frac{1}{n}\sum_{i=1}^n x_i, p\right) \leq \frac{\log\left(\frac{2}{\delta}\right)}{n}.$$

Theorem 6.5 *For every $\delta > 0, m \in \mathbb{N}$, distribution μ on $\mathbb{R}^k \times \{\pm 1\}$ and distribution P on $\mathcal{H}$, with probability at least $1-\delta$ over $S_m \sim \mu^m$, for all distributions Q on $\mathcal{H}$,*

$$\mathrm{KL}(\hat{e}(Q, S_m), e(Q)) \leq \frac{\mathrm{KL}(Q, P) + \log\left(\frac{m}{\delta}\right)}{m-1}.$$

Theorem 6.5 motivates the following PAC learning algorithm:

- Fix a probability $\delta > 0$ and a distribution P on $\mathcal{H}$,
- Collect an i.i.d sample S_m of size m,
- Compute the optimal distribution Q of $\mathcal{H}$ that minimizes

$$\mathrm{KL}^{-1}\left(\hat{e}(Q, S_m), \frac{\mathrm{KL}(Q, P) + \log\left(\frac{m}{\delta}\right)}{m-1}\right), \tag{3}$$

- Then return the randomized classifier given by Q.

6.3 PAC Generalization Bounds

For a parametric family of classifiers H let $h_w = H(w, \cdot) \in H$. Here w can represent the weights and biases with h_w being the corresponding neural network. Fix $\delta \in (0, 1)$, distribution P on $\mathbb{R}^d$, and i.i.d sample $S_m \sim \mu^m$. To minimize (3) with respect to Q. Consider a constrained setting so that the optimization is tractable. For $w \in \mathbb{R}^d$ and $s \in \mathbb{R}^d_+$ let $\mathcal{N}_{w,s} = \mathcal{N}(w, \mathrm{diag}(s))$. Apply the simple upper bound $\mathrm{KL}^{-1}(q, c) \leq q + \sqrt{\frac{c}{2}}$ to (3), replace the loss with the (convex surrogate) logistic loss and restrict Q to a multivariate normal distribution of the form $\mathcal{N}_{w,s}$ to obtain the optimization problem

$$\min_{w \in \mathbb{R}^d, s \in \mathbb{R}^d_+} \tilde{e}\left(\mathcal{N}_{w,s}, S_m\right) + \sqrt{\frac{\mathrm{KL}(\mathcal{N}_{w,s}, P) + \log\left(\frac{m}{\delta}\right)}{2(m-1)}}.$$

Consider prior P of the form $\mathcal{N}(0, \lambda I)$. To choose λ practically the problem is discretized at the cost of expanding the eventual generalization bound. Let λ have the form $c \exp\left(-\frac{j}{b}\right)$ for $j \in \mathbb{N}$, so that c is an upper bound and b controls precision. By ensuring Theorem 6.5 holds with probability $1 - \frac{6}{\pi^2 j^2}$ for each $j \in \mathbb{N}$ Theorem 6.3 applies to get results that hold for probability $1 - \delta$. Treat λ as continuous during optimization and then discretize at the point of evaluating the bound. Hence, replace j with $b \log\left(\frac{c}{\lambda}\right)$ during optimization to yields the minimization problem

$$\min_{w \in \mathbb{R}^d, s \in \mathbb{R}^d_+, \lambda \in (0, c)} \tilde{e}(\mathcal{N}_{w,s}, S_m) + \sqrt{\frac{1}{2} B_{\mathrm{RE}}(w, s, \lambda; \delta)}, \tag{4}$$

where

$$B_{\mathrm{RE}}(w, s, \lambda; \delta) = \frac{\mathrm{KL}(\mathcal{N}_{w,s}, \mathcal{N}(w_0, \lambda I)) + 2\log\left(b \log\left(\frac{c}{\lambda}\right)\right) + \log\left(\frac{\pi^2 m}{6\delta}\right)}{m-1}.$$

The goal is therefore to solve the optimization problem (4). As computing the gradient of $\tilde{e}(\mathcal{N}_{w,s}, S_m)$ is not feasible in practice the gradient of

$$\tilde{e}\left(h_{w+\xi \odot \sqrt{s}}, S_m\right)$$

where $\xi \sim \mathcal{N}_{0,I_d}$ is computed instead. Once good candidates for this optimization problem are found (how to do this will be outlined in Section 6.4) then return to (3) to calculate the final error bound. With the choice of λ it follows that with probability $1-\delta$, uniformly over all $w \in \mathbb{R}^d, s \in \mathbb{R}^d_+$ and λ (of the discrete form) the error rate of the random classifier $Q = \mathcal{N}_{w,s}$ is bounded by

$$\mathrm{KL}^{-1}(\hat{e}(Q, S_m), B_{\mathrm{RE}}(w, s, \lambda; \delta)).$$

It is often not possible to compute $\hat{e}(Q, S_m)$ and so instead an unbiased estimate is obtained by estimating Q. Given n i.i.d samples $w_1, \ldots, w_n$ from Q consider the Monte Carlo approximation $\hat{Q}_n = \sum_{i=1}^n \delta_{w_i}$, to get the bound

$$\hat{e}(Q, S_m) \leq \overline{\hat{e}_{n,\delta'}}(Q, S_m) := \mathrm{KL}^{-1}\left(\hat{e}\left(\hat{Q}_n, S_m\right), \frac{1}{n}\log\left(\frac{2}{\delta'}\right)\right),$$

which holds with probability $1-\delta'$. Finally, by Theorem 6.3

$$e(Q) \leq \mathrm{KL}^{-1}\left(\overline{\hat{e}_{n,\delta'}}(Q, S_m), B_{\mathrm{RE}}(w, s, \lambda; \delta)\right),$$

holds with probability $1-\delta-\delta'$.

6.4 Optimizing PAC Generalization Bounds

To ensure that the solution to (4) provides a bound that is non-vacuous the values of w, s, λ are optimized. To do this first train the network via SGD to get a value of w. Then instantiate a stochastic neural network with a multivariate normal distribution $Q = \mathcal{N}_{w,s}$ over the weights, with $s = |w|$ being the diagonal entries of the covariance matrix. Next apply Algorithm 1 to deduce the values of w, s and λ that give a tighter bound.

6.5 Details of the Optimization Process

To compute $\overline{\hat{e}_{n,\delta'}}(Q, S_m) := \mathrm{KL}^{-1}\left(\hat{e}\left(\hat{Q}_n, S_m\right), \frac{1}{n}\log\left(\frac{2}{\delta'}\right)\right)$. Note that

$$\hat{e}(\hat{Q}_n, S_m) = \sum_{i=1}^n \delta_{w_i}\left(\frac{1}{m}\sum_{j=1}^m l(h_{w_i}(x_j), y_j)\right).$$

Then as inverting KL divergence has no closed form use Newton's method to find an approximation, this is outlined in Algorithm 2. Code available here.

Algorithm 1 Optimizing the PAC Bounds

Require:
$w_0 \in \mathbb{R}^d$, the network parameters at initialization.
$w \in \mathbb{R}^d$, the network parameters after SGD.
S_m, training examples.
$\delta \in (0,1)$, confidence parameter.
$b \in \mathbb{N}, c \in (0,1)$, precision and bound for λ.
$\tau \in (0,1), T$, learning rate.
Ensure: Optimal w, s, λ.
$\zeta = \text{abs}(w)$ ▷ $s(\zeta) = e^{2\eta}$
$\rho = -3$ ▷ $\lambda(\rho) = e^{2\rho}$
$B(w, s, \lambda, w') = \tilde{e}(h_{w'}, S_m) + \sqrt{\frac{1}{2} B_{\text{RE}}(w, s, \lambda)}$
for $t = 1 \rightarrow T$ **do**
Sample $\xi \sim \mathcal{N}(0, I_d)$
$w'(w, \zeta) = w + \xi \odot \sqrt{s(\zeta)}$

$$\begin{pmatrix} w \\ \zeta \\ \rho \end{pmatrix} = -\tau \begin{pmatrix} \nabla_w B(w, s(\zeta), \lambda(\rho), w'(w, \zeta)) \\ \nabla_\zeta B(w, s(\zeta), \lambda(\rho), w'(w, \zeta)) \\ \nabla_\rho B(w, s(\zeta), \lambda(\rho), w'(w, \zeta)) \end{pmatrix}$$

end for
return $w, s(\zeta), \lambda(\rho)$

Algorithm 2 Newton's Method for Inverting KL Divergence

Require: q, c, initial estimate p_0 and $N \in \mathbb{N}$
Ensure: p such that $p \approx \text{KL}^{-1}(q, c)$
for $n = 1 \rightarrow N$ **do**
if $p \geq 1$ **then**
Return 1
else
$p_0 = p_0 - \frac{q \log\left(\frac{q}{c}\right) + (1-q) \log\left(\frac{1-q}{1-c}\right) - c}{\frac{1-q}{1-p} - \frac{q}{p}}$
end if
end for
return p_0

7 Compression

The discussion of the survey revolves around the observation that neural networks are instantiated with many more parameters than are required to overfit the data. However, it is seen empirically that trained networks do not overfit and with a high probability generalize well to unseen data. In classical statistical theory, only as many parameters as training samples are required to overfit, which suggests that some of the extra capacity of the network is redundant in expressing the learned function. In [7] compression frameworks are constructed that aims to reduce the effective number of parameters required to express the function of a trained network whilst maintaining its performance. By capitalizing on how the network responds to noise injected into the weights of the network, algorithms can be devised to compress the network. Consequently, generalization bounds can be proven by utilizing compression mechanisms. In this section, these methods are introduced by considering linear classifiers and then fully connected ReLU neural networks.
The setup is that of [7]. Consider some training data and a learned classifier f that has a high complexity but achieves a low empirical loss. The aim is to find a simpler classifier g that attains a similar empirical loss as f on the training data.

7.1 Establishing the Notion of Compression

Suppose a learned classifier f has high complexity but achieves a low empirical loss. Let $S_m = \{(x_i, y_i)\}_{i=1}^m$ be the training data with x_i being data points and $y_i \in \{1, \dots, k\}$. The multi-class classifier f maps x to $f(x) \in \mathbb{R}^k$. The classification loss for distribution $\mathcal{D}$ is given by $\mathbb{P}_{(x,y)\sim\mathcal{D}}\left(f(x)[y] \leq \max_{i\neq y} f(x)[i]\right)$ and the expected margin loss for margin $\gamma > 0$ is given by $L_\gamma(f) = \mathbb{P}_{(x,y)\sim\mathcal{D}}\left(f(x)[y] \leq \gamma + \max_{i\neq y} f(x)[i]\right)$. Now $\mathcal{D}$ is often unknown so the empirical margin loss $\hat{L}_\gamma$ is used as a proxy. Use d to denote the number of fully connected layers in the network and x^i for the vector before the activation at layer $i = 0, \dots, d$ and as x^0 is the input it x. The network will have A^i as the weight matrix at layer i and h^i hidden units at layer i, let $h = \max_{i=1}^d h^i$. The function calculated by the network will be denoted $f_A(x)$. For layers $i \leq j$ the operator for the composition of the layers is denoted $M^{i,j}$, the Jacobian of the input x is denoted $J_x^{i,j}$ and $\phi(\cdot)$ is component-wise ReLU then the following hold,

$$x^i = A^i\phi\left(x^{i-1}\right),\ x^j = M^{i,j}\left(x^i\right),\ \text{and } M^{i,j}\left(x^i\right) = J_{x^i}^{i,j}x^i.$$

Note that for a matrix B, $\|B\|_F$ is the Frobenius norm, $\|B\|_2$ is the spectral norm and $\frac{\|B\|_F^2}{\|B\|_2^2}$ is its stable rank.

Definition 7.1 *Let f be a classifier and $G_{\mathcal{A}} = \{g_A : A \in \mathcal{A}\}$ be a class of classifiers. Say that f is (γ, S)-compressible via $G_{\mathcal{A}}$ if there exists $A \in \mathcal{A}$ such that for any $x \in S$,*

$$|f(x)[y] - g_A(x)[y]| \leq \gamma$$

for all y.

Definition 7.2 *Suppose $G_{\mathcal{A},s} = \{g_{A,s} : A \in \mathcal{A}\}$ is a class of classifiers indexed by trainable parameters A and fixed string s. A classifier f is (γ, S)-compressible with respect to $G_{\mathcal{A},s}$ using helper string s if there exists $A \in \mathcal{A}$ such that for any $x \in S$,*

$$|f(x)[y] - g_{A,s}(x)[y]| \leq \gamma$$

for all y.

Theorem 7.3 *Suppose $G_{\mathcal{A},s} = \{g_{A,s} : A \in \mathcal{A}\}$ where A is a set of q parameters each of which has at most r discrete values and s is a helper string. Let S be a training set with m samples. If the trained classifier f is (γ, S)-compressible via $G_{\mathcal{A},s}$ with helper string s, then there exists $A \in \mathcal{A}$ with high probably such that*

$$L_0(g_A) \leq \hat{L}_\gamma(f) + O\left(\sqrt{\frac{q \log(r)}{m}}\right)$$

over the training set.

Remark 7.4 *The theorem only gives a bound for g_A which is the compression of f. However, there are no requirements of a hypothesis class, assumptions are only made on f and its properties on a finite training set.*

Corollary 7.5 *If the compression works for $1 - \xi$ fraction of the training sample, then with a high probability*

$$L_0(g_A) \leq \hat{L}_\gamma(f) + \xi + O\left(\sqrt{\frac{q \log r}{m}}\right).$$

7.2 Compression of a Linear Classifier

The compression framework can be used to compress the decision vector of a linear classifier. Let $c \in \mathbb{R}^h$ be a decision vector that classifies an input x as $\operatorname{sgn}(c^\top x)$. Let $\mathcal{D}$ be a distribution on the inputs (x, y) where $\|x\| = 1$ and $y = \{\pm 1\}$. Define the margin, γ, of c to be such that $y\,(c^\top x) \geq \gamma$ for all (x, y) in the training set. To the vector c apply Algorithm 3.

Algorithm 3 (γ, c)

Require: vector c with $\|c\| \leq 1$, η.

Ensure: vector $\hat{c}$ such that for any fixed vector $\|u\| \leq 1$, with probability at least $1 - \eta$, $|c^\top u - \hat{c}^\top u| \leq \gamma$.

Vector $\hat{c}$ has $O\left(\frac{\log h}{\eta \gamma^2}\right)$ non-zero entries.

for $i = 1 \to d$ **do**

Let $z_i = 1$ with probability $p_i = \frac{2c_i^2}{\eta \gamma^2}$ and 0 otherwise.

Let $\hat{c}_i = \frac{z_i c_i}{p_i}$.

end for

return $\hat{c}$

Lemma 7.6 *Algorithm 3* (γ, c) *returns a vector* $\hat{c}$ *such that for any fixed* u*, with probability* $1-\eta$*,* $|\hat{c}^\top - c^\top u| \leq \gamma$*. The vector* $\hat{c}$ *has at most* $O\left(\frac{\log h}{\eta\gamma^2}\right)$ *non-zero entries with high probability.*

In the discrete case, a similar result holds.

Lemma 7.7 *Let Algorithm 3* $\left(\frac{\gamma}{2}, c\right)$ *return vector* $\tilde{c}$*. Let,*

$$\hat{c}_i = \begin{cases} 0 & |\tilde{c}_i| \geq 2\eta\gamma\sqrt{h} \\ \textit{rounding to nearest multiple of } \frac{\gamma}{2\sqrt{h}} & \textit{Otherwise.} \end{cases}$$

Then for any fixed u *with probability at least* $1-\eta$*,* $|\hat{c}^\top u - c^\top u| \leq \gamma$*.*

Using Lemmas 7.6 and 7.7 along with Corollary 7.5 the following holds.

Lemma 7.8 *For any number of samples* m*, there is an efficient algorithm to generate a compressed vector* $\hat{c}$*, such that*

$$L(\hat{c}) \leq \tilde{O}\left(\left(\frac{1}{\gamma^2 m}\right)^{\frac{1}{3}}\right).$$

Remark 7.9 *The rate is not optimal as it depends on* $m^{1/3}$ *and not* $\sqrt{m}$*. To resolve this employ helper strings.*

Algorithm 4 (γ, c)

Require: vector c with $\|c\| \leq 1$, η.

Ensure: vector $\hat{c}$ such that for any fixed vector $\|u\| \leq 1$, with probability at least $1-\eta$, $|c^\top u - \hat{c}^\top u| \leq \gamma$.

Let $k = \frac{16\log\left(\frac{1}{\eta}\right)}{\gamma^2}$.

Sample the random vectors $v_1, \ldots, v_k \sim \mathcal{N}(0, I)$.

Let $z_i = \langle v_i, c\rangle$.

(In Discrete Case) Round z_i to closes multiple of $\frac{\gamma}{2\sqrt{hk}}$.

return $\hat{c} = \frac{1}{k}\sum_{i=1}^{k} z_i v_i$

Remark 7.10 *The vectors* v_i *are the helper string.*

Lemma 7.11 *For any fixed vector* u*, Algorithm 4* (γ, c) *returns a vector* $\hat{c}$ *such that with probability at least* $1-\eta$*, we have* $|\hat{c}^\top - c^\top u| \leq \gamma$*.*

Lemma 7.12 *For any number of sample* m*, there is an efficient algorithm with helper string to generate a compressed vector* $\hat{c}$*, such that*

$$L(\hat{c}) \leq \tilde{O}\left(\sqrt{\frac{1}{\gamma^2 m}}\right).$$

Code that explores these examples can be found here.

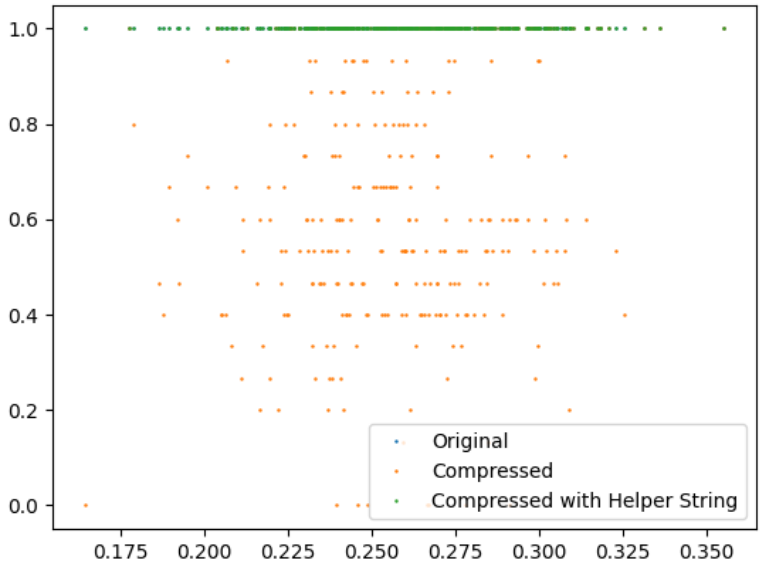

Figure 1: The test error of a linear classifier before and after applying compression with and without helper strings.

7.3 Compression of a Fully Connected Network

In a similar way, the layer matrices of a fully connected network can be compressed in such a way as to maintain a reasonable level of performance. A compression method is detailed in Algorithm 5:

Algorithm 5 (A, ϵ, η)

Require: Layer matrix $A \in \mathbb{R}^{h_1 \times h_2}$, error parameters ϵ, η.
Ensure: Returns $\hat{A}$ such that for all vectors u, v we have that

$$\mathbb{P}\left(\left|u^\top \hat{A} v - u^\top A v\right| \geq \epsilon \|A\|_F \|u\| \|v\|\right) \leq \eta$$

Sample $k = \frac{\log(\frac{1}{\eta})}{\epsilon^2}$ random matrices $M_1, \ldots, M_k$ with i.i.d entries ± 1.
for $k' = 1 \to k$ **do**
  Let $Z_{k'} = \langle A, M_{k'} \rangle M_{k'}$
end for
return $\hat{A} = \frac{1}{k} \sum_{k'=1}^{k} Z_{k'}$

Definition 7.13 *If M is a mapping from real-valued vectors to real-valued vectors, and $\mathcal{N}$ is a noise distribution. Then the noise sensitivity of M at x with respect to $\mathcal{N}$ is*

$$\psi_{\mathcal{N}}(M, x) = \mathbb{E}\left(\frac{\|M(x + \eta\|x\|) - M(x)\|^2}{\|M(x)\|^2}\right),$$

and let $\psi_{\mathcal{N},S}(M) = \max_{x \in S} \psi_{\mathcal{N}}(M, x)$.

Remark 7.14 *When $x \neq 0$ and the noise distribution is the Gaussian distribution $\mathcal{N}(0, I)$ then the noise sensitivity of matrix M is exactly $\frac{\|M\|_F^2}{\|Mx\|^2}$. Hence, it is at most the stable rank of M.*

Definition 7.15 *The layer cushion of layer i is defined as the largest μ_i such that for any $x \in S$*

$$\mu_i \left\|A^i\right\|_F \left\|\phi\left(x^{i-1}\right)\right\| \leq \left\|A^i \phi\left(x^{i-1}\right)\right\|.$$

Remark 7.16 *Note that $\frac{1}{\mu_i^2}$ is equal to the noise sensitivity of A^i at $\phi\left(x^{i-1}\right)$ with respect to noise $\eta \sim \mathcal{N}(0, I)$.*

Definition 7.17 *For layers $i \leq j$ the inter-layer cushion $\mu_{i,j}$ is the largest number such that*

$$\mu_{i,j} \left\|J_{x^i}^{i,j}\right\|_F \left\|x^i\right\| \leq \left\|J_{x^i}^{i,j} x^i\right\|$$

for any $x \in S$. Furthermore, let $\mu_{i\to} = \min_{i \leq j \leq d} \mu_{i,j}$.

Remark 7.18 *Note that $J_{x^i}^{i,i} = I$ so that*

$$\frac{\left\|J_{x^i}^{i,i} x^i\right\|}{\left\|J_{x^i}^{i,j}\right\|_F \|x^i\|} = \frac{1}{\sqrt{h^i}}.$$

Furthermore, $\frac{1}{\mu_{i,j}^2}$ is the noise sensitivity of $J_x^{i,j}$ with respect to noise $\eta \sim \mathcal{N}(0, I)$.

Definition 7.19 *The activation contraction c is defined as the smallest number such that for any layer i*

$$\left\|\phi\left(x^i\right)\right\| \geq \frac{\left\|x^i\right\|}{c}$$

for any $x \in S$.

Definition 7.20 *Let η be the noise generated as a result of applying Algorithm 5 to some of the layers before layer i. Define the inter-layer smoothness ρ_δ to be the smallest number such that with probability $1 - \delta$ we have that for layers $i < j$*

$$\left\|M^{i,j}\left(x^i + \eta\right) - J_{x^i}^{i,j}\left(x^i + \eta\right)\right\| \leq \frac{\|\eta\| \left\|x^j\right\|}{\rho_\delta \left\|x^i\right\|}$$

for any $x \in S$.

Lemma 7.21 *For any $0 < \delta, \epsilon \leq 1$ let $G = \left\{\left(U^i, x^i\right)\right\}_{i=1}^m$ be a set of matrix-vector pairs of size m where $U \in \mathbb{R}^{n \times h_1}$ and $x \in \mathbb{R}^{h_2}$, let $\hat{A} \in \mathbb{R}^{h_1 \times h_2}$ be the output of Algorithm 5 $\left(A, \epsilon, \eta = \frac{\delta}{mn}\right)$. With probability at least $1 - \delta$ we have for any $(U, x) \in G$ that $\left\|U(\hat{A} - A)x\right\| \leq \epsilon \|A\|_F \|U\|_F \|x\|$.*

Remark 7.22 *For a neural network let x be the input, A be the layer matrix and U the Jacobian of the network output with respect to the layer input. Then the network output before compression is given by UAx and after compression the output is given by $U\hat{A}x$.*

Use Lemma 7.21 to prove the following.

Lemma 7.23 *For any fully connected network f_A with $\rho_\delta \geq 3d$, any probability $0 < \delta \leq 1$ and any $0 < \epsilon \leq 1$, Algorithm 5 can generate weights $\tilde{A}$ for a network with*

$$\frac{72c^2d^2 \log\left(\frac{mdh}{\delta}\right)}{\epsilon^2} \cdot \sum_{i=1}^{d} \frac{1}{\mu_i^2 \mu_{i\rightarrow}^2}$$

total parameters such that with probability $1 - \frac{\delta}{2}$ over the generated weights $\tilde{A}$, for any $x \in S$

$$\left\|f_A(x) - f_{\tilde{A}}(x)\right\| \leq \epsilon \|f_A(x)\|,$$

where $\mu_i, \mu_{i\rightarrow}, c$ and ρ_δ are the layer cushion, inter-layer cushion, activation contraction and inter-layer smoother for the network.

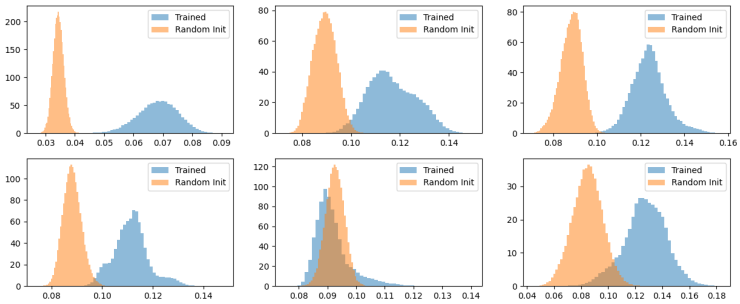

Figure 2: The cushion values for different layers of a fully connected network before and after training.

Subsequently, the following result holds.

Lemma 7.24 *For any fully connected network f_A with $\rho_\delta \geq 3d$, any probability $0 < \delta \leq 1$ and any margin $\gamma > 0$, f_A can be compressed (with respect to a random string) to another fully connected network $f_{\tilde{A}}$ such that for $x \in S$, $\hat{L}_0(f_{\hat{A}}) \leq \hat{L}_\gamma(f_A)$ and the number of parameters in $f_{\tilde{A}}$ is at most*

$$\tilde{O}\left(\frac{c^2 d^2 \max_{x \in S} \|f_A(x)\|_2^2}{\gamma^2} \sum_{i=1}^{d} \frac{1}{\mu_i^2 \mu_{i\rightarrow}^2}\right).$$

Lemma 7.25 *Let f_A be a d-layer network with weights $A = \{A^1, \ldots, A^d\}$. Then for any input x, weights A and $\hat{A}$, if for any layer i, $\|A^i - \hat{A}^i\| \leq \frac{1}{d}\|A^i\|$, then*

$$\|f_A(x) - f_{\hat{A}}(x)\| \leq e\|x\| \left(\prod_{i=1}^{d} \|A^i\|_2\right) \sum_{i=1}^{d} \frac{\|A^i - \hat{A}^i\|_2}{\|A^i\|_2}$$

Using Lemmas 7.23,7.24 and 7.25 it can be shown that Algorithm 5 provides a generalization bound for suitable fully connected neural networks.

Theorem 7.26 *For any fully connected network f_A with $\rho_\delta \geq 3d$, any probability $0 < \delta \leq 1$ and any margin γ. Algorithm 5 generates weights $\tilde{A}$ such that with probability $1 - \delta$ over the training set,*

$$L_0(f_{\tilde{A}}) \leq \hat{L}_\gamma(f_A) + \tilde{O}\left(\sqrt{\frac{c^2 d^2 \max_{x \in S} \|f_A(x)\|_2^2 \sum_{i=1}^{d} \frac{1}{\mu_i^2 \mu_{i\rightarrow}^2}}{\gamma^2 m}}\right).$$

Code that explores this example can be found here.

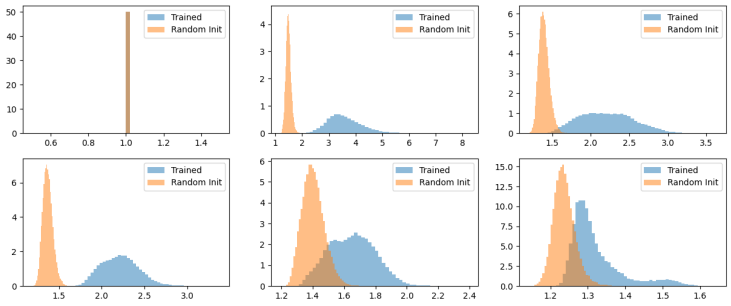

Figure 3: The activation contraction values for different layers of a fully connected network before and after training.

8 PAC Compression Bounds

This section will explore the work of [15], which uses compression ideas to tighten PAC-Bayes bounds on generalization error. The work evaluates generalization bounds by measuring the effective compressed size of networks and then substituting this into the bounds. For example, neural networks tend to be robust to perturbations of their weights, which lead to compression algorithms. Accounting for this can reduce the effective complexity of our networks a yield tighter bounds on the generalization error. Intuitively, if a model tends to overfit then there is a limit as to how much it can be compressed, thus affecting the tightness of the bound.

8.1 Applying Compression to PAC-Bayes Bounds

Consider learning a classifier using examples (x, y) where $x \in \mathcal{X}$ is a feature and $y \in \mathcal{Y}$ is a label. Assume that $(X_i, Y_i) \overset{\text{i.i.d}}{\sim} \mathcal{D}$ for some distribution $\mathcal{D}$. The aim is to choose a hypothesis $h : \mathcal{X} \to \mathcal{Y}$ that minimizes the risk,

$$L(h) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}(L(h(X), Y)).$$

Generally, $\mathcal{D}$ is unknown and so instead an i.i.d sample $S_m = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ is used to minimize the empirical risk

$$\hat{L}(h) = \frac{1}{m} \sum_{(x,y)\in S} L(h(x), y).$$

Suppose that $\hat{h}$ minimizes the empirical risk, there is the possibility that it overfits the training sample, $\hat{L}(\hat{h}) \ll L(\hat{h})$. The generalization error is given by $L(\hat{h}) - \hat{L}(\hat{h})$ and measures the degree to which the hypothesis overfits.

Theorem 8.1 *Let L be a 0-1 valued loss function. Let π be a probability measure on the hypothesis class, and let $\alpha > 1, \epsilon > 0$. Then, with probability at least $1 - \epsilon$ over the distribution of the sample:*

$$L(\rho) = \mathbb{E}_{h\sim\rho}(L(h)) \leq \inf_{\lambda>1} \Phi^{-1}_{\lambda/n}\left(\hat{L}(\rho) + \frac{\alpha}{\lambda}\left(\mathrm{KL}(\rho,\pi) - \log(\epsilon) + 2\log\left(\frac{\log\left(\alpha^2\lambda\right)}{\log(\alpha)}\right)\right)\right)$$

where

$$\Phi^{-1}_{\gamma}(x) = \frac{1 - e^{-\gamma x}}{1 - e^{-\gamma}}.$$

Remark 8.2 *Refer to π as the prior and ρ as the posterior. The challenge is to find a prior for which $\mathrm{KL}(\rho,\pi)$ is small and computable. The choice of π can be motivated by ideas of compressibility.*

The idea is to choose the prior π such that a greater probability mass is associated with models with short code length.

Theorem 8.3 *Let $|h|_c$ denote the number of bits required to represent hypothesis h using some pre-specified coding c. Let ρ denote the point mass at the compressed model $\hat{h}$. Let m denote any probability measure on the positive integers. Then there exists a prior π_c such that*

$$\mathrm{KL}(\rho,\pi_c) \leq \left|\hat{h}\right|_c \log(2) - \log\left(m\left(\left|\hat{h}\right|_c\right)\right).$$

Remark 8.4 *An example of a coding scheme c could be Huffman encoding. However, such compression schemes are agnostic to any structure of the hypothesis class $\mathcal{H}$. By exploiting structure in the hypothesis class the bound can be improved substantially.*

A formalisation of compression schemes is required to refine Theorem 8.3. Denote a compression procedure by a triple (S, C, Q) where

- $S = \{s_1, \ldots, s_k\} \subseteq \{1, \ldots, p\}$ is the location of the non-zero weights,
- $C = \{c_1, \ldots, c_r\} \subseteq \mathbb{R}$, is a codebook, and
- $Q = (q_1, \ldots, q_k)$ for $q_i \in \{1, \ldots, r\}$ are the quantized values.

Define the corresponding weights $w(S, Q, C) \in \mathbb{R}^p$ as,

$$w_i(S,Q,C) = \begin{cases} c_{q_j} & i = s_j \\ 0 & \text{otherwise.} \end{cases}$$

Training a neural network is a stochastic process due to the randomness of SGD. So to analyse the generalization error consider applying Gaussian noise to weights. This is motivated by the work of the previous section. For this we use $\rho \sim \mathcal{N}\left(w, \sigma^2 J\right)$, with J being a diagonal matrix,

Theorem 8.5 *Let (S, C, Q) be the output of a compression scheme, and let $\rho_{S,C,Q}$ be the stochastic estimator given by the weights decoded from the triplet and variance σ^2. Let c denote an arbitrarily fixed coding scheme and let m denote an arbitrary distribution on the positive integers. Then for any $\tau > 0$, there is a PAC-Bayes prior π such that*

$$\begin{aligned}\mathrm{KL}(\rho_{S,C,Q}, \pi) \leq & (k\lceil \log(r) \rceil + |S|_c + |C|_c) \log(2) - \log(m)(k\lceil \log(r) \rceil + |S|_c + |C|_c) \\ & + \sum_{i=1}^{k} \mathrm{KL}\left(\mathcal{N}\left(c_{q_i}, \sigma^2\right), \sum_{j=1}^{r} \mathcal{N}\left(c_j, \tau^2\right) \right).\end{aligned}$$

Choosing the prior alluded to by Theorem 8.5 and utilizing Theorem 8.1 one can obtain a PAC generalization bounds that exploits notions of compressibility.

8.2 Extension

Using the framework for defining compression mechanisms, Algorithm 3 from the previous section can be formalized. Recall, that the algorithm compresses a vector $c \in \mathbb{R}^d$ to a vector $\hat{c}$. As $\hat{c}$ is a random variable there must be some randomness to either S, Q or C. For $i \in \{1, \ldots, d\}$ let $i \in S$ with probability $\frac{2c_i^2}{\eta\gamma^2}$, let $C_i = \frac{c_i}{p_i}$ and $q_i = s_i$. Recalling, that

$$w_i(S, Q, C) = \begin{cases} c_{q_j} & i = s_j \\ 0 & \text{otherwise,} \end{cases}$$

it follows that $\hat{c}_i = 0$ when $i \notin S$ which happens with probability $1 - \frac{2c_i^2}{\eta\gamma^2}$. On the other hand, $\hat{c}_i \neq 0$ only when $i \in S$ which happens with probability $\frac{2c_i^2}{\eta\gamma^2}$, in which case $\hat{c}_i = C_i = \frac{c_i}{p_i}$. Note that in this compression framework, there is no quantization and thus the codebook will have a length equal to the number of non-zero entries in $\hat{c}$. Hence, Huffman encoding will provide no benefit, as entries occur singly. However, the bound of Theorem 8.1 can be used to get a bound on the KL divergence that can then be inserted into Theorem 8.1 to get a bound on the generalization error.

9 Data Driven PAC-Bayes Bounds

Neural networks are stable to noise injected in their weights which motivated compression methods that consequently lead to tighter generalization bounds. In a similar way, the structure of the data used to train a particular network can be exploited to get tighter generalization bounds. A lot of work to obtain non-vacuous PAC-Bayes generalization bounds is to develop priors that reduce the size of the KL divergence between the prior and the posterior. The idea behind the work of [17] is to hold out some of the training data to obtain a data-inspired prior. Note that training a neural network corresponds to a learning rule, that takes a sample S and returns a network $Q(S)$ which is considered

to be the posterior. Recall the PAC-Bayes bound involves $\mathrm{KL}(Q(S), P)$, and so our prior P should be such that this term is small. Minimizing $\mathbb{E}(\mathrm{KL}(Q(S), P))$ gives the oracle prior P^*. In the case where the prior is formed before seeing the training data, S, the oracle prior is optimal in expectation. In [17] it is demonstrated that if the prior is formed after seeing a subset of the training data then the prior that the oracle prior may not yield the tightest bound in expectation.
To investigate this hypothesis consider the setup of [17]. There is a space of labelled examples Z, and probability distributions $\mathcal{M}_1(Z)$ on this space. The hypothesis class $\mathcal{H}$ is the space of neural networks parameterised by weights w. The quality of which is determined by the loss function $l : \mathcal{H} \times Z \to [0, 1]$ and the risk of a particular hypothesis is given by

$$L_{\mathcal{D}}(w) = \mathbb{E}_{z\sim\mathcal{D}}(l(w, z)).$$

Similarly, the empirical risk for an i.i.d training set $S \sim \mathcal{D}^n$ is

$$L_S(w) = L_{\hat{\mathcal{D}}_n}(w) = \frac{1}{n}\sum_{i=1}^{n} l(w, z_i).$$

The linear PAC-Bayes bound is summarized in the following theorem.

Theorem 9.1 *Let $\beta, \delta \in (0, 1), n \in \mathbb{N}, \mathcal{D} \in \mathcal{M}_1(Z)$ and $P \in \mathcal{M}_1(\mathcal{H})$. With probability at least $1 - \delta$ over $S \sim \mathcal{D}^n$, for all $Q \in \mathcal{M}_1(\mathcal{H})$,*

$$L_{\mathcal{D}}(Q) \leq \Psi_{\beta,\delta}(Q, P; S) := \frac{1}{\beta}L_S(Q) + \frac{\mathrm{KL}(Q, P) + \log\left(\frac{1}{\delta}\right)}{2\beta(1-\beta)|S|}$$

Remark 9.2 *This bound is valid for all priors independent from S.*

Theorem 9.3 *Let $n \in \mathbb{N}$ and fix a probability kernel $Q : Z^n \to \mathcal{M}_1(\mathcal{H})$. For all $\beta, \delta \in (0, 1)$ and $\mathcal{D} \in \mathcal{M}_1(Z)$, $\mathbb{E}_{S\sim\mathcal{D}^n}(\Psi_{\beta,\delta}(Q(S), P; S))$ is minimized in P by the oracle prior $P^* = \mathbb{E}_{S\sim\mathcal{D}^n}(Q(S))$.*

9.1 Data-Dependent Oracle Priors

Let J be a subset of $[n]$ of size $m < n$ that is possibly random and use this as a set of indices to define a subset of S, denoted S_J. Using this subset define the data-dependent oracle prior $P^*(S_J) = \mathbb{E}(Q(S)|S_J)$ which arises as the solution of the optimization problem

$$\inf_{P\in Z^{|J|}\to\mathcal{M}_1(\mathcal{H})} \mathbb{E}(\mathrm{KL}(Q(S), P(S_J))). \tag{5}$$

Remark 9.4 *With $\hat{w}$ being a random element of $\mathcal{H}$ such that $\mathbb{P}(\hat{w}|S, J) = Q(S)$ a.s then (5) is the conditional mutual information $I(\hat{w}; S|S_J)$.*

With a restricted family of prior distributions $\mathcal{F} \subseteq \mathcal{M}_1(\mathcal{H})$ the optimization problem (5) is over kernels $Z^{|J|} \to \mathcal{F}$. A solution to which is denoted $P^*_{\mathcal{F}}(S_J)$ with (5) referred to as the conditional $\mathcal{F}$-mutual information, $I_{\mathcal{F}}(\hat{w}; S|S_J)$.

Definition 9.5 *With fixed $\mathcal{F}$ the information rate gain and the excess bias are*

$$R_{\mathcal{F}}\left(\hat{w}; S|S_J\right) = \frac{I_{\mathcal{F}}\left(\hat{h}; S\right)}{|S|} - \frac{I_{\mathcal{F}}\left(\hat{h}; S|S_J, J\right)}{|S \setminus S_J|} \text{ and } B(\hat{w}; S|S_J) = \mathbb{E}\left(L_{S\setminus S_J}(\hat{w}) - L_S(\hat{w})\right).$$

Proposition 9.6 *Let $\beta, \delta \in (0,1), n \in \mathbb{N}$ and $\mathcal{D} \in \mathcal{M}_1(Z)$. Fix $Q : Z^n \to \mathcal{M}_1(\mathcal{H})$ and let $J \subseteq [n]$ be a subset of nonrandom cardinality $m < n$, independent from $S \sim \mathcal{D}^n$. Conditional on S and J, let $\hat{w}$ have distribution $Q(S)$. Then*

$$\mathbb{E}_J \mathbb{E}_{S\sim\mathcal{D}^n}\left(\Psi_{\beta,\delta}(Q(S), P^*_{\mathcal{F}}(S_J); S \setminus S_J)\right) \leq \mathbb{E}_{S\sim\mathcal{D}^n}\left(\Psi_{\beta,\delta}(Q(S), P^*_{\mathcal{F}}; S)\right)$$

if and only if

$$R_{\mathcal{F}}(\hat{w}; S|S_J) \geq 2(1-\beta)B(\hat{w}; S|S_J) + \frac{\log\left(\frac{1}{\delta}\right)}{n}\frac{m}{n-m}.$$

Therefore, data-dependent priors, minimize linear PAC-Bayes bounds in expectation.

9.2 Data-Dependent Priors Using SGD

Now use this data to gain insight into the subsequent training process conducted in the full training set to encode information about training into the prior.

Restrict the optimization of our prior to the family of Gaussian priors, $\mathcal{F}$, that generate Gaussian posteriors. Let $(\Omega, \mathcal{F}, \nu)$ be a probability space and focus on the kernels

$$Q : \Omega \times Z^n \to \mathcal{M}_1(\mathcal{H}), \text{ for } Q(U,S) = \mathcal{N}(w_s, \Sigma),$$

where w_S are the learned weights of SGD on the full data set S. Fix some non-negative integer $m \leq n$ and let $\alpha = \frac{m}{n}$. Let S_α be a subset of S with size m and containing the first m indices processed by SGD. Let $\mathbb{E}^{S_\alpha, U}[\cdot]$ be the conditional expectation operator given S_α, U so that the tightest bound in expectation for Theorem 9.1 is the minimizer of

$$\mathbb{E}^{S_\alpha, U}(\mathrm{KL}(Q(U,S), P). \tag{6}$$

Restrict the analyses further to solving (6) for priors P of form $\mathcal{N}(w_\alpha, \sigma_P I)$, such that with σ_P fixed (6) reduces to

$$\mathrm{argmin}_{w_\alpha} \mathbb{E}^{S_\alpha, U}\left(\|w_S - w_\alpha\|\right),$$

it follows that $w_\alpha = \mathbb{E}^{S_\alpha, U}(w_S)$.

9.2.1 Ghost Samples As $\mathcal{D}$ is unknown $\mathbb{E}^{S_\alpha,U}(w_S)$ is approximated using a ghost sample, S^G, which is an independent sample equal in distribution to S. Combine a $1-\alpha$ fraction of S^G with S_α to obtain the sample S^G_α. Let w^G_α be the mean of $Q\left(U, S^G_\alpha\right)$. By construction SGD with randomness U on S^G_α will process S_α then process the combined portion of S^G, hence, $w^G_\alpha = w_S$ are equal in distribution when conditioned on S_α and U. Therefore, w^G_α is an unbiased estimator of $\mathbb{E}^{S_\alpha,U}(w_S)$.

- The SGD run on S is the base run.
- The SGD run on S_α is the α-prefix run.
- The SGD run on S^G_α is the α-prefix+ghost run and obtains the parameters w^G_α.

The resulting parameters of the α-prefix and α-prefix+ghost run can be used as the centre of the Gaussian prior used to obtain the tightened generalization bound. However, sometimes the ghost sample is not attainable in practice, and hence one simply relies upon α-prefix runs to obtain the mean of the prior. It is not clear whether α-prefix+ghost run will always obtain a parameter that leads to a tighter generalization bound. Recall, that σ_P is assumed to be fixed in the optimization process. Algorithm 7 is independent of this parameter and so it can be optimized afterwards without requiring a re-run of the optimization process.

Algorithm 6 Stochastic Gradient Descent

Require: Learning rate η
  function SGD($w_0, S, b, t, \mathcal{E} = -\infty$)
    $w \leftarrow w_0$
    for $i \leftarrow 1$ to t **do**
      Sample $S' \in S$ with $|S'| = b$
      $w \leftarrow w - \eta \nabla L_{S'}(w)$
      if $L_S^{0\text{-}1}(w) \leq \mathcal{E}$ **then**
        break
      end if
    end for
  end function

Algorithm 7 Obtaining Bound Using SGD Informed Prior

Require: Stopping criteria $\mathcal{E}$, Prefix fraction α, Ghost Data S^G (If available), Batch size b.

function GETBOUND($\mathcal{E}, \alpha, T, \sigma_P$)

$S_\alpha \leftarrow \{z_1, \ldots, z_{\alpha|S|} \subset S\}$

$w_\alpha^0 \leftarrow \text{SGD}\left(w_0, S_\alpha, b, \frac{|S_\alpha|}{b}\right)$

$w_S \leftarrow \text{SGD}(w_\alpha^0, S, b\infty, \mathcal{E})$ ▷ Base Run

$w_\alpha^G \leftarrow \text{SGD}(w_\alpha^0, S_\alpha^G, b, T, \cdot)$ ▷ Ghost run if data available, otherwise prefix run

$P \leftarrow \mathcal{N}\left(w_\alpha^G, \sigma_P I_p\right)$

$Q \leftarrow \mathcal{N}\left(w_S, \sigma_P I_p\right)$

Bound$\leftarrow \Psi_\delta^*(Q, P; S \setminus S_\alpha)$

return Bound

end function

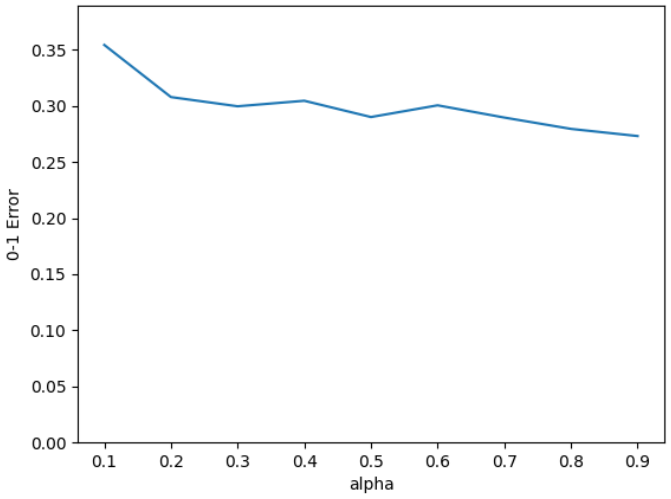

Figure 4: Shows the generalization error for varying levels of α.

Part III
Information Theoretic Approach

10 Introduction

Training neural network is essentially transferring information from the training data into the weights of the network. The goal is to effectively represent data from the training data such that it can be extrapolated to make inferences in a different setting. However, there exist multiple representations of information, some of which are richer than others. Analysing networks from an information theoretic perspective provides insight into the utility of the learned representations beyond their performance on a singular metric, the loss. The framework allows the tracking of information transfer from the input data to the outputs of the network and can indicate different stages of the learning process. Information-theoretic approaches are inherently related to the data inputted into the network and hence can provide more robust metrics for bounding the generalization errors of the network.

11 Controlling Bias in Data Analysis

One of the first efforts to contextualize learning bias in information theory was done in [3]. The main result of [3] is an information-theoretic bound on the bias of adaptively choosing a function from data. A dataset D that is drawn from a probability distribution $\mathcal{P}$ defined over $\mathcal{D}$. There is a set of analyses

$$\phi_1, \ldots, \phi_m : \mathcal{D} \to \mathbb{R}$$

that can be run on the data. Each ϕ_i is a random variable dependent on the realization $D \sim \mathcal{P}$. When the realization is made a particular $\phi_T(D)$ is reported for $T \in [m]$. The selection rule $T : \mathcal{D} \to [m]$ determines how the realization relates to the reported result. Bias may be present in the selection rule as it is a function of the realization which is itself a proxy of the underlying distribution $\mathcal{D}$. Let

$$\Phi = \begin{pmatrix} \phi_1 \\ \vdots \\ \phi_m \end{pmatrix} : \Omega \to \mathbb{R}^m \text{ and } T : \Omega \to [m]$$

and

$$\mu = \begin{pmatrix} \mu_1 \\ \vdots \\ \mu_m \end{pmatrix} := \mathbb{E}(\Phi).$$

Definition 11.1 *A real-valued random variable X is σ-sub-Gaussian if for all $\lambda \in \mathbb{R}$*

$$\mathbb{E}\left(e^{\lambda X}\right) \leq e^{\frac{\lambda^2\sigma^2}{2}}.$$

Definition 11.2 ([5]) *For two random variables X and Y, with joint distribution $p(x, y)$, their Mutual Information is defined as,*

$$\begin{aligned} I(X;Y) = \mathrm{KL}(p(x,y), p(x)p(y)) &= \sum_{x\in X, y\in Y} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right) \\ &= H(X) - H(X|Y), \end{aligned}$$

where $H(X)$ and $H(X|Y)$ are the entropy and conditional entropy of X and Y.

Mutual information quantifies the number of relevant bits that the input variable X contains about the label Y on average.

Theorem 11.3 *Suppose that for each $i \in [m]$, $\phi_i - \mu_i$ is σ-sub-Gaussian. Then,*

$$|\mathbb{E}(\phi_T) - \mathbb{E}(\mu_T)| \leq \sigma\sqrt{2I(T;\Phi)},$$

Remark 11.4

- $\mathbb{E}(\phi_T)$ *is taken jointly over the realized values of the ϕ_i and T.*
- $\mathbb{E}(\mu_T)$ *is taken over the selection procedure T.*
- $|\mathbb{E}(\phi_T) - \mathbb{E}(\mu_T)|$ *quantifies the bias due to T.*
- $I(T;\phi)$ *quantifies the dependence of the selection process on the noise in the test statistics.*

Proposition 11.5 *Let $\Phi = (\phi_1\ \phi_2\ \ldots)^\top$ be a collection of independent normally distributed random variables with mean 0 and variance σ^2. For $B > 1$, let $T_B = \mathrm{argmax}_{1\leq i\leq \lfloor e^B\rfloor}\phi_i$. Then, $I(T_B;\Phi) \leq B$ and*

$$\mathbb{E}(\Phi_{T_B}) - \sigma\sqrt{2B} \to 0$$

as $B \to \infty$. Furthermore, there exists a $c > 0$ such that

$$\mathbb{E}(\phi_{T_B}) \geq c\sigma\sqrt{2B}, \quad \forall B \geq 2.$$

The above are bounds on the bias involving specific assumptions on the distributions and the selection procedure. The next section considers [2] which shows how some of these assumptions can be relaxed.

12 Generalizing the Framework

In [2] Theorem 11.3 is extended to distributions with non-trivial moment generating functions. Furthermore, a new measure is introduced which generalizes mutual information.

Definition 12.1 *The cumulant generating function of a random variable X is*

$$\psi(\lambda) = \log\left(\mathbb{E}\left(e^{\lambda X}\right)\right), \quad \lambda \geq 0.$$

In what follows assume that for all considered random variables there exists a $\lambda > 0$ such that $\mathbb{E}\left(e^{\lambda X}\right) < \infty$.

Definition 12.2 *A random variable X is sub-Exponential with parameters (σ, b) if*

$$\mathbb{E}\left(e^{\lambda X}\right) \leq e^{\frac{\lambda^2\sigma^2}{2}}, \quad 0 \leq \lambda < \frac{1}{b}.$$

Definition 12.3 *A random variable X is sub-Gamma on the right tail with variance factor σ^2 and scale parameter c if*

$$\psi(\lambda) \leq \frac{\lambda^2\sigma^2}{2(1-c\lambda)}, \quad 0 < \lambda < \frac{1}{c}.$$

Definition 12.4 *The β-norm of a random variable X for $\beta \geq 1$ is*

$$\|X\|_\beta = \begin{cases} \left(\mathbb{E}\left(|X|^\beta\right)\right)^{\frac{1}{\beta}} & 1 \leq \beta < \infty \\ \operatorname{ess\,sup}|X| & \beta = \infty, \end{cases}$$

where $\operatorname{ess\,sup} = \inf\{M : \mathbb{P}(X > M) = 0\}$.

Proposition 12.5 *For any function f, lets its convex conjugate, f^*, be defined as*

$$f^*(y) = \sup_{x \in X}(\langle x, y\rangle - f(x)).$$

Definition 12.6 *For $\phi : \mathbb{R}_{\geq 0} \to \mathbb{R}$ a convex lower semi-continuous function satisfying $\phi(1) = 0$, the ϕ-divergence of two probability distributions is*

$$D_\phi(Q, P) = \int \phi\left(\frac{dQ}{dP}\right) dP.$$

Remark 12.7 *For $\phi(x) = x\log(x) - x + 1$, it follows that $D_\phi(Q, P) = \mathrm{KL}(Q, P)$.*

For non-negative sequences $\{a_n\}$ and $\{b_n\}$ let $a_n \lesssim b_n$ if there is a constant $C > 0$ such that $\limsup_n \frac{a_n}{b_n} \leq C$.

Theorem 12.8 *Suppose that $\phi_i - \mu_i$ has cumulant generating function upper bounded by $\psi_i(\lambda)$ over domain $[0, b_i)$ where $b_i \in (0, \infty]$. Suppose $\psi_i(\lambda)$ is convex with $\psi_i(0) = \psi_i'(0) = 0$. Define the expected cumulant generating function $\bar{\psi}(\lambda)$ as*

$$\bar{\psi}(\lambda) = \mathbb{E}_T\left(\psi_T(\lambda)\right), \quad \lambda \in \left[0, \min_i b_i\right).$$

Then,

$$\mathbb{E}(\phi_T - \mu_T) \leq \left(\bar{\psi}^*\right)^{-1} I(T; \Phi).$$

Definition 12.9 *For $1 \leq \alpha < \infty$ let*

$$I_\alpha(X; Y) = D_{\phi_\alpha}(P_{XY}, P_X P_Y)$$

where $\phi_\alpha(x) = |x - 1|^\alpha$.

Remark 12.10

- *$I_\alpha(X; Y) \geq 0$, and*
- *$I_\alpha(X; Y) = 0$ if and only if X ad Y are independent.*

Theorem 12.11 *Suppose $\phi_i - \mu_i$ has its β-norm upper bounded by σ_i, for $1 < \beta \leq \infty$. Let α be such that $\frac{1}{\alpha} + \frac{1}{\beta} = 1$. Then,*

$$|\mathbb{E}(\phi_T - \mu_T)| \leq \|\sigma_T\|_\beta I_\alpha(T; \Phi)^{\frac{1}{\alpha}}.$$

For $\beta = 2$ we have

$$|\mathbb{E}(\phi_T - \mu_T)| \leq \|\sigma_T\|_2 \sqrt{n - 1},$$

and for $2 < \beta \leq \infty$ and $1 \leq \alpha < 2$ we have

$$|\mathbb{E}(\phi_T - \mu_T)| \leq \|\sigma_T\|_\beta \left(1 + n^{\alpha - 1}\right)^{\frac{1}{\alpha}} \leq 2^{\frac{1}{\alpha}} \|\sigma_T\|_\beta n^{\frac{1}{\beta}}.$$

Corollary 12.12 *Suppose $\phi_i - \mu_i$ is σ_i-sub-Gaussian. Then,*

$$\mathbb{E}(\phi_T - \mu_T) \leq \|\sigma_T\|_2 \sqrt{2I(T; \Phi)}.$$

Corollary 12.13 *Suppose $\phi_i - \mu_i$ are sub-Gamma random variables with parameters (σ^2, c). Then,*

$$\mathbb{E}(\phi_T - \mu_T) \leq \sigma \sqrt{2I(T; \Phi)} + cI(T; \Phi).$$

Corollary 12.14 *Suppose $\phi_i - \mu_i$ are sub-Exponential random variables with parameters (σ, b). Then,*

$$\mathbb{E}_T(\phi_T - \mu_T) \leq \begin{cases} \sigma\sqrt{2I(T; \Phi)} & I(T; \Phi) \leq \frac{\sigma^2}{2b} \\ bI(T; \Phi) + \frac{\sigma^2}{2b^2} & \text{otherwise.} \end{cases}$$

13 Evolution of Mutual Information

It has been shown that layered neural networks form a Markov chain, which suggests studying them from the perspective of mutual information. That is, consider each layer to be a single random variable and look at its mutual information with the input X and the output Y. The properties of neural network training are explored using this approach in [5]. In this work, the learning is formulated as finding a good representation $T(X)$ of the input patterns $x \in X$ that generates good predictions for label $y \in Y$. Training is conducted through stochastic gradient descent (SGD), where at each step we aim to minimize the empirical error over the weights of the network. As this minimization occurs layer-by-layer the whole layer is treated as a single representation T which is characterized by the encoder $P(T|X)$ and the decoder $P(Y|T)$ distributions.

Theorem 13.1 *For any invertible functions ϕ and ψ,*

$$I(X;Y) = I(\psi(X), \phi(Y)).$$

Theorem 13.2 *For any three variables that form a Markov chain $X \to Y \to Z$,*

$$I(X;Y) \geq I(X,Z).$$

13.1 Information Plane

Given $P(X,Y)$, any representation T corresponds to a unique point in the information plane with coordinates $(I(X;T), I(T;Y))$. Consider a K-layered deep neural network, with T_i denoting the representation of the i^{th} layer then there is a unique information path which satisfies Theorem 13.2,

$$\begin{aligned} I(X;Y) \geq I(T_1;Y) \geq \cdots \geq I(T_k;Y) \geq I\left(\hat{Y};Y\right), \\ H(X) \geq I(X;T_1) \geq \cdots \geq I(X;T_k) \geq I\left(X;\hat{Y}\right). \end{aligned}$$

Due to Theorem 13.1 it is possible that many different deep neural networks correspond to an information path.

13.2 Information Bottleneck

Definition 13.3 *Let X and Y be two random variables. A sufficient statistic $S(X)$ is map or partition of X that captures all the information that X has on Y,*

$$I(S(X);Y) = I(X,Y).$$

A minimal sufficient statistic, $T(X)$, induces the coarsest such partition on X. Consequently, one can form the Markov Chain

$$Y \to X \to S(X) \to T(X).$$

Using Theorem 13.2 finding T can be formulated as the optimization problem,

$$T(X) = \arg \min_{S(X);I(S(X);Y)=I(X;Y)} I(S(X);X).$$

The Information Bottleneck trade-off enables a framework for finding approximate minimal sufficient statistics. Let $t \in T$ be the compressed representation of $x \in X$, so that x is represented as $p(t|x)$. The Information Bottleneck trade-off is captured in the optimization problem

$$\min_{p(t|x),p(y|t),p(t)} (I(X;T) - \beta I(T;Y)).$$

Where β determines the level of relevant information captured by T. The solution to this problem is given by

$$\begin{cases} p(t|x) = \frac{p(t)}{Z(x;\beta)} \exp\left(-\beta \mathrm{KL}(p(y|x), p(y|t))\right) \\ p(t) = \sum_x p(t|x)p(x), \\ p(y|t) = \sum_x p(y|x)p(x|t), \end{cases} \tag{7}$$

where $Z(x;\beta)$ is the normalized function.

13.3 Student-Teacher Analysis

Work done by [11] implements this theory in different settings. In this first setting, there is a linear teacher network that generates training examples for a deep linear student network to learn. The teacher network has an input size of N_i and an output size of 1. The input is a multi-variate normal, $X \sim \mathcal{N}(0, \frac{1}{N_i} I_{N_i})$, and the weights of the network, W_o, are sampled independently from $\mathcal{N}(0, \sigma_2^2)$. The output for a given input is given by $Y = W_o X + \epsilon_o$ where $\epsilon_o \sim \mathcal{N}\left(0, \sigma_o^2\right)$. Take P samples from this teacher network to train a student network using gradient descent to minimize the mean squared error. The student network consists of an input layer, hidden layers and a single output neuron. The activation function on the hidden layer neurons is just the identity function. This setup can be represented as $\hat{Y} = W_{D+1} \dots W_1 X$ when the network has depth D. The activity of the i^{th} hidden layer is given by $T = \bar{W} X = W_i \dots W_1 X$. The true generalization error at training step t is given by

$$E_g(t) = \|W_o - W_{\text{tot}}(t)\|_F^2 + \sigma_o^2.$$

To calculate the mutual information some noise needs to be added otherwise, it would be infinite. Hence, let $T = \bar{W} X + \epsilon_{MI}$ for $\epsilon_{MI} \sim \mathcal{N}(0, \sigma_{MI}^2)$. With these assumptions, it follows that

$$I(T;X) = \log\left|\bar{W}\bar{W}^\top + \sigma_{MI}^2 I_{N_h}\right| - \log\left|\sigma_{MI}^2 I_{N_h}\right|,$$

where $|\cdot|$ denotes the determinant of a matrix and N_h is the number of hidden units in the layer. Similarly, the mutual information with the output Y can be calculated as

$$
\begin{aligned}
H(Y) &= \frac{N_o}{2}\log(2\pi e) + \frac{1}{2}\log\left|W_o W_o^\top + \sigma_o^2 I_{N_o}\right|, \\
H(T) &= \frac{N_h}{2}\log(2\pi e) + \frac{1}{2}\log\left|\bar{W}\bar{W}^\top + \sigma_{MI}^2 I_{N_h}\right|, \\
H(Y;T) &= \frac{N_o + N_h}{2}\log(2\pi e) + \frac{1}{2}\log\begin{vmatrix} \bar{W}\bar{W}^\top + \sigma_{MI}^2 I_{N_h} & \bar{W}W_o^\top \\ W_o\bar{W}^\top & W_o W_o^\top + \sigma_o^2 I_{N_o} \end{vmatrix}, \\
I(Y;T) &= H(Y) + H(T) - H(Y;T),
\end{aligned}
$$

where N_o is the size of the input. For the implementation, there will only be a single hidden layer so that $\bar{W} = W_1$, refer to Figure 5 for the results.

13.4 Non-Linear Setting via KDE

In another setting developed by [11] the mutual information for non-linear neural networks is investigated. This is done by appealing to Kernel Density Estimation. To apply KDE the hidden activity is assumed to be distributed as a mixture of Gaussians. As the layer activity is a deterministic function of the input noise needs to be added to get finite mutual information. Hence, we let $T = h + \epsilon$ where h is the activity of the hidden layer and $\epsilon \sim \mathcal{N}(0, \sigma^2 I)$. Under these assumptions, it follows that

$$
\begin{aligned}
I(T;X) &\leq -\frac{1}{P}\sum_i \log\left(\frac{1}{P}\sum_j \exp\left(-\frac{1}{2}\frac{\|h_i - h_j\|_2^2}{\sigma^2}\right)\right) \\
I(T;Y) &\leq -\frac{1}{P}\sum_i \log\left(\frac{1}{P}\sum_j \exp\left(-\frac{1}{2}\frac{\|h_i - h_j\|_2^2}{\sigma^2}\right)\right) \\
&\quad - \sum_{l=1}^{L} p_l \left(-\frac{1}{P_l}\sum_{Y_i = l}\log\left(\frac{1}{P_l}\sum_{Y_j = l}\exp\left(-\frac{1}{2}\frac{\|h_i - h_j\|_2^2}{\sigma^2}\right)\right)\right),
\end{aligned}
$$

where

- P is the number of training samples,
- h_i is the hidden activity in response to sample i,
- L is the number of output labels,
- P_l is the number of samples with label l,
- $p_l = \frac{P_l}{P}$, and

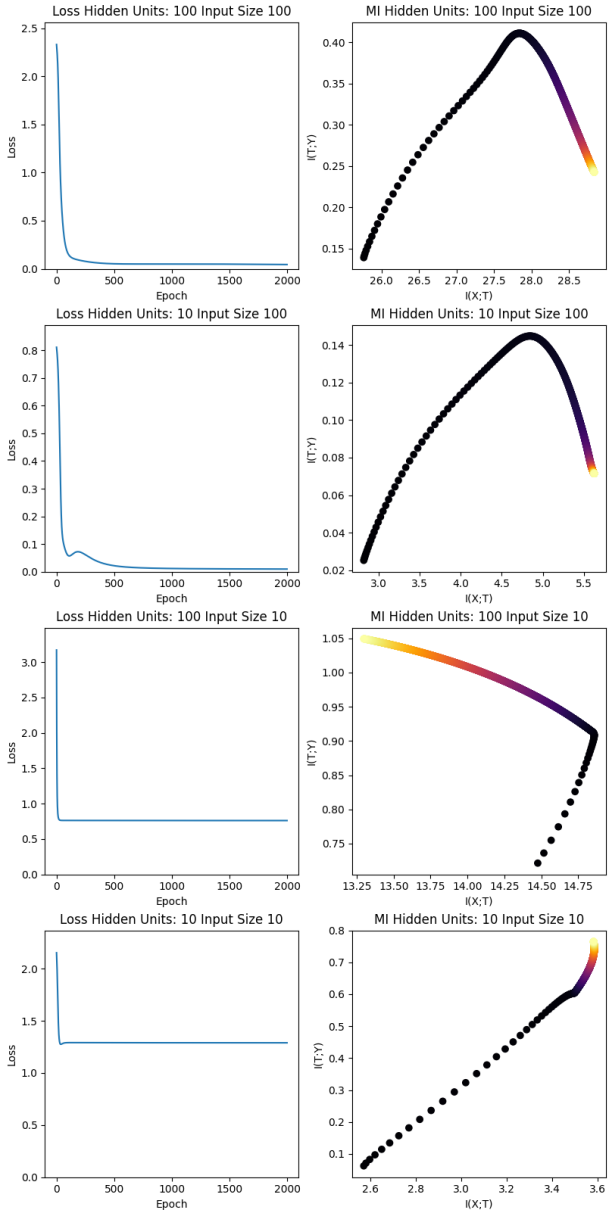

Figure 5: Explores the Mutual Information in the teacher-student scenario with different architectures and training data.

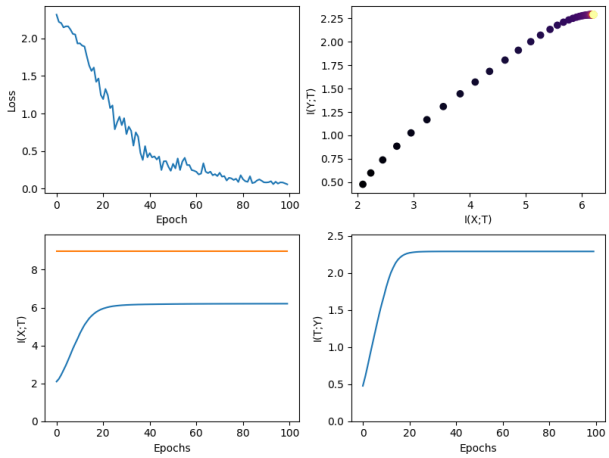

Figure 6: The Mutual Information of a ReLU neural network estimated using Kernel Density Estimation.

- $Y_i = l$ is the sum over examples with output l.

See the results of this implementation in Figure 6. Where a network with layers $784 \to 1024 \to 20 \to 20 \to 20 \to 10$ is trained on 1000 images from the MNIST dataset. The activations of the third hidden layer are observed and used to estimate the mutual information between the inputs and outputs. The orange line corresponds to the entropy of a uniform distribution across the 1000 samples ($\log_2(1000)$).

14 Generalization Bounds for Learning Algorithms

Return to the framework set out by [3] and [2] and translate it into the domain of learning algorithms, to get generalization bounds. This is the work of [6]. Recall, that generalization error is the difference between the population risk of the output hypothesis and its empirical risk on the training data. In learning problems, generalization error is equivalent to the bias in data analysis discussed previously. This is a promising approach to generating generalization bounds as mutual information is strongly dependent on the input dataset, and generalization error is also impacted by the input dataset. Hence, consider an input space $\mathcal{Z}$ and a hypothesis space $\mathcal{W}$. Define a non-negative loss function $l : \mathcal{W} \times \mathcal{Z} \to \mathbb{R}^+$ and characterize the learning algorithm by a Markov kernel $P_{W|S}$ where $S = (Z_1, \ldots, Z_n)$ is the training set. Assume that S is an i.i.d sample of elements from an unknown distribution μ. The learning algorithm is a random variable W on $\mathcal{W}$ with distribution $P_{W|S}$. For

$w \in \mathcal{W}$, the population risk is then

$$L_\mu(w) := \mathbb{E}(l(w,Z)) = \int_Z l(w,z)\mu(dz).$$

Ideally, the learning algorithm will be such that the excess risk

$$L_\mu(W) - \inf_{w\in\mathcal{W}} L_\mu(w)$$

and the expected excess risk

$$R_{\text{excess}}(\mu, P_{W|S})$$

are small. However, in practice one uses the risk proxy

$$L_S(w) := \frac{1}{n}\sum_{i=1}^{n} l(w,Z_i),$$

which is the empirical risk for $w \in \mathcal{W}$ on dataset S. As empirical risk is a proxy a source of potential error is introduced known as the generalization error,

$$\text{gen}(\mu, P_{W|S}) := \mathbb{E}(L_\mu(W) - L_S(W)),$$

where the expectation is over $P_{S,W} = \mu^{\otimes n} \otimes P_{W|S}$. The expected population risk can be decomposed as

$$\mathbb{E}(L_\mu(W)) = \mathbb{E}(L_S(W)) + \text{gen}(\mu, P_{W|S}).$$

Definition 14.1 *A learning algorithm is (ϵ,μ)-stable in input-output mutual information if, under the data-generating distribution μ,*

$$I(S;W) \leq \epsilon$$

Definition 14.2 *A learning algorithm is ϵ-stable in input-output mutual information if*

$$\sup_\mu I(S;W) \leq \epsilon.$$

The learning algorithm $P_{W|S}$ can be viewed as a channel from $\mathcal{Z}^n$ to $\mathcal{W}$ with $\sup_\mu I(S;W)$ being the information capacity of the channel. Therefore, a learning algorithm is more stable if its information capacity is smaller. Now we proceed to try and bound generalization using $I(S;W)$.

Consider a pair of random variables X and Y with joint distribution $P_{X,Y}$. Let $\bar{X}$ and $\bar{Y}$ be independent copies of X and Y respectively such that $P_{\bar{X},\bar{Y}} = P_X \otimes P_Y$.

Lemma 14.3 *If $f : \mathcal{X}\times\mathcal{Y} \to \mathbb{R}$ is such that $f\left(\bar{X},\bar{Y}\right)$ is σ-sub-Gaussian under $P_{\bar{X},\bar{Y}}$, then*

$$\left|\mathbb{E}(f(X,Y)) - \mathbb{E}\left(f\left(\bar{X},\bar{Y}\right)\right)\right| \leq \sqrt{2\sigma^2 I(X;Y)}.$$

Let $X = S, Y = W$ and $f(s,w) = \frac{1}{n}\sum_{i=1}^{n} l(w, z_i)$. For $w \in W$ the empirical risk can be written as $L_S(w) = f(S,w)$ and the population risk can be written as $L_\mu(w) = \mathbb{E}(f(S,w))$. Therefore, generalization error is

$$\text{gen}(\mu, P_{W|S}) = \mathbb{E}\left(f\left(\bar{S}, \bar{W}\right)\right) - \mathbb{E}(f(S,W)),$$

where the joint distribution of S and W is $P_{S,W} = \mu^{\otimes n} \otimes P_{W|S}$. Using the fact that if $l(w,Z)$ is σ-sub-Gaussian then $f(S,w)$ is $\frac{\sigma}{\sqrt{n}}$-sub-Gaussian and Lemma 14.3 yields the following.

Theorem 14.4 *Suppose $l(w,Z)$ is σ-sub-Gaussian under μ for all $w \in \mathcal{W}$, then*

$$\left|\text{gen}(\mu, P_{W|S})\right| \leq \sqrt{\frac{2\sigma^2 I(S;W)}{n}}.$$

Theorem 14.5 *Let the hypothesis space $\mathcal{W}$ be finite. Suppose that $l(w,Z)$ is σ-sub-Gaussian under μ for all $w \in \mathcal{W}$, then*

$$\left|\text{gen}(\mu, P_{W|S})\right| \leq \sqrt{\frac{2\sigma^2 I(\Lambda_{\mathcal{W}}(S);W)}{n}},$$

where $\Lambda_W(S) := (L_S(w))_{w\in\mathcal{W}}$.

Theorem 14.6 *Suppose $l(w,Z)$ is σ-sub-Gaussian under μ for all $w \in \mathcal{W}$. If a learning algorithm satisfies $I(\Lambda_{\mathcal{W}}(S);W) \leq \epsilon$, then for any $\alpha > 0$ and $0 < \beta \leq 1$ it follows that*

$$\mathbb{P}(|L_\mu(W) - L_S(W)| > \alpha) \leq \beta$$

can be guaranteed by a sample complexity of

$$n = \frac{8\sigma^2}{\alpha^2}\left(\frac{\epsilon}{\beta} + \log\left(\frac{2}{\beta}\right)\right).$$

Theorem 14.7 *Suppose $l(w,Z)$ is σ-sub-Gaussian under μ for all $w \in \mathcal{W}$. If a learning algorithm satisfies $I(\Lambda_{\mathcal{W}}(S);W) \leq \epsilon$, then*

$$\mathbb{E}(|L_\mu(W) - L_S(W)|) \leq \sqrt{\frac{2\sigma^2(\epsilon + \log(2))}{n}}.$$

14.1 Application to Binary Classification

For binary classification $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ where $\mathcal{Y} = \{0,1\}$. With $\mathcal{W}$ being a collection of classifiers and $l(w,z) = \mathbb{I}\{w(x) \neq y\}$. Given a data set S split it into S_1 and S_2 with sizes n_1 and n_2 respectively. Choose a subset of hypothesis $\mathcal{W}_1 \subset \mathcal{W}$ such that $(w(X_1), \ldots, w(X_{n_1}))$ for $w \in \mathcal{W}_1$ are all distinct and

$$\left\{(w(X_1), \ldots, w(X_{n_1}) : w \in \mathcal{W}_1\right\} = \left\{(w(X_1), \ldots, w(X_{n_1}) : w \in \mathcal{W}\right\}.$$

Next, choose a hypothesis from $\mathcal{W}_1$ such that

$$W = \text{argmin}_{w\in\mathcal{W}_1} L_{S_2}(w).$$

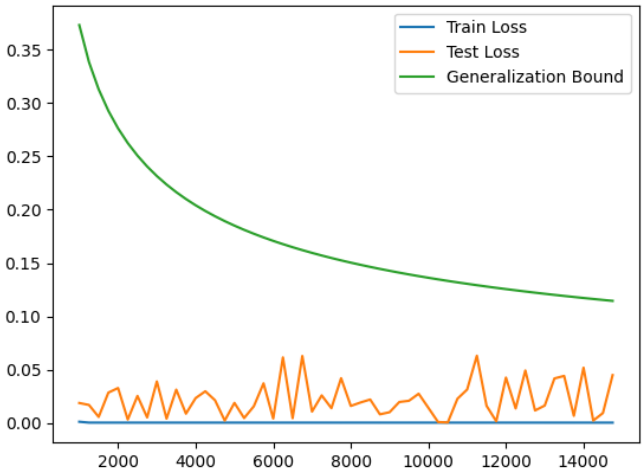

Figure 7: Bounding the test loss of a linear classifier.

Denote the n^{th} shatter coefficient and the VC dimension of $\mathcal{W}$ by $\mathbb{S}_n$ and V respectively. Then,

$$\mathbb{E}(L_\mu(W)) - \mathbb{E}(L_{S_2}(W)) \leq \sqrt{\frac{V \log(n_1 + 1)}{2n_2}}. \tag{8}$$

Furthermore,

$$\mathbb{E}(L_{S_2}(W)) \leq \inf_{w \in \mathcal{W}} L_\mu(w) + c\sqrt{\frac{V}{n_1}}, \tag{9}$$

for some constant c. Combining (8) and (9) for $n_1 = n_2 = \frac{n}{2}$ gives the bound

$$\mathbb{E}(L_\mu(W)) \leq \inf_{w \in \mathcal{W}} L_\mu(w) + c\sqrt{\frac{V \log(n)}{n}}.$$

15 Chaining Mutual Information

The bounds illustrated above have some key limitations. Firstly, they ignore the dependencies between the hypotheses within the sample space, and secondly, they ignore the dependencies between the input data and the output. To resolve these [8] applies a method known as chaining that was developed to tighten uniform bounds on random processes. The method works by first capturing the dependencies between hypotheses through a metric d on a set T, and then to discretize T to determine

the maximum values of bounds on the random process. To develop this formally consider the setting of supervised learning, where there is an input domain $\mathcal{X}$ and a label domain $\mathcal{Y}$ with $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. The hypothesis set is $\mathcal{H} = \{h_w : w \in \mathcal{W}\}$ and the non-negative loss function is $l : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$. The learning algorithm receives a training set $S = (Z_1, \ldots, Z_n)$ with examples drawn i.i.d from distribution μ over $\mathcal{Z}$. The algorithm picks $h_W \in \mathcal{H}$ according to a random transformation $P_{W|S}$. For any $w \in \mathcal{W}$ let

$$L_\mu(w) := \mathbb{E}(l(h_w, Z)), \quad Z \sim \mu$$

denote population risk of h_w and let

$$L_S(w) := \frac{1}{n} \sum_{i=1}^{n} l(h_w, Z_i)$$

denote empirical risk of h_w over S. The generalization error of h_w is given by

$$\text{gen}(w) := L_\mu(w) - L_S(w)$$

and the expected (absolute) generalization error are given by

$$\text{gen}(\mu, P_{W|S}) := \mathbb{E}(L_\mu(w) - L_S(w))$$

and

$$\text{gen}^+(\mu, P_{W|S}) := \mathbb{E}(|L_\mu(w) - L_S(w)|)$$

respectively. Let $X_\mathcal{N} := \{X_i : i \in \mathcal{N}\}$, $\mathbf{0}$ be the zero function, $H(x)$ denote the Shannon entropy of discrete random variable X, and $h(Y)$ the differential entropy of an absolutely continuous random variable Y.

Definition 15.1 *Let d be a metric on the set T*

1. *A finite set $\mathcal{N}$ is an ϵ-net for (T, d) if there exists a function $\pi_\mathcal{N}$ which maps every point $t \in T$ to $\pi_\mathcal{N}(t) \in \mathcal{N}$ such that $d(t, \pi_\mathcal{N}(t)) \leq \epsilon$.*
2. *The covering number for a metric space (T, d) is the smallest cardinality of an ϵ-net for that space denoted $N(T, d, \epsilon)$. That is,*

$$N(T, d, \epsilon) := \inf\{|\mathcal{N}| : \mathcal{N} \text{ is an } \epsilon\text{-net for } (T, d)\}.$$

3. *An ϵ-net $\mathcal{N}$ for the metric space (T, d) is called minimal if $|\mathcal{N}| = N(T, d, \epsilon)$.*

Definition 15.2 *The random process $\{X_t\}_{t \in T}$ on the metric space (T, d) is called sub-Gaussian if $\mathbb{E}(X_t) = 0$ for all $t \in T$ and*

$$\mathbb{E}\left(e^{\lambda(X_t - X_s)}\right) \leq e^{\frac{1}{2}\lambda^2 d^2(t,s)}$$

for all $t, s \in T, \lambda \geq 0$.

Definition 15.3 *The random process $\{X_t\}_{t\in T}$ is called separable if there is a countable set $T_0 \subseteq T$ such that $X_t \in \lim_{s\to t, s\in T_0} X_s$ for all $t \in T$* a.s*, where $x \in \lim_{s\to t, s\in T_0} x_s$ means that there is a sequence (s_n) in T_0 such that $s_n \to t$ and $x_{s_n} \to x$.*

Definition 15.4 *Call a partition $\mathcal{P} = \{A_1, \dots, A_m\}$ of the set T an ϵ-partition of the metric space (T, d) if for all $i = 1, \dots, m$ A_i can be contained within a ball of radius ϵ. A sequence of partitions $\{\mathcal{P}_k\}_{k=m}^{\infty}$ of a set T is called an increasing sequence if for all $k \geq m$ and each $A \in \mathcal{P}_{k+1}$ there exists $B \in \mathcal{P}_k$ such that $A \subseteq B$. For any such sequence and any $t \in T$ let $[t]_k$ denote the unique set $A \in \mathcal{P}_k$ such that $t \in A$.*

From now on assume that (T, d) is a bounded metric space, with $k_1(T)$ an integer such that $2^{-(k_1(T)-1)} \geq \operatorname{diam}(T)$.

Theorem 15.5 *Assume that $\{\operatorname{gen}(w)\}_{w\in\mathcal{W}}$ is a separable sub-Gaussian process on the bounded metric space $(\mathcal{W}, d)$. Let $\{\mathcal{P}_k\}_{k=k_1(\mathcal{W})}$ be an increasing sequence of partitions of $\mathcal{W}$, where for each $k \geq k_1(\mathcal{W})$, $\mathcal{P}_k$ is a 2^{-k}-partition of $(\mathcal{W}, d)$.*

1. $$\operatorname{gen}(\mu, P_{W|S}) \leq 3\sqrt{2} \sum_{k=k_1(\mathcal{W})}^{\infty} 2^{-k} \sqrt{I([W]_k; S)},$$

2. *If $\mathbf{0} \in \{l(h_w, \cdot) : w \in \mathcal{W}\}$, then*
$$\operatorname{gen}^+(\mu, P_{W|S}) \leq 3\sqrt{2} \sum_{k=k_1(\mathcal{W})}^{\infty} 2^{-k} \sqrt{I([W]_k; S) + \log(2)}.$$

Theorem 15.6 *Assume that $\{X_t\}_{t\in T}$ is a separable sub-Gaussian process on the bounded metric space $(\mathcal{W}, d)$. Let $\{\mathcal{P}_k\}_{k=k_1(\mathcal{W})}$ be an increasing sequence of partitions of $\mathcal{W}$, where for each $k \geq k_1(T)$, $\mathcal{P}_k$ is a 2^{-k}-partition of (T, d).*

1. $$\mathbb{E}(X_W) \leq 3\sqrt{2} \sum_{k=k_1(T)}^{\infty} 2^{-k} \sqrt{I([W]_k; X_T)},$$

2. *For arbitrary $t_0 \in T$,*
$$\mathbb{E}(|X_W - X_{t_0}|) \leq 3\sqrt{2} \sum_{k=k_1(T)}^{\infty} 2^{-k} \sqrt{I([W]_k; X_T) + \log(2)}.$$

16 Conditional Mutual Information

When investigating learning algorithms using mutual information noise had to be introduced into our observations to get finite values of mutual information. The work of [19] resolves this by considering conditional mutual information instead (CMI). CMI measures how well the learning algorithm can recognize the input given the output. This is calculated by using a "supersample", which consists of regular data points and "ghost" data points. CMI measures the ability to distinguish the regular inputs from their ghosts.

Definition 16.1 *Let $A : \mathcal{Z}^n \to \mathcal{W}$ be a randomized or deterministic algorithm. Let $\mathcal{D}$ be a probability distribution on $\mathcal{Z}$ and let $\tilde{Z} \in \mathcal{Z}^{2\times n}$ consist of $2n$ samples drawn independently from $\mathcal{D}$. Let $S \in \{0,1\}^n$ be uniformly random and independent from $\tilde{Z}$ and the randomness of A. Define $\tilde{Z}_S \in \mathcal{Z}^n$ by $(\tilde{Z}_S)_i = \tilde{Z}_{i,S_i+1}$ for all $i \in [n]$.*

- *The conditional mutual information (CMI) of A with respect to $\mathcal{D}$ is,*

$$\mathrm{CMI}_{\mathcal{D}}(A) := I\left(A\left(\tilde{Z}_S\right); S|\tilde{Z}\right).$$

- *The (distribution-free) conditional mutual information (CMI) of A is*

$$\mathrm{CMI}(A) := \sup_{\tilde{z}\in\mathcal{Z}^{n\times 2}} I\left(A\left(\tilde{z}_S\right); S\right).$$

Remark 16.2

- *For any A and $\mathcal{D}$ it follows that* $0 \le \mathrm{CMI}_{\mathcal{D}}(A) \le \mathrm{CMI}(A) \le n\log(2)$.
- *If* $\mathrm{CMI}_{\mathcal{D}}(A) = 0$ *then the output of A is independent of its input.*
- *If* $\mathrm{CMI}_{\mathcal{D}}(A) = n\log(2)$ *then the output of A reveals all of the input.*
- CMI *is always finite.*

16.1 CMI and VC Dimension

Recall that VC dimension is a property of a hypothesis class, whereas, CMI depends also on the algorithm. However, there is a connection between the two. Which enables one to utilize the theory of VC dimension and learnability to the generalization results provided by CMI. Let $\mathcal{W}$ be a class of function $h : \mathcal{X} \to \{0,1\}$. Consider the 0-1 loss $l : \mathcal{W} \times (\mathcal{X} \times \{0,1\}) \to \{0,1\}$ defined by

$$l(h,(x,y)) = \begin{cases} 0 & h(x) = y \\ 1 & \text{otherwise.} \end{cases}$$

The population loss is $l(h, \mathcal{D}) = \mathbb{E}_{(X,Y)\sim\mathcal{D}}(l(h,(X,Y)))$, where $\mathcal{D}$ is a distribution on $\mathcal{X} \times \{0,1\}$. For $z \in (\mathcal{X} \times \{0,1\})^n$ and $h : \mathcal{X} \to \{0,1\}$ let

$$l(h,z) := \frac{1}{N}\sum_{i=1}^{n} l(h, z_i).$$

For $A : (\mathcal{X} \times \{0,1\})^n \to \mathcal{W}$ is an empirical risk minimizer for $\mathcal{W}$ if

$$l(A(z), z)) = \inf_{h\in\mathcal{W}} l(h,z)$$

for all $z \in (\mathcal{X} \times \{0,1\})^n$.

Theorem 16.3 *Let $\mathcal{Z} = \mathcal{X} \times \{0,1\}$ and $\mathcal{H} = \{h : \mathcal{X} \to \{0,1\}\}$ a hypothesis class with VC dimension d. Then, there exists an empirical risk minimizer $A : \mathcal{Z}^n \to \mathcal{H}$ such that* $\mathrm{CMI}(A) \leq d\log(n) + 2$.

16.2 Generalization via CMI

CMI can be used to generate generalization bounds.

Theorem 16.4 *Let $\mathcal{D}$ be a distribution on $\mathcal{Z}$. Let $A : \mathcal{Z}^n \to \mathcal{W}$ be a randomized algorithm. Let $l : \mathcal{W}\times\mathcal{Z} \to \mathbb{R}$ be an arbitrary (deterministic and measurable) function. Suppose there exists $\Delta : \mathcal{Z}^2 \to \mathbb{R}$ such that $|l(w,z_1) - l(w,z_2)| \leq \Delta(z_1,z_2)$ for all $z_1, z_2 \in \mathcal{Z}$ and $w \in \mathcal{W}$. Then,*

$$\left|\mathbb{E}_{Z\sim\mathcal{D}^n,A}(l(A(Z),Z) - l(A(Z),D))\right| \leq \sqrt{\frac{2}{n}\mathrm{CMI}_{\mathcal{D}}(A)\mathbb{E}_{(Z_1,Z_2)\sim\mathcal{D}^2}\left(\Delta\left(Z_1,Z_2\right)^2\right)}.$$

A tighter stronger statement can be obtain by losing a factor in the bound.

Theorem 16.5 *Let $\mathcal{D}$ be a distribution on $\mathcal{Z}$. Let $A : \mathcal{Z}^n \to \mathcal{W}$ be a randomized algorithm. Let $l : \mathcal{W}\times\mathcal{Z} \to \mathbb{R}$ be an arbitrary function. Suppose there exists $\Delta : \mathcal{Z}^2 \to \mathbb{R}$ such that $|l(w,z_1) - l(w,z_2)| \leq \Delta(z_1,z_2)$ for all $z_1, z_2 \in \mathcal{Z}$ and $w \in \mathcal{W}$. Then,*

$$\mathbb{E}_{Z\sim\mathcal{D}^n,A}\left(|l(A(Z),Z) - l(A(Z),D)|\right) \leq \sqrt{\frac{2}{n}(\mathrm{CMI}_{\mathcal{D}}(A) + \log(2))\mathbb{E}_{(Z_1,Z_2)\sim\mathcal{D}^2}\left(\Delta(Z_1,Z_2)^2\right)}$$

Part IV
Appealing to Gradients

17 Introduction

Gradients form the foundation of implementing deep neural network architectures. It is through gradient descent methods that one is able to train these large networks to perform well on data sets. These methods work by understanding the gradient of a loss function at training examples with respect to the training examples. Therefore, any of the learned features of the network are discovered within these gradients. The properties of the trained network are a product of the evolution of these gradients. Designing tools to investigate how the learning algorithm manipulates these gradients can provide a lot of insight into the features learned by the network, the processes underlying the training process, the network's capacity to generalize, and areas of exploitation to refine the training process.

18 Stiffness

An idea introduced in [13] is that of stiffness, which is used to

1. explain network generalization,
2. uncover semantically meaningful groups of data points,
3. explore the effects of learning rates on the function learned by the network, and
4. defining and measuring so-called dynamical critical length, ξ.

It will turn out that the definition of stiffness will

1. relate directly to network generalization,
2. will be sensitive to the semantically meaningful content of inputs, and
3. capture information about the loss landscape.

18.1 Sign and Cosine Stiffness

Consider a functional approximation f, parameterized by a trainable parameter W used to conduct classification tasks, with data points X and true labels y. Let $\mathcal{L}(f_W(X), y))$ be the loss function, and let the gradient of this loss function with respect to the parameters be

$$\bar{g} = \nabla_W \mathcal{L}(f_W(X), y).$$

Gradient descent uses this quantity to update the weights of the functional approximation.

Definition 18.1 *For two data points (X_1, y_1) and (X_2, y_2) define the sign stiffness to be*

$$S_{\text{sign}}\left((X_1, y_1), (X_2, y_2); f\right) = \mathbb{E}\left(\text{sign}\left(\bar{g}_1 \cdot \bar{g}_2\right)\right).$$

Definition 18.2 *For two data points (X_1, y_1) and (X_2, y_2) define the cosine stiffness to be*

$$S_{\cos}\left((X_1, y_1), (X_2, y_2); f\right) = \mathbb{E}\left(\cos\left(\bar{g}_1 \cdot \bar{g}_2\right)\right),$$

where

$$\cos\left(\bar{g}_1 \cdot \bar{g}_2\right) = \frac{\bar{g}_1 \cdot \bar{g}_2}{|\bar{g}_1||\bar{g}_2|}.$$

Remark 18.3 *Note the stiffness is dependent on the underlying distributions of the dataset from which X_1 and X_2 are drawn.*

18.2 Intuition for Stiffness

To understand what these definitions mean intuitively consider two data points and their ground truth labels (X_1, y_1) and (X_2, y_2). Compute the gradient of the loss function with respect to the parameters at these data points as

$$\bar{g}_1 \nabla_W \mathcal{L}(f_W(X_1), y_1) \text{ and } \bar{g}_2 = \nabla_W \mathcal{L}(f_W(X_2), y_2).$$

Now consider

$$\Delta\mathcal{L}_1 = -\epsilon \nabla_\epsilon \mathcal{L}(f_{W-\epsilon\bar{g}_1}(X_1), y_1) = -\epsilon \bar{g}_1 \cdot \bar{g}_1 + O(\epsilon^2).$$

By construction as $\epsilon \to 0$ it follows that $\Delta\mathcal{L}_1 < 0$. However, stiffness concerns itself with the behaviour of

$$\Delta\mathcal{L}_2 = -\epsilon \nabla_\epsilon \mathcal{L}(f_{W-\epsilon\bar{g}_1}(X_2), y_2) = -\epsilon \bar{g}_1 \cdot \bar{g}_2 + O(\epsilon^2),$$

as ϵ (the same as that used in $\Delta\mathcal{L}_1$) tends to 0. A positive stiffness means that in this limit $\Delta\mathcal{L}_2 < 0$, and so the parameter updates caused by SGD for these two data points support each other. The degree to which they support each other is directly related to the quantity $\bar{g}_1 \cdot \bar{g}_2$.

18.3 Class Membership Stiffness

It turns out that sign stiffness will be more suitable for identifying stiffness between classes, whereas cosine stiffness will be more useful for identifying within-class stiffness. Stiffness can be evaluated in different ways. Firstly, there is the train-train method where the two points are chosen from the train set. In the train-val evaluation, one point is taken from the train set and the other from the validation set. Then, in the val-val evaluation, both points are taken from the validation set. Clearly, generalization is directly related to train-val stiffness. Suppose that our set of data points X contains semantic classes $c_1, c_2, \ldots, c_n$.

Definition 18.4 *Define the class stiffness, C, to be a matrix with entries*

$$[C]_{ij} = \mathbb{E}_{X_1 \in c_i, X_2 \in c_j, X_1 \neq X_2}(S((X_1, y_1), (X_2, y_2))).$$

The entries of this matrix can be interpreted as follows,

- The on-diagonal elements correspond to the suitability of the gradient update to the members of the class itself. Hence, these entries give an idea of within-class generalization.
- The off-diagonal elements express the amount of improvement transferred from one class to another.

Using these interpretations we make the following definitions.

Definition 18.5 *For a set of data points X with n semantic classes summarize the between-class stiffness by*

$$S_{\text{between classes}} = \frac{1}{n(n-1)} \sum_i \sum_{i \neq j} [C]_{ij}.$$

Definition 18.6 *For a set of data points X with n semantic classes summarize the within-class stiffness by*

$$S_{\text{within classes}} = \frac{1}{n} \sum_i [C]_{ii}.$$

When within-class stiffness drops below 1, the generality of the features improved does not extend to even the class itself, suggesting that overfitting may be occurring.

18.4 Super-Classes

From the context available data set classes may be categorised into groups that share more meta semantic properties. These categories are called super-classes, and similarly, we call categories of super-classes super-super-classes. Empirically, it is been observed that stiffness between these clustered groups is higher than expected, suggesting that stiffness can be used as a measure of related semantic content between classes.

18.5 Stiffness and Distance

The relationship between stiffness and absolute distance between the points in the input space can also be investigated. If we normalize our data to the unit sphere we can use the metric

$$\text{distance}\left(\bar{X}_1, \bar{X}_2\right) = 1 - \frac{\bar{X}_1 \cdot \bar{X}_2}{|\bar{X}_1||\bar{X}_2|}.$$

Note that at a distance of zero, the stiffness is one. Define the threshold distance ξ to be the point at which on average the stiffness goes to zero. Observing ξ as a function of distance and learning rate enables one to estimate the size of the stiff regions of a neural network.

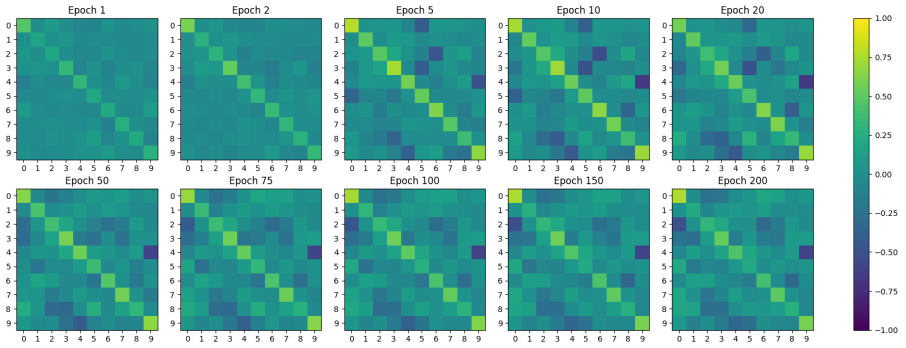

Figure 8: Evolution of the stiffness between the classes of MNIST data sets during training.

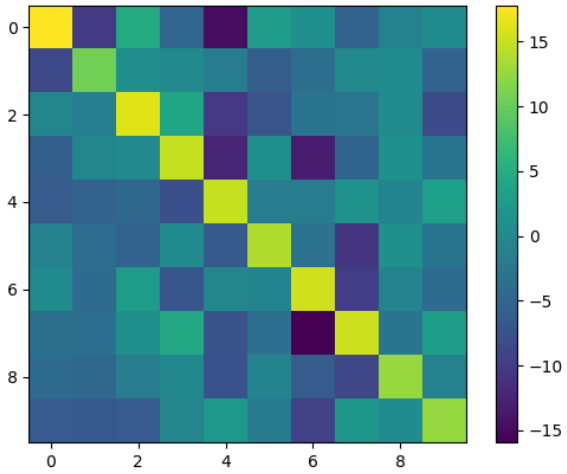

Figure 9: The Last layer activity of a neural network trained on MNIST.

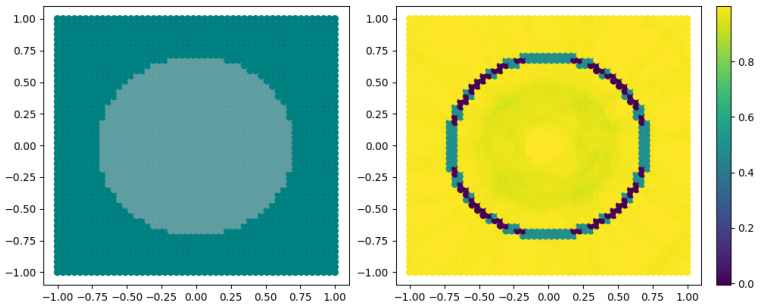

Figure 10: The stiffness of points within a neighbourhood can be used to identify the learned boundary of a classification problem.

18.6 Neighbourhoods Stiffness

The stiffness in neighbourhoods can be used to identify features of the dataset. Points along the boundary of a class for example will experience differing gradients to its neighbours. Performing weighted averages of the stiffness between points in a neighbourhood can therefore identify the boundary of features within a dataset.

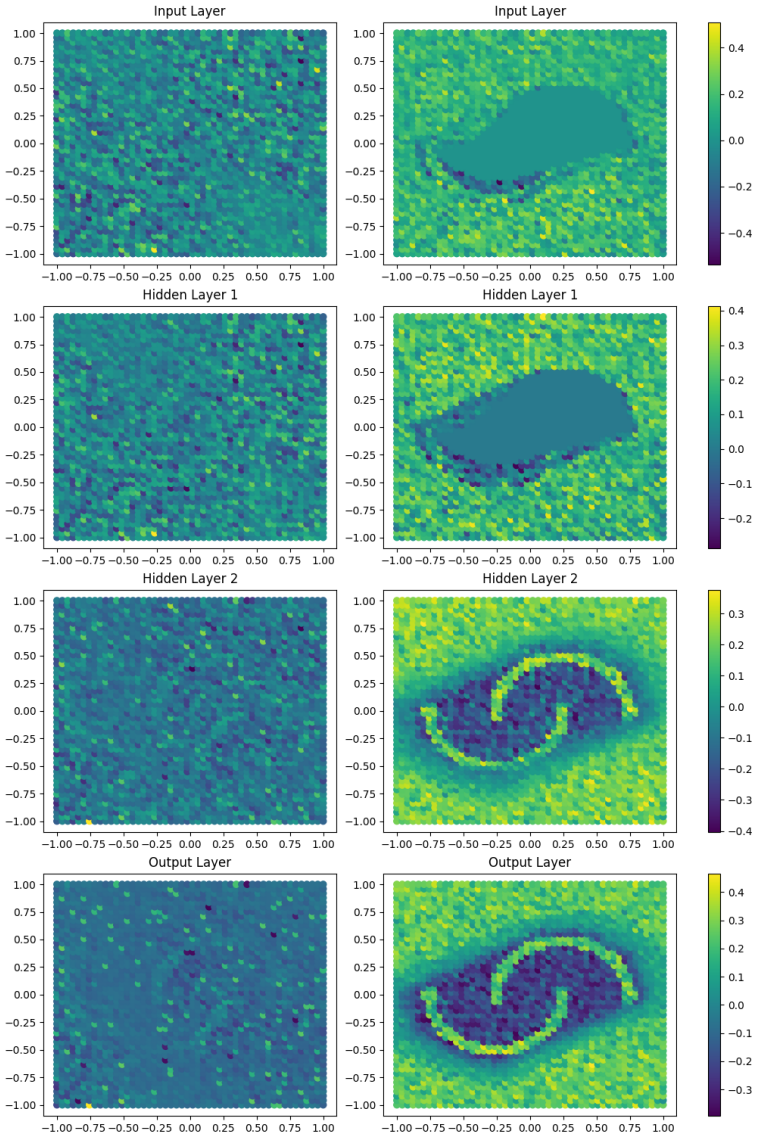

Figure 11: Average stiffness between an example and a random sample from the training data containing samples from both categories.

19 Neural Tangent Kernels

Tangent kernels are used to understand what function the neural networks are learning. As neural networks improve performance by altering their parameters in response to how a loss function evaluated at different data points changes with respect to those parameters. Therefore, to discern the features captured by the network it is important to look at how the gradients of the functions represented by the network interact at different points. Intuitively, at data points with similar features, these functions should have gradients that interact strongly in the parameter space as these are reinforced by the learning algorithm. In [16] this intuition is formalized and then also applied to give of neural network complexity.

19.1 Tangent Features and Kernels

Let $\mathcal{F}$ be a class of scalar functions parameterised by $\mathbf{w} \in \mathbb{R}^P$, $f_{\mathbf{w}} : \mathcal{X} \to \mathbb{R}$.

Definition 19.1 *The tangent features of a scalar function $f_{\mathbf{w}} : \mathcal{X} \to \mathbb{R}$ is the gradient with the respect to the parameters,*

$$\Phi_{\mathbf{w}}(\mathbf{x}) = \nabla_{\mathbf{w}} f_{\mathbf{w}}(\mathbf{x}) \in \mathbb{R}^P.$$

With the corresponding tangent kernel given by

$$k_{\mathbf{w}}(\mathbf{x}, \tilde{\mathbf{x}}) = \langle \Phi_{\mathbf{w}}(\mathbf{x}), \Phi_{\mathbf{w}}(\tilde{\mathbf{x}}) \rangle$$

One can then quantify the changes in the output of the function as a result of perturbations in the weights as

$$\delta f_{\mathbf{w}}(\mathbf{x}) = \langle \delta\mathbf{w}, \Phi_{\mathbf{w}}(\mathbf{x}) \rangle + O\left(\|\delta\mathbf{w}\|^2 \right).$$

To characterise the prominent directions in parameter space look at the eigenvalue decomposition of the covariance matrix

$$g_{\mathbf{w}} = \mathbb{E}_{\mathbf{x}\sim\rho} \left(\Phi_{\mathbf{w}}(\mathbf{x}) \Phi_{\mathbf{w}}(\mathbf{x})^\top \right)$$

where ρ is the distribution of the input. That is, consider

$$g_{\mathbf{w}} = \sum_{j=1}^{P} \lambda_{\mathbf{w}j} \mathbf{v}_{\mathbf{w}j} \mathbf{v}_{\mathbf{w}j}^\top.$$

Given inputs $\mathbf{x}_i$, and the corresponding vector outputs $f_{\mathbf{w}}(\mathbf{x}_i)$ for $i = 1, \ldots, n$, the gradient descent update on the weights is given by

$$\delta\mathbf{w}_{\mathrm{GD}} = -\eta \nabla_{\mathbf{w}} L$$

for some loss function L. The resulting function updates can be linearly approximated by

$$\delta f_{\mathrm{GD}}(\mathbf{x}_i) = \langle \delta\mathbf{w}_{\mathrm{GD}}, \Phi_{\mathbf{w}}(\mathbf{x}_i) \rangle,$$

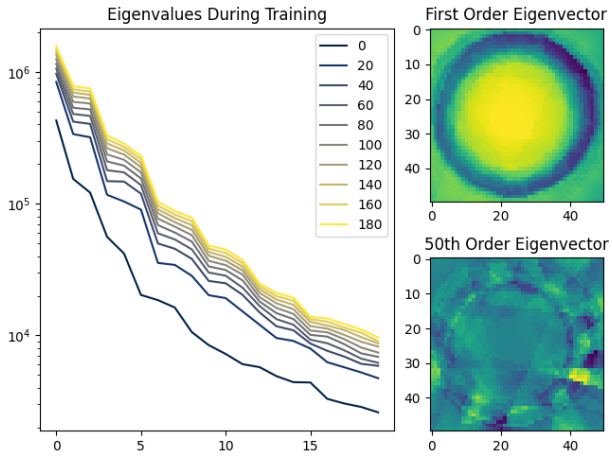

Figure 12: **Left:** Eigenvalues of tangent kernel during training. **Right:** Eigenvectors of the tangent kernel at the end of training.

which can be decomposed in the eigenbasis of the tangent kernel,

$$u_{\mathbf{w}j}(\mathbf{x}) = \frac{1}{\sqrt{\lambda_{\mathbf{w}j}}} \langle \mathbf{v}_{\mathbf{w}j}, \Phi_{\mathbf{w}}(\mathbf{x}) \rangle.$$

Lemma 19.2 *The function updates decompose as* $\delta f_{GD} = \sum_{j=1}^{P} \delta f_j u_{\mathbf{w}j}(\mathbf{x})$ *with*

$$\delta f_j = -\eta \lambda_{\mathbf{w}j} (\mathbf{u}_{\mathbf{w}j})^\top (\nabla_{f_{\mathbf{w}}} L),$$

where $\mathbf{u}_{\mathbf{w}j} = (u_{\mathbf{w}j}(\mathbf{x}_1) \dots u_{\mathbf{w}j}(\mathbf{x}_n))^\top \in \mathbb{R}^n$ *and* $\nabla_{f_{\mathbf{w}}}$ *is the gradient with respect to sample outputs.*

Illustrating how the eigenvalues of the tangent kernel act as mode-specific re-scaling of the learning rate. Introducing a local bias to prioritize learning functions within the top eigenspaces of the kernel. Hence, knowing the properties of the tangent kernel enables the understanding of what the network has learned. An implementation of the tangent kernel to do just that can be found here.

19.2 Application as a Complexity Measure

Consider scalar functions $f_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \Phi(\mathbf{x}) \rangle$ that are linearly parameterised by $\mathbf{w} \in \mathbb{R}^p$. When given n inputs, the tangent features $\Phi(\mathbf{x}_i) \in \mathbb{R}^P$ yield a feature matrix $\Phi \in \mathbb{R}^{n \times P}$. Recall, the empirical Rademacher complexity for a sample set S and a class of functions $\mathcal{F}$ is given by

$$\hat{\mathfrak{R}}_S(\mathcal{F}) = \frac{1}{m} \mathbb{E}_{\xi \in \{\pm 1\}^m} \left(\sup_{f \in \mathcal{F}} \sum_{i=1}^{m} \xi_i f(\mathbf{x}_i) \right).$$

Therefore, by controlling the capacity of networks, tighter generalization bounds can be obtained as the Rademacher complexity depends on the size of the class $\mathcal{F}$. Common methods of restricting capacity appeal to bounding the norm of the weight vector. However, the implicit bias induced by the training algorithm identified by the tangent kernel can be used.

Definition 19.3 *Let $A \in \mathbb{R}^{P\times P}$ be an invertible matrix. The vector norm of $\mathbf{w} \in \mathbb{R}^P$ induced by A is given by $\|\mathbf{w}\|_A := \sqrt{\mathbf{w}^\top g_A \mathbf{w}}$ where $g_A = AA^\top$.*

Proposition 19.4 *For the restricted class of functions $\mathcal{F}^A_{M_A} = \{f_\mathbf{w} : \mathbf{x} \mapsto \langle \mathbf{w}, \Phi(\mathbf{x})\rangle : \|\mathbf{w}\|_A \leq M_A\}$ and sample set S,*

$$\hat{\mathfrak{R}}_S\left(\mathcal{F}^A_{M_A}\right) \leq \frac{M_A}{n}\left\|A^{-1}\Phi^\top\right\|_F,$$

where $\left\|A^{-1}\Phi^\top\right\|_F$ is the Frobenius norm of the re-scaled feature matrix.

As the training algorithm is iterative, consider $f_\mathbf{w} = \sum_t \delta f_{\mathbf{w}_t}$ with $\delta f_{\mathbf{w}_t} = \langle \delta\mathbf{w}_t, \Phi(\mathbf{x})\rangle$. Apply a local constraint on the parameter updates by considering the restricted class of functions

$$\mathcal{F}^\mathbf{A}_\mathbf{m} = \left\{ f_\mathbf{w} : \mathbf{x} \mapsto \sum_t \langle \delta\mathbf{w}_t, \Phi(\mathbf{x})\rangle : \|\delta\mathbf{w}_t\|_{A_t} \leq m_t \right\}.$$

Theorem 19.5 *Given any sequences $\mathbf{A}$ and $\mathbf{m}$ of invertible matrices $A_t \in \mathbb{R}^{P\times P}$ and positive numbers $m_t > 0$ the following bound holds*

$$\hat{\mathfrak{R}}\left(\mathcal{F}^\mathbf{A}_\mathbf{m}\right) \leq \sum_t \frac{m_t}{n}\left\|A_t^{-1}\Phi^\top\right\|_F.$$

Remark 19.6 *Using the linear re-parameterisation $\mathbf{w} \mapsto A^\top\mathbf{w}$ and $\Phi \mapsto A^{-1}\Phi$ consider the restricted function class*

$$\mathcal{F}^\Phi_\mathbf{m} = \left\{ f_\mathbf{w} : \mathbf{x} \mapsto \sum_t \left\langle \tilde{\delta}\mathbf{w}_t, \Phi_t(\mathbf{x})\right\rangle : \left\|\tilde{\delta}\mathbf{w}_t\right\|_2 \leq m_t \right\}.$$

In this case a similar bound holds,

$$\hat{\mathfrak{R}}_S \leq \sum_t \frac{m_t}{n}\|\Phi_t\|_F.$$

These bounds are for fixed sequences of feature maps. However, the results can be extrapolated to the non-deterministic setting for which [16] proposes the following heuristic measure for the complexity of neural networks,

$$C(f_\mathbf{w}) = \sum_t \|\delta\mathbf{w}_t\|\|\Phi_t\|_F.$$

Where Φ_t is the learned tangent feature matrix at iteration t, and $\|\delta\mathbf{w}_t\|_2$ is the norm of the SGD update.

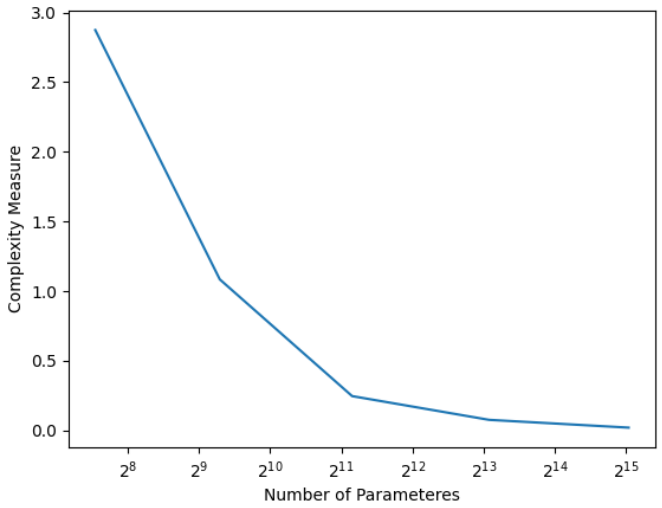

Figure 13: Complexity of fully connected networks with a certain number of parameters trained up to the point of 0.01 Cross Entropy Loss on a synthetic data set.

20 Geometric Complexity Measure

The Rademacher complexity is a measure of complexity that focuses on the entire hypothesis space rather than focusing on a function. Complexity notions such as VC dimension and parameter counting also focus on quantifying complexity for the entire hypothesis space. On the other hand, measures such as counting the number of linear pieces of ReLU networks and matrix norms, measure the complexity of the function independently from the task at hand. The work of [21] focuses on capturing function complexity over proposed datasets through geometrical arguments with the following proposed complexity measure.

Definition 20.1 *Let $g_\theta : \mathbb{R}^d \to \mathbb{R}^k$ be a neural network parameterised by θ. Let $g_\theta(x) = \alpha(f_\theta(x))$ where α is the last layer activation, and f_θ is its logit network. The Geometric Complexity (GC) of the network over a dataset D is the discrete Dirichlet energy of its logit network,*

$$\langle f_\theta, D \rangle = \frac{1}{|D|} \sum_{x \in D} \|\nabla_x f_\theta(x)\|_F^2,$$

where $\|\nabla_x f_\theta(x)\|_F$ is the Frobenius norm of the network Jacobian.

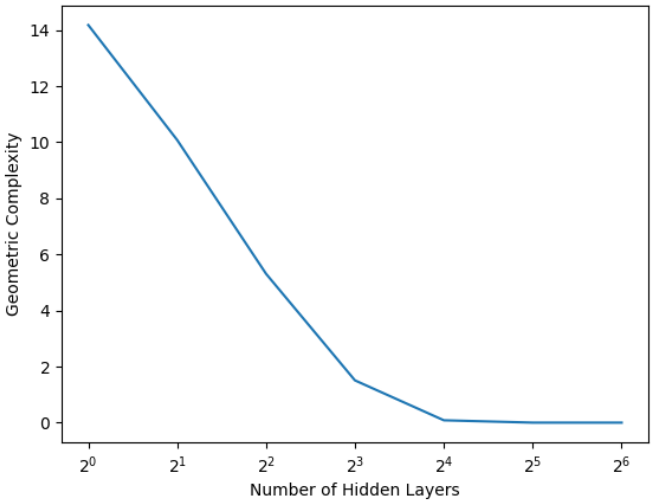

Figure 14: A plot of the geometric complexity of a fully connected ReLU network with a varying number of hidden layers.

20.1 GC and ReLU Networks

Let $g_\theta : \mathbb{R}^d \to \mathbb{R}^k$ be a ReLU network. As g_θ parameterises piece-wise linear function, the domain can be partitioned by subsets $X_i \subset \mathbb{R}^d$ such that f_θ is an affine map $A_i x + b_i$. Let $D_i = D \cap X_i$, then for every $x \in X_i$ it follows that $\|\nabla_x f_\theta(x)\|_F^2 = \|A_i\|_F^2$ so that

$$\langle f_\theta, D \rangle_G = \sum_i \left(\frac{n_i}{|D|} \right) \|A_i\|_F^2,$$

where $n_i = |D_i|$. Therefore, a stratified batch $B \subset D$ has a GC that coincides with the network. Hence, the GC on large batches can be used as a proxy for the GC of a network, making this a tractable measure of complexity to attain during training.

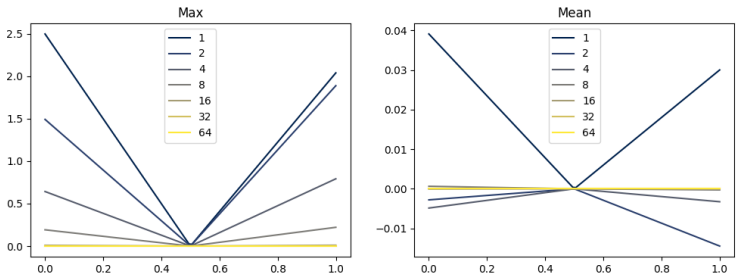

Figure 15: Determining the function of a ReLU network with a variable number of layers when initialized using the Glorot Initialization scheme.

21 Algorithmic Stability

The work of [20] focuses on the learning algorithm to understand network generalization. When taking an algorithmic perspective it is important to understand how gradient descent exploits patterns in the training data as networks are usually trained through gradient-based methods. The only point at which this exploitation could occur is in the parameter update step. A general update rule takes the form

$$w_{t+1} = w_t - \frac{\eta}{m} \sum_{i=1}^{m} g_i(w_t),$$

where $g_i(w_t)$ is the gradient of the loss function at example i evaluated at the parameter w_t, and η is the learning rate.

1. The average gradient $g(w_t)$ is strong in directions where per-example gradients are "similar".
2. Parameters that correspond to stronger directions change more.

When per-example gradients reinforce each other they are said to be coherent. A learning process is stable if changing an example in the training set doesn't affect the learned model in drastic ways. Strong directions in the average gradient are stable, as multiple examples reinforce each other. The stability of the learning process relates to the ability to generalize. That is, stable updates in the parameters should lead to good generalization. To approach generalization from the perspective of learning algorithms requires a quantitative notion of coherence.

21.1 Motivating Example

Consider the linear model $\hat{y} = w \cdot x = \sum_{i=1}^{6} w_i x_i$ which is fitted using the square loss $l(w) = \frac{1}{2}(y - \hat{y})$. Consider the datasets

$$\begin{bmatrix} i & & & x_i & & & & y_i \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 2 & 0 & -1 & 0 & 0 & 0 & -1 & -1 \\ 3 & 0 & 0 & -1 & 0 & 0 & -1 & -1 \\ 4 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 5 & 0 & 0 & 0 & 0 & -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} i & & & x_i & & & & y_i \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 0 & -1 & 0 & 0 & 0 & 0 & -1 \\ 3 & 0 & 0 & -1 & 0 & 0 & 0 & -1 \\ 4 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 5 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \end{bmatrix},$$

where the difference between the two only arises in the last entry of the x_i. Refer to the first one as "real" data and the second as "random" data, as with the first data set the last entry of the x_i correlates with the label y_i. However, with the other dataset, there is no correlation between any one entry of the x_i with the labels y_i and hence appears "random". Use the first 4 rows for training and the last for testing. Consider the average gradient of the loss function on each of these datasets. For the real data set the 6^{th} entry is reinforced by each example, on the other hand, in the random dataset this component is not enforced. Hence, during gradient descent, the real dataset causes a greater change in the 6^{th} component of the parameter compared to the random data set. It is reassuring then that the gradients of the data points reinforce the learnable feature so that the model can learn this. Refer to Figure 16

21.2 Coherence

Let $\mathcal{D}(z)$ denote the distribution of examples z from a finite set Z. Suppose the network has d trainable parameters. Let $l_z(w)$ be the loss of example $z \sim \mathcal{D}$ for parameter vector $w \in \mathbb{R}^d$ and let $l(w) = \mathbb{E}_{z\sim\mathcal{D}}(l_z(w))$ be the expected loss. Denote the gradient of the loss at example z by $g_z := (\nabla l_z)(w)$ and the average gradient over all examples by $g := (\nabla l)(w)$. Throughout η will denote the learning rate. The learning problem is trying to minimize the expected loss through a gradient descent algorithm where small descent steps $h = -\eta g$ are taken. Using Taylor approximations to get that

$$l(w+h) - l(w) \approx g \cdot h = -\eta g \cdot g = -\eta \mathbb{E}_{z\sim\mathcal{D}}(g_z) \cdot \mathbb{E}_{z\sim\mathcal{D}}(g_z) = -\eta \mathbb{E}_{z,z'\sim\mathcal{D}}(g_z \cdot g_{z'}).$$

Which motivates using the average pairwise dot product as a metric for coherence. With this setup, there are clear directions for improving the generalization of networks by modifying the process of gradient descent.

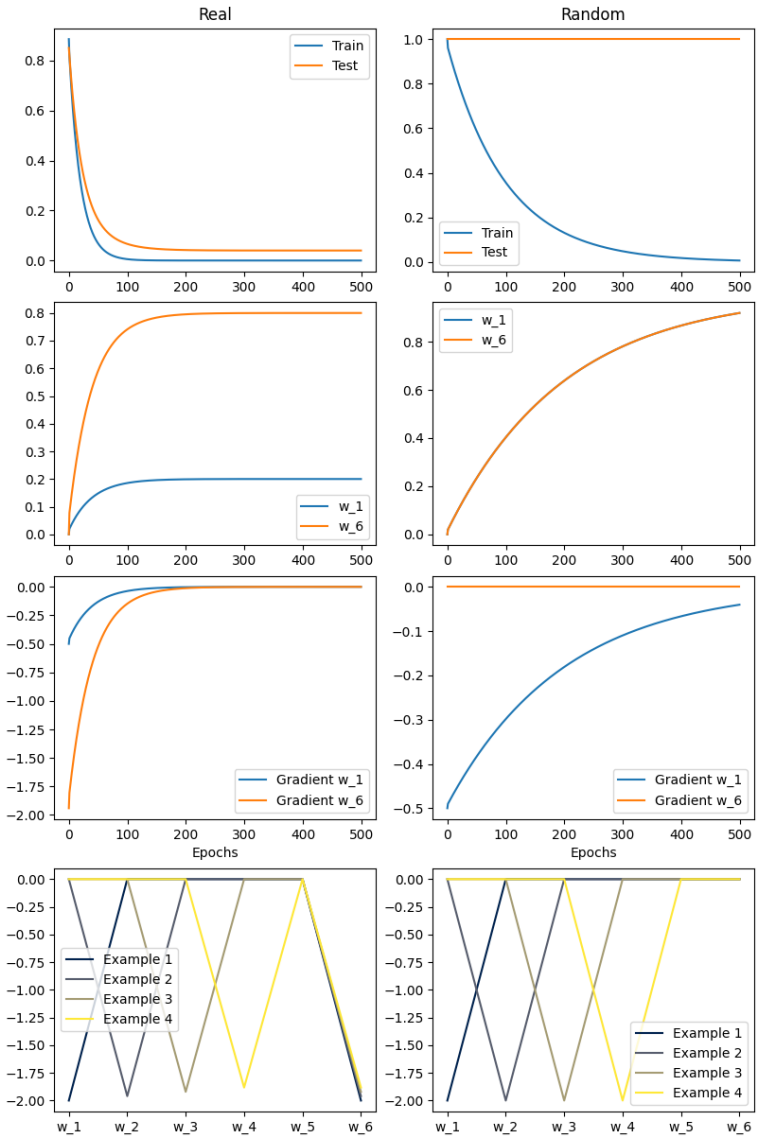

Figure 16: **First Row:** The training and test loss. **Second Row:** The value of the 6^{th} parameter during training. **Third Row:** Gradient of the 6^{th} parameter during training. **Fourth Row:** The value of the loss function at the different parameters for each example.

1. Making gradient descent more stable by combining per-example gradients using robust mean estimation techniques (i.e. use the median rather than sample mean) to eliminate weak directions in the gradients.
2. Using l^2 regularization as a means to reduce movement in weak directions.
3. As training progresses the gradients of fitted examples become negligible. Therefore, early stopping may prevent the fewer examples in the latter parts of training from dominating the average gradient and triggering over-fitting.

To use the average pairwise dot product as an interpretable metric it has to be normalized. For an individual loss l_z consider a step h_z down its gradient g_z at a parameter point w,

$$l_z(w + h_z) - l_z(w).$$

Using Taylor approximations and taking expectations over z yields

$$\mathbb{E}_{z\sim\mathcal{D}}\left(l_z(w + h_z) - l_z(w)\right) = -\eta\mathbb{E}_{z\sim\mathcal{D}}(g_z \cdot g_z).$$

This is the idealized reduction in loss, as per-example examples tend not to be aligned so steps are usually not taken down an individual losses gradient. Therefore, it serves as a reasonable scaling factor for the metric. Let α be the normalized metric for coherence defined as

$$\alpha(\mathcal{D}) := \frac{\mathbb{E}_{z,z'\sim\mathcal{D}}(g_z \cdot g_{z'})}{\mathbb{E}_{z\sim\mathcal{D}(g_z\cdot g_z)}}.$$

Theorem 21.1 *Let $\mathcal{V}$ be a probability distribution on a collection of m vectors in Euclidean space. Then, $0 \leq \alpha(\mathcal{V}) \leq 1$, where*

1. *$\alpha(\mathcal{V}) = 0$ if and only if $\mathbb{E}_{v\sim\mathcal{V}}(v) = 0$, and*
2. *$\alpha(\mathcal{V}) = 1$ if and only if all vectors are equal.*

Furthermore, for $0 \neq k \in \mathbb{R}$ we have

$$\alpha(k\mathcal{V}) = \alpha(\mathcal{V})$$

where $k\mathcal{V}$ is the distribution of random variables kv for $v \sim \mathcal{V}$.

Remark 21.2 *Typically one does not have access to the underlying distribution of the samples is unknown. Let α_m denote the coherence of a sample S where $|S| = m$.*

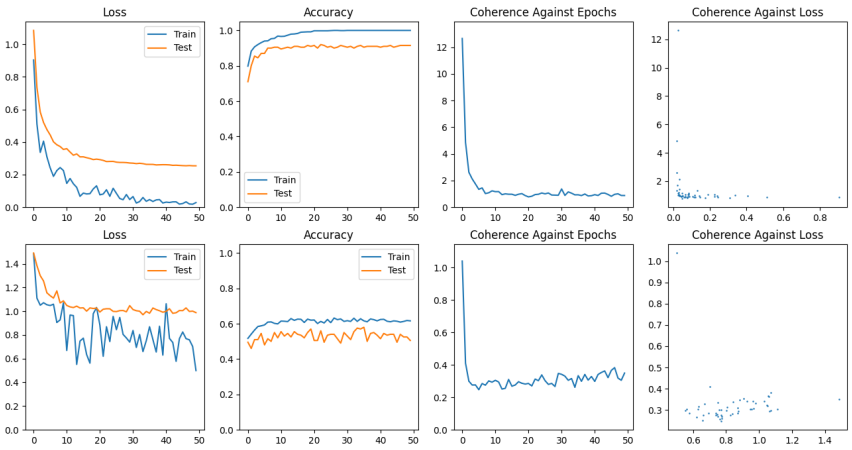

Figure 17: Coherence during training on MNIST where the plots on the bottom row are a result of 50% randomized labels.

Example 21.3 *Let S be a sample of size m, whose gradients g_i $(1 \leq i \leq m)$ are pairwise orthogonal. Then,*

$$\alpha_m = \frac{\frac{1}{m}\mathbb{E}_i(g_i \cdot g_i)}{\mathbb{E}_i(g_i \cdot g_i)} = \frac{1}{m}.$$

Therefore, in this case, α_m is independent of the g_i. Call $\frac{1}{m}$ the orthogonal limit of a sample of size m and denote it $\alpha_m^{\perp}$.

The quantity $\alpha_m/\alpha_m^{\perp}$ can be interpreted as the average number of samples that a particular gradient helps fit.

1. For an orthogonal sample $\alpha_m/\alpha_m^{\perp} = 1$. Examples are fitted independently.
2. For a perfectly coherent sample $\alpha_m/\alpha_m^{\perp} = m$. Examples help fit all other examples.
3. When $\alpha_m/\alpha_m^{\perp} = 0$, we have $\alpha_m = 0$ so on average no example helps fit any other example.
4. In the under-parameterized setting $d \ll m$ with a sample of $k \gg 1$ copies of orthogonal gradients in d-dimensional space (i.e. $m = kd \gg d$). We have that $\alpha_m = \frac{1}{d}$ and $\alpha_m/\alpha_m^{\perp} = k$. Hence, each example helps k other examples in the sample as expected.

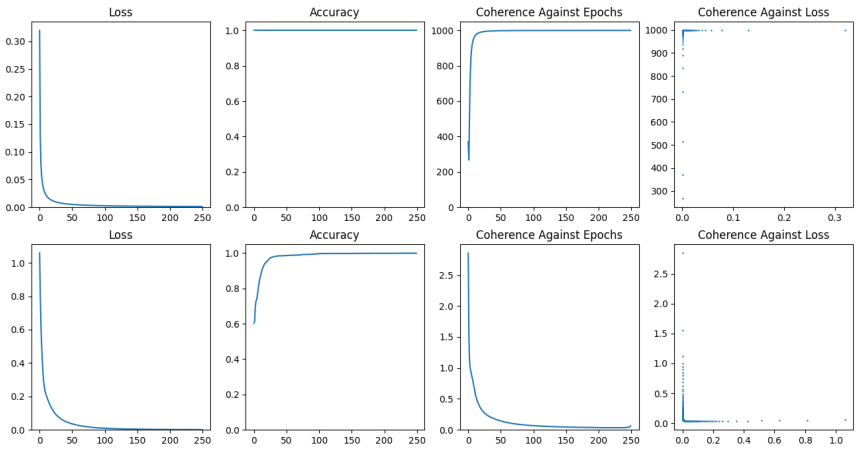

Figure 18: Training and coherence plots for (**Top Row**) data consisting of 1000 identical points and (**Bottom Row**) of distinct points forming two classes.

21.3 Application to the Generalization Gap

Small batch stochastic gradient descent is stable as each individual example is looked at so rarely when training is not run for too long. This is a dataset-independent argument for stability. In [20] a dataset-dependent argument for stability is given where coherence implies stability which in turn implies generalization. The expected generalization gap is the expected difference between training and test loss over samples of size m from $\mathcal{D}$, $\text{gap}(\mathcal{D}, m)$.

Theorem 21.4 *If stochastic gradient descent is run for T steps on a training set consisting of m examples drawn from distribution $\mathcal{D}$, then,*

$$|\text{gap}(\mathcal{D}, m)| \leq \frac{L^2}{m} \sum_{t=1}^{T} (\eta_k \beta)_{k=t+1}^{T} \cdot \eta_t \cdot \sqrt{2(1 - \alpha(w_{t-1}))}$$

where

- $\alpha(w)$ *denotes coherence at point w in parameter space,*
- w_t *is the parameter value seen at step t of gradient descent,*
- η_t *is the learning rate a step t of gradient descent,*
- $(\eta_k \beta)_{k=t_0}^{t_1} = \prod_{k=t_0}^{t_1} (1 + \eta_k \beta)$, *and*

- *L and β are Lipschitz constants.*

Remark 21.5

- *The bound is dependent on the training length, and size of the training set.*
- *High coherence early on in training is better than high coherence later on.*
- *The bound applies uniformly to stochastic and full-batch cases.*
- *The bound is only useful in a qualitative sense and is loose.*

Theorem 21.4 requires a formal development. First, recall that there is a domain of samples Z with $\mathcal{D}(Z)$ a distribution on Z from which training and test examples are sampled. With $l(w, z)$ the loss on example z with model parameters w. Machine learning then aims to minimize the population risk $R(w) = \mathbb{E}_{z\sim\mathcal{D}} l(w, z)$. When equipped with training samples $S = (z_1, \ldots, z_m)$ the empirical risk is minimized,

$$\hat{R}(w, S) := \frac{1}{m} \sum_{i\in[m]} l(w, z_i).$$

The aim is to bound expected generalization error using ideas in algorithmic stability,

$$\text{gap}(\mathcal{D}, m) := \mathbb{E}_{S\sim\mathcal{D}^m}\mathbb{E}_\theta \left(R(A_\theta(S)) - \hat{R}(A_\theta(S), S)\right)$$

where $A_\theta(S)$ is the model obtained by running the randomized training algorithm A on S, with θ being the sequence of decisions made by the stochastic processes of A. If we have a second sample $S' := (z'_1, \ldots, z'_m)$ then the expected stability of A is given by

$$\text{stab}(\mathcal{D}, m) := \mathbb{E}_{S\sim\mathcal{D}^m}\mathbb{E}_{S'\sim\mathcal{D}^m}\mathbb{E}_\theta \left(\frac{1}{m} \sum_{i\in[m]} \left(l\left(A_\theta\left(S^{(i)}\right), z_i\right) - l(A_\theta(S), z_i)\right)\right)$$

where $S^{(i)} = \left(z_1, \ldots, z_{i-1}, z'_i, z_{i+1}, \ldots, z_m\right)$.

Theorem 21.6 *When training with m samples from a distribution $\mathcal{D}$,*

$$\text{gap}(\mathcal{D}, m) = \text{stab}(\mathcal{D}, m).$$

To proceed we make some assumptions on the smoothness of the loss function. With $g(w, z) = \nabla_w l(w, z)$ assume that for all $z \in Z$,

- $|l(w, z) - l(w', z)| \leq L\|w - w'\|$,
- $\|g(w, z)\| \leq L$, and

- $\|g(w,z) - g(w',z)\| \leq \beta\|w - w'\|$.

Suppose during T steps of gradient descent training on the datasets S and $S^{(i)}$ the weight parameters $w_0, \ldots, w_T$ and $w'_0, \ldots, w'_T$ respectively are observed. In the context of stochastic gradient descent suppose that the mini-batches of size b are sampled randomly without-replacement. Use $I_t(\theta)$ to denote the indicator random variable for the event that the i^{th} training example is selected in the mini-batch of time-step t. Now starting from the update rule

$$w_t = w_{t-1} - \eta_t \frac{1}{b} \sum_{j \in [b]} g(w_{t-1}, z_{tj})$$

we deduce that

$$\delta_t \leq \delta_{t-1} + \eta_t \|\Delta''_t(b)\|$$

where

- $\delta_t = \|w_t - w'_t\|$, and
- $\Delta''_t(b) = \frac{1}{b} \sum_{j \in [b]} \left(g\left(w_{t-1}, z_{tj}\right) - g\left(w'_{t-1}, z'_{tj}\right)\right)$.

Lemma 21.7 *For $t \in [T]$, it follows that*

$$\delta_t \leq (1 + \eta_t \beta) \cdot \delta_{t-1} + \frac{\eta_t I_t(\theta)}{b} \|g(w_{t-1}, z_i) - g(w_{t-1}, z'_i)\|.$$

Lemma 21.8 *It follows that,*

$$\delta_T \leq \sum_{t \in [T]} \left(\frac{\eta_t I_t(\theta)}{b} \|g(w_{t-1}, z_i) - g(w_{t-1}, z'_i)\| \prod_{k=t+1}^{T} (1 + \eta_k \beta) \right).$$

Lemma 21.9 *It follows that,*

$$\mathbb{E}_{S \sim \mathcal{D}^m} \mathbb{E}_{z'_i \sim \mathcal{D}} \mathbb{E}_\theta \left(\left| l\left(A_\theta\left(S^{(i)}\right), z_i\right) - l(A_\theta(S), z_i) \right| \right) \leq \frac{L^2}{m} \sum_{t \in [T]} (\eta_k \beta)_{k=t+1}^{T} \eta \sqrt{2(1 - \alpha(w_{t-1}))}.$$

Theorem 21.10

$$|\text{stab}(\mathcal{D}, m)| \leq \frac{L^2}{m} \sum_{t=1}^{T} (\eta_k \beta)_{k=t+1}^{T} \cdot \eta_t \cdot \sqrt{2(1 - \alpha(w_{t-1}))}.$$

Theorem 21.4 is a direct consequence of Theorem 21.10 by applying Theorem 21.6. Theorem 21.4 can be applied in different learning regimes to yield the following corollaries.

Corollary 21.11 *If the steps are fixed, $\eta_t = \eta$, then*

$$|\text{gap}(\mathcal{D}, m)| \leq \frac{L^2 \eta}{m} \sum_{t=1}^{T} \exp((T - t)\eta\beta) \cdot \sqrt{2(1 - \alpha(w_{t-1}))}.$$

Corollary 21.12 *If the step size decay linearly, that is for some $\eta > 0$ we have $\eta_t \leq \frac{\eta}{t}$, then*

$$|\text{gap}(\mathcal{D}, m)| \leq \frac{L^2 \eta T^{\eta\beta}}{m} \sum_{t=1}^{T} \sqrt{2(1 - \alpha(w_{t-1}))}.$$

21.4 Coherence of Fully Connected Neural Networks

In [20] coherence of fully connected networks is investigated. A fully connected one-layer neural network with 2048 neurons is trained using SGD and a learning rate of 0.1. The evolution of $\alpha_m/\alpha_m^{\perp}$ is investigated during training on the MNIST dataset. The evolution of $\alpha_m/\alpha_m^{\perp}$ is also noted for the same network but trained on random data. The random data is generated by assigning random labels to the MNIST dataset. There are several issues that inhibit the ability to compute coherence through training:

1. Batch normalization means that it is not possible to recover per-example gradients.
2. Large training sets mean it would be impractical to consider computing the coherence even if access to per-example gradients was granted.

To alleviate these issues a method to approximate the $\alpha_m/\alpha_m^{\perp}$ from the batch data is employed.

Theorem 21.13 *Let $v_1, \ldots, v_k$ be k* i.i.d *variables drawn from $\mathcal{V}$. Let $\mathcal{W}$ denote the distribution of the random variable $w = \frac{1}{k}\sum_{i=1}^{k} v_i$. Then,*

$$\alpha(\mathcal{W}) = \alpha(k\mathcal{W}) = \frac{k\alpha(\mathcal{V})}{1 + (k-1)\alpha(\mathcal{V})}$$

Furthermore, $\alpha(\mathcal{W}) \geq \alpha(\mathcal{V})$ with equality if and only if $\alpha(\mathcal{V}) = 0$ or $\alpha(\mathcal{V}) = 1$.

Using this Theorem 21.13 a tractable method for approximating coherence from gradients can be computed at the batch level rather than the per-example level.

1. Compute the gradients for each batch.
2. Compute coherence of the $\frac{m}{k}$ batch in the usual way, $\alpha(\mathcal{W})$.
3. Re-arrange the identity of Theorem 21.13,

$$\alpha(\mathcal{V}) = \frac{\alpha(\mathcal{W})}{k - (k-1)\alpha(\mathcal{W})}$$

   and use this to get a value for $\alpha(\mathcal{V})$.
4. Compute $\alpha_m/\alpha_m^{\perp}$ as $m\alpha(\mathcal{V})$.

The code for this implementation can be founder here.

21.5 Suppressing Weak Gradients

In theory, suppressing weak directions in the average gradient should lead to less over-fitting and better generalization. Indeed current regularization techniques perform this implicitly, however, winsorized gradient descent (WGD) aims to do it explicitly.

21.5.1 Winsorized Gradient Descent Let $w_t^{(j)}$ be the j^{th} component of the trainable parameter w_t and let $g_i^{(j)}(w_t)$ be the j^{th} component of the gradient of the i^{th} example at w_t. Let $c \in [0, 50]$ be the level of winsorization. Define $l^{(j)}$ and $u^{(j)}$ be the c^{th} and $(100 - c)^{\text{th}}$ percentile of $g_i^{(j)}(w_t)$ respectively. Then the update rule for WGD is

$$w_{t+1}^{(j)} = w_t^{(j)} - \frac{\eta}{m} \sum_{i=1}^{m} \text{clip}\left(g_i^{(j)}(w_t), l^{(j)}, u^{(j)}\right)$$

where $\text{clip}\left(g_i^{(j)}(w_t), l^{(j)}, u^{(j)}\right) := \min(\max(x, l), u)$. The parameter c acts as a threshold as to what to consider an outlier. A similar stochastic version, winsorized stochastic gradient descent (WSGD), can also be employed to reduce the computational costs. In any case, WGD incurs greater computational costs as it necessitates storing all per-example (batch) gradients to perform the update. Furthermore, higher winsorization values lead to optimization instability as training accuracy is observed to fall after a certain point. To address these issues the median of means algorithm is used:

1. Divide the sample into k groups,
2. Compute sample mean of each group,
3. Return the median of these k means.

Part V
Other Approaches to Investigating Generalization

22 Unit-Wise Capacity Measures

The work of [10] looks at two-layer fully connected ReLU networks. The inputs have dimension d, outputs have dimension c, and layers have h hidden units. The function of the network is represented as $f_{\mathbf{V},\mathbf{U}}(\mathbf{x}) = \mathbf{V}(\mathbf{U}\mathbf{x})_+$ where $\mathbf{x} \in \mathbb{R}^d, \mathbf{U} \in \mathbb{R}^{h\times d}$ and $\mathbf{V} \in \mathbb{R}^{c\times h}$. With $\mathbf{u}_i$ denoting the incoming weights to hidden unit i and $\mathbf{v}_i$, the outgoing weights to hidden unit i. The network is initialized with weights $\mathbf{U}_0$ and $\mathbf{V}_0$ where $\mathbf{u}_i^0$ and $\mathbf{v}_i^0$ denote the corresponding weights as defined above. The network will be used to perform c-class classification, where the maximum output of a score function gives the predicted label. Define this score function to be the margin operator $\mu : \mathbb{R}^c \times [c] \to \mathbb{R}$ which scores an output $f(\mathbf{x})$ for each label $y \in [c]$ according to $\mu(f(\mathbf{x}), y) = f(\mathbf{x})[y] - \max_{i\neq y} f(\mathbf{x})[i]$. To train the network we use the ramp loss,

$$l_\gamma(f(\mathbf{x}), y) = \begin{cases} 0 & \mu(f(\mathbf{x}), y) > \gamma \\ \frac{\mu(f(\mathbf{x}),y)}{\gamma} & \mu(f(\mathbf{x}), y) \in [0, \gamma] \\ 1 & \mu(f(\mathbf{x}), y) < 0. \end{cases}$$

Hence, the following definitions of error emerge,

- $L_\gamma(f) = \mathbb{P}_{(\mathbf{x},y)\sim\mathcal{D}}(l_\gamma(f(\mathbf{x}, y))$, the expected margin loss of a predictor $f(\cdot)$, for distribution $\mathcal{D}$ and margin $\gamma > 0$.
- $\hat{L}_\gamma(f)$, the empirical margin loss.
- $L_0(f)$, the expected risk.
- $\hat{L}_0(f)$, the expected training error.

For a hypothesis class $\mathcal{H}$, a training set S and with $l_\gamma \circ \mathcal{H} := \{l_\gamma \circ h : h \in \mathcal{H}\}$ Theorem 4.3 can be applied to obtain.

Theorem 22.1 *With probability* $1 - \delta$ *over a training set of size* m*,*

$$L_0(f) \leq \hat{L}_\gamma(f) + 2\mathfrak{R}_S(l_\gamma \circ \mathcal{H}) + 3\sqrt{\frac{\log\left(\frac{2}{\delta}\right)}{2m}}$$

holds for any $f \in \mathcal{H}$*.*

Definition 22.2 *The unique capacity of a hidden unit* i *is* $\beta_i = \left\|\mathbf{u}_i - \mathbf{u}_i^0\right\|_2$*.*

Definition 22.3 *The unit impact of a hidden unit i is $\alpha_i = \|\mathbf{v}_i\|_2$.*

Definition 22.4 *Let $\mathcal{W}$ be the restricted set of parameters*

$$\mathcal{W} = \left\{(\mathbf{V}, \mathbf{U}) : \mathbf{V} \in \mathbb{R}^{c\times h}, \mathbf{U} \in \mathbb{R}^{h\times d}, \|\mathbf{v}_i\| \leq \alpha_i, \left\|\mathbf{u}_i - \mathbf{u}_i^0\right\|_2 \leq \beta_i\right\}$$

and let $\mathcal{F}_{\mathcal{W}}$ be the corresponding class of neural networks

$$\mathcal{F}_{\mathcal{W}} = \{f(\mathbf{x}) = \mathbf{V}(\mathbf{U}\mathbf{x})_+ : (\mathbf{V}, \mathbf{U}) \in \mathcal{W}\}$$

Theorem 22.5 *Given a training set $S = \{\mathbf{x}_i\}_{i=1}^m$ and $\gamma > 0$, then*

$$\begin{aligned}\mathfrak{R}_S(l_\gamma \circ \mathcal{F}_{\mathcal{W}}) &\leq \frac{2\sqrt{2c}+2}{\gamma m}\sum_{j=1}^h \alpha_j\left(\beta_j\|\mathbf{X}\|_F + \left\|\mathbf{u}_j^0\mathbf{X}\right\|_2\right)\\ &\leq \frac{2\sqrt{2c}+2}{\gamma m}\|\alpha\|_2\left(\|\beta\|_2\sqrt{\frac{1}{m}\sum_{i=1}^m\|\mathbf{x}_i\|_2^2} + \sqrt{\frac{1}{m}\sum_{i=1}^m\left\|\mathbf{U}^0\mathbf{x}_i\right\|_2^2}\right).\end{aligned}$$

Theorem 22.6 *For any $h \geq 2, \gamma > 0, \delta \in (0,1)$ and $\mathbf{U}^0 \in \mathbb{R}^{h\times d}$ with probability $1-\delta$ over the training set $S = \{\mathbf{x}_i\}_{i=1}^m \subset \mathbb{R}^d$, for any function $f(\mathbf{x}) = \mathbf{V}(\mathbf{U}\mathbf{x})_+$ such that $\mathbf{V} \in \mathbb{R}^{c\times h}$ and $\mathbf{U} \in \mathbb{R}^{h\times d}$,*

$$\begin{aligned}L_0(f) &\leq \hat{L}_\gamma(f) + \tilde{O}\left(\frac{\sqrt{c}\|\mathbf{V}\|_F\left(\left\|\mathbf{U}-\mathbf{U}^0\right\|_F\|\mathbf{X}\|_F + \left\|\mathbf{U}^0\mathbf{X}\right\|_F\right)}{\gamma m} + \sqrt{\frac{h}{m}}\right)\\ &\leq \hat{L}_\gamma(f) + \tilde{O}\left(\frac{\sqrt{c}\|\mathbf{V}\|_F\left(\left\|\mathbf{U}-\mathbf{U}^0\right\|_F + \left\|\mathbf{U}^0\right\|_2\right)\sqrt{\frac{1}{m}\sum_{i=1}^m\|\mathbf{x}_i\|_2^2}}{\gamma\sqrt{m}} + \sqrt{\frac{h}{m}}\right)\end{aligned}$$

Therefore, the term $\|\mathbf{V}\|_F\left(\left\|\mathbf{U}-\mathbf{U}^0\right\|_F + \left\|\mathbf{U}^0\right\|_2\right) + \sqrt{h}$ can be used as a heuristic for complexity.

23 Validation Paradigm

The work of [22] looks into how the training-validation paradigm leads to deep neural networks that generalize well. In this paradigm, a validation set of data is held out to optimize the model architecture and hyper-parameters. Giving rise to the hypothesis that *deep neural networks can obtain good generalization error by performing a model search on the validation set.*
Consider an input $x \in \mathcal{X}$ and a label $y \in \mathcal{Y}$. The loss function is denoted $\mathcal{L}$ and $\mathcal{R}[f] = \mathbb{E}_{x,y\sim\mathbb{P}_{(\mathcal{X},\mathcal{Y})}}(\mathcal{L}(f(x), y))$ is the expected risk of a function f for $\mathbb{P}_{(\mathcal{X},\mathcal{Y})}$ being the true distribution. Let $f_{\mathcal{A}(S)} : \mathcal{X} \to \mathcal{Y}$ denote the function learnt by a learning algorithm $\mathcal{A}$ on training set $S := \{(x_1, y_1), \ldots, (x_m, y_m)\}$. The set of possible learned functions is characterized by the hypothesis space $\mathcal{F}$. Associated with this space we have the family of loss functions $\mathcal{L}_{\mathcal{F}} := \{g : g \in \mathcal{F}, g(x, y) = \mathcal{L}(f(x), y)\}$.

Machine learning aims to minimize $\mathcal{R}\left(f_{\mathcal{A}(S)}\right)$. However, this is non-computable as $\mathbb{P}_{(\mathcal{X},\mathcal{Y})}$ is unknown. Therefore, one minimizes the empirical risk

$$\mathcal{R}_S\left(f_{\mathcal{A}(S)}\right) = \frac{1}{|S|}\sum_{(x,y)\in S}\mathcal{L}(f_{\mathcal{A}(S)}(x), y),$$

where the generalization gap is defined to be $\mathcal{R}\left(f_{\mathcal{A}(S)}\right) - \mathcal{R}_S\left(f_{\mathcal{A}(S)}\right)$.

Proposition 23.1 *Let $S^{(\mathrm{val})}_{m_{\mathrm{val}}}$ be a held-out validation set, where $\left|S^{(\mathrm{val})}_{m_{\mathrm{val}}}\right| = m_{\mathrm{val}}$. Assume that m_{val} is an* i.i.d *sample from $\mathbb{P}_{(X,Y)}$. Let $\kappa_{f,i} = \mathcal{R}(f) - \mathcal{L}(f(x_i), y_i)$ for $(x_i, y_i) \in S^{(\mathrm{val})}_{m_{\mathrm{val}}}$. Suppose that $\mathbb{E}\left(\kappa^2_{f,i}\right) \leq \gamma^2$ and $|\kappa_{f,i}| \leq C$ almost surely for all $(f, i) \in \mathcal{F}_{\mathrm{val}} \times \{1, \ldots, m_{\mathrm{val}}\}$. Then, for $\delta \in (0, 1]$, with probability $1 - \delta$*

$$\mathcal{R}(f) \leq \mathcal{R}_{S^{(\mathrm{val})}_{m_{\mathrm{val}}}}(f) + \frac{2C\log\left(\frac{|\mathcal{F}_{\mathrm{val}}|}{\delta}\right)}{3m_{\mathrm{val}}} + \sqrt{\frac{2\gamma^2\log\left(\frac{|\mathcal{F}_{\mathrm{val}}|}{\delta}\right)}{m_{\mathrm{val}}}}$$

holds for all $f \in \mathcal{F}_{\mathrm{val}}$.

Remark 23.2

- *$\mathcal{F}_{\mathrm{val}}$ is independent of $S^{(\mathrm{val})}_{m_{\mathrm{val}}}$.*
- *The bound is only dependent on the validation error on $S^{(\mathrm{val})}_{m_{\mathrm{val}}}$.*

The dependence on $|\mathcal{F}_{\mathrm{val}}|$ can be alleviated in the following corollary of Theorem 4.3.

Corollary 23.3 *Assume $S^{(\mathrm{val})}_{m_{\mathrm{val}}}$ is an* i.i.d *sample from $\mathbb{P}_{(\mathcal{X},\mathcal{Y})}$. Let $\mathcal{L}_{\mathcal{F}_{\mathrm{val}}} = \{g : f \in \mathcal{F}_{\mathrm{val}}, g(x, y) := \mathcal{L}(f(x), y)\}$. Then when $\mathcal{L}$ has co-domain $[0, 1]$ it follows that,*

$$\mathcal{R}(f) \leq \mathcal{R}_{S^{(\mathrm{val})}_{m_{\mathrm{val}}}}(f) + 2\mathfrak{R}_m\left(\mathcal{L}_{\mathcal{F}_{\mathrm{val}}}\right) + \sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{m_{\mathrm{val}}}}.$$

Some code for implementing this paradigm can be found here.

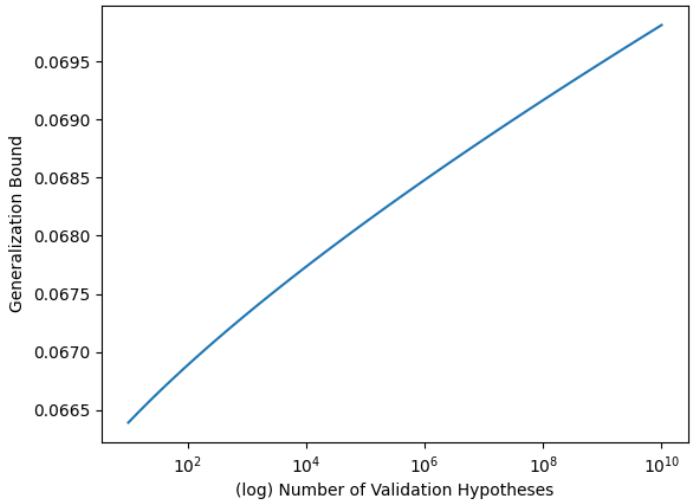

Figure 19: Scaling of generalization error bound with the number of models used for validation. Models used for validation are those that achieve low training loss.

Part VI
A Topological Perspective of Neural Networks

24 Topological Evolution of the Training Data

In [18] deep neural networks are investigated by understanding how the layers manipulate the input's topology. Persistent homology is used to understand how topological features of the input data evolve through the layers of the network, and how architectural properties of the network contribute to this evolution. In [18] a framework for this investigation is developed and some preliminary conclusions are given. It does not attempt to explain network generalization from the topological lens.

24.1 Persistent Homology

Definition 24.1 *A k-dimensional simplex σ in $\mathbb{R}^d$ is the convex hull of $k+1$ affinely independent points $v_0, \ldots, v_k \in \mathbb{R}^d$, denoted $\sigma = [v_0, \ldots, v_k]$.*

The faces of a k-simplex are simplicies of dimension 0 to $k-1$ formed by convex hulls of proper subsets of the vertex set $\{v_0, \ldots, v_k\}$.

Definition 24.2 *An m-dimensional geometrical simplicial complex K in $\mathbb{R}^d$ is a finite collection of simplices in $\mathbb{R}^d$ of dimension at most m that are*

1. *Any intersection between two simplices is K is necessarily a face of both of them, and*
2. *includes all faces of all its simplices.*

Definition 24.3 *An abstract simplicial complex is a list of simplicies $K = \{\sigma_1, \ldots, \sigma_n\}$ such that if $\tau \subseteq \sigma \in K$ then $\tau \in K$.*

Definition 24.4 *Let $C_0, \ldots, C_d$ be vector spaces over $\mathbb{F}_2$. Let boundary operators $\delta_k : C_k \to C_{k-1}$ be linear maps satisfying*

$$\delta_k \circ \delta_{k-1} = 0$$

for all $k = 1, \ldots, d$.

Definition 24.5 *A chain complex is a sequence*

$$0 \overset{\delta_{d+1}}{\to} C_d \ldots C_1 \overset{\delta_0}{\to} 0,$$

where $C_{d+1} = C_{-1} = 0$. The elements in the image of δ_k are called boundaries and elements of the kernel of δ_{k-1} are called cycles,

$$B_k := \operatorname{im}(\delta_{k+1}) \subseteq \ker(\delta_k) =: Z_k.$$

Definition 24.6 *The k^{th} homology group is the quotient vector space*

$$H_k := Z_k / B_k, \quad \text{for } k = 0, \ldots, d.$$

The homology classes are the equivalence classes

$$[z] = z + B_k = \{z + b \in Z_k : b \in B_k\}.$$

Definition 24.7 *The k^{th} Betti number of K is $\beta_k := \dim(H_k)$.*

Remark 24.8 *The k^{th} Betti number of K counts the number of k-dimensional holes in K.*

Let $K^{(k)} = \{\sigma_1, \ldots, \sigma_m\}$ be the set of all k-dimensional simplicies in K. When working over $\mathbb{F}_2$, let $C_k(K)$ be the vector space with basis $K^{(k)}$. Define the boundary operator $\delta_k : C_k(K) \to C_{k-1}(K)$ for a k-simplex $\sigma = [v_0, \ldots, v_k]$ as

$$\delta_k(\sigma) = \delta_k\left(\sum_{j=1}^{m} n_j \sigma_j\right) = \sum_{j=1}^{m} n_j \delta_k(\sigma_j).$$

Working over $\mathbb{F}_2$ ensures that $H_k(K) \cong \mathbb{F}_2^{\beta_k}$ and

$$\beta_k(K) = \dim(H_k(K)) = \text{null}(\delta_k) - \text{rank}(\delta_{k+1}) \quad k = 0, \dots, d.$$

Let $m_k = \left|K^{(k)}\right|$, then the number of simplicies in a d-dimensional simplicial complex is bounded by

$$|K| \leq \sum_{i=1}^{d} \binom{m_0}{i+1}.$$

Definition 24.9 *For K_1 and K_2 two abstract simplicial complexes a simplicial map $f : K_1^{(0)} \to K_2^{(0)}$ is such that for $\sigma = [v_0, \dots, v_k] \in K_1$ it follows that $[f(v_0), \dots, f(v_k)] \in K_2$. This induces a map between complexes*

$$f : C_k(K_1) \to C_k(K_2), \quad \sum_{j=1}^{m} n_j \sigma_j \mapsto \sum_{j=1}^{m} n_j f(\sigma_j),$$

which induces a map between homologies

$$H_k(f) : H_k(K_1) \to H_k(K_2), \quad \left[\sum_{j=1}^{m} n_j \sigma_j\right] \mapsto \left[n_j f(\sigma_j)\right]$$

for all $k = 0, \dots, d+1$.

Definition 24.10 *Let δ be a metric on $\mathbb{R}^d$. The Vietoris-Rips complex at scale $\epsilon \geq 0$ on $X \subseteq \mathbb{R}^d$ is the abstract simplicial complex*

$$\text{VR}_\epsilon(X) := \left\{[x_0, \dots, x_k\} : \delta\left(x_i, x_j\right) \leq 2\epsilon, x_0, \dots, x_k \in X, k = 0, \dots, n\right\}.$$

If X is a dense enough sample from a manifold $M \subseteq \mathbb{R}^d$, then $\text{VR}_\epsilon(X)$ can recover in some sense the true topology of M.

- At scale $\epsilon = 0$, then $\text{VR}_0(X) = \{[x] : x \in X\}$, that is VR_0 overfits the data X.
- As $\epsilon \to \infty$ all the points of X become vertices of a single $|X|$-dimensional simplex, giving a contractible topological space.

A persistence barcode is an interval $[\epsilon, \epsilon')$ where at the left end-point a new feature appears (is born) and at the right end-point the feature disappears (or dies). The length of the interval, $\epsilon' - \epsilon$, is the persistence of that feature. Computing persistence barcodes for homology groups is known as persistent homology. In the following consider the finite set of scales $\epsilon_0 < \cdots < \epsilon_m$ and the simplicial complexes $K_j = \text{VR}\left(X, \epsilon_j\right)$.

Definition 24.11 *The filtration of simplicial complexes is the chain of nested simplicial complexes*

$$K_0 \subseteq \cdots \subseteq K_m,$$

with the inclusion maps $f_j : K_j \hookrightarrow K_{j+1}$ *for* $j = 0, \ldots, m-1$.

Remark 24.12 *The index* j *is referred to as time. If a homology class is in* $H_k(K_{j+1})$ *is not in the image* $H_k(K_j)$*, then the class is born at time* $j+1$*. If simplicies of different homology classes in* $H_k(K_i)$ *are mapped to the same homology class of* $H_k(K_j)$ *for* $i < j$ *then one class is said to have died, and the other is said to have persisted from time* i *to* j*.*

To identify classes that persist from time j to $j+p$ it is sufficient to consider the p-persistent k^{th} homology group

$$H_k^{j,p} = Z_k^j \Big/ \left(B_k^{j+p} \cap Z_k^j\right).$$

24.2 Binary Classification

For binary classification, the aim is to classify two different probability distributions supported on disjoint manifolds $M_a, M_b \subseteq \mathbb{R}^d$. Suppose there exists a classifier with zero prediction error. Sample a large finite set of points $T \subseteq M_a \cup M_b$ uniformly and densely and deduce the Betti numbers of M_a and M_b. Consider the feed-forward neural network $\nu : \mathbb{R}^d \to [0,1]$ given by

$$\nu = s \circ f_l \circ \cdots \circ f_1,$$

where $f_j : \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}$ is a network layer involving an affine map $\rho_j : \mathbb{R}^{n_j} \to \mathbb{R}^{n_{j+1}}, x \mapsto A_j x + b_j$ composed with an activation function $\sigma : \mathbb{R}^{n_{j+1}} \to \mathbb{R}^{n_{j+1}}$. The function $s : \mathbb{R}^{n_{l+1}} \to [0,1]$ is the score function . Let

$$\nu_j = f_j \circ \ldots f_1 \text{ and } \nu = s \circ \nu_l.$$

The output of the network is interpreted to be the probability that the input is on M_a. A well-trained network correctly classifies all $x \in T$. In reality, the topologies M_a and M_b may be intertwined in a complicated manner. Using the theory built up in the previous section it can be shown experimentally that the neural network unravels the decision boundary to opposite ends of $[0,1]$ for effective classification.

24.3 Architectural Implications on Data Topology

Every network trained with differing activation functions shows a decrease in β_0 across the layers of the network. With tanh activation the reduction is less effective, whereas ReLU demonstrated the largest decay due to the fact that ReLU is a non-homeomorphic activation function.
A Bottleneck layer forces large topological changes, and a narrow network changes topology faster than a wider one. Reducing the depth of a constant-width network makes it increasingly difficult to

train the network to high accuracy. As the burden of changing topology does not spread evenly across the layers, but instead is concentrated at the final layers. Which reaches the saturation limits of the layers.

Part VII
Conclusion

25 Conclusion

Various methodologies for understanding deep neural networks have been put forth in this survey. A culmination of theoretical and empirical techniques have been employed to try and develop explanations for deep neural networks' impressive capacity to generalize. The Bayesian machine learning framework is useful for formalizing learning algorithms in a probabilistic setting. It enables the construction of probabilistic bounds that incorporate the inherent randomness of the training data. Despite giving a statistical perspective on deep neural networks in practice the results are vacuous and require considerable fine-tuning to be useful. It falls short of explaining the phenomena seen in practice. It is a technique for investigation more traditional learning algorithms that were developed before the advent of deep neural networks and has struggled to adapt to the current state of the field. The information-theoretic approach is motivated by the transfer of information from the dataset to the network and has yielded remarkable insights into the stages of the training process. It has provided a theoretical setting to investigate the training process of deep neural networks. It focuses on information contained in the data and does not take into account the architecture or the learning algorithm. In large part, the success of deep neural networks is attributed to the architecture and the learning algorithm, and hence the information-theoretic framework perhaps is insufficient to fully understand deep neural networks.

The most promising and practical approach involves the investigation of the gradients involved in the networks. Computing the gradient of the network with respect to the parameters at an output, or calculating the gradient of the loss with respect to the parameters at training examples encapsulates information about the learning algorithm and the training data. In the case of the neural tangent kernel, the gradient is with respect to the weights and the output of the network and does not involve the loss function. Therefore, it reveals what the network has learned but lacks detail on generalization. On the other hand, coherence and stiffness involve gradients of the loss function and hence can be directly linked to generalization. Using gradients appeals to the central component of deep neural networks, stochastic gradient descent. This is how the network learns and the learned structures are what give rise to the network's generalizing abilities. The techniques involving gradients discussed in this survey require are agnostic to any properties of the network and so receive few issues when employed practically.

The topological perspective is a sophisticated approach to understanding the evolution of the geometrical properties of the training data through the layers of the neural network. The approach reveals why certain architectural properties of the network are effective by understanding how they are able to manipulate the structure of the data. A lot of insight has been gained from trying to understand

the various aspects of a deep neural network through a geometrical perspective and the topological arguments explored in this survey provide a compelling framework to do this. A limitation however is that implementing the framework in practice is computationally expensive and it neglects the properties of the learning algorithm.

26 Future Work

It has been noted that gradients of the loss function at different parameter values can be effectively correlated to generalization, as it is a rich source of information on the training process of deep neural networks. However, the current form that they are explored in this survey, lacks a strong geometrical analysis and do not explicitly involve any information on the network's architecture. Future work in the field could look into investigating stiffness and coherence in a topological framework to develop notions that involve the data, the learning algorithm and the architecture.

References

[1] David A. McAllester. "Some PAC-Bayesian Theorems". In: *Machine Learning* 37 (1998), pp. 355–363.

[2] Jiantao Jiao, Yanjun Han, and Tsachy Weissman. "Dependence Measures Bounding the Exploration Bias for General Measurements". In: *CoRR* (2016).

[3] Daniel Russo and James Zou. "Controlling Bias in Adaptive Data Analysis Using Information Theory". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Ed. by Arthur Gretton and Christian C. Robert. Vol. 51. Proceedings of Machine Learning Research. PMLR, 2016, pp. 1232–1240.

[4] Gintare Karolina Dziugaite and Daniel M. Roy. "Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data". In: *CoRR* (2017).

[5] Ravid Shwartz-Ziv and Naftali Tishby. "Opening the Black Box of Deep Neural Networks via Information". In: *CoRR* (2017).

[6] Aolin Xu and Maxim Raginsky. "Information-theoretic analysis of generalization capability of learning algorithms". In: *CoRR* (2017).

[7] S. Arora, R. Ge, B. Neyshabur, and Y. Zhang. "Stronger generalization bounds for deep nets via a compression approach". In: *CoRR* (2018).

[8] Amir R. Asadi, Emmanuel Abbe, and Sergio Verdú. "Chaining Mutual Information and Tightening Generalization Bounds". In: *CoRR* (2018).

[9] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2018.

[10] Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. "Towards Understanding the Role of Over-Parametrization in Generalization of Neural Networks". In: *CoRR* (2018).

[11] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. "On the Information Bottleneck Theory of Deep Learning". In: *International Conference on Learning Representations*. 2018.

[12] Lei Wu, Chao Ma, and Weinan E. "How SGD Selects the Global Minima in Over-parameterized Learning: A Dynamical Stability Perspective". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018.

[13] Stanislav Fort, Pawel Krzysztof Nowak, and Srini Narayanan. "Stiffness: A New Perspective on Generalization in Neural Networks". In: *CoRR* (2019).

[14] Benjamin Guedj. *A Primer on PAC-Bayesian Learning*. 2019.

[15] Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, and Peter Orbanz. *Non-Vacuous Generalization Bounds at the ImageNet Scale: A PAC-Bayesian Compression Approach*. 2019.

[16] Aristide Baratin, Thomas George, César Laurent, R. Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. "Implicit Regularization via Neural Feature Alignment". In: *CoRR* (2020).

[17] Gintare Karolina Dziugaite, Kyle Hsu, Waseem Gharbieh, and Daniel M. Roy. "On the role of data in PAC-Bayes bounds". In: *CoRR* (2020).

[18] Gregory Naitzat, Andrey Zhitnikov, and Lek-Heng Lim. "Topology of deep neural networks". In: *CoRR* (2020).

[19] Thomas Steinke and Lydia Zakynthinou. "Reasoning About Generalization via Conditional Mutual Information". In: *CoRR* (2020).

[20] Satrajit Chatterjee and Piotr Zielinski. *On the Generalization Mystery in Deep Learning*. 2022.

[21] Benoit Dherin, Michael Munn, Mihaela Rosca, and David G. T. Barrett. *Why neural networks find simple solutions: the many regularizers of geometric complexity*. 2022.

[22] K. Kawaguchi, Y. Bengio, and L. Kaelbling. "Generalization in Deep Learning". In: *Mathematical Aspects of Deep Learning*. Cambridge University Press, 2022, pp. 112–148.

[23] Patrick Rebeschini. *Algorithmic Foundations of Learning*. Nov. 2022.