Generalization Bounds for MLP's (Multilayer Perceptron)

Kushal Tirumala
California Institute of Technology

Abstract

In recent years, there have been many efforts to provide non vacuous generalization bounds for different neural networks. Generally, these different bounds either arise from (1) considering specific deep learning architectures (MLP, CNN, etc.) or (2) interpreting generalization of a network in a different way (probabilistic, compression, etc.) Here, we take a step back from trying to bound generalization using complicated analysis, instead focusing on intuitive model-specific bounds. We are inspired by simpler notions of learning algorithm stability, and analyze the bounds to come up with improvements to SGD.

1 Background

Deriving generalization bounds for machine learning has been a topic of interest for sometime, and has recently become more popular. Generalization is usually approached from the lens of complexity measures, wherein we look for correlation with generalization. For example, standard ML intuition leans towards low weight norms, because low weight norms tend to allow models to generalize — therefore, weight norms are a simple example of a "complexity measure."

More recently, a lot of work has been done to get tight bounds on generalization for MLPs. This can be traced back to VC dimension bounds and statistical learning theory. The main problem with these methods is that neural networks (broadly speaking) are universal function approximators (as evidenced by [1] wherein neural networks can fit even randomly labeled data) — they can fit almost any dataset given enough iterations. So the VC dimension of neural networks (or MLP's for that matter) is naturally very big, leading to vaccuous bounds on generalization. There has obviously been improvements (with probabilistic techniques and compression techniques) on bounding the generalization gap for different models; however, a lot of them involve a somewhat involved framework. Here, we try to approach bounding generalization gap of MLPs in an intuitive way — if we know how sensitive the MLP is to input changes and parameter changes, we should have all the information we need to bound generalization.

2 Motivation

One of the mysteries of machine learning is how variable even simple models can be in terms of generalization. One of the most simple models — multilayer perceptron — seemed to have nonintuitive generalization behavior. Regularization usually helps with this (whether it is weight regularization or some other form of implicit regularization such as dropout).

Instead of trying to dive into complex models and complex frameworks for generalization, we can consider simple models — multilayer perceptrons (MLP). First, we survey the field of theoretical generalization bounds for MLPs. Then, we consider a simple idea — if you can bound how sensitive an MLP is to input changes, and how sensitive it is to parameter changes, that should encompass all information needed to predict how well an MLP generalizes. This is because generalization of a model class is, broadly speaking, a measure of how well a model class does when we change input data and its parameters.

One thing to note is that most generalization bounds do incorporate model weights/parameters in some way (whether through VC dimension, weight norms, etc.) and there has been promising results in coming up with non vacuous bounds. However, a lot of these analyses require stepping into some framework, which on some level feels unintuitive. Therefore, the main motivation of this report is to see if we can use practical intuition from machine learning (i.e input and parameter sensitivity and the idea of stability), to both bound generalization and improve SGD to help models avoid overfiting.

3 Previous Work

First, we survey some other MLP bounds found in current literature:

(VC dimension bound): An MLP with ReLU activiation, W parameters, and L layers has VC dimension $O(WL\log W)$ Harvey et al. [2], giving that with probability $\geq 1-\delta$ we have:

$$L_{true}(f) \leq L_{train}(f;S) + O\left(\sqrt{\frac{d\log\frac{2em}{d} + 2\log\frac{2}{\delta}}{m}}\right)$$

where $m = |S|$ is the number of data points, d is the VC dimension so $d = O(WL\log W)$. The link between VC dimension and generalization is well known, but for a quick proof refer to Corollary 6-2 in [3]. Although this bound incorporates model (through d), we note that neural networks have long been known as universal function approximators. For example, in [1] it was shown that neural networks can fit even random labels. Since VC dimension is the size of the largest dataset the learning algorithm can shatter, this means VC dimension of neural networks can become very big (remember the analysis shows the VC dimension grows asymptotically as the number of layers and number of

parameters, so the constant can become very big). This implies that generalization bounds based on VC dimension can very quickly become vacuous.

(Covering numbers bound): A slightly different bound in 2017 from Bartlett et al. [4] approached generalization based using covering numbers to upper bound Rademacher complexity, which can be thought of as a complexity measure which encompasses how well a sample represent the underlying distribution; it differs from VC dimension in that it is dependent on the distribution at hand. They assert that for a MLP with d layers and inputs bounded by $|x|_2 \leq B$, we have that (by Neyshabur et al.'s analysis of the bound in [5]):

$$L_{true}(f_w) \leq \hat{L}_\gamma(f_w) + \tilde{O}\left(\sqrt{\frac{B^2 \prod_i ||W_i||_2^2 (\sum_i \frac{||W_i||_1}{||W_i||_2})}{\gamma^2 m}}\right)$$

where B is the bound for the inputs, $||W_i||_2^2$ is the square of the $l2$ norm, and γ is a defined margin, where L_γ is margin loss (which essentially is an upper bound on the 0/1 loss, since it counts a misclassification as a wrong classification by the model, or a right classification by the model with too low confidence $\leq \gamma$), and we ignore log terms. This also incorporates model information, and is a little less likely to become vacuous with larger nets (since there is no VC dimension term, just norm of weights); however, it should be noted that norms naturally grow during training as shown in figure 1 in [6].

(PAC-Bayes bound): Finally, we have recent PAC-Bayes bound from Neyshabur et al. [5] PAC-Bayes is one of the methodologies receiving a lot of attention, due to its ability to give tighter bounds on generalization. The analysis is very similar to the notion of parameter sensitivity that will be discussed in the following sections (in fact lemma 2 in the paper is essentially parameter sensitivity bound). For MLP's with relu nonlinearity, and margin $\gamma > 0$ we have that:

$$L_{true}(f_w) \leq \hat{L}_\gamma(f_w) + O\left(\sqrt{\frac{B^2 d^2 h \, log(dh) \prod_i^d ||W_i||_2^2 \sum_i^d \frac{||W_i||_F^2}{||W_i||_2^2} + ln\frac{dm}{\gamma}}{\gamma^2 m}}\right)$$

where d is the number of layers, B is the bound on inputs, h is the number of parameters, and m is the number of data points. We see some similarity between the covering numbers bound above, and as mentioned in [5] this bound dominates when weights are dense and uniform magnitude.

4 Novel Generalization Bound

We can also define the true loss of our MLP wrt dataset x as :

$$L_{true}(f) = \mathbb{E}_{z \sim P}[L(f, z)]$$

where P is distribution from which our samples x are drawn, and L is the loss of f on a single data point x. Similarly we can define training (empirical) loss as:

$$L_{train}(f; S) = \frac{1}{|S|} \sum_i L(f, z_i)$$

where z_i is the i'th data point in the sample S. Technically, a generalization gap is given by $L_{true} - L_{train}$; however, in practice, generalization gap is estimated by looking at the difference in in sample loss (L_{train}) and out of sample loss (L_{test}) (because we have no handle on the true distribution P).

Theorem 1 *Take L-layer perceptron f with weights $\{W_i\}$ with bounded input, output, normalized weight change matrices (at every step of optimization procedure), and gaussian initialization of weight matrices. Assume loss L is Lipschitz and bounded, and nonlinearity φ is s.t. $\alpha||x-y||_2^2 \leq ||\varphi(x) - \varphi(y)||_2^2 \leq \beta||x-y||_2^2$, where $0 \leq \alpha, \beta \leq 1$, $\varphi(0) = 0$, and φ has lipshictz constant. Let κ be a bound on the condition number of the weight matrices $\{W_i\}$. Then with probability $\geq 1 - \delta$ the*

generalization gap for f (for fixed input dataset X) is given by

$$L_{true}(f) \leq L_{train}(f;X) + O\left(\sqrt{\frac{\prod_{i=1}^{L}\lambda_i + \sqrt{\left(\prod_{i=1}^{L}\left(1+\frac{K}{||W_l||_F^2}\right)-1\right)}}{2m\delta}}\right)$$

where m is the number of data points, K is the number of iterations, and λ_i is the spectral norm of W_i.

Proof: Consider the following problem setup. Say we have a L-layer perceptron with nonlinearity σ. The layers will be denoted by W_i and the corresponding nonlinearities will be denoted by σ_i. We can define the output of the multilayer perceptron as:

$$f(W;x) = \sigma_L(W_L(\sigma_{L-1}W_{L-1}(\dots(W_1(x)))))$$

or more concisely, we can see the above as $f(W;x) = \circ_{i=1}^{n}\sigma_i W_i(x)$ i.e the composition of the layers with nonlinearity. We are not assuming any Lipschitz constants on the weight matrices, because weight matrices in the context of deep learning can become rather arbitrary. However, we can specify a bit about nonlinearity (as shown later).

Specifically, assume the nonlinerity σ has a Lipschitz constant of C. Let λ_i be the maximum eigenvalue of matrix W_i. Then since weight matrices W_i are nothing but linear transformations, we have for any $x, y \in \mathbb{R}^k$ that $||W_i(x) - W_i(y)||_2 = ||W_i(x-y)|| \leq ||\lambda_i(x-y)||_2 = \lambda_i||x-y||_2$. So we have that:

$$\begin{aligned}
||f(W;x+\delta x) - f(W;x)||_2 &= ||\circ_{i=1}^{L}\sigma_i W_i(x+\delta x) - \circ_{i=1}^{L}\sigma_i W_i(x)||_2 \\
&\leq C * \lambda_n||\circ_{i=1}^{L-1}\sigma_i W_i(x+\delta x) - \circ_{i=1}^{L-1}\sigma_i W_i(x)||_2 \\
&\dots \\
&\leq C^L \prod_{i=1}^{L}\lambda_i||x+\delta x - x||_2 \\
&= C^L \prod_{i=1}^{n}\lambda_i||\delta x||_2
\end{aligned}$$

Then from Theorem 1 in [7] (Bernstein et al.), we can see that the parameter sensitivity of a MLP is given by:

$$||f(W+\delta W;x) - f(W;x)||_2^2 \leq G\left(\prod_{i=1}^{L}\left(1+\frac{||\delta W_l||_F^2}{||W_l||_F^2}\right)-1\right)||f(W;x)||_2^2$$

where $G = \left(2\frac{\beta}{\alpha}n^*\kappa^2\right)^L$, where n^* is the maximum width of the network, κ is a bound on the condition number of the weight matrices $\{W_l\}$, and α, β are lower and upper bounds on the nonlinearity i.e $\alpha||x-y||_2^2 \leq ||\varphi(x) - \varphi(y)||_2^2 \leq \beta||x-y||_2^2$

As a quick note, note that Theorem 1 in [7] requires the condition number to be bounded, and this may not necessarily be the case. However as noted in [7], smoothed analysis of condition number of MLP's have shown that κ is finite with probability 1 for an iid gaussian initialization (and stays finite with small amount of noise added in training process).

Now we can draw from statistical learning theory in [8], specifically from the concept of hypothesis stability. We call a learning algorithm A to have hypothesis stability of β if the expected leave-one-out error is bounded with respect to sample is bounded, or if for samples $S \in \mathbb{R}^k$ and all $i \in \{1,\dots,m\}$:

$$\mathbb{E}_S[|L(f_S, z_i) - L(f_{S^{\setminus i}}, z_i)|] \leq \beta$$

for every data point $z_i \in S$. $S^{\setminus i}$ is the same as sample S with the ith data point removed, and f_S represents the learned function with respect to S.

We can recast the above inequality in terms of input and parameter sensitivity, since the difference between f_S and $f_{S^{\setminus i}}$ is nothing but change in parameters, and similarly the difference between S and $S^{\setminus i}$ is nothing but the change in input. Specifically, assume that the loss function L is L'-lipschitz, so that:

$$\begin{aligned}|L(f_S, S) - L(f_{S^{\setminus i}}, S^i)| &\leq L'|f_S(S) - f_{S^{\setminus i}}(S^{\setminus i})| \\ &= L'|f(W + \delta W; x + \delta x) - f(W; x)| \\ &= L'|f(W + \delta W; x + \delta x) - f(W + \delta W; x) + f(W + \delta W; x) - f(W; x)| \\ &\leq L'\bigg(|f(W + \delta W; x + \delta x) - f(W + \delta W; x)| + |f(W + \delta W; x) - f(W; x)|\bigg)\end{aligned}$$

where the second term in the chain of equalities is clearly an upper bound on $L'|f_S(z_i) - f_{S^{\setminus i}}(z_i)|$ for all i (this difference is discussed in Appendix 1). Now from above, we can bound the two terms as follows:

$$|f(W + \delta W; x + \delta x) - f(W + \delta W; x)| \leq \sqrt{t}\bigg(C^L \prod_{i=1}^{n} \lambda_i\bigg)||\delta x||_2$$

and

$$|f(W + \delta W; x) - f(W; x)| \leq \sqrt{t}\sqrt{G\bigg(\prod_{i=1}^{L}\bigg(1 + \frac{||\delta W_l||_F^2}{||W_l||_F^2}\bigg) - 1\bigg)||f(W; x)||_2^2}$$

where G is the constant defined above, and t is the dimension of our output. This follows since for any vector $v \in \mathbb{R}^n$ we have that $||v||_1 \leq \sqrt{n}||v||_2$, and then the statement follows by the previous input and parameter sensitivity bounds.

Since we normalize input and output so that $||\delta x||_2 = 1, ||f(W; x)||_2^2 = 1$, we arrive at the following bound:

$$|L(f_S, S) - L(f_{S^i}, S^i)| \leq L'\bigg(C^L\sqrt{t}\prod_{i=1}^{L}\lambda_i + \sqrt{t}\sqrt{G\bigg(\prod_{i=1}^{L}\bigg(1 + \frac{||\delta W_l||_F^2}{||W_l||_F^2}\bigg) - 1\bigg)}\bigg) = \beta_{S,S^i,\theta}$$

Then take $\beta_{S,S^i,\theta} = L'\sqrt{t}\bigg(C^L\prod_{i=1}^{L}\lambda_i + \sqrt{G * \prod_{i=1}^{L}\bigg(1 + \frac{||\delta W_l||_F^2}{||W_l||_F^2}\bigg) - 1}\bigg)$ where θ holds information about the parameterization of f (i.e how many layers of MLP).

The only problem with the above is that the bound depends on the networks f_S, f_{S^i} to determine $||\delta W_l||_F^2$. Since we normalize the change matrices used to update the network (at every step of the optimization procedure), at every update we have $||\delta W_1||_F^2 \leq 1$, so for any samples S, S^i, we have $||\delta W_l||_F^2 \leq 1$, so our bound becomes:

$$\beta_\theta = L'\sqrt{t}\bigg(C^L\prod_{i=1}^{L}\lambda_i + \sqrt{Gt * \bigg(\prod_{i=1}^{L}\bigg(1 + \frac{K}{||W_l||_F^2}\bigg) - 1\bigg)}\bigg)$$

where K is the number of iterations the network is trained for.

Clearly this leads to a pointwise hypothesis stability using the same β_θ bound (since the above takes the loss with respect to the whole sample). Then from Theorem 11 in [8] , we have that with probability $\geq 1 - \delta$:

$$L_{true}(f) \leq L_{train}(f; X) + \sqrt{\frac{M^2 + 12MmL'\sqrt{t}\bigg(C^L\prod_{i=1}^{L}\lambda_i + \sqrt{G * \bigg(\prod_{i=1}^{L}\bigg(1 + \frac{K}{||W_l||_F^2}\bigg) - 1\bigg)}\bigg)}{2m\delta}}$$

where M is the bound of our cost function, m is the size of our sample X, and the constants M, L' depend on the loss function chosen. m depends on the size of out sample, and t is output dimension, and G is a constant depending on choice of model. Note that all of these are constants (since the input dataset X is fixed), so we have that with probability $\geq 1 - \delta$ the generalization gap goes as:

$$O\left(\sqrt{\frac{\prod_i \lambda_i + \sqrt{\left(\prod_{i=1}^{L}\left(1 + \frac{K}{||W_i||_F^2}\right) - 1\right)}}{2m\delta}}\right)$$

and the desired result follows.

5 Generalization Bound Discussion

The bound above is a bit tasking to analyze directly, but the interesting thing is that it incorporates model information and training information. Some generalization bounds depend only on m, M and other Lipschitz constants, which works well when trying to prove generalization for learning algorithms (instead of specific models), which you can see with VC dimension bounds. Other bounds include model information, but not training procedure information. However, practitioners usually have specific model architectures and training procedure in mind, and would want to know how to provably reduce their generalization gap. This is why more recent bounds consider weight norms, or spectral norms, margin, or other model-specific attributes.

Note that up until this point, the only actual restriction we have imposed on the network is normalization (apart from gaussian initialization of weight matrices to keep condition number bounded). We normalized input, output, and change in weight matrices at an update step. As mentioned in [5], normalization of weight matrices does not affect the learning capability of a neural network, so we are not shrinking the model class. To see this intuitively, just note that normalization just squeezes the range of input/output to be between some restricted range, but it is the relative difference in inputs/outputs that a model captures.

The interesting parts of this bound are the dependence on the product spectral norm, and the product of $1 + \frac{K}{||W_i||_F^2}$ ratio. This bound can very quickly become vacuous if the $\frac{K}{||W_i||_F^2}$ is not suitably small (or number of layers is not suitably small), since the term is exponential in the number of layers (with base > 1).

Intuitively, spectral norm measures how much a weight matrix can stretch or shrink an input vector, so it makes sense that the lower the spectral norm product, the better the network generalizes (because the less a model can stretch or shrink an input vector, the more stable it is to input perturbation). Basic intuition also tells us that as you increase the number of updates, your model can overfit (generalization error can go up); however, parameter frobenius norms are usually minimized to prevent overfitting whereas the above suggests otherwise. But according to [9], extensive experiments in generalization correlation finds that spectral norm strongly correlates negatively with generalization, and frobenius norm of parameters is slightly postively correlated with generalization. This aligns with our bound, and gives a bit of outside empirical evidence to support the asymptotic behavior of MLP generalization.

In the next section, we use this analysis of the bound in the context of nonconvex optimization (to improve SGD).

6 Training Procedure (nonconvex optimization)

Loss landscapes for neural networks (with respect to parameter space) are nonconvex due to nonlinearity in the network. Nonlinearity in a model allows it to find nonlinear decision boundaries (which is why machine learning is so capable). Most models in ML (MLPs included) are optimized via standard SGD:

$$W_{t+1} = W_t - \eta \frac{\partial O}{\partial W_t}$$

where O is the procedure-defined objective function, η is the learning rate (or step size), and we compute the gradient with respect to a given batch. In the above, we did specify that the δW (weight change matrices) must be normalized, so our gradient descent is of the form:

$$W_{t+1} = W_t - \eta \frac{\partial O}{\partial W_t} * \frac{1}{||\frac{\partial O}{\partial W_t}||_F^2}$$

Note that in our bound above, we want to minimize $\prod_{i=1}^{L} \lambda_i + \prod_{i=1}^{L}(1 + \frac{K}{||W_i||_F^2}) \geq \prod_{i=1}^{L} \lambda_i + \prod_{i=1}^{L}(\frac{1}{||W_i||_F^2})$ since $K > 0$. By AM-GM, we can instead consider minimizing:

$$\prod_{i=1}^{L} \lambda_i \prod_{i=1}^{L} \frac{1}{||W_i||_F^2} = \prod_{i=1}^{L} \lambda_i (\frac{1}{||W_i||_F^2})$$

So our bound from above suggests that good generalization with training sample S comes from minimizing the following objective:

$$\min_W \frac{1}{n} \sum_{i=1}^{n} L(f_S, z_i) + a_1 \sum_{j=1}^{L} \frac{\lambda_j}{||W_j||_F^2}$$

where W is what parameterizes our function f (as an MLP), λ_j represents the largest eigenvalue of weight matrix j, and a_1 is a regularization coefficient. To do this, we can consider SGD with a modified update. Specifically, we need to find the gradient of the regularization term $R = \frac{\lambda_j}{||W_j||_F^2}$ (with respect to W_j). First note that by quotient rule we have:

$$\frac{\partial R}{\partial W_j} = \frac{\lambda_j' ||W_j||_F^2 - (||W_j||_F^2)' \lambda_j}{(||W_j||_F^2)^2}$$

Note that $\frac{\partial ||W_j||_F^2}{\partial W_j} = \frac{\partial}{\partial W_j} Tr(W_j W_j^T) = 2W_j$. Note that the gradient of the spectral norm is given by $\frac{\partial \lambda_j}{\partial W_j} = u_1 v_1^T$, where u_1 is the first left singular vector, v_1 is the first right singular vector (note that this is in the case the two largest eigenvalues are not the same).

So we have our SGD update as:

$$W_{t+1} = W_t - \eta \left(\frac{\partial L}{\partial W_t} + \left(\frac{u_1 v_1^T ||W_t||_F^2 - 2W_j \lambda_j}{(||W_j||_F^2)^2} \right) \right) * \frac{1}{N}$$

where $N = ||\frac{\partial (L+R)}{\partial W_t}||_F^2$ is the change normalization factor.

7 Generalization Bound Analysis

One quick way to test the usefulness of the bound is to empirically track the results of MLP during training, with generalization.

To test this bound, we considered the following experiments:

1. MLP with the following layers Linear(32*32*3) layer, Linear(512), Linear(10) with ReLU activation for 6000 iterations (on CIFAR 10) with batch size 512, and learning rate 0.01.
2. MLP with the following layers Linear(32*32*3) layer, Linear(2048), Linear(512), Linear(256), Linear(10) with ReLu activation for 10000 iterations (on CIFAR 10) with batch size 512, and learning rate 0.01.
3. MLP with following layers Linear(28*28), Linear(256), Linear(10), with ReLu activation for 1000 iterations (on MNIST) with batch size 250, learning rate 0.01.

CIFAR10 is a dataset of 60,000 images from 10 different classes. The training set consists of 50,000 images, and the test set consists of 10,000 images. MNIST is a dataset of 50000 handwritten images, with a test set of 10000 images. Models were all trained all with cross entropy loss, to keep consistency across results. We also track metrics as per update step, instead of epoch (since plotting versus epoch scales plots as per batch size, leading to misinterpretation). We then tracked the following metrics:

1. Train, Test Loss
2. Train, Test Accuracy
3. Inverse frobenius norm term ("term 3") $\prod_{i=1}^{L} 1 + \frac{K}{||W_i||_F^2}$
4. Spectral norm term ("term 4") $\prod_{i=1}^{L} ||W_i||_*$
5. Generalization term ("term 5") $\sqrt{\sqrt{\prod_{i=1}^{L}\left(1 + \frac{K}{||W_i||_F^2}\right)} + \prod_{i=1}^{L} ||W_i||_*}$

We note that since we work on a finite set of data points, the loss function is bounded, and has bounded gradient (and therefore is Lipschitz). Another concern is that we use ReLU in the following, although the parameter sensitivity bound will diverge for $\alpha = 0$ (the G constant in section 4). Since this is an empirical analysis, we leave reconciling this for future work (using leaky ReLU for example). Also, since we are more interested in the model-specific terms in the generalization bound, we chose to focus on the terms 3, 4, and 5 above (instead of the whole bound).

We first walk through an analysis of the experiment (1) results. We note that the model does not perform that well overall, as evidenced by the train/test accuracies in figure 1

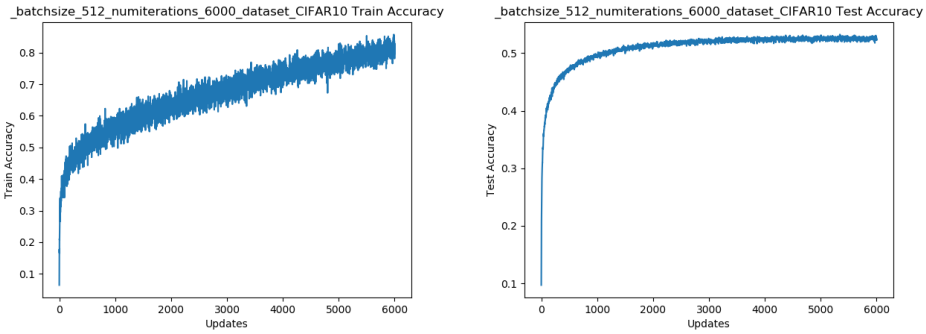

Figure 1: Experiment 1, train and test accuracies plotted against update count. Shown on the left is the training accuracy over 6000 iterations, and on the right is the test accuracy over 6000 iterations

This is due to the fact that CIFAR10 is a relatively harder image dataset to classify, in the sense that simple networks (such as MLPs) might not be fully capable of reaching high test accuracy. Practitioners typically use some form of CNN (for example, VGGnet). To see examples of different MLP architectures and more complex architectures see Table 1 in [1].

To get a measure of the empirical generalization, the standard is to look at the difference between test and training loss in figure 2:

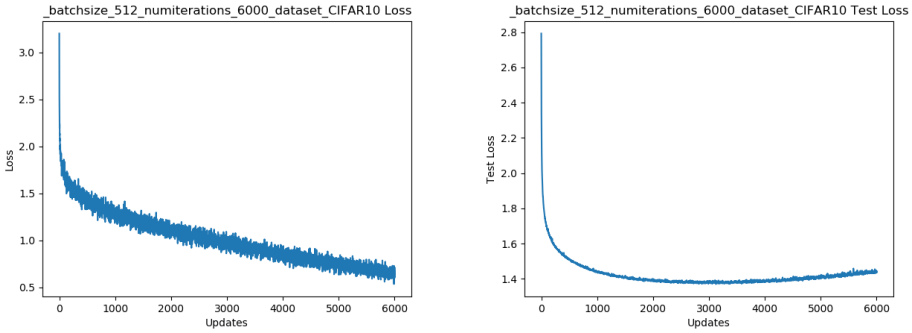

Figure 2: Experiment 1, train and test losses plotted against update count. Shown on the left is the training loss over 6000 iterations, and on the right is the test loss over 6000 iterations

We see that the network is able to learn, although it has not fully converged yet (since the training loss is still decreasing). Usually, we would allow this training loss to completely converge, but we notice that the model already begins to overfit — near the 3000 iteration, we see the test loss starts to increase. This is a classic example of overfitting, and gives us a good base point to see how much information our generalization terms give us in an empirical setting. Looking at the generalization bound terms:

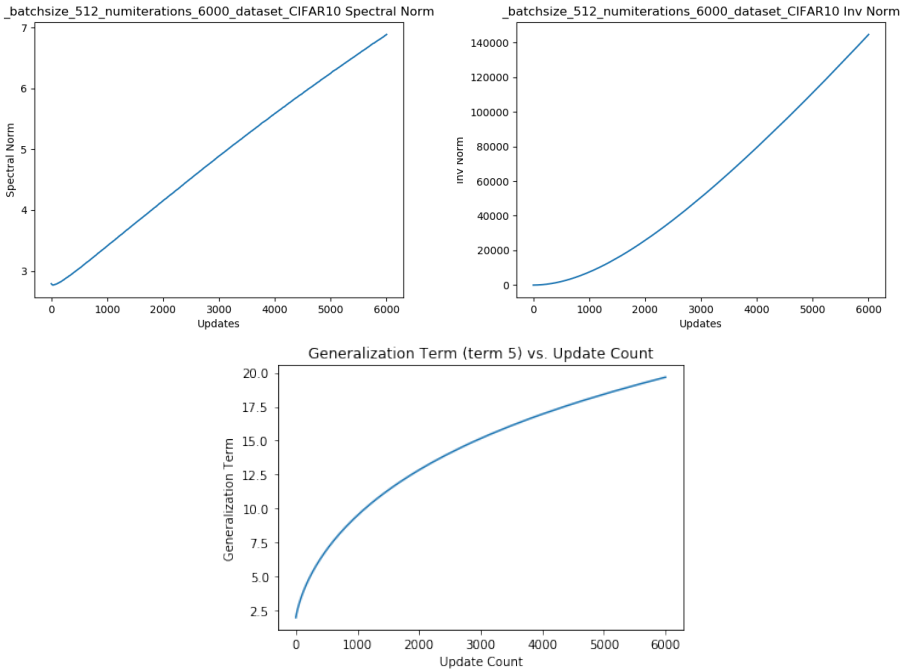

Figure 3: Experiment 1, terms in novel generalization bound tracked against update step. Plotted on the top left is the product of spectral norm (see term (3)), on the top right of the inverse frobenius term (see term (4)). On the bottom plotted is the generalization term (see term (5))

The "gradient" in figure 4 is $g_i^2 - g_j^2$ where g_i is the generaliziation term at update step i, and g_j the generalization term at update step j. This works as the closest linear approximation (to the square of the gradient term) at every point since the change in the update step is always 1. What we see in figure 4 is that the gradient of the generalization term has low slope (i.e the second derivative of the gradient term is 0) around the same time the empirical generalization gap is lowest (from the data, this occurs at iteration 153):

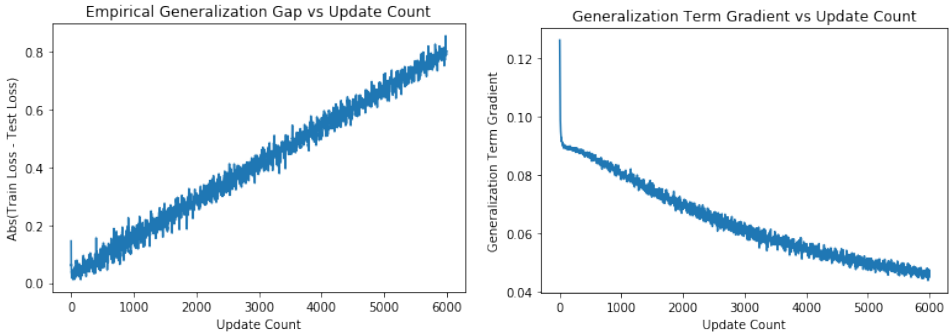

Figure 4: Experiment 1, on the right plotted is the gradient of the generalization term, and on the left plotted is the absolute value of generalization gap (subtracting train loss from test loss)

So we see that the generalization term (term (5)) has some correlation with the actual empirical generalization gap. What is more interesting is that the analysis of the individual terms (terms 3, 4) shows little correlation with actual empiricial generalization gap both in figure 3 and in the context of the generalization gradient analysis done above. You can see this since in figure 3 we see that the inverse norm term is concave up (which means the gradient will have no decrease), and the spectral norm term is very slightly concave down after being concave up for the first few iterations. Specifically, the difference graph goes as:

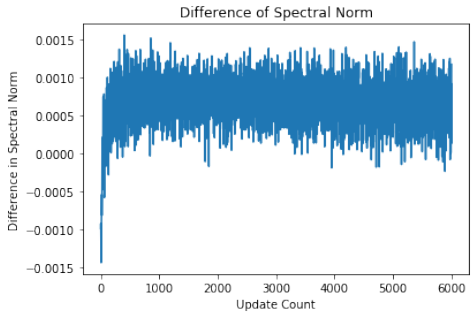

Figure 5: Experiment 1, gradient of spectral norm plotted against update count ("gradient" taken as sequential difference, since change in update count is always 1)

which does not have any clear trend with the empirical model generalization gap. So we see that individually, terms (3) and (4) might not have much meaning for the empirical generalization of a model. However, combining then in term (5) gives us some actual information on empirical model generalization.

In figures 7, 8, and 9 we show the same results for experiment (2), wherein we used an MLP with more parameters and trained for a higher number of iterations. The purpose is to introduce more overfitting, and analyze whether there is a similarity in the behavior we saw with experiment (1). We see that the model overfits much more, as expected. We also see that as expected, the scale of the inverse frobenius term, spectral norm term, and generalization term (since we added more parameters). However, we note that very similar behavior is apparent, where we have the 0 slope of the gradient of the generalization term correlating with where the model starts to overfit (in experiment (2), the model starts to overfit at iteration 179).

As a control experiment, we also ran the architecture specified in experiment (3). The purpose of this experiment was to observe the behavior with respect to terms (3), (4), and (5) with a model that does not overfit. You can observe the results in figures 10, 11, 12. We see that there is very little overfitting occurring in 2, and the model is able to reach very high train/test accuracy. We see that in figure 11, there is very different behavior in the gradient of the generalization term, since there is no clear change in the trend of the generalization gradient, and the empirical gap hovers around 0.04 (there is no trend to empirical generalization gap). When we observe the generalization terms, we see that the

spectral norm and inverse frobenius term scales very similarly to experiments 1 and 2, but there is less curvature to the generalization term.

In fact, if we plot the square of the generalization term for the three experiments:

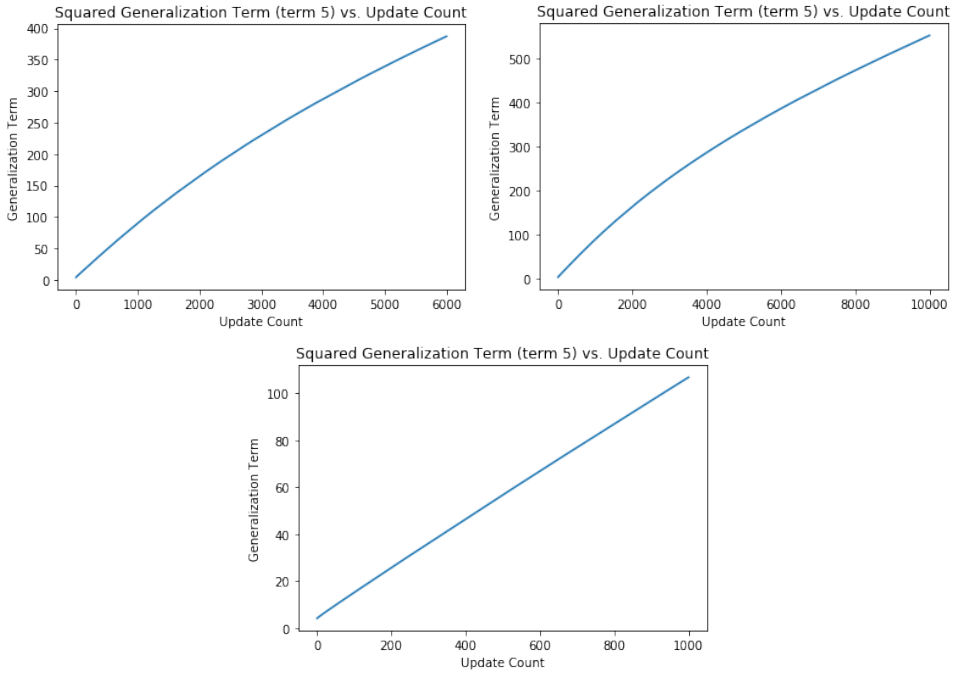

Figure 6: (Generalization Term)^2 plotted for experiment 1 (top left), experiment 2 (top right), and experiment 3 (bottom)

we see that the model that generalizes well (right most graph) has a straight line trend, whereas the overfitting models (two left graphs) have a slight curvature. This suggests that the generalization term is non-linear in the number of updates for overfitting models, and might be linear in the number of updates for non-overfitting (good generalization) models. This is a bit of a surprise, since the K-terms in the generalization bound (if you expand out the product over layers) is dominated by a $K^{L/4}$ term, so a linear relationship is not expected between the generalization term and the number of updates (however, there could be curvature to the bottom graph if extended for more iterations).

The conclusion from this portion is that the combined generalization term (term (5)) actually has some correlation with empirical generalization gap. Specifically, slope of gradient of generalization term is near 0, around where the model begins to overfit. In contrast, individually the terms in the generalization bound do not seem to have any trend with empirical generalization gap. We also see that overfitting models have curvature in this generalization term (over number of updates), whereas generalizing models do not.

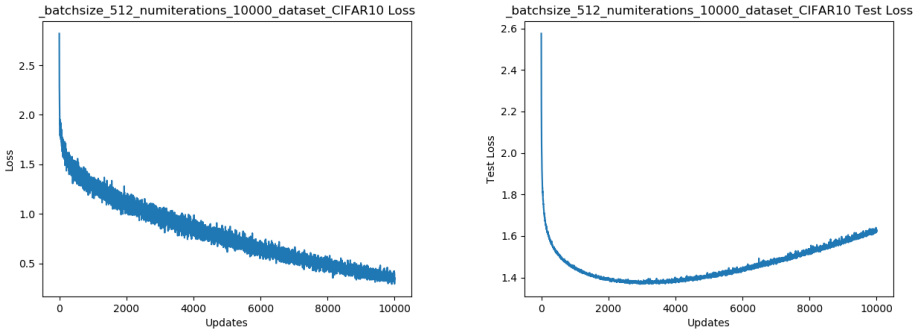

Figure 7: Experiment 2, train and test losses plotted against update count. Shown on the left is the training loss over 6000 iterations, and on the right is the test loss over 6000 iterations

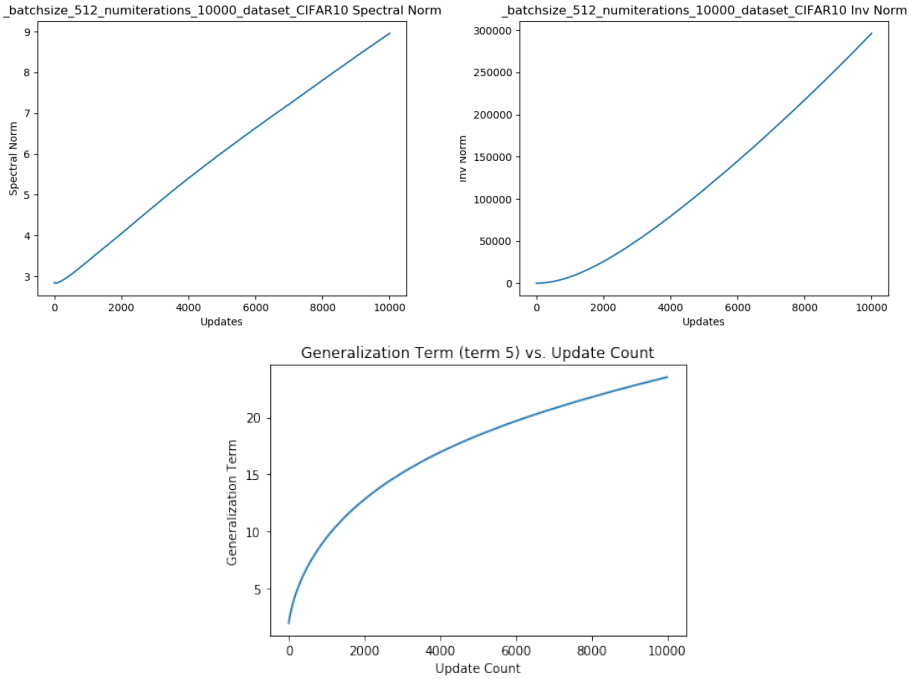

Figure 8: Experiment 2, generalization terms 3, 4, 5 plotted against update step. Plotted on the top left is the product of spectral norm (see term (3)), on the top right of the inverse frobenius term (see term (4)). On the bottom plotted is the generalization term (see term (5))

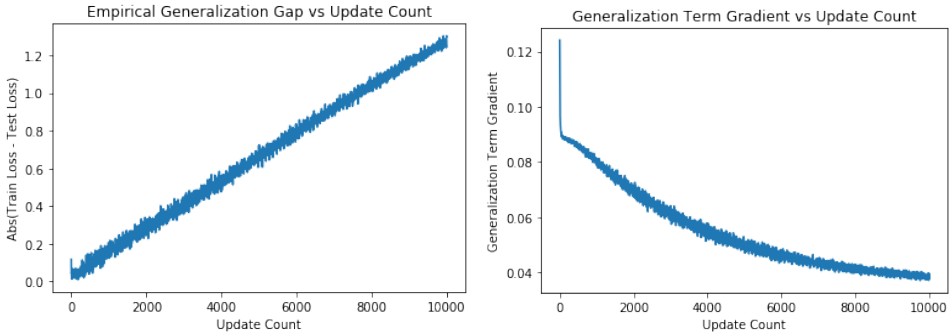

Figure 9: Experiment 2, on the right plotted is the gradient of the generalization, and on the left plotted is the absolute value of generalization gap (subtracting train loss from test loss)

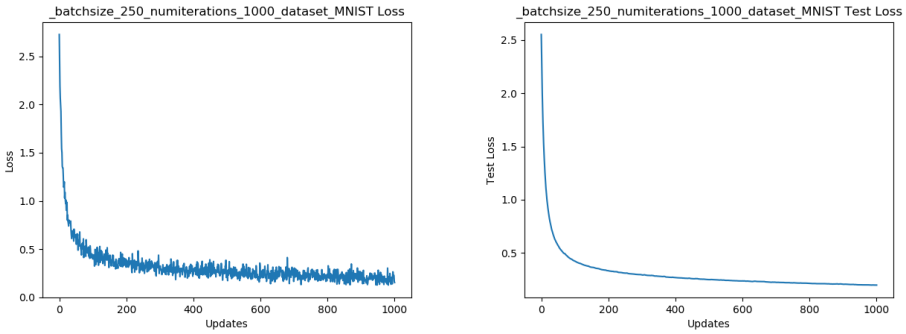

Figure 10: Experiment 3, train and test losses plotted against update count. Shown on the left is the training loss over 6000 iterations, and on the right is the test loss over 6000 iterations

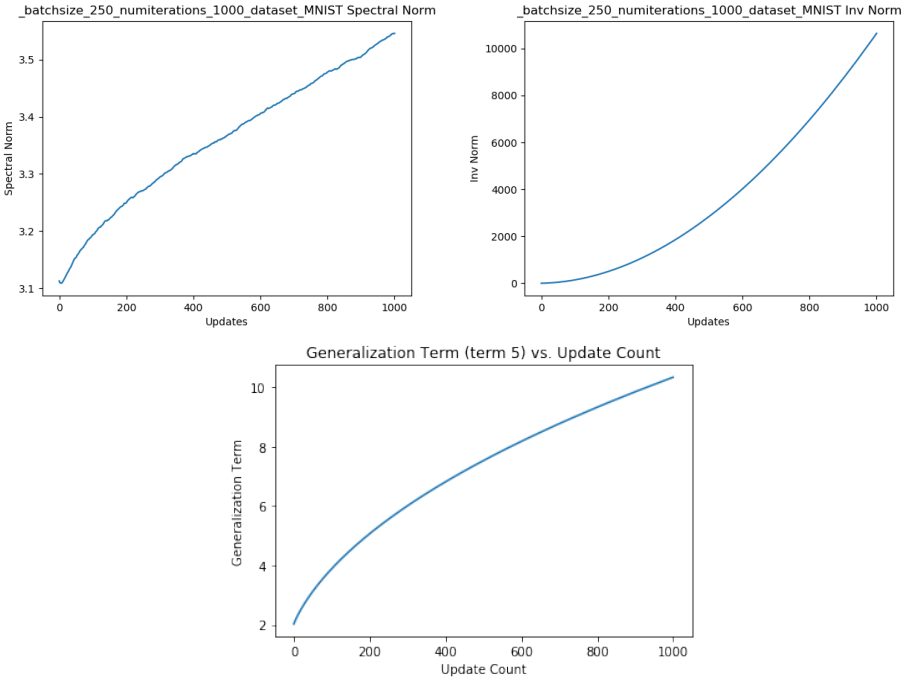

Figure 11: Experiment 3, generalization terms 3, 4, 5 plotted against update step. Plotted on the top left is the product of spectral norm (see term (3)), on the top right of the inverse frobenius term (see term (4)). On the bottom plotted is the generalization term (see term (5))

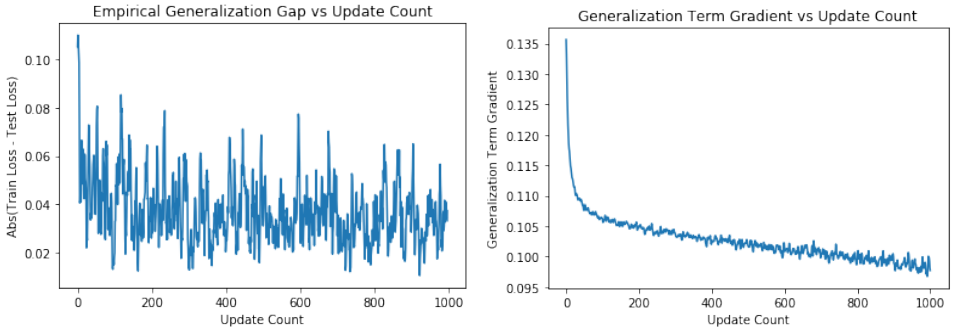

Figure 12: Experiment 3, on the right plotted is the pairwise difference of the generalization, and on the left plotted is the absolute value of generalization gap (subtracting train loss from test loss)

8 Improved SGD Analysis

To empirically test the improvements of the new regularization term in section 6, we used the pytorch 'Optimizer' base class to implement a new optimizer. We run the same setup as experiment (1), with 3 different optimizers:

1. Normal SGD with learning rate 0.01
2. Normal SGD with $L2$ regularization with regularization coefficient 1 and learning rate 0.01
3. Normal SGD with regularization as derived in section 6 with regularization coefficient 1 with learning rate 0.01

We show the test loss from the three trials below:

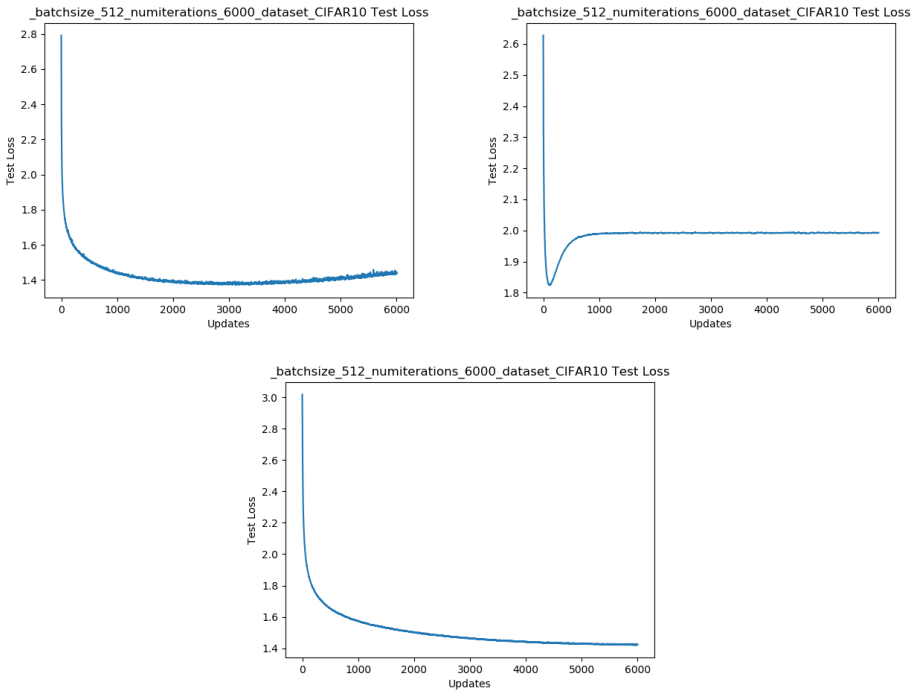

Figure 13: Comparison of test loss curves, with different optimizers. On the top left, we have no regularization. On the top right, we have L2 regularization. On the bottom, we have regularization from section 6

We see that without regularization, we see overfitting around the 3000'th iteration. With L2 regularization, we have that in the first 100 iterations the model starts to overfit, and then stabilizes around the 1000th iteration. Finally, we see that with the regularization in 6, there is no overfitting. This suggests that this new regularization might actually be an improvement over standard regularization techniques used in applied machine learning.

It is also worth noting that computing u_1, v_1 for every weight matrix at every optimization step can become computationally expensive, and as of right now the implementation manually finds the SVD decomposition to retrieve u_1, v_1 and the largest eigenvalue. It might be worth exploring power iteration methods to estimate these quantities instead of finding the entire SVD decomposition at every optimization step for every weight matrix.

A quick note: after doing my analysis, I found a similar analysis done only for spectral norm from 2017 ([10]), where they also bound input sensitivity by product of spectral norm and try to improve SGD. You see from figure 1 that DenseNet trained on CIFAR100 with spectral norm regularization and cross entropy loss has the worst test accuracy but the lowest generalization gap. This is interesting, since it suggests that while spectral norm regularization helps reduce overfitting, it also restricts

the model from learning; however, our regularization achieves around the same test accuracy as the non-regularized model, but still prevents overfitting.

9 Conclusion

There are three main conclusions to take away from this work:

1. Considering input sensitivity and parameter sensitivity can lead to stability bounds, which can be used to bound generalization. Using these stability bounds, we can get a bound on generalization that scales as $\sqrt{\sqrt{\prod_{i=1}^{L}\left(1+\frac{K}{||W_i||_F^2}\right)}+\prod_{i=1}^{L}||W_i||_*}$, where $||\cdot||_*$ denotes the spectral norm, $||W_i||_F$ is the frobenius norm, and K is the number of iterations the model is trained for. However, this bound can become vacuous with a large number of layers (deep learning), or if the $\frac{K-1}{||W_i||_F^2}$ is not sufficiently small.
2. Incorporating sensitivity-based generalization bounds in SGD can empirically improve MLP performance.
3. Considering spectral norm and inverse frobenius norm holds some information in the context of the empirical generalization gap. Specifically, we see that there is an inflection point in the gradient of the generalization term, around when the model empirically starts to overfit. We also see that the square of generalization term has curvature (over update count) only for overfitting models (in the experiments we tried) (see figure 6). In other words, considering spectral norm and inverse frobenius norm holds some information in the context of empirical generalization gap.

Another ending remark is that the regularization term we consider:

$$\sum \frac{||W_i||_*}{||W_i||_F^2}$$

is somewhat related to condition number. We can rewrite $||W_i||_F^2$ as the sum of singular values of W_i. So by trying to minimize $\frac{||W_i||_*}{||W_i||_F^2}$, we are essentially trying to bring the largest eigenvalue down as much as possible, while keeping the sum of eigenvalues large. This is closely related to minimizing the condition number of the weight matrices since $\kappa(W)=\frac{\lambda_{max}}{\lambda_{min}}$. In other words, considering input and parameter sensitivity could be equivalent to considering condition number of weight matrices (for MLPs), which could lead to tighter bounds.

One thing I would like to point out is that in Appendix I, we can actually make the generalization bound tighter, although the terms in the bound do not track well with empirical generalization gap (i.e there is little correlation between the theoretical generalization gap term, and empirical generalization gap). This suggests that the slightly stronger notion of stability of:

$$|L(f_S(S)) - f_{S^{\setminus i}}(S^{\setminus i})|$$

might be fruitful in coming up with meaningful generalization bounds. In the context of [8], this falls within pointwise hypothesis stability and uniform stability (since this implies pointwise hypothesis stability).

9.1 Future Research

There is plenty of work to be done in line with the above. Some include:

(1) Incorporating the notion of uniform stability. As you might have noticed, we proved the β bound for every such sample S and every data point i, which is usually only done to prove uniform stability of a learning algorithm. The above gets a bit close, but cannot prove anything for the L-infinity norm across all samples and data points yet.

(2) Exploring the stability with respect sample leave-one-out error (as opposed to point-wise) as discussed in the conclusion, seeing what specific generalization bounds come from it

algorithms that are sample leave-one-out-error stable, and reconciling this notion with uniform stability in some way (we know it implies pointwise hypothesis stability, but there may be some implications for uniform stability).

(3) More testing of the improved SGD update. We note that when run on slightly more complicated architectures (such as MLPs with a larger number of parameters), the regularization does not show as much as an improvement as seen in figure 6. Along those lines, different datasets apart from CIFAR-10/MNIST should be considered.

(4) Comparing the above analysis to PAC-bayes frameworks. Seeing as how the notion of stability can be introduced by considering parameter senstivity, and PAC-bayes methods use only parameter senstivity to bound generalization, it would be interesting to see if using the above parameter sensitivity in a PAC-bayes frameworks leads to better bounds

10 Acknowledgments

I would like to thank Jeremy Bernstein for exploring and surveying the current state of generalization bounds with me throughout the term.

References

[1] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2018.

[2] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks. *Journal of Machine Learning Research*, 20(63):1–17, 2019.

[3] Jian Zhang. Generalization bound for infinite function class. *STAT 598Y Purdue Lecture Notes*, link: https://www.stat.purdue.edu/~jianzhan/STAT598Y/NOTES/slt06.pdf.

[4] Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[5] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *arXiv preprint arXiv:1707.09564*, 2017.

[6] Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

[7] Jeremy Bernstein, Arash Vahdat, Yisong Yue, and Ming-Yu Liu. On the distance between two neural networks and the stability of learning. *arXiv preprint arXiv:2002.03432*, 2020.

[8] Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of machine learning research*, 2(Mar):499–526, 2002.

[9] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[10] Yuichi Yoshida and Takeru Miyato. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

https://github.com/kushaltirumala/generalization_bound_emp

11 Appendix I

11.1 Leave-one-out-error bound

We discuss the slight discrepancy between the classical definition of leave-one-out-error, and the sample leave-one-out-error loss we used in section 4. Formally, pointwise hypothesis stability is

given by if for samples $S \in \mathbb{R}^k$ and all $i \in \{1, \ldots, m\}$:

$$\mathbb{E}_S[|L(f_S, z_i) - L(f_{S^{\setminus i}}, z_i)|] \leq \beta$$

However, note that we analyzed the $|L(f_S, S) - L(f_{S^{\setminus i}}, S^i)|$. To reconcile this, we note that for an L'-lipschitz function we have:

$$|L(f_S, S) - L(f_{S^{\setminus i}}, S^{\setminus i})| \leq L'|f_S(S) - f_{S^{\setminus i}}(S^{\setminus i})|$$

Then note that:

$$|L(f_S, z_i) - L(f_{S^{\setminus i}}, z_i)| \leq |L(f_S, S) - L(f_{S^{\setminus i}}, S^{\setminus i})|$$

since $z_i \in S$. So it clearly follows that:

$$|L(f_S, z_i) - L(f_{S^{\setminus i}}, z_i)| \leq L'(|f_S(S) - f_{S^{\setminus i}}(S)| + L'(|f_{S^{\setminus i}}(S) + f_{S^{\setminus i}}(S^{\setminus i})|)$$

where the RHS is exactly what we analyze in section 4. The desired result follows.

11.2 Tightening Generalization Bounds

From above, it is clear that what we actually analyze in section 4 is much stronger than what we need for pointwise hypothesis stability. In fact, we can react pointwise hypothesis stability purely as a parameter sensitivity problem since we have for all sample S and data points i that:

$$|L(f_S, z_i) - L(f_{S^{\setminus i}}, z_i)| \leq L'|f_S(z_i) - f_{S^{\setminus i}}(z_i)|$$

$$= L'(f(W + \delta W; z_i) - f(W; z_i)| \quad \leq L'\sqrt{t}\sqrt{G\left(\prod_{i=1}^{L}\left(1 + \frac{||\delta W_l||_F^2}{||W_l||_F^2}\right) - 1\right)}$$

where the t is the model output dimension, G is constant is the same as defined in section 4. δW here is the same difference in model weights due to training on S versues $S^{\setminus i}$. Using the same change matrix normalization trick, we can get a sample-agnostic bound of:

$$|L(f_S, z_i) - L(f_{S^{\setminus i}}, z_i) \leq L'\sqrt{t}\sqrt{G\left(\prod_{i=1}^{L}\left(1 + \frac{K}{||W_l||_F^2}\right) - 1\right)}$$

with K being the number of optimizer steps. Then we have a tighter generalization bound with probability $\geq 1 - \delta$

$$L_{true}(f) \leq L_{train}(f) + O\left(\frac{\prod_{i=1}^{L}\left(1 + \frac{K}{||W_i||_F^2}\right)^{1/4}}{(2m\delta)^{1/2}}\right)$$

where m is the number of data points, K is the number of updates iterations. However note that from empirical analysis, that there might not be good use for this bound (for empirical generalization). This may suggest that the slightly stronger notion of stability we introduced (by considering difference in loss from evaluating f_S on S and $f_{S^{\setminus i}}$ on $S^{\setminus i}$) might contain more useful information than pointwise hypothesis stability. This is discussed in the conclusion.