Understanding Generalization

by

Ming Yang Ong

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Masters of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2017

Author . . . .
Department of Electrical Engineering and Computer Science
May 26, 2017

Certified by . . . .
Pablo A. Parrilo
Professor, Department of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by . . . .
Christopher Terman
Chairman, Masters of Engineering Thesis Committee

Understanding Generalization

by

Ming Yang Ong

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 2017, in partial fulfillment of the
requirements for the degree of
Masters of Engineering in Electrical Engineering and Computer Science

Abstract

An important goal in machine learning is to understand how to design models that can generalize. This thesis follows a venerable line of research aimed at understanding generalization through the lens of stability- the study of how variations on the inputs of a system can cause its outputs to change. We explore stability and generalization in two different directions. In the first direction we look at proving stability using a proof technique provided by Hardt et al [HRS16]. We apply this technique to stochastic gradient descent with momentum and investigate the resulting stability bounds under some assumptions. In the second direction, we explore the effectiveness of stability in obtaining generalization bounds under the violation of some model assumptions. In particular, we show that stability is insufficient for generalization under domain adaptation. We introduce a sufficient condition and show that some properties can imply this condition.

Thesis Supervisor: Pablo A. Parrilo
Title: Professor, Department of Electrical Engineering and Computer Science

Acknowledgments

Can I begin by confessing that before I started my M'Eng, I'd barely read a single academic paper? So the learning curve was definitely steep[1]! Nevertheless I feel I've grown in these two semesters of the M'Eng program figuring out how to navigate the not always friendly waters of academic research, and also learning about the interesting and exciting things out there to be learned in the vast beyond. For this I would like to thank my advisor Pablo who not only offered invaluable comments and advice regarding my thesis but also taught me the ways of the grad student, encouraging me to read papers and attend talks. I would also like to thank all my friends who took the time off their busy schedules to share with me stories of research, math, and grad student life.

[1]As we are all machine learning people, let us not get into this debate about what a steep learning curve actually means

Contents

List of Figures

Chapter 1

Introduction

The decision making systems of today are as data driven as ever, and are deployed in many wide ranging technologies from medical diagnoses to self driving cars. As we begin to integrate these technologies into our daily lives, however, the failure of such systems to make reasonable decisions could result in devastating consequences such as the loss of human life. When we rely on these systems to help us perform increasingly ambitious tasks, a central question is how to design systems that can respond accurately to new situations, given that we can only train them on a finite number of examples. How do we guarantee the performance of our system on unseen circumstances? How do we know if our systems can generalize?

Over the years, understanding how to control generalization has become a basic question in statistical learning theory. To this end, many approaches have been proposed to achieve this control. One prominent approach is to restrict the capacity of the hypothesis space, which provides a guarantee on the uniform convergence of empirical quantities to their mean [Vap98]. In this thesis, we will be concerned with another method, that is, by controlling the stability of a learning algorithm. Roughly speaking, stability is a measure of input-output sensitivity of our learning algorithm—by how much does our model change when we slightly perturb our training set? Our starting point is a definition of uniform stability introduced by Bousquet and Elisseeff in 2002 [BE02]. This particular definition has appealed to many researchers because of its convenient characterization and also data and distribution independence. Within this framework, two natural questions can be asked:

1. How do we prove a learning algorithm satisfies uniform stability?

2. How useful is a uniformly stable algorithm if strict model assumptions are violated in a minor way?

We investigate these two questions separately, dividing our findings into chapters. First of all, we begin this thesis in Chapter 2 with a gentle introduction to the problem of machine learning and empirical risk minimization. We assume the reader has no prior machine learning knowledge, and so we will formulate our problems of interest from first principles. In particular, we show how risk, generalization, and consistency naturally arise in the general setting of learning, how they relate to each other, and how they can be controlled by the property of stability.

In Chapter 3, we discuss a technique proposed by Hardt et al [HRS16] that can be used to analyze stability bounds of models that are trained by a stochastic method. Even though this technique relies on only elementary convex analysis, it can be applied to a large range of learning algorithms. In particular, we will look at a specific algorithm: stochastic gradient descent with momentum. We present some results, but with the downside that the bounds attained from this analysis are meaningful only under very strict conditions. We show that the stability can potentially be upper bounded by the joint spectral radius of a class of matrices.

In Chapter 4, we explore some of the limitations of stability as a standard for achieving our original goal- to guarantee performance under unseen circumstances. We consider the setting of domain adaptation [BDBCP07], and show that stability is insufficient for generalization when our training data is drawn from a distribution that differs slightly from the test distribution. We show that the generalization error can be decomposed into two terms, the first is the original generalization error, and the second is some measure of robustness, which can be upper bounded by properties such as the algorithmic robustness [XM10] or the Wasserstein distance between the distributions.

Chapter 2

The General Setting of Learning

What is machine learning? In some sense, machine learning is the study of systems that learn how to make decisions from data. When a system makes decisions, it incurs some kind of risk in relation to its true environment. Within this framework, thinking about machine learning is equivalent to thinking about the problem of risk minimization:

2.1 Machine Learning as Risk Minimization

Suppose we want to understand some relationship between observations x and y. Imagine there being some unknown functional dependency $x \to y$ that represents the true relationship between these two observations. Then the goal is to construct a learning machine that attempts to imitate the true relationship by enacting $x \to y'$. To help the learning machine, we give it access to previous realizations of x and y, which are referred to as *training data.* We can formalize this as a model of searching for functional dependency, introduced by Vapnik [Vap98], comprising three components:

1. A generator component which produces examples x
2. A supervisor that enacts the true relationship $x \to y$
3. A learning machine whose goal is to imitate the supervisor given training data $\{(x_i, y_i)\}$.

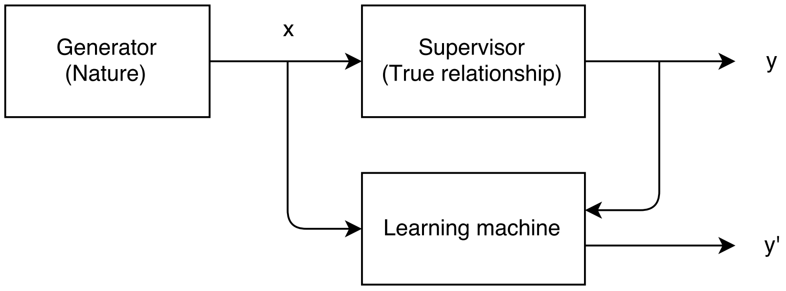

Figure 2-1: A model of supervised learning

Here, we assume that our training data $z_i = (x_i, y_i) \in Z$ are generated i.i.d. according to some unknown but fixed probability distribution, and that our learning machine is given access to a finite training set $S = \{z_1, \ldots, z_n\}$ of cardinality n. To imitate the supervisor, the learning machine is tasked to pick the best function $x \to y'$ out of a set of possible functions parameterized by model parameters $w \in W$.

What does it mean to pick the "best" function? We assume a given loss function $f : W \times Z \to \mathbb{R}$ that provides a measure of how inaccurate our predictions are on a new example $z \in Z$. Define the risk:

$$R(w) = \mathbb{E}_z \ f(w, z)$$

Then our goal is to find the model parameters $w \in W$ that minimizes the risk:

$$w^* = \arg\min_{w \in W} R(w) \qquad R^* = R(w^*)$$

To solve this minimization problem, we apply a *learning algorithm.* Formally, a learning algorithm is a function $A : Z^n \to W$ that receives a training data set $S \in Z^n$ and returns model parameters w. We sometimes say that A learns w from S.

Because our distribution is unknown, the risk often cannot be explicitly calculated, so we have to estimate it from other quantities. Given training data S, a simple estimator is the average of sample losses, known as the empirical risk:

$$R_S(w) = \frac{1}{n} \sum_{z_i \in S} f(w, z_i)$$

Since the empirical distribution is an approximation of the true distribution, certainly a method that attempts to minimize the empirical risk is a good place to start. The problem of finding w that minimizes R_S is known as the problem of empirical risk minimization (ERM) [Vap98]. We can similarly denote:

$$w_S^* = \arg\min_{w \in W} R_S(w) \qquad R_S^* = R_S(w^*)$$

ERM is sufficiently general that it is the underlying principle of many popular machine learning methods such as:

Example 2.1.1 (Regression). *In the model of linear regression we can consider prediction functions of the form $h_w : x \to \langle w, x \rangle$. The sum of squares residue can then be expressed as the empirical risk associated with loss function $f : w \times z \to (h_w(x) - y)^2$*

Example 2.1.2 (Support Vector Machines). *An SVM is a type of binary classification model. Prediction functions take the form $h_w : x \to \mathbb{1}_{\langle w,x \rangle > 0}$, and a possible loss function is $f(w, z) = g(y \cdot \langle w, x \rangle)$ where the hinge function g is*

$$g(v) = \begin{cases} 0 & v > 1 \\ 1 - v & v \leq 1 \end{cases}$$

And the problem is to minimize $\frac{1}{n} \sum_{i=1}^{n} f(w, z_i) + ||w||^2$, where the first term is the empirical risk and the last term is a regularization parameter.

In fact, ERM is general enough that it even encompasses some unsupervised learning problems (even though we described ERM in terms of supervised learning earlier in Figure 2-1):

Example 2.1.3 (k-means clustering). *Let W be the set of all subsets of $\mathbb{R}^n$ of size k. Then the k-means clustering procedure is to find a hypothesis $w \in W$ (corresponding to k centroids) that minimizes the risk associated with loss function $f(w, z) = \inf_{c \in w} ||c - z||^2$.*

Now we have rather surreptitiously claimed that solving ERM is a good way to approach the learning problem. This claim needs to be supported, so we ask: suppose our learning algorithm returns the solution w, how close are we to our original goal of minimizing $R(w)$? This is essentially a question of consistency and generalization which we will discuss in Section 2.2 below.

2.2 Consistency and Generalization

Recall from Section 2.1 that our original goal is to find the minimizer of the risk. Suppose our learning algorithm gives us a model w. One way to evaluate the performance of our model is to consider $R(w) - R^*$. This motivates a notion of consistency:

Definition 2.2.1 (Consistency [SSSSS10])**.** *We say that a learning algorithm $\mathcal{A}$ that learns w from S is consistent with rate $\epsilon(n)$ with respect to a distribution $\mathcal{D}$ if,*

$$\mathbb{E}_{S\sim\mathcal{D}^n}\left[R(w) - R^*\right] = \epsilon(n)$$

If $\lim_{n\to\infty}\epsilon(n) = 0$, we say that $\mathcal{A}$ is consistent. If the convergence holds for all distributions $\mathcal{D}$, then we say that $\mathcal{A}$ is universally consistent. We also say that a learning problem is learnable if there exists a learning algorithm that is universally consistent for the problem.

We note that some authors define consistency in terms of a probability bound [MNPR06]. This doesn't really matter as we can always translate an expectation bound $E[|X|] \leq \epsilon$ to a probability guarantee e.g. with Markov's inequality we have $\Pr[X \geq \epsilon/\delta] \leq \delta$[1].

There are some authors who refer to consistency as generalization [BCN16]. In some sense, a high probability guarantee that the predictions of our model do not differ much from the best possible hypothesis is essentially a guarantee on the performance of our model in the real world.

But a more common way people think about generalization is how different our model performs on the true distribution compared to the empirical distribution. Indeed, the difference $R(w) - R_S(w)$ provides us with a measure of cross validation

[1]A discussion on this topic can be found in Section 7 of [SSSSS10]

performance of our model (in practice, this is often referred to as the "test error minus training error"). We can think of this generalization as a guarantee of our model's performance in the real world given its performance in a training environment.

Both of these characterizations can contribute to our understanding of generalization, which is the informal idea that we want our systems to make reasonable decisions under some uncertainty. Unfortunately this ambiguity results in some confusion as to what is formally defined as generalization in existing literature, and the term has ended up referring to several different quantities. Nevertheless, they can fortunately all be related by one equation. Suppose our algorithm learns w from S, then we can write the following tautology:

$$R(w) - R^* = [R(w) - R_S(w)] + [R_S(w) - R_S(\tilde{w})] + [R_S(\tilde{w}) - R^*] \qquad (2.2.0.1)$$

This decomposition equation is not new [LCR16, BB07]. Regardless this helps us understand the relationship between the following terms that we will define below:

Term	What others have called this	What we will call this
$R(w)$	Generalization error [BE02, XM10]	Risk
$R(w) - R^*$	Generalization error [BCN16]	Excess risk
$R(w) - R_S(w)$	Generalization error [HRS16, LCR16]	Generalization error

Figure 2-2: The ambiguous use of the term "generalization" in existing literature

The remaining terms in Equation 2.2.0.1 can be analyzed depending on our choice of $\tilde{w}$. For example, if we choose $\tilde{w}$ to be the ERM solution, then the second term $R_S(w) - R_S(\tilde{w})$ is the optimization error, and the last term $R_S(\tilde{w}) - R^*$ can be seen as the approximation error between ERM and risk minimization. In this framework, balancing the terms in Equation 2.2.0.1 result in a few tradeoffs. For example, it is possible to construct a naïve algorithm that has vanishing expected generalization error, but with no guarantees on the optimization error:

Example 2.2.1. *Consider the algorithm that returns w independently of S. Then the expected generalization error is always zero.*

Proof. Since A is independent of S, so

$$\begin{aligned}
\mathbb{E}_S[R(w) - R_S(w)] &= \mathbb{E}_z f(w,z) - \mathbb{E}_S[\frac{1}{n}\sum_{i=1}^{n} f(w,z_i)] \\
&= \mathbb{E}_z f(w,z) - \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{z_i}[f(w,z_i)] \\
&= 0
\end{aligned}$$

□

So we would like to investigate an approach that can explore this tradeoff. One prominent research direction is the study of notions of capacity of the hypothesis class, e.g. Rademachar complexity (see references within [ZBH+17]) or VC dimension [VC71]. When a learning problem has finite VC dimension, then there exists an algorithm that can provide consistency and generalization guarantees. However, this condition is pretty strict as it requires assumptions on the entire hypothesis class. It turns out that this is not necessary, and that a notion of stability on our learning algorithm is sufficient to provide the same guarantees. Using stability without recourse to notions of capacity like VC dimension is nice as 1) since stability is a property of a learning algorithm, so proving stability explicitly provides a consistent algorithm, as opposed to just a certificate of existence, and 2) a learning problem with a stable algorithm generalizes and can be consistent under some assumptions even if its hypothesis class has infinite capacity.

We will explore the relationship between stability and generalization in the next section.

2.3 Stability

What is stability? Roughly speaking, an algorithm is stable if its outputs vary only slightly if we change one element in its training set. We can think of stability as a property of a learning algorithm that characterizes its sensitivity to its training data. In many papers that concern stability [BE02, MNPR06, SSSSS10], stability is often described as the ability of an algorithm to be robust against a few data points which might deviate grossly from the structure of the bulk of the dataset.

A number of results have shown that stability is sufficient for generalization [BE02] and also consistency [MNPR06]. We will cover a few of these results in this section. But first, we need a few definitions. Let us construct a modified training set obtained from S:

Definition 2.3.1 (Adjacent datasets)**.** *Denote*

$$S^{(i)} = \{z_1, \dots, z_{i-1}, z_i', z_{i+1}, \dots z_n\}$$

to be the training set that is identical to S except for the i^{th} element z_i replaced with z_i' which is drawn i.i.d. from D. Then a training set S' is said to be adjacent to S if there exists an i and $S^{(i)}$ such that $S' = S^{(i)}$.

Definition 2.3.2 (Uniform stability)**.** *A (possibly random) learning algorithm A is ϵ-uniformly stable with respect to a loss function f if for all adjacent datasets S and S',*

$$\sup_z \mathbb{E}_A[f(A(S), z) - f(A(S'), z)] \leq \epsilon$$

Where the expectation is taken over the internal randomness of A. When $\epsilon \leq O(\frac{1}{n})$, we sometimes simply say that A is uniformly stable.

This definition allows us to think of stability as some notion of continuity of f with respect to S, induced by A.

An interesting result is that if a model is obtained from an ϵ-uniformly stable algorithm, the empirical risk and the risk are within ϵ error in expectation. In other words, stability implies generalization. This can be shown with the following theorem:

Theorem 2.3.1 (Generalization in Expectation[2]). *Let A be ϵ-uniformly stable. Then,*

$$|\mathbb{E}_{S,A}[R(A(S)) - R_S(A(S))]| \leq \epsilon$$

Proof. Let $S = (z_1 \dots z_n)$ and $S^{(i)}$ be defined as in Definition 2.3.1. We can compute the expectation of $R_S(A(S))$:

$$\begin{aligned}
\mathbb{E}_{S,A}[R_S(A(S))] &= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{S,A}[f(A(S), z_i)] \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{S,A,z_i'}[f(A(S^{(i)}), z_i')] \\
&= \mathbb{E}_{S,A,z_i'}[f(A(S^{(i)}), z_i')]
\end{aligned}$$

with the second equality arising from mathematical substitution— the two quantities are indistinguishable! Putting these together we have:

$$\begin{aligned}
|\mathbb{E}_{S,A}[R(A(S)) - R_S(A(S))]| &= |\mathbb{E}_{S,A,z_i'}[f(A(S), z_i') - f(A(S^{(i)}), z_i')]| \\
&\leq |\sup_z \mathbb{E}_{S,A,z_i'}[f(A(S), z) - f(A(S^{(i)}), z)]| \\
&\leq \sup_{S,z_i',z} \mathbb{E}_A[f(A(S), z) - f(A(S^{(i)}), z)] \\
&\leq \epsilon
\end{aligned}$$

□

Besides generalizing in expectation, several other high probability bounds also exist [BE02]. Besides generalization, a more surprising result is that when our algorithm is ERM, uniform stability also implies consistency:

Theorem 2.3.2 (Stability of ERM implies consistency[3]). *Uniform stability is sufficient for universal consistency of ERM.*

[2]Proof adapted from [HRS16]

[3]Adapted from [MNPR06]

Proof. If our learning algorithm is ERM, this means that our algorithm learns w_S^* from S. From Theorem 2.3.1, we know that uniform stability implies that

$$\lim_{n\to\infty} \mathbb{E}[R(w_S^*)] = \lim_{n\to\infty} \mathbb{E}[R_S(w_S^*)]$$

Then $R(w^*) \leq R(w_S^*)$ and $R_S(w_S^*) \leq R_S(w^*)$. Therefore,

$$R(w^*) \leq \lim_{n\to\infty} \mathbb{E}[R(w_S^*)] = \lim_{n\to\infty} \mathbb{E}[R_S(w_S^*)] \leq \lim_{n\to\infty} \mathbb{E}[R_S(w^*)] = R(w^*)$$

with the last equality arising from the law of large numbers. This results in

$$\lim_{n\to\infty} \mathbb{E}[R(w_S^*)] = R^*$$

$\square$

Theorem 2.3.2 requires our algorithm to be explicitly an ERM, but not all learning algorithms are of this form (the converse isn't true either; not all ERM algorithms are uniformly stable [SSSSS10]). Especially with numerical optimizers and stochastic methods, an exact minimizer is rarely found. To allow for a more general discussion, we can also consider the case when w is not an exact minimizer of ERM, but within some ϵ region of the exact minimizer.

Corollary 2.3.1 (Stability of almost ERM implies almost consistency[4])**.** *Suppose an ϵ_{stab}-uniformly stable learning algorithm learns w from S such that $R_S(w)$ is within an ϵ_{opt} neighborhood of the optimum R_S^*. Then the algorithm is universally consistent with rate $\epsilon_{opt} + \epsilon_{stab}$.*

Proof. This is simply a consequence of Equation 2.2.0.1, Theorem 2.3.1 and Theorem 2.3.2. We observe that $\mathbb{E}[R(w)] = \mathbb{E}[R_S(w_S^*)] + \epsilon_{stab}$ and thus:

[4] Very similar results have been proposed by [MNPR06, SSSSS10]

$$\begin{aligned}
\mathbb{E}[R(w)] - R^* &= \mathbb{E}[R_S(w_S^*)] + \epsilon_{stab} + \epsilon_{opt} - R^* \\
&\leq \mathbb{E}[R_S(w^*)] + \epsilon_{stab} + \epsilon_{opt} - R^* \\
&= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{z_i}[f(w^*, z_i)] + \epsilon_{stab} + \epsilon_{opt} - R^* \\
&= R(w^*) + \epsilon_{stab} + \epsilon_{opt} - R^* \\
&= \epsilon_{stab} + \epsilon_{opt}
\end{aligned}$$

□

We are now ready to explore a few questions in the coming chapters: how do we prove an algorithm is uniformly stable? How useful in stability under small violations of our model assumptions?

Chapter 3

Proving Stability: Growth Divergence of Iterative Methods

Recall that in the previous chapter we observed that a uniformly stable algorithm implied generalization. However, a definition of uniform stability is not very useful if we cannot design algorithms that satisfy this condition. Therefore, in this chapter, we explore some techniques that can be used to design uniformly stable algorithms.

3.1 Related work

Stability and regularization Bousquet and Elisseeff showed that some explicit regularization techniques such as Hilbert space regularization and Kullback-Leibler regularization corresponded to models that were uniformly stable [BE02]. More recently, Shalev-Shwartz showed that in general loss functions that were lipschitz and strongly convex were uniformly stable [SSSSS10] (note that L2 regularization turns a convex function into a strongly convex one)! These results on stability provide a fresh and exciting new perspective on the role of regularization in machine learning.

On the same theme of regularization, research has suggested that tuning optimization hyperparameters such as the number of training epoches serve as a form of implicit regularization that can potentially improve generalization [LCR16]. This builds on previous work by Hardt et al [HRS16] that showed that one can control uniform stability bounds of models trained with stochastic gradient descent by only training for a finite number of epoches. In this approach, the *algorithm* is stochastic gradient descent (SGD) with respect to the loss function as the objective. In this chapter, we investigate what happens when the optimization algorithm is chosen to be different from SGD. In particular, the algorithm we investigate is stochastic gradient descent with momentum (SGDM).

Accelerated methods and momentum Acccelerated gradient descent or momentum methods have been well studied in the setting of a fixed objective function. Some popular versions include Nesterov's accelerated gradient [Nes98] and also the heavy ball method by Polyak [Pol64]. In this chapter we will work with a variation on the latter.

One advantage of momentum is that it allows us to escape "shallow" local minima, and navigate other peculiar features of nonlinear cost functions e.g. when training neural networks (see [Ber99] and references within). Additionally, a problem that arises in gradient descent methods is that the basic algorithm has trouble navigating ravines, i.e. surfaces that are alternately very steep and very flat along the path of the algorithm. The result is that the iteration vector descends quickly in the first direction but makes very slow progress in the other. Intuitively, momentum helps us avoid this weakness by accelerating the iteration vector in the relevant direction. This advantage can be easily seen from Figure 3-1.

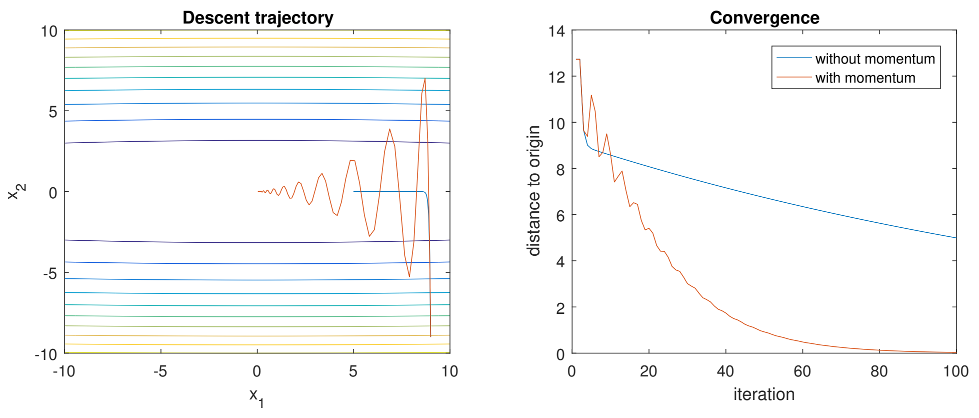

Figure 3-1: Behavior of gradient descent on a fixed objective along a ravine with and without inclusion of a momentum term. In this example, the objective is a quadratic function $f(x) = x^T Qx$ where $Q = [0.1, 0; 0, 1]$ and the hyperparameters of the descent algorithm are hand tuned with learning rate $\alpha = 0.3$ and momentum rate $\gamma = 0.85$.

Clearly, momentum methods have some obvious benefits over gradient descent on a fixed objective. In the stochastic setting however, less is clear [Ber15]. Research has suggested that momentum methods are less robust to noise, which might affect their convergence performance in the stochastic setting (see references within [HRS16]). In either case, the tradeoff between the benefits and weaknesses makes SGDM a fascinating method to study, and the deep connection between robustness, stability and generalization motivates our research in this direction.

The rest of this chapter is lined up as follows: in Section 3.2, we derive stability bounds for models trained with SGDM and note that these bounds depend on the joint spectral radius of a set of matrices associated with our training set. In Section 3.3, we analyze bounds on the joint spectral radius.

3.2 Uniform stability of momentum methods

3.2.1 Setup and assumptions

Here we finally introduce the algorithm that we will be analyzing in this chapter. SGDM is an iterative algorithm defined by the following procedure:

1. The algorithm is initialized with a starting vector w_0 and no initial momentum, i.e. $w_{-1} = w_0$.

2. At each step i of the algorithm, we select an element z_i from S uniformly at random[1]. Then defining $f_i = f(\cdot, z_i)$ we update w_{i+1} with the following rule:

$$w_{i+1} = w_i - \alpha \nabla f_i(w_i) + \gamma(w_i - w_{i-1}) \tag{3.2.1.1}$$

   We assume the algorithm is initialized with no momentum, i.e. $w_{-1} = w_0$.

3. After t iterations, the algorithm outputs w_t

We refer to α as the learning rate, γ as the momentum rate, and t the epoch or iteration number of our algorithm. In our analysis, we consider the case where α, γ are fixed constants that will be determined later.

Now, up to this point, we haven't quite mentioned anything about the nature of the loss functions f. For our proof, we require a few assumptions, which we will formally state below.

Assumption 3.2.1 (Quadratic loss). *$f(\cdot, z)$ is quadratic. This means that for every training example z_i, we have*

$$f_i : w \to w^T Q_i w + b_i w$$

This assumption is a benign one— quadratic functions are commonly used in convergence analysis in nonlinear programming ([Ber15], page 59). The rationale is that a twice differentiable cost function behaves like a quadratic cost function in the neighborhood of a minimum where the Hessian matrix is positive definite.

We observe that with this assumption, $\nabla f_i(w_i) = 2Q_i w_i + b_i$, and so we can explicitly write the recurrence relation in Equation 3.2.1.1 as:

[1]This is a common assumption in the analysis of stochastic gradient methods in many machine learning applications [BCN16, LCR16]. Indeed, there are many different ways in which one can pick z_i, for example, by generating a random permutation and picking each $z_i \in S$ according to the order given by the permutation, i.e. in [HRS16].

$$w_{i+1} = w_i - 2\alpha Q_i w_i - \alpha b_i + \gamma(w_i - w_{i-1})$$

Which is a linear relationship, i.e.

$$\begin{bmatrix} w_{i+1} \\ w_i \\ c_{i+1} \end{bmatrix} = \begin{bmatrix} (1+\gamma)I - 2\alpha Q_i & -\gamma I & -\alpha b_i \\ I & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w_i \\ w_{i-1} \\ c_i \end{bmatrix} \tag{3.2.1.2}$$

$$\mathbf{w}_{i+1} = M_i \mathbf{w}_i$$

Notice that if our initial vector is set with $c_0 = 1$, then $c_{i+1} = c_i = 1$ for all $i \in \mathbb{N}$. So it suffices to consider optimization over the domain $\mathbb{W} \subset \{[a, b, 1] : a, b \in W\}$.

Our second assumption is some notion of continuity of our loss function with respect to our model parameter space.

Definition 3.2.1 (Lipschitz continuity [Nes98])**.** *We say a function f is L-Lipschitz with respect to $||\cdot||$ if for all points u on the domain of f we have*

$$|f(u) - f(v)| \leq L||u - v||$$

Assumption 3.2.2 (Lipschitz continuity)**.** *$f(\cdot, z)$ is L-Lipschitz with respect to the euclidean norm*

A small observation is that quadratic loss functions are not L-Lipschitz if the domain $\mathbb{W}$ is unbounded. Therefore, in order to work with both of these assumptions simultaneously it is implied that $\mathbb{W}$ is bounded. However, for the rest of the proof we will simply assume both assumptions hold and work from there.

A final remark is that we will be working with matrix and vector norms extensively in our proof. So we need to standardize the kind of norms we are working with.

Remark 3.2.1 (Choice of norm)**.** *Unless it is otherwise stated, we write $||\cdot||$ to refer to the operator norm when applied to a matrix and the euclidean norm when applied to a vector*

3.2.2 Proof sketch

Here we describe our proof technique inspired by the methods used by Hardt et al for proving stability of SGD. In a nutshell, we will analyze the growth divergence of the outputs of two instances of the algorithm run on two data sets that differ in precisely one location. Recall that to prove stability we want to bound $\mathbb{E}|f(w,z) - f(w',z)|$. Observe that if $f(\cdot, z)$ is L-Lipschitz then $\mathbb{E}|f(w,z) - f(w',z)| \leq L\mathbb{E}||w - w'||$ for all w and w'. Therefore it suffices to analyze the growth divergence of w_i and w_i' at the i^{th} iteration[2]. Note that we can treat w_i as the output of a switched linear system in discrete time. Therefore we can express the growth as the sum of a product of linear operators, which itself is a linear operator! Bounding the norm of this linear operator will do the trick. We will analyze these bounds in Section 3.3.

We skipped a small detail, which is that the randomness involved in selecting the initial conditions of the algorithm and also each linear operator M_i at each step of the algorithm, is shared between the two running instances of the algorithm. This means that at each step of the algorithm, with probability $1 - \frac{1}{n}$ our two instances pick the same M_i, and with probability $\frac{1}{n}$, our two instances pick a different M_i. We refer the reader to a note on shared randomness in Appendix A.1 if this is confusing, but the general idea is that this property enables a clever way of factorizing out the linear operators so that the final expression for the stability bound is simple and elegant.

3.2.3 Growth recursion of SGDM

Suppose we run SGDM on two adjacent sample sets S and S' for t steps each. This corresponds to applying Equation 3.2.1.2 for t iterations to $\mathbf{w}_0$ and $\mathbf{w}_0'$ respectively. Furthermore, by our independence assumption and the law of total expectation we have:

[2]In Hardt et al, it also sufficed to bound $||w_i - w_i'||$ recursively in expectation as a function of $||w_{i-1} - w_{i-1}'||$. Here the analysis differs slightly. For accelerated methods, we can't bound the growth like in Hardt et al, as accelerated methods can have unbounded expansiveness.

$$\begin{aligned}
\mathbb{E}[\mathbf{w}_t - \mathbf{w}'_t] &= \mathbb{E}\left[\left(\prod_{i=0}^{t-1} M_i\right)\mathbf{w}_0 - \left(\prod_{i=0}^{t-1} M'_i\right)\mathbf{w}'_0\right] \\
&= \mathbb{E}\left[\left(\prod_{i=0}^{t-1} M_i\right) - \mathbb{E}\left[\prod_{i=0}^{t-1} M'_i \middle| M_0 \dots M_{t-1}\right]\right]\mathbb{E}[\mathbf{w}_0] \\
&= \mathbb{E}\left[\left(\prod_{i=0}^{t-1} M_i\right) - \left(\prod_{i=0}^{t-1} \mathbb{E}[M'_i|M_i]\right)\right]\mathbb{E}[\mathbf{w}_0]
\end{aligned}$$

As we discussed earlier, a key observation is that at each step i of the SGDM, the samples drawn from both sets are the same with probability $1 - \frac{1}{n}$, and different with probability $\frac{1}{n}$. Note also that whenever the drawings are different, they will always correspond to the same pair of examples, since S and S' differ by exactly one element. So without loss of generality, we can denote M_i to be the drawing that was the same and $M'' \neq M_i$ to be the exact drawing that was different.

$$\mathbb{E}\left[\left(\prod_{i=0}^{t-1} M_i\right) - \left(\prod_{i=0}^{t-1} \mathbb{E}[M'_i|M_i]\right)\right] = \mathbb{E}\left[\prod_{i=0}^{t-1} M_i - \prod_{i=0}^{t-1}\left(\frac{n-1}{n} M_i + \frac{1}{n} M''\right)\right] \tag{3.2.3.1}$$

We can also denote $V_i = M'' - M_i$ for convenience, then

$$\mathbb{E}[\mathbf{w}_t - \mathbf{w}'_t] = \mathbb{E}\left[\prod_{i=0}^{t-1} M_i - \prod_{i=0}^{t-1}\left(M_i + \frac{1}{n} V_i\right)\right]\mathbb{E}[\mathbf{w}_0]$$

Notice that we can expand the second product into a sum of terms:

$$\begin{aligned}
\prod_{i=0}^{t-1}\left(M_i + \frac{1}{n}V_i\right) &= \prod_{i=0}^{t-1} M_i + \frac{1}{n}\sum_{j=0}^{t-1}\left(\prod_{i=0}^{j-1} M_i\right) V_j \left(\prod_{i=j+1}^{t-1} M_i\right) \\
&\quad + \frac{1}{n^2}\sum_{0\le j<k\le t-1}\left(\prod_{i=0}^{j-1} M_i\right) V_j \left(\prod_{i=j+1}^{k-1} M_i\right) V_k \left(\prod_{i=k+1}^{t-1} M_i\right) + \dots
\end{aligned} \tag{3.2.3.2}$$

Combining these expressions together we obtain

$$\mathbb{E}[\mathbf{w}_t - \mathbf{w}'_t] = -\mathbb{E}\left[\frac{1}{n}\sum_{j=0}^{t-1}\left(\prod_{i=0}^{j-1} M_i\right) V_j \left(\prod_{i=j+1}^{t-1} M_i\right) + \frac{1}{n^2}\dots\right]\mathbb{E}[\mathbf{w}_0] \tag{3.2.3.3}$$

Which is a vector expression that characterizes the growth divergence between the two vectors $\mathbf{w}_t$ and $\mathbf{w}'_t$. Notice that there are $2^t - 1$ products of matrices which are each t matrices long. This analysis has the rather interesting interpretation that we can think of SGDM as some unnormalized distribution of $2^t - 1$ possible "paths" taken by a initial vector. The expected growth is then proportional to the expectation over the final destinations of these paths.

Now taking norms[3] on both sides and applying the triangle inequality we get:

$$\mathbb{E}||\mathbf{w}_t - \mathbf{w}'_t|| \leq \mathbb{E}\left[\frac{1}{n}\sum_{j=0}^{t-1}\left|\left|\left(\prod_{i=0}^{j-1} M_i\right) V_j \left(\prod_{i=j+1}^{t-1} M_i\right)\right|\right| + \frac{1}{n^2}\dots\right]\mathbb{E}||\mathbf{w}_0|| \tag{3.2.3.4}$$

So, in order to bound the growth it suffices to bound the expression on the right. Note that this expression depends on a constructed vector $\mathbf{w}_i = [w_i, w_{i-1}, 1]^T$, but we can always express $||\mathbf{w}_0||$ as $\sqrt{2 \cdot ||w_0||^2 + 1}$ when $||\mathbf{w}_0||$ is the euclidean norm and when we have $w_0 = w_{-1}$ i.e. no initial momentum.

Upper bounds

Recall that there is a bijection between training examples $z_i \in S$ and operators M_i. So we can denote Σ to be the set of all M_i, V_i corresponding to S. Note that Σ is data-dependent and finite with cardinality $2n$, containing n elements $M_0 \dots M_n$ and n elements $V_0 \dots V_n$.

We adopt the following notation from [Jun09] to denote the product of matrices of length t:

[3]Note that the pair of norms chosen in Remark 3.2.1 have been chosen to be consistent

$$\Sigma^t \triangleq \{A_0 \dots A_{t-1} : A_i \in \Sigma\}$$

And also to denote the "maximum size" of products of length t:

$$\hat{\rho}_t(\Sigma, ||\cdot||) \triangleq \sup_{A \in \Sigma^t} ||A||^{\frac{1}{t}}$$

When it is clear from context we write $\hat{\rho}_t(\Sigma)$ or sometimes simply $\hat{\rho}_t$. Indeed, in this section we have already established that $||\cdot||$ refers to the operator norm and that Σ is defined above. Anyway, we observe that we can bound the norm of every product of matrices in Equation 3.2.3.4 with the supremum of the norm over all products of matrices drawn from Σ^t, e.g.

$$\left|\left|\left(\prod_{i=0}^{j-1} M_i\right) V_j \left(\prod_{i=j+1}^{t-1} M_i\right)\right|\right| \leq \hat{\rho}_t^t$$

Then we can simplify Equation 3.2.3.4 into:

$$\begin{aligned}
\mathbb{E}||\mathbf{w}_t - \mathbf{w}'_t|| &\leq \left(\frac{\hat{\rho}_t^t}{n}\binom{t}{1} + \frac{\hat{\rho}_t^t}{n^2}\binom{t}{2} + \dots\right)\mathbb{E}||\mathbf{w}_0|| \\
&= \hat{\rho}_t^t \left(\sum_{k=1}^{t} \frac{1}{n^k}\binom{t}{k}\right) \cdot \mathbb{E}||\mathbf{w}_0|| \\
&= \hat{\rho}_t^t((1 + \frac{1}{n})^t - 1) \cdot \mathbb{E}||\mathbf{w}_0|| \qquad (3.2.3.5)
\end{aligned}$$

Now we are ready to prove the uniform stability of SGDM:

Theorem 3.2.1. *SGDM with respect to a loss function f that is quadratic and L-Lipschitz satisfies uniform stability with rate*

$$\epsilon = L((1 + \frac{1}{n})^t - 1) \cdot \mathbb{E}||\mathbf{w}_0|| \cdot \sup_{\Sigma} \hat{\rho}_t(\Sigma)^t$$

Proof. Firstly, $\mathbb{E}||\mathbf{w}_t - \mathbf{w}'_t||$ is also an upper bound on growth of w in our original parameter space, i.e.

$$\mathbb{E}||w_t - w'_t|| \leq \mathbb{E}||\mathbf{w}_t - \mathbf{w}'_t||$$

So for all $z \in Z$,

$$\begin{aligned}
\mathbb{E}|f(w_t, z) - f(w'_t, z)| &\leq L\ \mathbb{E}||w_t - w'_t|| \\
&\leq L\ \mathbb{E}||\mathbf{w}_t - \mathbf{w}'_t|| \\
&\leq L\hat{\rho}_t^t((1 + \frac{1}{n})^t - 1) \cdot \mathbb{E}||\mathbf{w}_0||
\end{aligned}$$

Where the first inequality comes from the Lipschitz continuity of f, and the last inequality comes from Equation 3.2.3.5. Note that since $w_t = A(S)$, the above bound is data-dependent. To obtain uniform stability bounds, we need to take the supremum over all adjacent datasets S, S':

$$\sup_{S,S'} \mathbb{E}|f(A(S), z) - f(A(S'), z)| \leq L((1 + \frac{1}{n})^t - 1) \cdot \mathbb{E}||\mathbf{w}_0|| \cdot \sup_{\Sigma} \hat{\rho}_t(\Sigma)^t$$

□

Note that $\epsilon = O(\frac{1}{n})$, meaning that our generalization bounds converge as our training sets get larger. Even though this "satisfies the stability condition" so to speak, it is important to point out that converging bounds are not very meaningful when the algorithm is essentially a finite series of updates— as the size of the dataset increases, the probability that we encounter the bad sample vanishes anyway! Generally we want n and t to be related in some meaningful way. In practice, large scale machine learning models are sometimes trained with an online learning procedure that takes a single pass at the data (this was noted in [LCR16]). With such a procedure, the size of the training set is equal to the number of SGDM epoches, i.e. $t = n$.

Corollary 3.2.1. *With a single pass of the data, SGDM satisfies uniform stability with rate*

$$\epsilon = L(e-1) \cdot \mathbb{E}||\mathbf{w}_0|| \cdot \sup_{\Sigma} \hat{\rho}_t(\Sigma)^t$$

Proof. The proof is simply to note that $e = \lim_{m\to\infty}(1+\frac{1}{m})^m > (1+\frac{1}{n})^n$ for any $n \in \mathbb{N}$ □

It remains to find bounds on $\hat{\rho}_t$. Indeed, if $\hat{\rho}_t < 1$, then we get strong stability bounds that vanish exponentially with t. If $\hat{\rho}_t > 1$, then we get weak stability bounds that grow exponentially with t. In light of this behavior we would like to analyze the asymptotic behavior of $\hat{\rho}_t(\Sigma)$, a quantity known as the joint spectral radius of Σ:

Definition 3.2.2 (The Joint Spectral Radius [RS60])**.** *The joint spectral radius of a set of matrices Σ is the limit:*

$$\hat{\rho}(\Sigma) \triangleq \lim_{t\to\infty} \hat{\rho}_t(\Sigma, ||\cdot||)$$

An important result in spectral theory allows us to relate the joint spectral radius to some other more convenient quantities. Let us denote

$$\rho_t(\Sigma) \triangleq \sup_{A\in\Sigma^t} \rho(A)^{\frac{1}{t}} \qquad \rho(\Sigma) = \lim_{t\to\infty} \rho_t(\Sigma)$$

Where $\rho(A)$ is the spectral radius of a matrix A, i.e. the maximum modulus of its eigenvalues. Then it is known that for any bounded set Σ we have the following theorem:

Theorem 3.2.2 (The Joint Spectral Radius Theorem [Jun09][4])**.** *For bounded sets Σ, we have*

$$\rho(\Sigma) = \hat{\rho}(\Sigma)$$

From here on we can simply denote the joint spectral radius as $\rho(\Sigma)$ also, due to the above equality. So our problem is now reduced to finding bounds on $\rho(\Sigma)$.

[4][Jun09] attributes this result to Berger and Wang (1992)

3.3 The joint spectral radius

In the previous section the stability bounds we obtained depended on $\rho(\Sigma)$. At this junction, it makes sense to look at the structure of Σ. Recall that Σ was constructed as the set of linear operators associated with the inhomogeneous quadratic form, i.e. $\Sigma = \{M_i\} \cup \{V_i\}$, with

$$M_i = \begin{bmatrix} (1+\gamma)I - 2\alpha Q_i & -\gamma I & -\alpha b_i \\ I & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad V_i = \begin{bmatrix} 2\alpha(Q - Q_i) & 0 & \alpha(b - b_i) \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

For convenience, we can also construct another set of linear operators Σ' that is associated with the homogeneous quadratic form, i.e. $\Sigma' = \{M_i'\} \cup \{V_i'\}$, with

$$M_i' = \begin{bmatrix} (1+\gamma)I - 2\alpha Q_i & -\gamma I \\ I & 0 \end{bmatrix} \qquad V_i' = \begin{bmatrix} 2\alpha(Q - Q_i) & 0 \\ 0 & 0 \end{bmatrix}$$

As it turns out, we can relate $\rho(\Sigma)$ and $\rho(\Sigma')$ with the following lemma holds:

Lemma 3.3.1. $\rho(\Sigma) = \max\{\rho(\Sigma'), 1\}$.

Proof. Observe that

$$M_i = \left[\begin{array}{cc|c} (1+\gamma)I - 2\alpha Q_i & -\gamma I & -\alpha b_i \\ I & 0 & 0 \\ \hline 0 & 0 & 1 \end{array}\right] = \begin{bmatrix} M_i' & C_i \\ 0 & D_i \end{bmatrix}$$

$$V_i = \left[\begin{array}{cc|c} 2\alpha(Q - Q_i) & 0 & \alpha(b - b_i) \\ 0 & 0 & 0 \\ \hline 0 & 0 & 0 \end{array}\right] = \begin{bmatrix} V_i' & C_i \\ 0 & D_i \end{bmatrix}$$

And consider that for every $B_i, B_j \in \Sigma$ corresponding to $B_i', B_j' \in \Sigma'$,

$$B_i B_j = \begin{bmatrix} B_i' & C_i \\ 0 & D_i \end{bmatrix} \begin{bmatrix} B_j' & C_j \\ 0 & D_j \end{bmatrix} = \begin{bmatrix} B_i'B_j' & B_i'C_j + C_iD_j \\ 0 & D_iD_j \end{bmatrix}$$

Finally

$$\rho(B_i B_j) = \max\{\rho(B_i' B_j'), \rho(D_i D_j)\}$$
$$\rho_t(\Sigma) = \max\{\rho_t(\Sigma'), 1\}$$

Taking the limit $t \to \infty$ completes the proof. □

Since $\rho(\Sigma) \geq 1$, so the bad news is that we will never get strong stability bounds. However, this makes sense as a stochastic method never really converges to a minimum, but instead to a region of confusion [BCN16, Ber15], therefore, we don't expect the stability error to vanish. For quadratic loss functions, the region of confusion is characterized by the affine offsets $b_i w$.

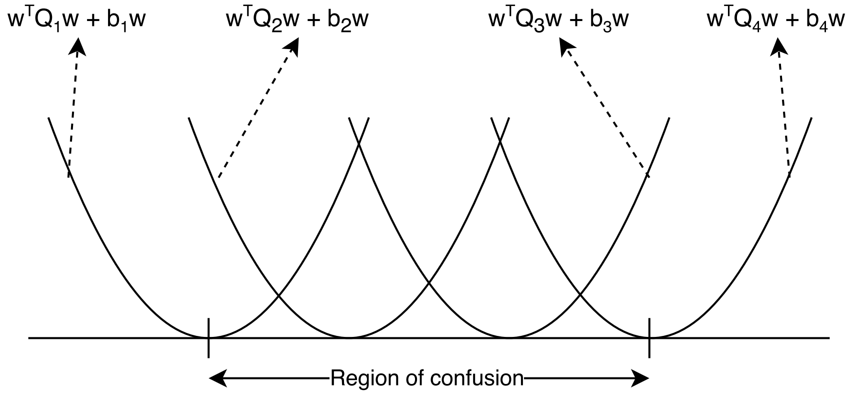

Figure 3-2: Region of confusion characterized by affine offsets $b_i w$. If we suppose $\rho(\Sigma) < 1$, then $\rho(\{M_i\}) < 1$. This implies that the stochastic method converges to a minimum, and so the region of confusion must be zero, a contradiction.

Even though we cannot get strong stability bounds, there is still a silver lining. By combining Lemma 3.3.1 with Corollary 3.2.1, we note that we can get a constant stability bound:

Corollary 3.3.1. *If $\rho(\Sigma') \leq 1$, then SGDM satisfies uniform stability with rate $\epsilon = L(e-1) \cdot \mathbb{E}||\mathbf{w}_0||$.*

Therefore, it is perhaps meaningful to investigate the spectral radius of the set of operators associated with only the homogeneous quadratic form. For the rest of this chapter we will thus denote Σ as Σ', i.e. we will only need to consider Σ that is of the form:

1. Σ is finite with cardinality $2n$
2. Σ contains exactly n elements of the form

$$V_i = \begin{bmatrix} 2\alpha(Q - Q_i) & 0 \\ 0 & 0 \end{bmatrix}$$

   for each $Q_i \in \mathbf{Q}$, and Q corresponds to the element in S' that differs from S.

3. The remaining n elements of Σ are of the form

$$M_i = \begin{bmatrix} (1+\gamma)I - 2\alpha Q_i & -\gamma I \\ I & 0 \end{bmatrix}$$

   for each $Q_i \in \mathbf{Q}$

3.3.1 Some experimental results

Recall that in order to get stability bounds, we needed to find $\sup_\Sigma \rho(\Sigma)$. In order to gain some intuition about the behavior of this quantity, it can be helpful to numerically compute $\rho(\Sigma)$ under some computer generated Σ. This will be the focus of this section. We will begin our generation with a restricted set of matrices and see what happens when we relax these restrictions. A starting point is to control the condition number of Q_i from our quadratic loss function in Assumption 3.2.1:

Assumption 3.3.1. *Let Q_i be positive definite with $\eta I \preceq Q_i \preceq \eta_L I$.*

Computing $\rho(\Sigma)$ poses a challenge too, at least, on first glance it does not appear to be explicitly computable from its definition. Fortunately, we can rely on a theorem from Rota and Strang [RS60] to help us:

Theorem 3.3.1 (The joint spectral radius is the infinum over all norms [RS60])**.**

$$\rho(\Sigma) = \inf_{||\cdot||} \sup_{A\in\Sigma} ||A||$$

This theorem is surprising and also very powerful, as it implies that to find $\rho(\Sigma)$, we don't need to compute long products of matrices anymore, potentially requiring an exponential number of computations in the size of the dataset. Indeed, since we are only concerned with finding an upper bound on $\rho(\Sigma)$, it suffices to find a "good enough" norm. The theorem implies that for any given norm $||\cdot||$,

$$\rho(\Sigma) \leq \sup_{A\in\Sigma} ||A||$$

One particular candidate norm is the norm under a similarity transformation:

$$||A||_T \overset{\triangle}{=} ||T^{-1}AT||$$

Where the norm on the r.h.s. is the operator norm. If we can find a matrix T such that conjugation under T simultaneously reduces the operator norm for all elements $A \in \Sigma$, then $||\cdot||_T$ can be a good norm.

Our results from this endeavor are inconclusive, but we have experimented with several quantities, and found that taking $T = \frac{1}{n}\sum_{i=1}^{n} M_i$ yields pretty good results some of the time, depending on our condition number $\frac{\eta_L}{\eta}$. We present below a table and also a linear regression plot:

η	η_L	$\frac{\eta_L}{\eta}$	α	γ	$\sup_{A\in\Sigma} \|\|A\|\|_T$
3	4	1.33	0.1436	0.0718	0.4279
5	8	1.6	0.0780	0.1170	0.6311
3	5	1.67	0.1270	0.1270	0.6688
5	9	1.8	0.0729	0.1459	0.8453
1	2	2	0.3431	0.1716	0.9871
5	11	2.2	0.0649	0.1946	1.1116
3	7	2.33	0.1044	0.2087	1.2014

Figure 3-3: Some values of $\sup_{A\in\Sigma} ||A||_T$ under some hyperparameter combinations. We generate Σ based on 100 random 10×10 PSD matrices that satisfy $\eta I \preceq Q_i \preceq \eta_L I$, and choose $T = \frac{1}{n}\sum_{i=1}^{n} M_i$. The remaining hyperparameters are computed as $\alpha = \frac{2}{(\sqrt{\eta}+\sqrt{\eta_L})^2}$ and $\gamma = \frac{\sqrt{\eta}-\sqrt{\eta_L}}{\sqrt{\eta}+\sqrt{\eta_L}}$.

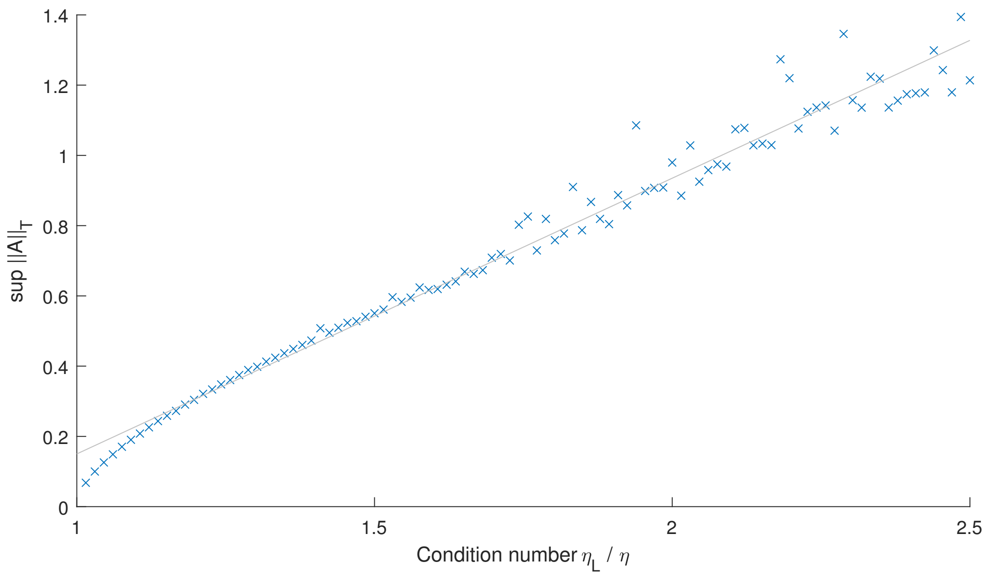

Figure 3-4: A regression plot of condition number $\frac{\eta_L}{\eta}$ vs $\sup_{A\in\Sigma} ||A||_T$, and with $T, \Sigma, \alpha, \gamma$ generated and computed similar to Figure 3-3.

These plots give us some intuition as to the behavior of SGDM in the limit. Since we know $\sup_{A\in\Sigma} ||A||_T \geq \rho(\Sigma) = \lim_{t\to\infty} \hat{\rho}_t(\Sigma)$, so this also gives us some idea of the maximum growth of two instances of SGDM run on specifically generated adjacent datasets, under certain hyperparameter combinations. From Equation 3.2.3.5 we can see that we can get good bounds on the growth when $\lim_{t\to\infty} \hat{\rho}_t(\Sigma) < 1$.

Unfortunately beyond these plots we don't have a very interesting theory to support our experimental results. Furthermore uniform stability bounds require taking the supremum over all adjacent datasets, but the plots that we have in Figure 3-4 are data-dependent, i.e. they depend on the generation of Σ.

3.4 Discussion

Let us recap some of the results in this chapter. We obtained an expression for the growth divergence of SGDM, which in the limit can be upper bounded by the joint spectral radius. We then showed that it sufficed to consider only the homogeneous quadratic form to analyze the joint spectral radius. Finally, we ran some experiments to compute the joint spectral radius for computer generated Σ.

One of the interesting implications of our results is that the stability bounds appear to be independent of the magnitude of the affine offsets. So even though there might be a very large region of confusion, the stability bounds can still be tight. In fact, Corollary 3.3.1 tells us that for well conditioned problems, the stability bounds that we obtained only depend on the Lipschitz continuity of the loss function and the initial starting point in the algorithm.

Secondly, in the larger picture, our work represents some progress in a nascent research direction of trying to obtain generalization bounds by analyzing the behavior of an algorithm, departing from previous work that simply analyzes the solution of an optimization problem. By comparing the results of SGDM that we obtain with the results of SGD from [HRS16], it is evident that different optimization methods can yield very different generalization bounds. This demonstrates potential benefit of this method of algorithmic analysis, as some algorithms can generalize better than others.

This leads to a couple of interesting questions and directions to pursue in future work:

1. Is it possible to find a class of algorithms such that all other classes of algorithms cannot generalize better than it? We note that this bears some similarity to the concept of admissibility in statistics. Though in statistics, it is much easier to analyze this because one can explicitly calculate the risk.

2. One thing that we did not discuss is the convergence rate of our methods. One of the appeals of SGDM is that it appears to converge much faster than SGD. So it is of theoretical and practical interest that we can design methods that can converge quickly and also generalize well.

Chapter 4

Stability and Domain Adaptation

4.1 Motivation

Stability is a limited measure of robustness Recall that in Section 2.3 we characterized stability as a measure of robustness to changes in data. This worked well under the assumption that training samples are drawn from the same distribution that is used to evaluate the risk. In many applications however we often want to think of generalization as the ability of our machine learning models to adapt to the real world, which may be different from the training environment. Can stability still provide generalization guarantees under this change of assumptions?

A recent experimental result by Zhang et al. [ZBH+17] showed that by partially corrupting some of the labels in training data, a neural network exhibits poor generalization ability. They contaminated a fraction of training labels by replacing them with labels picked uniformly at random from the label class (notably different from resampling from the true distribution), and observed poor testing error. It was concluded that a new theory of generalization was needed to account for the setting where the training distribution is affected by data contamination.

Previous work has examined the question of contamination from various angles as early as in 1960s. In the field of robust estimation, Huber studied a contamination model [Hub64] in which with some known probability, samples from training data are replaced from a contaminating distribution. Over the years, many methods have been developed to successfully estimate parameters of a distribution under contamination. As a simple example, it is known that for gaussian distributions the median is a robust approximation for the true mean even though the dataset contains a fraction of outliers that deviate grossly from the bulk of the dataset. These methods work well in estimation theory, but less is clear in the context of learning where we are instead concerned with finding a hypothesis that can approximate an unknown distribution.

In learning theory, this setting arises in the study of domain adaptation, where one assumes the model learns from a source domain and its performance is evaluated on a target domain. Ben-David et al. [BDBCP07] investigated the target generalization performance of a specific classifier trained on a source domain, and obtained bounds using a capacity argument that depended on the variational distance between the distributions of the two domains.

So there appears to be a disconnect between the way generalization is treated in estimation theory compared to learning theory. Notably, results in estimation theory have been able to produce methods that can work well under adversarial contamination whereas the generalization bounds obtained in learning theory depend on some notion of distance between the source and target domains. Furthermore, the fact that uniform stability is often touted as a distribution independent property that can guarantee generalization seems to suggest that there might be something worth looking at in this direction. So the goal of this chapter is to explore the setting of domain adaptation through the lens of stability.

4.2 Preliminaries

As a starting point for a better notion of generalization we consider the case where the training distribution P differs from the true distribution D. Let $S = \{z_1 \dots z_n\}$ be a training set of z_i drawn i.i.d. from P, and A a learning algorithm[1], then we can define the generalization error:

[1]In this chapter, the randomization of A is unimportant, so we drop the expectation over A that we used in Section 2.2 and assume A is a deterministic algorithm. The analysis is unaffected.

$$\epsilon_{gen} \triangleq R(A(S)) - R_S(A(S))$$

with

$$R(w) \triangleq \mathbb{E}_{z\sim D} f(w, z)$$

and R_S remains unchanged as the average of sample losses

$$R_S(w) \triangleq \frac{1}{n}\sum_{i=1}^{n} f(w, z_i)$$

Since ϵ_{gen} is defined with respect to P and D, so we can also explicitly write $\epsilon_{gen}(P, D)$ when it is not already clear from context. Note that when $P = D$, then we are back to the original generalization error defined in section 2.2, which we will denote as $\epsilon_{gen}(P, P)$ in this section. Now, we can write ϵ_{gen} as a sum of terms:

$$\begin{aligned}\epsilon_{gen} &= R(A(S)) - R_S(A(S)) \\ &= f(A(S), z) - R_S(A(S)) + \mathbb{E}_{z\sim D} f(A(S), z) - \mathbb{E}_{z\sim P} f(A(S), z) \\ &= \epsilon_{gen}(P, P) + \mathbb{E}_{z\sim D} f(A(S), z) - \mathbb{E}_{z\sim P} f(A(S), z)\end{aligned}$$

The first term is the original generalization error and the second term represents the sensitivity of the loss function to changes in the probability distribution. It is useful to define:

$$\epsilon_{rob} \triangleq \mathbb{E}_{z\sim D} f(A(S), z) - \mathbb{E}_{z\sim P} f(A(S), z)$$

Which we will refer to as the robustness error with respect to P and D. Same as before, we can also write $\epsilon_{rob}(P, D)$ or ϵ_{rob} depending on context. This enables us to write

$$\epsilon_{gen}(P, D) = \epsilon_{gen}(P, P) + \epsilon_{rob}(P, D) \tag{4.2.0.1}$$

This decomposition will simplify our analysis later.

4.2.1 A simple mixture model

Clearly ϵ_{gen} is not very interesting if P and D are allowed to be arbitrary distributions, as ϵ_{rob} can take on arbitrary values on the range of $f(w, \cdot)$. So instead we can expect that ϵ_{gen} depends on the similarity between P and D, and we can define a distance measure between P and D (this does not need to be a distance in the topological sense) and consider P, D that are close to each other. Several distance measures have been investigated in the field of domain adaptation, such as the variational distance, $\mathcal{A}$-distance [BDBCP07], and discrepancy distance [MMR09], but in this section we consider Huber's contamination model, which we briefly introduced in Section 4.1. This can be formalized as the following mixture model

$$P = (1 - \delta)D + \delta Q \tag{4.2.1.1}$$

Where P is our training distribution, D is our true distribution, and Q is picked to be some contaminating distribution (for example, in [ZBH+17], Q is uniformly random noise). Note that this implies bounded variational distance (see A.2). The mixture model not only has a good interpretation, but also provides a convenient expression of the generalization error in terms of δ. Observe that when drawing our training examples $z \sim P$, we are sampling with probability $1 - \delta$ from D, and with probability δ from Q. Let the probability density functions of P, D, Q be p_P, p_D, p_Q respectively, then the robustness error $\epsilon_{rob}(P, D)$ can be written as:

$$\begin{aligned}
\epsilon_{rob}(P, D) &= \mathbb{E}_{z\sim D} f(w, z) - \mathbb{E}_{z\sim P} f(w, z) \\
&= \int_{z\in Z} f(w, z) p_D(z) dz - \int_{z\in Z} f(w, z) p_P(z) dz \\
&= \int_{z\in Z} f(w, z) p_D(z) dz - \int_{z\in Z} f(w, z)[(1-\delta) p_D(z) + \delta p_Q(z)] dz \\
&= \delta \cdot \left[\int_{z\in Z} f(w, z) p_D(z) dz - \int_{z\in Z} f(w, z) p_Q(z) dz \right] \\
&= \delta \cdot [\mathbb{E}_{z\sim D} f(w, z) - \mathbb{E}_{z\sim Q} f(w, z)] \\
&= \delta \cdot \epsilon_{rob}(Q, D)
\end{aligned}$$

Putting this into Equation 4.2.0.1, the generalization error can be also expressed as

$$\epsilon_{gen}(P, D) = \epsilon_{gen}(P, P) + \delta \cdot \epsilon_{rob}(Q, D) \tag{4.2.1.2}$$

For the rest of this chapter, we will investigate generalization under contamination in the sense described above. How do we guarantee that our model is robust to contamination?

4.2.2 Measures of robustness

If generalization is the property that the results of a method are influenced slightly by small changes in the data, then robustness is the property that the results of a method are influenced slightly by small deviations from the assumed model. To begin characterizing robustness, a good starting point is to find bounds on the generalization performance of our algorithm with respect to contamination. Note that with the distance measure from Equation 4.2.1.1 we can define a δ-contamination neighborhood:

$$N_\delta(D) = \{(1-\delta)D + \delta Q : Q \text{ is any distribution}\}$$

We can also define is the worst possible generalization over all distributions in the δ-contamination neighborhood of D:

$$E(\delta) \overset{\triangle}{=} \sup_{P \in N_\delta(D)} \epsilon_{gen}$$

Which we refer to $E(\delta)$ as the δ-robust generalization error associated with algorithm A. Then one natural way to characterize the robustness of an algorithm in relation to $E(\delta)$ is a notion of breakdown:

Definition 4.2.1 (Finite-sample breakdown point [Hub81]). *The finite-sample breakdown point of a learning algorithm is the smallest K such that for every $\delta > K$ the δ-robust generalization error is unbounded.*

We have used this terminology somewhat irresponsibly as the original use case of finite-sample breakdown point is in estimation theory and is a property describing estimators. However, we feel this conveys a similar intuition when describing the behavior of our learning algorithm with respect to its generalization performance.

Note that unboundedness depends trivially on certain aspects of the problem instance, such as the loss function f. For example, if we have bounded f then every algorithm will never breakdown (i.e. have a finite-sample breakdown point of 1). To avoid these kinds of trivial examples, we need a more descriptive property. As a starting point, we can define $\hat{E}$, the worst δ-robust generalization error over all possible learning algorithms:

$$\hat{E}(\delta) \overset{\triangle}{=} \sup_{A} E(\delta)$$

Which we can also write as $\hat{E}$ when it is clear from context. In the context of the mixture model we think it is natural to introduce a new definition of breakdown:

Definition 4.2.2 (Contamination breakdown point). *The contamination breakdown point of a learning algorithm is the smallest K such that for every $\delta > K$ the δ-robust generalization error is at least $\delta \cdot \hat{E}$.*

We can intuitively think about this new breakdown as defining some linear threshold of badness and finding the smallest contamination that admits an instance that crosses that threshold. Under Huber's contamination model, the contamination breakdown point also admits a natural interpretation: the contamination breakdown point is where the algorithm starts to behave in a way that it completely ignores the robustness error (i.e. no better than assuming $P = D$ in the worst case). This relationship can be seen from Equation 4.2.1.2.

A short note on some previous work

Before we proceed with our proofs we would like to remark that, similar to generalization, the term "robustness" does not refer to an agreed upon formal concept or quantity. This creates a semantic problem as the terms "generalization" and "robustness" conveys similar meaning in common language to a general audience. There is indeed a body of literature that is dedicated to investigating measures of robustness that bear resemblance to the work we reviewed in Sections 2.2 and 2.3. For completeness, let us introduce a few of the tools used to explore robustness in this context: the influence function, proposed by Hampel, and the sensitivity curve, proposed by Tukey. Both of these definitions can be found standard references like Huber [Hub81].

Definition 4.2.3 (Influence function [Hub81])**.** *The influence function of T at a point z for a distribution D is the special Gâteaux derivative (if it exists)*

$$IF(z; T, D) = \lim_{\epsilon \to 0} \frac{T((1-\epsilon)D + \epsilon\Delta_z) - T(D)}{\epsilon}$$

where Δ_z is the Dirac distribution at the point z such that $\Delta_z(z) = 1$.

It is easy to see that the following definition of sensitivity curve is some kind of discretization of the influence function:

Definition 4.2.4 (Sensitivity curve [Hub81])**.** *The sensitivity curve of T at a point z given a data set $S = \{z_1, \ldots, z_n\}$ is defined by:*

$$SC(z, T) = n(T(S \cup \{z\}) - T(S))$$

Now, we haven't yet defined T. In the context of estimation theory, T is a map that assigns every distribution D an estimator $T(D)$. Previous work by Christmann and Steinwart has extended this to learning theory in which T is a map from the space of distributions to a Banach space of prediction functions [CS04]. In our notation, T is simply our learning algorithm A that maps training sets S to parameters in W. Then the robustness guarantees are provided by proving boundedness of the sensitivity curve or influence function over the domain $z \in Z$. Below we show that this kind of guarantee is essentially a stability argument.

Theorem 4.2.1 (Sensitivity implies stability). *Let f be a loss function with $f(\cdot, z)$ is L-Lipschitz in W for all $z \in Z$. Suppose $\sup_{z\in Z} ||SC(z, A)|| \leq \epsilon_{sc}$, then A is uniformly stable with rate $\frac{2L}{n-1}\epsilon_{sc}$.*

Proof. Let S have cardinality k. Firstly, observe that

$$\begin{aligned}
\sup_{z,z'} ||k(A(S \cup \{z\}) - A(S \cup \{z'\})|| &= \sup_{z,z'} ||k(A(S \cup \{z\}) - A(S) + A(S) - A(S \cup \{z'\})|| \\
&\leq \sup_{z} ||k(A(S \cup \{z\}) - A(S))|| + \sup_{z'} ||k(A(S \cup \{z'\}) - A(S))|| \\
&\leq 2\epsilon_{sc}
\end{aligned}$$

So for all adjacent datasets S, S' of cardinality $n = k + 1$, we have

$$||A(S) - A(S')|| \leq \frac{2}{n-1}\epsilon_{sc}$$

Finally, the Lipschitz condition tells us that for all $z \in Z$,

$$f(A(S), z) - f(A(S'), z) \leq \frac{2L}{n-1}\epsilon_{sc}$$

□

Since the influence function is simply a continuous analog of the sensitivity curve, so a similar result holds in the other case.

4.3 Uniform stability is insufficient for generalization under domain adaptation

Recall from Theorem 2.3.1 that under the traditional learning model where $P = D$, uniform stability implied tight bounds on $E(\delta)$. Then a natural question to ask is: when $P \neq D$, is uniform stability a sufficient condition to do the same? In this section, we claim that the answer is no, according to the measures of robustness provided in Section 4.2.2. We will provide counterexamples, but first, a lemma:

Lemma 4.3.1. *Fix loss function f and distributions D, Q and let $P = (1-\delta)D + \delta Q$. There exists a uniformly stable algorithm A such that the generalization error is equal to $\delta \cdot \sup_w [\mathbb{E}_{z\sim D} f(w,z) - \mathbb{E}_{z\sim Q} f(w,z)]$*

Proof. Note that any algorithm A that is independent of S is uniformly stable with respect to any loss function:

$$\begin{aligned} \sup_z [f(A(S), z) - f(A(S'), z)] &= \sup_z [f(w,z) - f(w,z)] \\ &= 0 \end{aligned}$$

By Theorem 2.3.1 this implies that $\epsilon_{gen}(P, P) = 0$. In particular, fix distributions D and Q for our problem instance, and pick the algorithm A that always outputs $w^* = \arg\max_w [\mathbb{E}_{z\sim D} f(w,z) - \mathbb{E}_{z\sim Q} f(w,z)]$, which we note is independent of S. Then from Equation 4.2.1.2,

$$\begin{aligned} \epsilon_{gen}(P, D) &= \delta \cdot \epsilon_{rob}(Q, D) \\ &= \delta \cdot [\mathbb{E}_{z\sim D} f(A(S), z) - \mathbb{E}_{z\sim Q} f(A(S), z)] \\ &= \delta \cdot \sup_w [\mathbb{E}_{z\sim D} f(w,z) - \mathbb{E}_{z\sim Q} f(w,z)] \end{aligned}$$

□

Theorem 4.3.1. *There exists some f, D and a corresponding uniformly stable algorithm A with a finite-sample breakdown point of 0.*

Proof. Picking f, D adversarially and applying Lemma 4.3.1:

$$\begin{aligned}
\sup_{f,D} E(\delta) &= \sup_{f,D,Q} \epsilon_{gen} \\
&= \delta \cdot \sup_{f,D,Q} \sup_{w} \left[\mathbb{E}_{z\sim D} f(w,z) - \mathbb{E}_{z\sim Q} f(w,z)\right] \\
&= \delta \cdot \sup_{w} \sup_{f,D,Q} \left[\mathbb{E}_{z\sim D} f(w,z) - \mathbb{E}_{z\sim Q} f(w,z)\right] \\
&\geq \delta \cdot \sup_{w,f} \left(\sup_{x} f(w,x) - \inf_{y} f(w,y)\right)
\end{aligned}$$

With the last inequality arising from picking candidate distributions D, Q that output the sup and inf respectively with probability 1. Since these are candidate distributions, so the sup is always larger. Furthermore, since A is uniformly stable with respect to any loss function, so we can just pick a loss function such that the r.h.s. is unbounded. For example, f that is the mean squared error satisfies this condition. □

The theorem above shows that there exists a uniformly stable algorithm with unbounded generalization error, so clearly uniform stability is an insufficient condition for generalization when the training and the true distributions differ.

Our proof relied on picking adversarial distributions and loss functions. In many settings however, the true distribution is often fixed and the loss function already given, and we are only allowed to choose our algorithm. In this situation, to evaluate generalization, we can compare our generalization bounds to $\hat{E}$, using the robustness measure in Definition 4.2.2

Corollary 4.3.1. *Suppose we are given a fixed f and distribution D. Then there exists a uniformly stable algorithm A such that the contamination breakdown point is 0.*

Proof. Recall from the definition of $\hat{E}$:

$$\begin{aligned}
\hat{E}(\delta) &= \sup_{Q,A}[R(A(S)) - R_S(A(S))] \\
&= \sup_{w} \left(\mathbb{E}_{z\sim D} f(w,z) - \frac{1}{n}\sum_{y\in S} f(w,y) \right) \\
&\leq \sup_{w} \left(\mathbb{E}_{z\sim D} f(w,z) - \inf_{y} f(w,y) \right)
\end{aligned}$$

From Lemma 4.3.1,

$$\begin{aligned}
E(\delta) &= \delta \cdot \sup_{Q} \sup_{w} \left[\mathbb{E}_{z\sim D} f(w,z) - \mathbb{E}_{z\sim Q} f(w,z)\right] \\
&= \delta \cdot \sup_{w} \sup_{Q} \left[\mathbb{E}_{z\sim D} f(w,z) - \mathbb{E}_{z\sim Q} f(w,z)\right] \\
&\geq \delta \cdot \sup_{w} \left(\mathbb{E}_{z\sim D} f(w,z) - \inf_{y} f(w,y) \right) \\
&\geq \delta \cdot \hat{E}(\delta)
\end{aligned}$$

□

Corollary 4.3.1 tells us the breakdown point is 0, which implies that there is really nothing about uniform stability that can guarantee control over the robustness; any algorithm that assumes $P = D$ will work just as well in the worst case.

The results in this section are surprising as we have noted earlier in Theorem 4.2.1 that standard notions of robustness imply uniform stability, but Corollary 4.3.1 tells us that this is insufficient to characterize robustness under a simple mixture model.

4.4 Some upper bounds

To recap, in the previous section we found lower bounds on $\mathbb{E}(\delta)$ to show uniform stability was insufficient for generalization. In this section, we will find upper bounds on $\mathbb{E}(\delta)$. First, we can upper bound the generalization error in Equation 4.2.0.1 using the stability argument in Theorem 2.3.1. Let ϵ_{stab} be the stability error. Observe that we can write $E(\delta)$ as

$$\begin{aligned} E(\delta) &= \sup_{P \in N_\delta(D)} [\epsilon_{gen}(P, P) + \epsilon_{rob}(P, D)] \\ &\leq \sup_P \epsilon_{gen}(P, P) + \sup_{P \in N_\delta(D)} \epsilon_{rob}(P, D) \\ &\leq \epsilon_{stab} + \sup_{P \in N_\delta(D)} \epsilon_{rob}(P, D) \\ &= \epsilon_{stab} + \delta \cdot \sup_Q \epsilon_{rob}(Q, D) \end{aligned} \tag{4.4.0.1}$$

We note that this bound is pretty tight in the ERM setting, as the generalization error is always positive:

Remark 4.4.1. *When our learning algorithm is ERM, our expected generalization error is always positive*

Proof. Let w_S^* be the minimizer of R_S. It suffices to prove that $\mathbb{E}_S[R_S(w_S^*)] \leq \mathbb{E}_S[R(w_S^*)]$.

$$\begin{aligned} \mathbb{E}_S[R_S(w_S^*)] &= \mathbb{E}_S\left[\inf_w \frac{1}{n}\sum_{i=1}^n f(w, z_i)\right] \\ &\leq \inf_w \mathbb{E}_S\left[\frac{1}{n}\sum_{i=1}^n f(w, z_i)\right] \\ &= \inf_w R(w) \\ &\leq \mathbb{E}_S[R(w_S^*)] \end{aligned}$$

□

Remark 4.4.2. *When our learning algorithm is a uniformly stable ERM algorithm, the bound in Equation 4.4.0.1 is tight in the limit.*

Proof. Firstly Remark 4.4.1 tells us that ERM implies $\epsilon_{gen}(P, P) > 0$. Furthermore, a uniformly stable algorithm implies ϵ_{stab} exists. Finally, $\sup_{P \in N_\delta(D)} \epsilon_{rob}(P, D) =$

$\delta \cdot \sup_Q \epsilon_{rob}(Q, D)$, so when combined with the above,

$$\delta \cdot \sup_Q \epsilon_{rob}(Q, D) \leq E(\delta) \leq \epsilon_{stab} + \delta \cdot \sup_Q \epsilon_{rob}(Q, D)$$

□

So it suffices to find a good bound for $\sup_Q \epsilon_{rob}(Q, D)$.

4.4.1 (K, γ)-robust algorithm

One way to bound the robustness error is to consider the definition of $(K, \gamma(S))$-robustness from [XM10].

Definition 4.4.1 (Robust algorithm [XM10])**.** *An algorithm A is $(K, \gamma(S))$-robust if Z can be partitioned into K disjoint sets, denoted by $\{C_i\}_{i=1}^{K}$, such that for all $s \in S$,*

$$s, z \in C_i \implies |f(A(S), s) - f(A(S), z)| \leq \gamma(S)$$

Remark 4.4.3. *If A is $(1, \gamma(S))$-robust, then $\sup_Q \epsilon_{rob}(Q, D) \leq \gamma(S)$.*

Proof. From the definition of ϵ_{rob} and taking the sup,

$$\begin{aligned}
\sup_Q \epsilon_{rob}(Q, D) &= \sup_Q \mathbb{E}_{z \sim D} f(A(S), z) - \mathbb{E}_{z \sim Q} f(A(S), z) \\
&\leq \sup_{s, z \in Z} |f(A(S), s) - f(A(S), z)| \\
&\leq \gamma(S)
\end{aligned}$$

□

Note that a $(K, \gamma(S))$-robust algorithm with $K = 1$ is somewhat of a strict requirement. However, this is necessary; Q is chosen adversarially, so it can be picked to output any z on the support. The strict requirement means that this bound is likely to be loose, and that it will also be difficult to find algorithms that satisfy this robustness criterion meaningfully. Finally, this measure is also data dependent; it is unclear how $\gamma(S)$ evolves as $|S| \to \infty$, so we don't know if our upper bound will be tight in the limit.

4.4.2 The Wasserstein distance

Suppose $f(A(S), \cdot)$ is L-Lipschitz. Then the robustness error $\epsilon_{rob}(Q, D)$ can be upper bounded by the Wasserstein distance between D and Q using the Kantorovich-Rubinstein duality theorem[2]. The idea is that the robustness error so happens to be a feasible solution to the dual form of the Wasserstein:

$$\begin{aligned}
\epsilon_{rob}(Q, D) &= \mathbb{E}_{z\sim D} f(A(S), z) - \mathbb{E}_{z\sim Q} f(A(S), z) \\
&\leq L \sup_{g \text{ is 1-Lipschitz}} \mathbb{E}_{z\sim D} g(z) - \mathbb{E}_{z\sim Q} g(z) && \text{"dual Wasserstein OPT"} \\
&= L \inf_{\gamma\in\Pi(D,Q)} \mathbb{E}_{(x,y)\sim\gamma} ||x - y|| && \text{"primal Wasserstein OPT"}
\end{aligned}$$

Where $\Pi(D, Q)$ is the set of all joint distributions $\gamma(x, y)$ whose marginal distributions are D and Q respectively. The Wasserstein distance is also known as the Earth Mover's distance and thus has the natural interpretation that $\gamma(x, y)$ is the amount of work required to transport "mass" from x to y in order to turn D into Q.

This interpretation allows us to see that the generalization error can be bounded by the sum of two terms: the stability error and some distance measure between the contaminating distribution and the true distribution, and is thus expressed in a way that bears similarity to the distribution dependent bounds obtained in previous work.

4.5 Discussion

Our analysis shows that the generalization error under domain adaptation can be decomposed into two terms, the first is the original generalization error, and the

[2] I first learned about the Wasserstein distance reading the Wasserstein GAN paper by Arjovsky et al. [ACB17]. For a complete reference, please refer to [Vil09], but we shall also include a proof of the duality theorem in Appendix A.3 for completeness.

second is the robustness error. While stability can bound the former, we would need a stronger property that can guarantee that both terms are small. Through the lens of stability, this chapter explores the failure of certain machine learning algorithms to generalize, and we hope that our work can motivate future research to design algorithms that are robust under a more natural interpretation of uncertainty.

Finally, we hope that future work will be able to explore a few questions more fully. Recall that in Section 2.3 we noted that uniform stability was a distribution independent property that corresponded to models that yielded a small generalization error. However, all the generalization bounds we obtained in this chapter are distribution dependent. So, an important question is, is it possible to find a distribution independent property that can bound the robustness error, such that practical algorithms can be designed around them? If not, can we prove otherwise?

Appendix A

Theorems and Proofs

A.1 A note on shared randomness

Recall that from Equation 3.2.3.1 we wanted to find an expression for $\mathbb{E}[\mathbf{w}_t - \mathbf{w}'_t]$ from running SGDM on two adjacent sample sets S and S'. Let us define a random sequence $\langle\sigma\rangle = \{\sigma_0 \dots \sigma_t : \sigma_i \in \{1 \dots n\}\}$. Then SGDM corresponds to, at each step i, sampling σ_i and applying $M : S[\sigma_i] \to M_i$. Since σ_i is a uniform random variable, so

$$P(M_i = M'_i) = P(S[\sigma_i] = S'[\sigma_i]) = 1 - \frac{1}{n}$$

Then by our independence assumption and the law of total expectation we can write

$$\begin{aligned}
\mathbb{E}\left[\prod_{i=0}^{t-1} M_i - \prod_{i=0}^{t-1} M'_i\right] &= \mathbb{E}\left[\prod_{i=0}^{t-1} M_i - \prod_{i=0}^{t-1} \mathbb{E}[M'_i|M_i]\right] \\
&= \mathbb{E}\left[\prod_{i=0}^{t-1} M_i - \prod_{i=0}^{t-1} \Big(P(M_i = M'_i)M_i + P(M_i \neq M'_i)M''\Big)\right] \\
&= \mathbb{E}\left[\prod_{i=0}^{t-1} M_i - \prod_{i=0}^{t-1} \left(\frac{n-1}{n}M_i + \frac{1}{n}M''\right)\right]
\end{aligned}$$

Where M'' corresponds to the precise element in S' that differs from S. Renaming

$M'' - M_i$ as V_i we can evaluate the r.h.s. as:

$$\mathbb{E}\left[\prod_{i=0}^{t-1} M_i - \prod_{i=0}^{t-1}\left(M_i + \frac{1}{n}V_i\right)\right]$$

A.2 Huber contamination and variational distance

Definition A.2.1 (Variational distance [BDBCP07])**.** *Let $\mathcal{Z}$ be the set of measurable subsets under Z, then the L_1 or variational distance of P and D is defined as*

$$d_{L_1}(P, D) = 2 \sup_{B\in\mathcal{Z}} |p_P(B) - p_D(B)|$$

Claim A.2.1 (Contamination implies bounded variational distance)**.** *If $P = (1-\delta)D + \delta Q$, then $d_{L_1}(P, D) \leq 2\delta$.*

Proof. Let the probability density functions of P, D, Q be p_P, p_D, p_Q respectively. From the definition of variational distance,

$$\begin{aligned}
d_{L_1}(P, D) &= 2 \sup_{B\in\mathcal{Z}} |p_P(B) - p_D(B)| \\
&= 2 \sup_{B\in\mathcal{Z}} |(1-\delta)p_D(B) + \delta p_Q(B) - p_D(B)| \\
&= 2\delta \sup_{B\in\mathcal{Z}} |p_Q(B) - p_D(B)| \\
&\leq 2\delta
\end{aligned}$$

□

A.3 Kantorovich-Rubinstein duality

The primal form of the Wasserstein distance between discrete distributions D and Q is written as

$$\inf_{\gamma\in\Pi(D,Q)} \mathbb{E}_{(x,y)\sim\gamma}||x-y||$$

Where $\Pi(D, Q)$ is the set of all joint distributions $\gamma(x, y)$ whose marginal distributions are D and Q respectively. This can also written as a linear program:

$$\begin{aligned}
\text{minimize} \qquad & \sum_i \sum_j \gamma(x_i, x_j)||x_i - x_j|| \\
\text{subject to} \qquad & \sum_j \gamma(x_i, x_j) = p_D(x_i) \\
& \sum_i \gamma(x_i, x_j) = p_Q(x_j) \\
& \gamma(x_i, x_j) \geq 0 \qquad \forall i, j
\end{aligned}$$

Then the dual is:

$$\begin{aligned}
\text{maximize} \qquad & \sum_i f(x_i) p_D(x_i) + \sum_j g(x_j) p_Q(x_j) \\
\text{subject to} \qquad & f(x_i) + g(x_j) \leq ||x_i - x_j|| \\
& f(x_i) \geq 0 \qquad \forall i \\
& g(x_i) \geq 0 \qquad \forall i
\end{aligned}$$

When $x_i = x_j$, then $||x_i - x_j|| = 0$. Therefore $f(x_i) + g(x_i) \leq 0$. Furthermore, since f, g, p_D, p_Q are positive, so the objective function is better when $f(x_i) + g(x_i) = 0$ vs when $f(x_i) + g(x_i) < 0$. So we can consider the equality case, and so $f = -g$ everywhere on the domain. Then our optimization problem becomes the maximization of

$$\sum_i f(x_i) p_D(x_i) - \sum_j f(x_j) p_Q(x_j) = \mathbb{E}_{z \sim D} f(z) - \mathbb{E}_{z \sim Q} f(z)$$

Subject to $f(x_i) - f(x_j) \leq ||x_i - x_j||$ i.e. f is 1-Lipschitz. This gives us the dual form we saw in Section 4.4.2. An analog exists for continuous distributions D and Q with essentially the same result [Vil09].

Bibliography

[ACB17] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. In *arXiv:1701.07875v2*, 2017.

[BB07] Léon Bottou and Olivier Bousquet. The tradeoffs of large scale learning. *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 161–168, 2007.

[BCN16] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. In *arXiv:1606.04838*, 2016.

[BDBCP07] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptaion. *Advances in Neural Information Processing Systems*, 19, 2007.

[BE02] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[Ber99] Dimitri P. Bertsekas. *Nonlinear Programming.* Athena Scientific, second edition, 1999.

[Ber15] Dimitri P. Bertsekas. *Convex Optimization Algorithms.* Athena Scientific, 2015.

[CS04] Andreas Christmann and Ingo Steinwart. On robustness properties of convex risk minimization methods for pattern recognition. *Journal of Machine Learning Research*, 5:1007–1034, 2004.

[HRS16] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *arXiv:1509.01240*, 2016.

[Hub64] Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.

[Hub81] Peter J. Huber. *Robust Statistics.* John Wiley and Sons, Inc, (2009 reprint) 2nd edition, 1981.

[Jun09] Raphaël Jungers. *The Joint Spectral Radius, Theory and Applications.* Springer, 2009.

[LCR16] Junhong Lin, Raffaello Camoriano, and Lorenzo Rosasco. Generalization properties and implicit regularization for multiple passes sgm. In *arXiv:1605.08375*, 2016.

[MMR09] Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *Proceedings of The 22nd Annual Conference on Learning Theory*, 2009.

[MNPR06] Sayan Mukherjee, Partha Niyogi, Tomaso Poggio, and Ryan Rifkin. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Advances in Computational Mathematics*, 25:161–193, 2006.

[Nes98] Yurii Nesterov. *Introductory Lectures on Convex Programming*, volume 1: Basic Course. 1998.

[Pol64] Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4:1–17, 1964.

[RS60] Gian-Carlo Rota and Gilbert Strang. A note on the joint spectral radius. *Proceedings of the Netherlands Academy*, 22:379–381, 1960.

[SSSSS10] Shai Shalev-Shwartz, Ohad Shamir, Nathan Srebro, and Karthik Sridharan. Learnability, stability and uniform convergence. *Journal of Machine Learning Research*, 11:2635–2670, 2010.

[Vap98] Vladimir N. Vapnik. *Statistical Learning Theory.* John Wiley and Sons, Inc, 1998.

[VC71] Vladimir N. Vapnik and Alexey Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264âĂŞ280, 1971.

[Vil09] Cédric Villani. *Optimal Transport: Old and New.* Springer, 2009.

[XM10] Huan Xu and Shie Mannor. Robustness and generalization. In *arXiv:1005.2243*, 2010.

[ZBH$^+$17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *5th International Conference on Learning Representations*, 2017.