Train faster, generalize better: Stability of stochastic gradient descent

Moritz Hardt MRTZ@GOOGLE.COM
Benjamin Recht BRECHT@BERKELEY.EDU
Yoram Singer SINGER@GOOGLE.COM

Abstract

We show that parametric models trained by a stochastic gradient method (SGM) with few iterations have vanishing generalization error. We prove our results by arguing that SGM is algorithmically stable in the sense of Bousquet and Elisseeff. Our analysis only employs elementary tools from convex and continuous optimization. We derive stability bounds for both convex and non-convex optimization under standard Lipschitz and smoothness assumptions.

Applying our results to the convex case, we provide new insights for why multiple epochs of stochastic gradient methods generalize well in practice. In the non-convex case, we give a new interpretation of common practices in neural networks, and formally show that popular techniques for training large deep models are indeed stability-promoting. Our findings conceptually underscore the importance of reducing training time beyond its obvious benefit.

1. Introduction

The most widely used optimization method in machine learning practice is *stochastic gradient method* (SGM). Stochastic gradient methods aim to minimize the empirical risk of a model by repeatedly computing the gradient of a loss function on a single training example, or a batch of few examples, and updating the model parameters accordingly. SGM is scalable, robust, and performs well across many different domains ranging from smooth and strongly convex problems to complex non-convex objectives.

In a nutshell, our results establish that: *Any model trained with stochastic gradient method in a reasonable amount of time attains small generalization error.*

As training time is inevitably limited in practice, our results help to explain the strong generalization performance of stochastic gradient methods observed in practice. More concretely, we bound the generalization error of a model in terms of the number of iterations that stochastic gradient method took in order to train the model. Our main analysis tool is to employ the notion of algorithmic stability due to Bousquet and Elisseeff (2002). We demonstrate that the stochastic gradient method is stable provided that the objective is relatively smooth and the number of steps taken is sufficiently small.

It is common in practice to perform a linear number of steps in the size of the sample and to access each data point multiple times. Our results show in a broad range of settings that, provided the number of iterations is linear in the number of data points, the generalization error is bounded by a vanishing function of the sample size. The results hold true even for complex models with large number of parameters and no explicit regularization term in the objective. Namely, fast training time by itself is sufficient to prevent overfitting.

Our bounds are algorithm specific: Since the number of iterations we allow can be larger than the sample size, an arbitrary algorithm could easily achieve small training error by memorizing all training data with no generalization ability whatsoever. In contrast, if the stochastic gradient method manages to fit the training data in a reasonable number of iterations, it is guaranteed to generalize.

Conceptually, we show that minimizing training time is not only beneficial for obvious computational advantages, but also has the important byproduct of decreasing generalization error. Consequently, it may make sense for practitioners to focus on minimizing training time, for instance, by designing model architectures for which stochastic gradient method converges fastest to a desired error level.

1.1. Our contributions

Our focus is on generating *generalization bounds* for models learned with stochastic gradient descent. Recall that the generalization bound is the expected difference between the error a model incurs on a training set versus the error incurred on a new data point, sampled from the same

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

distribution that generated the training data. Throughout, we assume we are training models using n sampled data points.

Our results build on a fundamental connection between the generalization error of an algorithm and its *stability* properties. Roughly speaking, an algorithm is *stable* if the training error it achieves varies only slightly if we change any single training data point. The precise notion of stability we use is known as *uniform stability* due to (Bousquet & Elisseeff, 2002). It states that a randomized algorithm A is uniformly stable if for all data sets differing in only one element, the learned models produce nearly the same predictions. We review this method in Section 2, and provide a new adaptation of this theory to *iterative* algorithms.

In Section 3, we show that stochastic gradient is uniformly stable, and our techniques mimic its convergence proofs. For convex loss functions, we prove that the stability measure decreases as a function of the sum of the step sizes. For strongly convex loss functions, we show that stochastic gradient is stable, even if we train for an arbitrarily long time. We can combine our bounds on the generalization error of stochastic gradient method with optimization bounds quantifying the convergence of the empirical loss achieved by SGM. In Section 5, we show that models trained for multiple epochs match classic bounds for stochastic gradient (Nemirovski & Yudin, 1978; 1983).

More surprisingly, our results carry over to the case where the loss-function is non-convex. In this case we show that the method generalizes provided the steps are sufficiently small and the number of iterations is not too large. More specifically, we show the number of steps of stochastic gradient can grow as n^c for a small $c > 1$. This provides some explanation as to why neural networks can be trained for multiple epochs of stochastic gradient and still exhibit excellent generalization. In Section 4, we furthermore show that various heuristics used in practice, especially in the deep learning community, help to increase the stability of stochastic gradient method. For example, the popular dropout scheme (Krizhevsky et al., 2012; Srivastava et al., 2014) improves all of our bounds. Similarly, ℓ_2-regularization improves the exponent of n in our non-convex result. In fact, we can drive the exponent arbitrarily close to $1/2$ while preserving the non-convexity of the problem.

1.2. Related work

There is a venerable line of work on stability and generalization dating back more than thirty years (Devroye & Wagner, 1979; Kearns & Ron, 1999; Bousquet & Elisseeff, 2002; Mukherjee et al., 2006; Shalev-Shwartz et al., 2010). The landmark work by Bousquet and Elisseeff (Bousquet & Elisseeff, 2002) introduced the notion of *uniform stability* that we rely on. They showed that several important classification techniques are uniformly stable. In particular, under certain regularity assumptions, it was shown that the *optimizer* of a regularized empirical loss minimization problem is uniformly stable. Previous work generally applies only to the exact minimizer of specific optimization problems. It is not immediately evident on how to compute a generalization bound for an approximate minimizer such as one found by using stochastic gradient. Subsequent work studied stability bounds for randomized algorithms but focused on random perturbations of the cost function, such as those induced by bootstrapping or bagging (Elisseeff et al., 2005). This manuscript differs from this foundational work in that it derives stability bounds about the learning *procedure*, analyzing algorithmic properties that induce stability.

Classic results by Nemirovski and Yudin show that the stochastic gradient method produces is nearly optimal for empirical risk minimization of convex loss functions (Nemirovski & Yudin, 1978; 1983; Nemirovski et al., 2009; Frostig et al., 2015). These results have been extended by many machine learning researchers, yielding tighter bounds and probabilistic guarantees (Hazan et al., 2006; Hazan & Kale, 2014; Rakhlin et al., 2012). However, there is an important limitation of all of this prior art. The derived generalization bounds only hold for single passes over the data. That is, in order for the bounds to be valid, each training example must be used no more than once in a stochastic gradient update. In practice, of course, one tends to run multiple *epochs* of the stochastic gradient method. Our results resolve this issue by combining stability with optimization error. We use the foundational results to estimate the error on the *empirical risk* and then use stability to derive a deviation from the true risk. This enables us to study the risk incurred by multiple epochs and provide simple analyses of regularization methods for convex stochastic gradient. We compare our results to this related work in Section 5. We note that Rosasco and Villa obtain risk bounds for least squares minimization with an incremental gradient method in terms of the number of epochs (Rosasco & Villa, 2014). These bounds are akin to our study in Section 5, although our results are incomparable due to various different assumptions.

Finally, we note that in the non-convex case, the stochastic gradient method is remarkably successful for training large neural networks (Bottou, 1998; Krizhevsky et al., 2012). However, our theoretical understanding of this method is limited. Several authors have shown that the stochastic gradient method finds a stationary point of nonconvex cost functions (Kushner & Yin, 2003; Ghadimi & Lan, 2013). Beyond asymptotic convergence to stationary points, little is known about finding models with low training or generalization error in the nonconvex case. There have recently

been several important studies investigating optimal training of neural nets. For example Livni *et al.* show that networks with polynomial activations can be learned in a greedy fashion (Livni et al., 2014). Janzamin *et al.* (Janzamin et al., 2015) show that two layer neural networks can be learned using tensor methods. Arora *et al.* (Arora et al., 2015) show that two-layer sparse coding dictionaries can be learned via stochastic gradient. Our work complements these developments: rather than providing new insights into mechanisms that yield low training error, we provide insights into mechanisms that yield low generalization error. If one can achieve low training error quickly on a nonconvex problem with stochastic gradient, our results guarantee that the resulting model generalizes well.

2. Stability of randomized iterative algorithms

Consider the following general setting of supervised learning. There is an unknown distribution $\mathcal{D}$ over examples from some space Z. We receive a sample $S = (z_1, \ldots, z_n)$ of n examples drawn i.i.d. from $\mathcal{D}$. Our goal is to find a model w with small *population risk*, defined as: $R[w] \stackrel{\text{def}}{=} \mathbb{E}_{z\sim\mathcal{D}}\, f(w; z)$. Here, where f is a *loss function* and $f(w; z)$ designates the *loss* of the model described by w encountered on example z.

Since we cannot measure the objective $R[w]$ directly, we instead use a sample-averaged proxy, the *empirical risk*, defined as $R_S[w] \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f(w; z_i)$,

The *generalization error* of a model w is the difference

$$R_S[w] - R[w]. \tag{2.1}$$

When $w = A(S)$ is chosen as a function of the data by a potentially randomized algorithm A it makes sense to consider the expected generalization error

$$\epsilon_{\text{gen}} \stackrel{\text{def}}{=} \mathbb{E}_{S,A}[R_S[A(S)] - R[A(S)]]\,, \tag{2.2}$$

where the expectation is over the randomness of A and the sample S.

In order to bound the generalization error of an algorithm, we employ the following notion of *uniform stability* in which we allow randomized algorithms as well.

Definition 2.1. *A randomized algorithm A is ϵ-*uniformly stable *if for all data sets $S, S' \in Z^n$ such that S and S' differ in at most one example, we have*

$$\sup_z \mathbb{E}_A\left[f(A(S); z) - f(A(S'); z)\right] \leq \epsilon\,. \tag{2.3}$$

Here, the expectation is taken only over the internal randomness of A. We will denote by $\epsilon_{\text{stab}}(A, n)$ the infimum over all ϵ for which (2.3) *holds. We will omit the tuple (A, n) when it is clear from the context.*

We recall the important theorem that uniform stability implies *generalization in expectation*. The proof is based on an argument in Lemma 7 of (Bousquet & Elisseeff, 2002) and very similar to Lemma 11 in (Shalev-Shwartz et al., 2010).

Theorem 2.2. *Let A be ϵ-uniformly stable. Then,* $\left|\mathbb{E}_{S,A}\left[R_S[A(S)] - R[A(S)]\right]\right| \leq \epsilon$.

Theorem 2.2 proves that if an algorithm is uniformly stable, then its generalization error is small. We now turn to some properties of iterative algorithms that control their uniform stability.

2.1. Properties of update rules

We consider general update rules of the form $G\colon \Omega \to \Omega$ which map a point $w \in \Omega$ in the parameter space to another point $G(w)$. The most common update is the gradient update rule $G(w) = w - \alpha\nabla f(w)$, where $\alpha \geq 0$ is a step size and $f\colon \Omega \to \mathbb{R}$ is a function that we want to optimize.

The canonical update rule we will consider in this manuscript is an incremental gradient update, where $G(w) = w - \alpha\nabla f(w)$ for some convex function f. We will return to a detailed discussion of this specific update in the sequel, but the reader should keep this particular example in mind throughout the remainder of this section.

The following two definitions provide the foundation of our analysis of how two different sequences of update rules diverge when iterated from the same starting point. These definitions will ultimately be useful when analyzing the stability of stochastic gradient descent.

Definition 2.3. *An update rule is η-expansive if for all $v, w \in \Omega$, $\|G(v) - G(w)\| \leq \eta\|v - w\|$. It is σ-*bounded *if $\|w - G(w)\| \leq \sigma$.*

With these two properties, we can establish the following lemma of how a sequence of updates to a model diverge when the training set is perturbed.

Lemma 2.4 (Growth recursion)**.** *Fix an arbitrary sequence of updates $G_1, \ldots, G_T$ and another sequence $G'_1, \ldots, G'_T$. Let $w_0 = w'_0$ be a starting point in Ω and define $\delta_t = \|w'_t - w_t\|$ where w_t, w'_t are defined recursively through*

$$w_{t+1} = G_t(w_t) \qquad w'_{t+1} = G'_t(w'_t)\,. \qquad (t > 0)$$

Then, we have the recurrence relation $\delta_0 = 0$,

$$\delta_{t+1} \leq \begin{cases} \eta\delta_t & G_t = G'_t \text{ is } \eta\text{-expansive} \\ \min(\eta, 1)\delta_t + 2\sigma_t & G_t \text{ and } G'_t \text{ are } \sigma\text{-bounded,} \\ & G_t \text{ is } \eta \text{ expansive} \end{cases}$$

3. Stability of Stochastic Gradient Method

Given n labeled examples $S = (z_1, \ldots, z_n)$ where $z_i \in Z$, consider a *decomposable* objective function $f(w) =$

$\frac{1}{n}\sum_{i=1}^{n} f(w; z_i)$, where $f(w; z_i)$ denotes the *loss* of w on the example z_i. The stochastic gradient update for this problem with *learning rate* $\alpha_t > 0$ is given by $w_{t+1} = w_t - \alpha_t \nabla_w f(w_t; z_{i_t})$. Stochastic gradient method (SGM) is the algorithm resulting from performing stochastic gradient updates T times where the indices i_t are randomly chosen. There are two popular schemes for choosing the examples' indices. One is to pick i_t uniformly at random in $\{1, \ldots, n\}$ at each step. The other is to choose a random permutation over $\{1, \ldots, n\}$ and cycle through the examples repeatedly in the order determined by the permutation. Our results hold for both variants.

In parallel with the previous section the stochastic gradient method is akin to applying the *gradient update rule* defined as follows.

Definition 3.1. *For a nonnegative step size $\alpha \geq 0$ and a function $f\colon \Omega \to \mathbb{R}$, we define the* gradient update rule $G_{f,\alpha}$ *as* $G_{f,\alpha}(w) = w - \alpha \nabla f(w)$.

3.1. Proof idea: Stability of stochastic gradient method

In order to prove that the stochastic gradient method is stable, we will analyze the output of the algorithm on two data sets that differ in precisely one location. Note that if the loss function is L-Lipschitz for every example z, we have $\mathbb{E}\,|f(w; z) - f(w'; z)| \leq L\,\mathbb{E}\,\|w - w'\|$ for all w and w'. Hence, it suffices to analyze how w_t and w'_t diverge in the domain as a function of time t. Recalling that w_t is obtained from w_{t-1} via a gradient update, our goal is to bound $\delta_t = \|w_t - w'_t\|$ recursively and in expectation as a function of δ_{t-1}.

There are two cases to consider. In the first case, SGM selects the index of an example at step t on which is identical in S and S'. Unfortunately, it could still be the case that δ_t grows, since w_t and w'_t differ and so the gradients at these two points may still differ. Below, we will show how to control δ_t in terms of the convexity and smoothness properties of the stochastic gradients.

The second case to consider is when SGM selects the one example to update in which S and S' differ. Note that this happens only with probability $1/n$ if examples are selected randomly. In this case, we simply bound the increase in δ_t by the norm of the two gradient $\nabla f(w_{t-1}; z)$ and $\nabla f(w'_{t-1}; z')$. The sum of the norms is bounded by $2\alpha_t L$ and we obtain $\delta_t \leq \delta_t + 2\alpha_t L$. Combining the two cases, we can then solve a simple recurrence relation to obtain a bound on δ_T.

This simple approach suffices to obtain the desired result in the convex case, but there are additional difficulties in the non-convex case. Here, we need to use an intriguing stability property of stochastic gradient method. Specifically, the first time step t_0 at which SGM even encounters the example in which S and S' differ is a random variable in $\{1, \ldots, n\}$ which tends to be relatively large. Specifically, for any $m \in \{1, \ldots, n\}$, the probability that $t_0 \leq m$ is upper bounded by m/n. This allows us to argue that SGM has a long "burn-in period" where δ_t does not grow at all. Once δ_t begins to grow, the step size has already decayed allowing us to obtain a non-trivial bound.

We now turn to making this argument precise.

3.2. Expansion properties of stochastic gradients

Let us now record some of the core properties of the stochastic gradient update. The gradient update rule is bounded provided that the function f satisfies the following common Lipschitz condition.

Definition 3.2. *We say that f is L-*Lipschitz *if for all points u in the domain of f we have $\|\nabla f(x)\| \leq L$. This implies that $|f(u) - f(v)| \leq L\|u - v\|$.*

Lemma 3.3. *Assume that f is L-Lipschitz. Then, the gradient update $G_{f,\alpha}$ is (αL)-bounded.*

We now turn to expansiveness. As we will see shortly, different expansion properties are achieved for non-convex, convex, and strongly convex functions.

Definition 3.4. *A function $f\colon \Omega \to \mathbb{R}$ is γ-*strongly convex *if for all $u, v \in \Omega$ we have $f(u) \geq f(v) + \langle \nabla f(v), u - v\rangle + \frac{\gamma}{2}\|u - v\|^2$.*

We say f is *convex* if it is 0-strongly convex. The following standard notion of smoothness leads to a bound on how expansive the gradient update is.

Definition 3.5. *A function $f\colon \Omega \to \mathbb{R}$ is β-*smooth *if for all for all $u, v \in \Omega$ we have $\|\nabla f(u) - \nabla f(v)\| \leq \beta\|u - v\|$.*

In general, smoothness will imply that the gradient updates cannot be overly expansive. When the function is also convex and the step size is sufficiently small the gradient update becomes non-expansive. When the function is additionally strongly convex, the gradient update becomes *contractive* in the sense that η will be less than one and u and v will actually shrink closer to one another. The majority of the following results can be found in several textbooks and monographs. Notable references are Polyak (Polyak, 1987) and Nesterov (Nesterov, 2004). We include proofs in the appendix for completeness.

Lemma 3.6. *Assume that f is β-smooth. Then, $G_{f,\alpha}$ is $(1 + \alpha\beta)$-expansive. If f is in addition convex, then for any $\alpha \leq 2/\beta$, the update $G_{f,\alpha}$ is 1-expansive. If f is in addition γ-strongly convex, then for $\alpha \leq \frac{2}{\beta+\gamma}$, $G_{f,\alpha}$ is $\left(1 - \frac{\alpha\beta\gamma}{\beta+\gamma}\right)$-expansive.*

Henceforth we will no longer mention which random selection rule we use as the proofs are almost identical for both rules.

3.3. Convex optimization

We begin with a simple stability bound for convex loss minimization via stochastic gradient method.

Theorem 3.7. *Assume that the loss function $f(\cdot\,; z)$ is β-smooth, convex and L-Lipschitz for every z. Suppose that we run SGM with step sizes $\alpha_t \leq 2/\beta$ for T steps. Then, SGM satisfies uniform stability with $\epsilon_{\mathrm{stab}} \leq \frac{2L^2}{n}\sum_{t=1}^{T}\alpha_t$.*

Proof. Let S and S' be two samples of size n differing in only a single example. Consider the gradient updates $G_1, \ldots, G_T$ and $G'_1, \ldots, G'_T$ induced by running SGM on sample S and S', respectively. Let w_T and w'_T denote the corresponding outputs of SGM.

We now fix an example $z \in Z$ and apply the Lipschitz condition on $f(\cdot\,; z)$ to get

$$\mathbb{E}\,|f(w_T; z) - f(w'_T; z)| \leq L\,\mathbb{E}\,[\delta_T]\ , \qquad (3.1)$$

where $\delta_T = \|w_T - w'_T\|$. Observe that at step t, with probability $1 - 1/n$, the example selected by SGM is the same in both S and S'. In this case we have that $G_t = G'_t$ and we can use the 1-expansivity of the update rule G_t which follows from Lemma 3.6 using the fact that the objective function is convex and that $\alpha_t \leq 2/\beta$. With probability $1/n$ the selected example is different in which case we use that both G_t and G'_t are $\alpha_t L$-bounded as a consequence of Lemma 3.3. Hence, we can apply Lemma 2.4 and linearity of expectation to conclude that for every t, $\mathbb{E}\,[\delta_{t+1}] \leq \left(1 - \frac{1}{n}\right)\mathbb{E}\,[\delta_t] + \frac{1}{n}\,\mathbb{E}\,[\delta_t] + \frac{2\alpha_t L}{n} = \mathbb{E}\,[\delta_t] + \frac{2L\alpha_t}{n}$. Unraveling the recursion gives $\mathbb{E}\,[\delta_T] \leq \frac{2L}{n}\sum_{t=1}^{T}\alpha_t$. Plugging this back into equation (3.1), gives the desired result. □

We refer the reader to the full version (**?**) for our results on strongly convex optimization.

3.4. Non-convex optimization

In this section we prove stability results for stochastic gradient methods that do not require convexity. We will still assume that the objective function is smooth and Lipschitz as defined previously.

The crux of the proof is to observe that SGM typically makes several steps before it even encounters the one example on which two data sets in the stability analysis differ.

Theorem 3.8. *Assume that $f(\cdot; z) \in [0, 1]$ is an L-Lipschitz and β-smooth loss function for every z. Suppose that we run SGM for T steps with monotonically non-increasing step sizes $\alpha_t \leq c/t$. Then, SGM has uniform stability with*

$$\epsilon_{\mathrm{stab}} \leq \frac{1 + 1/\beta c}{n - 1}(2cL^2)^{\frac{1}{\beta c+1}}T^{\frac{\beta c}{\beta c+1}}$$

In particular, omitting constant factors that depend on β, c, and L, we get $\epsilon_{\mathrm{stab}} \lessapprox \frac{T^{1-1/(\beta c+1)}}{n}$.

4. Stability-inducing operations

In light of our results, it makes sense to analyse for operations that increase the stability of the stochastic gradient method. We show in this section that pleasingly several popular heuristics and methods indeed improve the stability of SGM. Our rather straightforward analyses both strengthen the bounds we previously obtained and help to provide an explanation for the empirical success of these methods.

Weight Decay and Regularization. Weight decay is a simple and effective method that often improves generalization (Krogh & Hertz, 1992).

Definition 4.1. *Let $f\colon \Omega \to \Omega$, be a differentiable function. We define the* gradient update with weight decay at rate μ *as $G_{f,\mu,\alpha}(w) = (1 - \alpha\mu)w - \alpha\nabla f(w)$.*

It is easy to verify that the above update rule is equivalent to performing a gradient update on the *ℓ_2-regularized* objective $g(w) = f(w) + \frac{\mu}{2}\|w\|^2$.

Lemma 4.2. *Assume that f is β-smooth. Then, $G_{f,\mu,\alpha}$ is $(1 + \alpha(\beta - \mu))$-expansive.*

The above lemma shows as that a regularization parameter μ counters a smoothness parameter β. Once $r > \beta$, the gradient update with decay becomes contractive. Any theorem we proved in previous sections that has a dependence on β leads to a corresponding theorem for stochastic gradient with weight decay in which β is replaced with $\beta - \mu$.

Gradient Clipping. It is common when training deep neural networks to enforce bounds on the norm of the gradients encountered by SGD. This is often done by either truncation, scaling, or dropping of examples that cause an exceptionally large value of the gradient norm.

Consider for example gradient clipping. In this procedure, a stochastic gradient $\nabla f(w; z)$ is replaced with $\nabla_c f(w; z) = \mathrm{clip}(\nabla f(w; z))$ where $\mathrm{clip}(x) = x$ if $\|x\| \leq B$ and $B/\|x\|$ otherwise.

Lemma 4.3. *If f is β-smooth, then $\|\nabla_c f(w; z)\| \leq B$ and $\|\nabla_c f(v; z) - \nabla_c f(w; z)\| \leq \beta\|v - w\|$.*

Similar arguments would apply to the different variants of gradient warping. Any such heuristic where the warping operation is Lipschitz directly leads to a bound on the Lipschitz parameter L that appears in our bounds. It is also easy to introduce a varying Lipschitz parameter L_t to account for possibly different values.

Dropout. Dropout (Srivastava et al., 2014) is a popular and effective heuristic for preventing large neural networks from overfitting. Here we prove that, indeed, dropout improves all of our stability bounds generically.

At each iteration of dropout SGD, we sample a random diagonal matrix P with precisely s diagonal entries equal to 1 and the rest equal to 0. Instead of taking a stochastic gradient step $\nabla f(w;z)$ we instead update with the perturbed gradient $\nabla_d f(w;z) := P\nabla f(Pw;z)$. The dropout update is both more bounded *and* smoother than the standard stochastic gradient update.

Lemma 4.4. *Assume that f is L-Lipschitz and β-smooth. Then, the dropout gradient $\nabla_d f(w;z)$ with dropout rate s has norm bounded by L and satisfies $\mathbb{E}_P \|\nabla_d f(v;z) - \nabla_d f(w;z)\| \leq (\sqrt{s/d})\beta\|v-w\|$.*

5. Convex risk minimization

We now outline how our generalization bounds lead to bounds on the population risk achieved by SGM in the convex setting. We restrict our attention to the convex case where we can contrast against known results. The main feature of our results is that we show that one can achieve bounds comparable or perhaps better than known results on stochastic gradient for risk minimization by running for multiple passes over the data set.

The key to the analysis in this section is to decompose the risk estimates into an *optimization error* term and a stability term. The optimization error designates how closely we optimize the empirical risk or a proxy of the empirical risk. By optimizing with stochastic gradient, we will be able to balance this optimization accuracy against how well we generalize. These results are inspired by the work of Bousquet and Bottou who provided similar analyses for SGM based on uniform convergence (Bottou & Bousquet, 2008). However, our stability results will yield sharper bounds.

Throughout this section, our risk decomposition works as follows. We define the *optimization error* to be the gap between the empirical risk and minimum empirical risk in expectation: $\epsilon_{\mathrm{opt}}(w) \stackrel{\mathrm{def}}{=} \mathbb{E}\left[R_S[w] - R_S[w_\star^S]\right]$ where $w_\star^S = \arg\min_w R_S[w]$. By Theorem 2.2, the expected risk of a w output by SGM is bounded as $\mathbb{E}[R[w]] \leq \mathbb{E}[R_S[w]] + \epsilon_{\mathrm{stab}} \leq \mathbb{E}[R_S[w_\star^S]] + \epsilon_{\mathrm{opt}}(w) + \epsilon_{\mathrm{stab}}$. In general, the optimization error decreases with the number of SGM iterations while the stability increases. Balancing these two terms will thus provide a reasonable excess risk against the empirical risk minimizer. Note that our analysis involves the expected minimum empirical risk which could be considerably smaller than the minimum risk. However, as we now show, it can never be larger.

Lemma 5.1. *Let $w_\star$ denote the minimizer of the population risk and $w_\star^S$ denote the minimizer of the empirical risk given a sampled data set S. Then $\mathbb{E}[R_S[w_\star^S]] \leq R[w_\star]$.*

To analyze the optimization error, we will make use of a classical result due to Nemirovski and Yudin (Nemirovski & Yudin, 1983).

Theorem 5.2. *Assume we run stochastic gradient descent with constant stepsize α on a convex function $R[w] = \mathbb{E}_z[f(w;z)]$. Assume further that $\|\nabla f(w;z)\| \leq L$ and $\|w_0 - w_\star\| \leq D$ for some minimizer $w_\star$ of R. Let $\bar{w}_T$ denote the average of the T iterates of the algorithm. Then we have $R[\bar{w}_T] \leq R[w_\star] + \frac{1}{2}\frac{D^2}{T\alpha} + \frac{1}{2}L^2\alpha$.*

The upper bound stated in the previous theorem is known to be tight even if the function is β-smooth (Nemirovski & Yudin, 1983).

Theorem 5.2 directly provides a generalization bound for SGM that holds when we make a single pass over the data. The theorem requires fresh samples from the distribution in each update step of SGM. Hence, given n data points, we cannot make more than n steps, and each sample must not be used more than once.

Corollary 5.3. *Let f be a convex loss function satisfying $\|\nabla f(w,z)\| \leq L$ and let $w_\star$ be a minimizer of the population risk $R[w] = \mathbb{E}_z f(w;z)$. Suppose we make a single pass of SGM over the sample $S = (z_1, \ldots, z_n)$ with fixed step size $\alpha = \frac{D}{L\sqrt{n}}$ starting from a point w_0 that satisfies $\|w_0 - w_\star\| \leq D$. Then, the average $\bar{w}_n$ of the iterates satisfies $\mathbb{E}[R[\bar{w}_n]] \leq R[w_\star] + \frac{DL}{\sqrt{n}}$.*

We now contrast this bound with what follows from our results.

Proposition 5.4. *Let $S = (z_1, \ldots, z_n)$ be a sample of size n. Let f be a β-smooth convex loss function satisfying $\|\nabla f(w,z)\| \leq L$ and let $w_\star^S$ be a minimizer of the empirical risk $R_S[w] = \frac{1}{n}\sum_{i=1}^n f(w;z_i)$. Suppose we run T steps of SGM with step size*

$$\alpha = \frac{D}{L\sqrt{n}}\left(\frac{n}{T}\cdot\sqrt{\frac{T}{n+2T}}\right)$$

from a starting point w_0 that satisfies $\|w_0 - w_\star^S\| \leq D$. Then, the average $\bar{w}_T$ over the iterates satisfies $\mathbb{E}[R[\bar{w}_T]] \leq \mathbb{E}[R_S[w_\star^S]] + \frac{DL}{\sqrt{n}}\sqrt{\frac{n+2T}{T}}$.

Note that the bound from our stability analysis is not directly comparable to Corollary 5.3 as we are comparing against the expected minimum empirical risk rather than the minimum risk. Lemma 5.1 implies that the excess risk in our bound is at most worse by a factor of $\sqrt{3}$ compared with Corollary 5.3 when $T = n$. Moreover, the excess risk in our bound tends to a factor merely $\sqrt{2}$ larger than the Nemirovski-Yudin bound as T goes to infinity. In contrast, the classical bound does not apply when $T > n$.

6. Experimental Evaluation

The goal of our experiments is to isolate the effect of training time, measured in number of steps, on the stability of SGM. We evaluated broadly a variety of neural network architectures and varying step sizes on a number of different datasets.

To measure algorithmic stability we consider two proxies. The first is the Euclidean distance between the parameters of two identical models trained on the datasets which differ by a single example. In all of our proofs, we use slow growth of this *parameter distance* as a way to prove stability. Note that it is not necessary for this parameter distance to grow slowly in order for our models to be algorithmically stable. This is a strictly stronger notion. Our second weaker proxy is to measure the generalization error directly in terms of the absolute different between the test error and training error of the model.

We analyzed four standard machine learning datasets each with their own corresponding deep architecture. We studied the LeNet architecture for MNIST, the cuda-convnet architecture for CIFAR-10, the AlexNet model for ImageNet, and the LSTM model for the Penn Treebank Language Model (PTB). Full details of our architectures and training procedures can be found below.

In all cases, we ran the following experiment. We choose a random example from the training set and remove it. The remaining examples constitute our set S. Then we create a set S' by replacing a random element of S with the element we deleted. We train stochastic gradient descent with the same random seed on datasets S and S'. We record the Euclidean distance between the individual layers in the neural network after every 100 SGM updates. We also record the training and testing errors once per epoch.

Our experiments show four primary findings:

Step size dependence. Doubling the step size no more than doubles the generalization error (see Figure 1). This behavior is fairly consistent for both generalization error defined with respect to classification accuracy and cross entropy (the loss function used for training). It thus suggests that there is at most a linear dependence on the step size in the generalization error.

Parameter distance. We evaluate the *normalized Euclidean distance* $\sqrt{\|w - w'\|^2/(\|w\|^2 + \|w'\|^2)}$ between the parameters w and w' of two models trained on two copies of the data differing in a random substitution. We observe that the parameter distance grows sub-linearly even in cases where our theory currently uses an exponential bound. This shows that our bounds are pessimistic.

Parameter distance verus generalization. There is a close correspondence between the parameter distance and generalization error. *A priori*, it could have been the case that the generalization error is small even though the parameter distance is large. Our experiments show that these two quantities often move in tandem and seem to be closely related.

Late substitution. When measuring parameter distance it is indeed important that SGM does not immediately encounter the random substitution, but only after some progress in training has occurred. If we artificially place the corrupted data point at the first step of SGM, the parameter distance can grow significantly faster subsequently. This effect is most pronounced in the ImageNet experiments, as displayed in Figure 2.

Experiments. We evaluated convolutional neural networks for image classification on three datasets: MNIST, Cifar10 and ImageNet. Starting with Cifar10, we chose a standard model consisting of three convolutional layers each followed by a pooling operation. This model roughly corresponds to that proposed by Krizhevsky *et al.* (Krizhevsky et al., 2012) and available in the "cuda-convnet" code[1]. However, to make the experiments more interpretable, we avoid all forms of regularization such as weight decay or dropout. The learning rate was fixed at 0.01. We also do not employ data augmentation even though this would greatly improve the ultimate test accuracy of the model. Additionally, we use only constant step sizes in our experiments. With these restrictions the model we use converges to below 20% test error. While this is not state of the art on Cifar10, our goal is not to optimize test accuracy but rather a simple, interpretable experimental setup.

The situation on MNIST is largely analogous to what we saw on Cifar10. We trained a LeNet inspired model with two convolutional layers and one fully-connected layer. The first and second convolutional layers have 20 and 50 hidden units respectively. This model is much smaller and converges significantly faster than the Cifar10 models, typically achieving best test error in five epochs. We trained with minibatch size 60. As a result, the amount of overfitting is smaller. In the case of MNIST, we also repeated our experiments after replacing the usual cross entropy objective with a squared loss objective. The results are in the full version on arxiv. It turned out that this does not harm convergence at all, while leading to somewhat smaller generalization error and parameter divergence.

On ImageNet, we trained the standard AlexNet architecture (Krizhevsky et al., 2012) using data augmentation, regularization, and dropout. Unlike in the case of Cifar10, we were unable to find a setting of hyperparameters that yielded reasonable performance without using these tech-

[1] https://code.google.com/archive/p/cuda-convnet

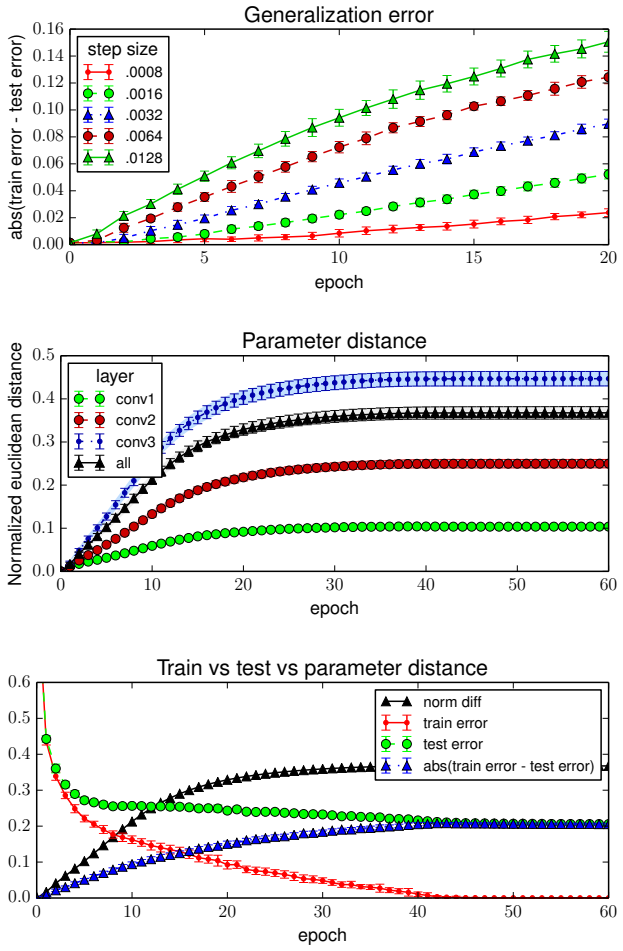

Figure 1. Experimental results on Cifar. Top: Generalization error. Middle: Parameter divergence. Bottom: Comparison of train, test, and generalization eror with parameter divergence.

niques. However, for Figure 2 (bottom), we did not use data-augmentation to exaggerate the effects of overfitting and demonstrate the impact scaling the model-size. This figure demonstrates that the model-size appears to be a second-order effect with regards to generalization error, and step-size has a considerably stronger impact.

6.1. Recurrent neural networks with LSTM

We also examined the stability of recurrent neural networks. Specifically, we looked at an LSTM architecture for language modeling (Zaremba et al., 2014). We focused on word-level prediction experiments using the Penn Tree Bank (PTB) (Marcus et al., 1993), consisting of 929,000 training words, 73,000 validation words, and 82,000 test words. PTB has 10,000 words in its vocabulary[2]. Following Zaremba *et al.*, we trained regularized LSTMs with two layers that were unrolled for 20 steps. We initialize the hidden states to zero. We trained with minibatch size 20. The LSTM has 200 units per layer and its parameters are initialized to have mean zero and standard deviation of 0.1. We did not use dropout to enhance reproducibility. Dropout would only increase the stability of our models. The results are displayed in Figure 3.

[2]Data source: `http://www.fit.vutbr.cz/~imikolov/rnnlm/simple-examples.tgz`

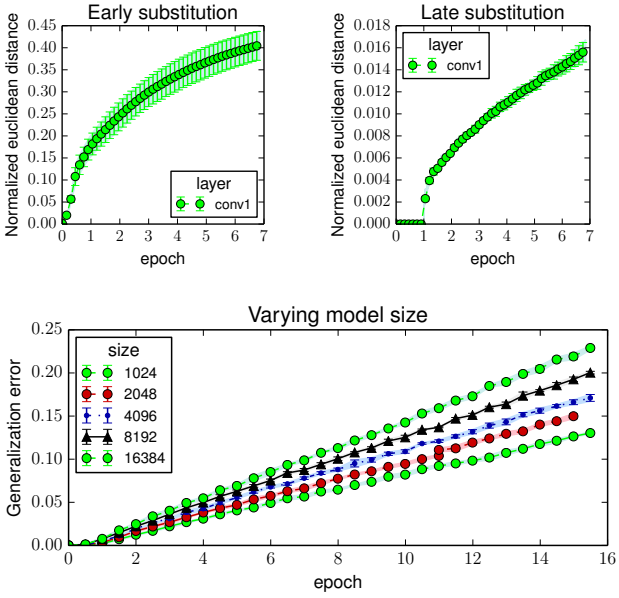

Figure 2. Experiments on ImageNet. Top left: Parameter divergence with early substitution. Top right: Late substitution. Bottom: Generalization error for varying model size.

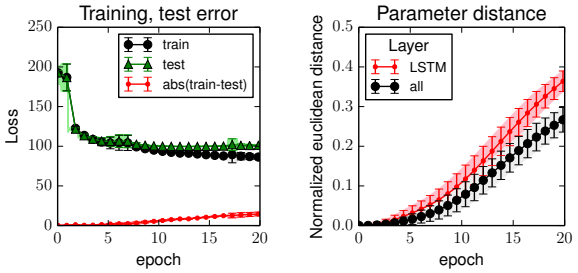

Figure 3. Training on PTB dataset with LSTM architecture.

Acknowledgements

The authors would like to thank Martin Abadi, Samy Bengio, Thomas Breuel, John Duchi, Vineet Gupta, Kevin Jamieson, Kenneth Marino, Giorgio Patrini, John Platt, Eric Price, Ludwig Schmidt, Nati Srebro, Ilya Sutskever, and Oriol Vinyals for their valuable feedback.

References

Arora, Sanjeev, Ge, Rong, Ma, Tengyu, and Moitra, Ankur. Simple, efficient, and neural algorithms for sparse coding. In *Proceedings of the Conference on Learning Theory (COLT)*, 2015.

Bottou, Léon. Online algorithms and stochastic approximations. In Saad, David (ed.), *Online Learning and Neural Networks*. Cambridge University Press, Cambridge, UK, 1998.

Bottou, Léon and Bousquet, Olivier. The tradeoffs of large scale learning. In *Neural Information Processing Systems*, 2008.

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

Devroye, Luc P. and Wagner, T.J. Distribution-free performance bounds for potential function rules. *Information Theory, IEEE Transactions on*, 25(5):601–604, Sep 1979. ISSN 0018-9448.

Elisseeff, Andre, Evgeniou, Theodoros, and Pontil, Massimiliano. Stability of randomized learning algorithms. *Journal of Machine Learning Research*, 6:55–79, 2005.

Frostig, R., Ge, R., Kakade, S. M., and Sidford, A. Competing with the empirical risk minimizer in a single pass. In *Conference on Learning Theory (COLT)*, 2015.

Ghadimi, Saeed and Lan, Guanghui. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

Hazan, Elad and Kale, Satyen. Beyond the regret minimization barrier: optimal algorithms for stochastic strongly-convex optimization. *Journal of Machine Learning Research*, 15(1):2489–2512, 2014.

Hazan, Elad, Kalai, Adam, Kale, Satyen, and Agarwal, Amit. Logarithmic regret algorithms for online convex optimization. In *Proceedings of the 19th Annual Conference on Learning Theory*, 2006.

Janzamin, Majid, Sedghi, Hanie, and Anandkumar, Anima. Generalization bounds for neural networks through tensor factorization. Preprint available at `arXiv:1506.08473`, 2015.

Kearns, Michael J. and Ron, Dana. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation*, 11(6):1427–1453, 1999.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*, pp. 1097–1105, 2012.

Krogh, A. and Hertz, J. A. A simple weight decay can improve generalization. In *Proc. NIPS*, pp. 950–957, 1992.

Kushner, Harold J. and Yin, G. George. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer-Verlag, New York, second edition, 2003.

Livni, Roi, Shalev-Shwartz, Shai, and Shamir, Ohad. On the computational efficiency of training neural networks. In *Advances in Neural Information Processing Systems*, pp. 855–863, 2014.

Marcus, Mitchell P, Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330, 1993.

Mukherjee, Sayan, Niyogi, Partha, Poggio, Tomaso, and Rifkin, Ryan M. Learning theory: stability is sufficient for generalization and necessary and sufficient for consistency of empirical risk minimization. *Adv. Comput. Math.*, 25(1-3):161–193, 2006.

Nemirovski, A, Juditsky, A, Lan, G, and Shapiro, A. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on Optimization*, 19(4): 1574–1609, 2009.

Nemirovski, Arkadi and Yudin, David B. On Cezari's convergence of the steepest descent method for approximating saddle point of convex-concave functions. In *Soviet Mathetmatics Doklady*, volume 19, 1978.

Nemirovski, Arkadi and Yudin, David B. *Problem complexity and method efficiency in optimization*. Wiley Interscience, 1983.

Nesterov, Y. *Introductory lectures on convex optimization*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, Boston, MA, 2004. ISBN 1-4020-7553-7. A basic course.

Polyak, B. T. Introduction to optimization. *Optimization Software, Inc.*, 1987.

Rakhlin, Alexander, Shamir, Ohad, and Sridharan, Karthik. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning*, 2012. Extended version at `arxiv:1109.5647`.

Rosasco, Lorenzo and Villa, Silvia. Learning with incremental iterative regularization. Preprint available at `arXiv:1405.0042v2`, 2014.

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

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Zaremba, Wojciech, Sutskever, Ilya, and Vinyals, Oriol. Recurrent neural network regularization. Technical report, 2014. Preprint available at `arxiv:1409.2329`.