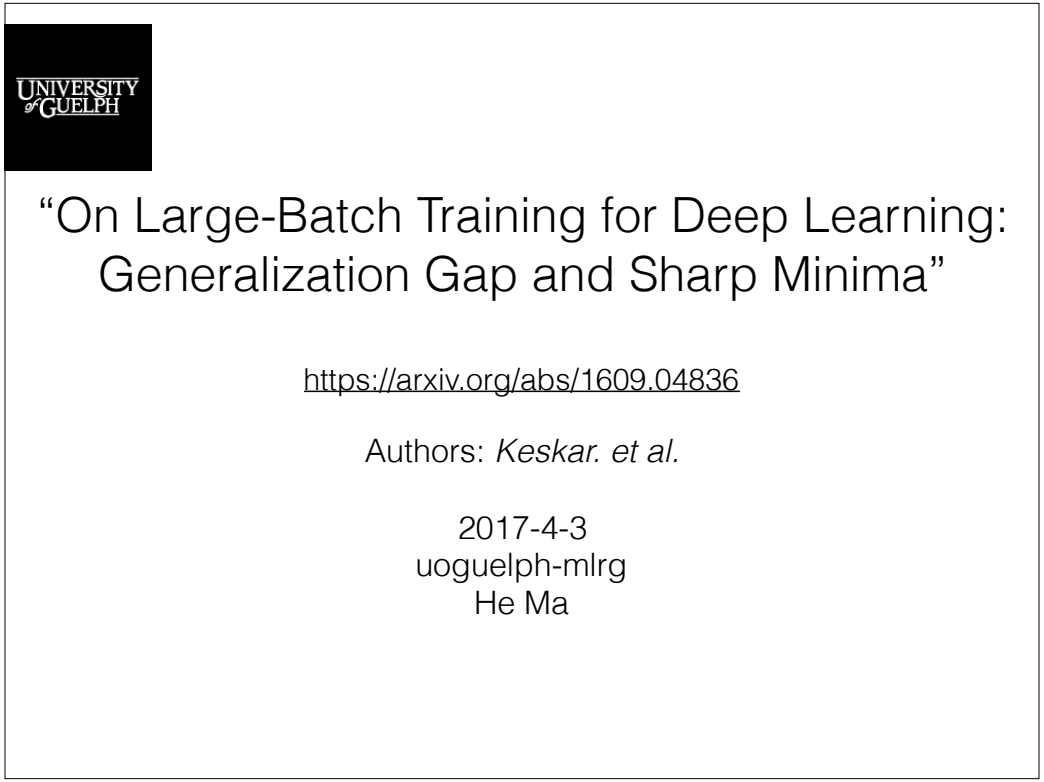


Northwestern University
Work was performed when author was an internship at Intel Corporation

There has been some interesting debates on the sharpness or flatness of local minima and their relation to generalization of a model.

This paper came out last fall was involved. This paper got accepted to ICLR 2017 as a conference paper.

The paper tries to empirically explain why LB SGD trained model generalize worse than SB counterpart. Although they didn't solve the problem, in fact they only showed some numerical evidence for their observation, but their work shed light on this problem which haven't been solved in recent years.

I'm interested in paper because in my experience of distributed training I also got this generalization gap. For example, when scaling AlexNet from one GPU to 8 GPUs with the same batch size, there will be consistently 1 or 2 percent accuracy drop on the validation set. I always wondered why.

Contents

My presentation will basically follow the process of thought of this paper.

I will first introduce the background knowledge about SGD. And then
I will show their experiment results and insight on why large batch SGD gives poor generalization.

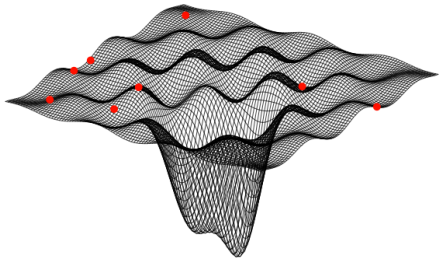
And finally I will wrap up and introduce another recent paper that go against the observation in this paper.

Introduction

- **Minimize the loss (objective)**

$$\min_{x \in \mathbb{R}^n} f(\mathbf{x}) := \frac{1}{M} \sum_{i=1}^{M} f_i(\mathbf{x})$$

$$\mathbf{x} = (x_0, x_1, ..., x_N)$$



credit:https://www.quora.com/Why-is-the-error-surface-convex-for-a-neural-network-that-uses-a-monotonic-activation-function

3

The paper starts by introducing the common routine in training neural networks is to find the best set of parameters x that can minimize the loss function f. they used mean cross entropy.

The loss function captures the deviation of the model prediction from the ground truth.

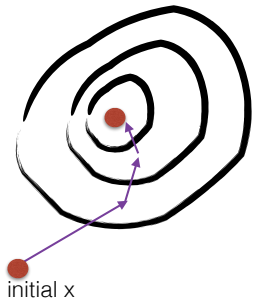
The loss function we are dealing with here is often high dimensional and non-convex. A two dimensional loss function example are shown here.

Introduction

- **Stochastic Gradient Descent**

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \left(\frac{1}{|B_k|} \sum_{i \in B_k} \nabla f_i(\mathbf{x}_k) \right)$$

$$B_k \subset \{1, 2, ..., M\}$$



- using the entire dataset (intractable or large memory cost) $B_k = \{1, 2, ..., M\}$
- using just one example each time (high variance) $B_1 = \{1\}, B_2 = \{2\}, ...$
- using a minibatch each time (best of both worlds) $B_1 = \{1, 2\}, B_2 = \{3, 4\}, ...$

4

To minimize the loss function, we often iteratively refine x in the parameter space by stochastic gradient descent or SGD over a number of training examples.

For example, over the entire training set that is when Bk={1,2,…,M}, or just one example (Bk=1, Bk=2, …) in the training set or a minibatch of examples in the training set (Bk={1,2}, Bk={3,4}).

Using the entire dataset tries to directly find out the solution with low noise by averaging out the noise though all the training examples. But it is often unpractical for large dataset. Because the computation involves an large summation.

Using just one example is another extreme is computationally easy to implement but introduces high variance.

Using a minibatch takes the best of both worlds

Typical batch size we use ranges from 32, 64 to 512. And minibatch training proved to be successful for a large number of applications.

Introduction

- **Some efforts to parallelize SGD:**
  - Dean et al., 2012; Zhang et al. 2015; Das et al., 2016
  - the speed-ups and scalability obtained are limited by small batch sizes
- **A major drawback of SGD:**
  - using large batch (**LB**) size leads to generalization gap (as high as as high as 5% even for small networks)

5

Some efforts were made to parallelize SGD training.

One intuitive way for improving parallelism is to increase the batch size |Bk|. This increases the computation per iteration that can be effective distributed and therefore improves speed-up. However, it is observed in several paper, this also leads to a loss in generalization.

So those efforts are all limited to small batch training.

Experiments and Insights

- main observation
- relation to overfitting
- experiment

6

Now I’m going to show some experiments and insights on why this is happening.

Experiments and Insights

- **Main observation**:

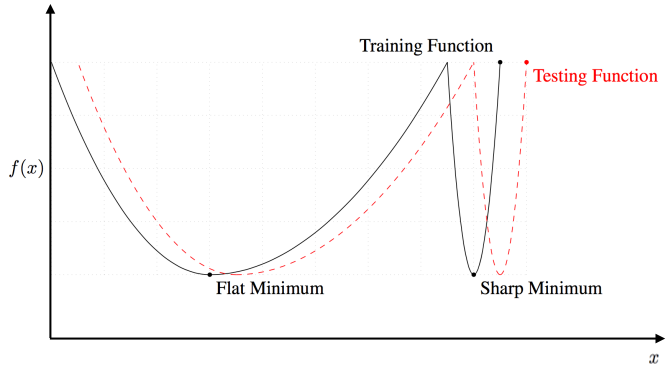
  1. LB methods are attracted to sharp minima
  2. Sharp minima tend to generalize worse.
  3. these minima are close to the initialization and LB methods are unable to escape from them.

7

The author did some experiments to support all three observations.

Experiments and Insights

- **The concept of sharp and flat minima**



8

The authors first show some insights on why sharp minima tend to generalize worse.

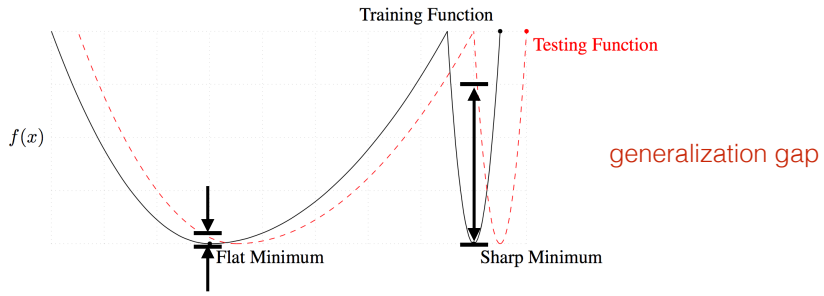
The concept of sharp and flat minima can be seen in this figure. where the Y-axis indicates value of the loss function and the X-axis the variables (parameters)

The solid black line is the loss function on the training set.
The dotted red line is the loss function on test set.

So when the model is optimized to minimize the training function. There are two local minima, a sharp one and a flat one. From this figure, we obviously want the x to be optimized to the flat one. Because it gives smaller test loss or generalize better.

Experiments and Insights

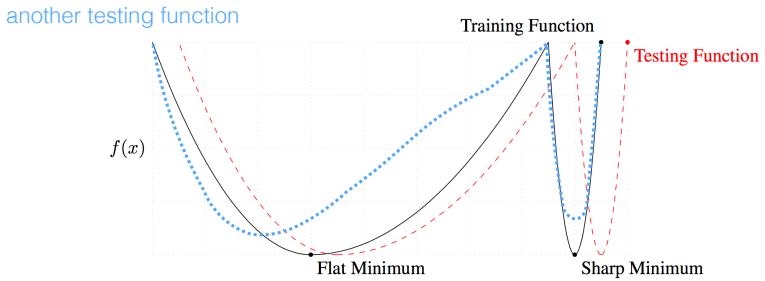
- **The concept of sharp and flat minima**



9

Experiments and Insights

- **Relation to over-fitting**



10

The author also points out that:
The generalization gap is different from the concept of overfitting.

Overfitting is between training set and test set.

LB generalization gap is between different local minima.

For example, if we have another test set, and the flat minima overfits the training sets because it should rather stop earlier.

Experiments and Insights

- **Numerical Experiments**:
  - experiment setup
  - visualization of LB minima vs. SB minima
  - heuristic metric of sharpness

11

The authors then try to support their observations by showing some experiment results.

Before talking about the experiment,
I will first go through their experiment setup, visualization of two kinds of minima and their way of measuring sharpness of minima.

Experiments and Insights

- **Experiment setup**:

Table 1: Network Configurations

Name	Network Type	Architecture	Data set
F_1	Fully Connected	Section B.1	MNIST (LeCun et al., 1998a)
F_2	Fully Connected	Section B.2	TIMIT (Garofolo et al., 1993)
C_1	(Shallow) Convolutional	Section B.3	CIFAR-10 (Krizhevsky & Hinton, 2009)
C_2	(Deep) Convolutional	Section B.4	CIFAR-10
C_3	(Shallow) Convolutional	Section B.3	CIFAR-100 (Krizhevsky & Hinton, 2009)
C_4	(Deep) Convolutional	Section B.4	CIFAR-100

Table 2: Performance of small-batch (SB) and large-batch (LB) variants of ADAM on the 6 networks listed in Table 1

	Training Accuracy		Testing Accuracy	
Name	SB	LB	SB	LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
C_2	$99.99\% \pm 0.04\%$	$99.99\% \pm 0.01\%$	$89.24\% \pm 0.12\%$	$87.26\% \pm 0.07\%$
C_3	$99.56\% \pm 0.44\%$	$99.88\% \pm 0.30\%$	$49.58\% \pm 0.39\%$	$46.45\% \pm 0.43\%$
C_4	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

12

The experiments were based on the snapshot parameters generated from 6 neural networks.

The baseline training and testing accuracy was shown in table 2.

Experiments and Insights

- **Experiment setup**:

Table 1: Network Configurations

Name	Network Type	Architecture	Data set
F_1	Fully Connected	Section B.1	MNIST (LeCun et al., 1998a)
F_2	Fully Connected	Section B.2	TIMIT (Garofolo et al., 1993)
C_1	(Shallow) Convolutional	Section B.3	CIFAR-10 (Krizhevsky & Hinton, 2009)
C_2	(Deep) Convolutional	Section B.4	CIFAR-10
C_3	(Shallow) Convolutional	Section B.3	CIFAR-100 (Krizhevsky & Hinton, 2009)
C_4	(Deep) Convolutional	Section B.4	CIFAR-100

Table 2: Performance of small-batch (SB) and large-batch (LB) variants of ADAM on the 6 networks listed in Table 1

	Training Accuracy		Testing Accuracy	
Name	SB	LB	SB	LB
F_1	$99.66\% \pm 0.05\%$	$99.92\% \pm 0.01\%$	$98.03\% \pm 0.07\%$	$97.81\% \pm 0.07\%$
F_2	$99.99\% \pm 0.03\%$	$98.35\% \pm 2.08\%$	$64.02\% \pm 0.2\%$	$59.45\% \pm 1.05\%$
C_1	$99.89\% \pm 0.02\%$	$99.66\% \pm 0.2\%$	$80.04\% \pm 0.12\%$	$77.26\% \pm 0.42\%$
C_2	$99.99\% \pm 0.04\%$	$99.99\% \pm 0.01\%$	$89.24\% \pm 0.12\%$	$87.26\% \pm 0.07\%$
C_3	$99.56\% \pm 0.44\%$	$99.88\% \pm 0.30\%$	$49.58\% \pm 0.39\%$	$46.45\% \pm 0.43\%$
C_4	$99.10\% \pm 1.23\%$	$99.57\% \pm 1.84\%$	$63.08\% \pm 0.5\%$	$57.81\% \pm 0.17\%$

SB=256
LB=10%

13

The experiments were based on the snapshot parameters generate from 5 neural networks.

The baseline training and testing accuracy was shown in table 2.

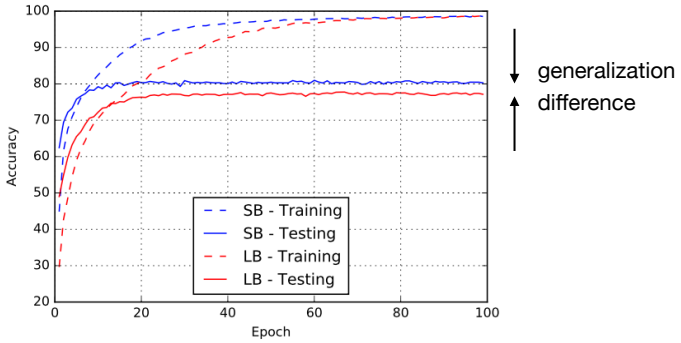
specifically, let's focus on the C1 which is the AlexNet on Cifar10. Conv-Pool, Conv-Pool, FC, Dropout, FC, Dropout, Softmax

Notice the C1 network training and testing accuracy. We see that the SB and LB trained snapshot has almost the same training accuracy. But there's 3% difference in testing accuracy.

Small batch size=256
Large batch size=10%

Experiments and Insights

- **Experiment setup**:



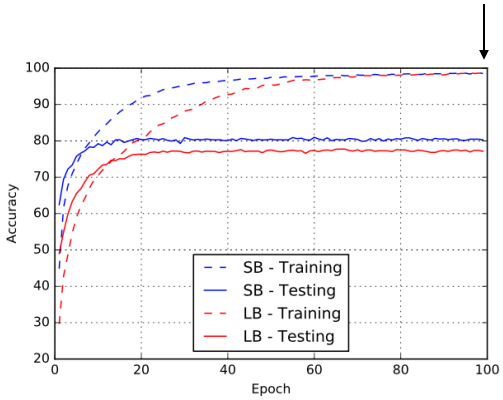
(b) Network C_1

14

We specific look at C1 for example. We can see there's the generalization difference.

Experiments and Insights

- **Experiment setup**:



snapshot SB $\mathbf{x}_s^*$

snapshot LB $\mathbf{x}_l^*$

(b) Network C_1

15

The author take the snapshot at the end of training.

Experiments and Insights

- **Visualization of LB minimizer vs SB minimizer**
  - interpolation and extrapolation on LB and SB snapshots

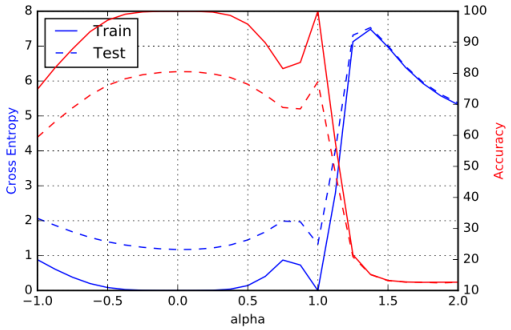
$$f(\alpha \mathbf{x}_l^* + (1-\alpha)\mathbf{x}_s^*)$$
$$\alpha \in [-1, 2]$$

16

To find out why there's a generalization difference, they visualize the difference of two solutions, by interpolating between the two snapshots when alpha is within (0,1) . they also extrapolate near each snapshot. That is when alpha is from -1 to 0 and from 1 to 2.

Experiments and Insights

- **Visualization of LB minimizer vs SB minimizer**
  - interpolation and extrapolation on LB and SB snapshots



(c) C_1

17

Just look at the red lines.

When the linear combination is close to each snapshot, which is alpha=0 and alpha=1, the training accuracy starts from 99 percent as we expect.

As soon as the combination goes far from the LB snapshot, which is alpha=1, the accuracy drops fast.

However, the accuracy drops slowly for the SB case.

Experiments and Insights

- **Sharpness of minima**

$$\phi_{\mathbf{x},f}(\epsilon, A) := \frac{(\max_{y \in \mathbb{C}_\epsilon} f(\mathbf{x} + Ay)) - f(\mathbf{x})}{1 + f(\mathbf{x})} \times 100$$

$$\mathbf{x} \in \mathbb{R}^n$$
$$y \in \mathbb{R}^p$$
$$A \in \mathbb{R}^{n \times p}$$

When p=N, this function explore the maximum loss in the full parameter space of x.

18

To better quantitatively describe the sharpness, the author proposed a metric to measure it and correlates to the generalization ability.

x is the N-dimension vector representing the parameter snapshot
y is a p-dimension vector

Since adjusting x in N-dimension, getting a solution will be expensive.

A is a N by p dimension matrix, mapping y from p dimension to N dimension.

So a small adjustment on y in p-dimension will results in a small adjustment in N dimension

they want to measure how this small adjustment around the original parameter x will result in an increase in the loss function f.

They defined the p-dimension space C to constraint y to be small. Within this space C, they are trying to find the y that can give the maximum loss. We can imagine the loss will be found on the edge of this space. That is the largest adjustment to x.

Experiments and Insights

- **Sharpness of minima**

Table 3: Sharpness of Minima in Full Space; ϵ is defined in (3).

	$\epsilon = 10^{-3}$		$\epsilon = 5 \cdot 10^{-4}$	
	SB	LB	SB	LB
F_1	1.23 ± 0.83	205.14 ± 69.52	0.61 ± 0.27	42.90 ± 17.14
F_2	1.39 ± 0.02	310.64 ± 38.46	0.90 ± 0.05	93.15 ± 6.81
C_1	28.58 ± 3.13	707.23 ± 43.04	7.08 ± 0.88	227.31 ± 23.23
C_2	8.68 ± 1.32	925.32 ± 38.29	2.07 ± 0.86	175.31 ± 18.28
C_3	29.85 ± 5.98	258.75 ± 8.96	8.56 ± 0.99	105.11 ± 13.22
C_4	12.83 ± 3.84	421.84 ± 36.97	4.07 ± 0.87	109.35 ± 16.57

p=N

Table 4: Sharpness of Minima in Random Subspaces of Dimension 100

	$\epsilon = 10^{-3}$		$\epsilon = 5 \cdot 10^{-4}$	
	SB	LB	SB	LB
F_1	0.11 ± 0.00	9.22 ± 0.56	0.05 ± 0.00	9.17 ± 0.14
F_2	0.29 ± 0.02	23.63 ± 0.54	0.05 ± 0.00	6.28 ± 0.19
C_1	2.18 ± 0.23	137.25 ± 21.60	0.71 ± 0.15	29.50 ± 7.48
C_2	0.95 ± 0.34	25.09 ± 2.61	0.31 ± 0.08	5.82 ± 0.52
C_3	17.02 ± 2.20	236.03 ± 31.26	4.03 ± 1.45	86.96 ± 27.39
C_4	6.05 ± 1.13	72.99 ± 10.96	1.89 ± 0.33	19.85 ± 4.12

p=100

19

They did some experiements to show their metric is valid and reflects what they observed in previous figure.

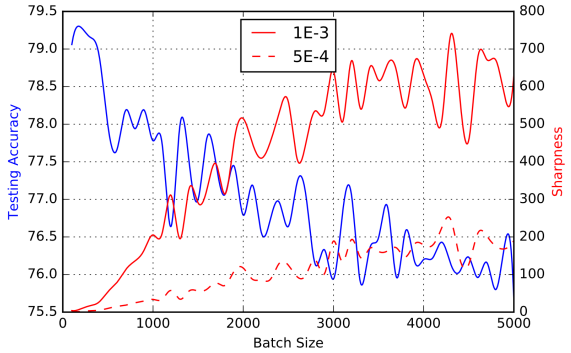
Table 3 and 4 shows in all cases, sharpness LB>SB by an order of magnitude.

////
sharp minimizers identified in our experiments do not resemble a cone

i.e., the function does not increase rapidly along all (or even most) directions. In fact, only 5% neighbourhood directions are steep.

Experiments and Insights

- **Sharpness negatively correlates to testing accuracy**



(b) C_1

20

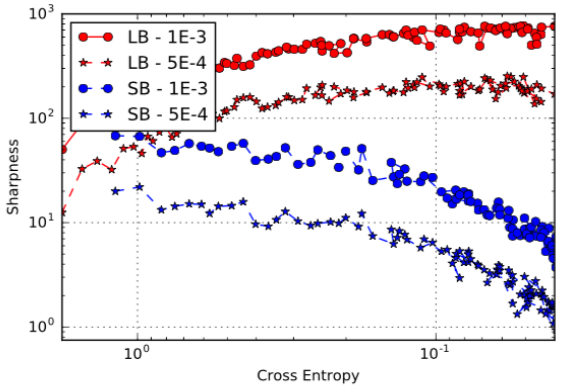
To show that their sharpness measure correlates to generalization loss

The larger batch size used, the worse testing accuracy, and the higher the sharpness.

It seems there's a negative correlation between sharpness and testing accuracy.

Insights and Experiment

- **Sharpness-Loss (LB vs SB)**



(b) C_1

21

To further strengthen their point…

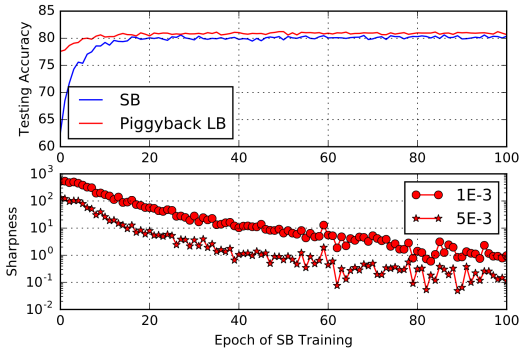
For one random trial of C1, larger cross entropy, the LB and SB yields similar sharpness, however when the mode is being optimized through epochs, the LB minima sharpness stays high.

This is test cross entropy.

This experiments tries to support the observation: LB training are attracted to sharp minima.

Experiments and Insights

- **Piggyback (warm-starting) experiment**



(b) C_1

22

Note that when warm-started with only a few initial epochs, the LB method does not yield a generalization improvement.

after certain number of epochs of warm-starting, the accuracy improves and sharpness of the large-batch iterates drop

It has been speculated that LB methods tend to be attracted to minimizers close to the starting point x0, whereas SB methods move away and locate minimizers that are farther away

Discussion

- **Contribution of the paper:**
  - Empirical conjecture:
    - “generalization gap is correlated with a marked sharpness of the minimizers obtained by large-batch (**LB**) methods.”
  - large number of numerical evidence

23

Let me summarize the contribution of the paper.

Discussion

- In summary
  1. Large Batch training converges to sharp minima
  2. Sharp minima tend to generalize worse
- Hessian based measure can explain (1)
- Lacking theoretical proof for (2), only numerical evidence.

24

We can explain the first observation theoretically using Hessian based measure.

For example, the parabola is sharper when the norm of hessian is larger.

A parabola surface is sharper when the norm of the hessian is larger.

However, it is difficult to theoretically prove the second observation.

Discussion

- This is only a conjecture. [1]

[1] Dinh et al. “Sharp Minima Can Generalize For Deep Nets”. March 2017

25

And in fact, this is only a conjecture mentioned by a recent paper called: Sharp minima can generalize.

This paper does not agree with the idea that sharp minima generalize worse. They say that sharp minima can generalize as well as flat minima.

So the sharpness really doesn’t matter. We can’t just use sharpness to measure the generalization.

Discussion

- Rethinking what sharpness actually means.[1]

$$\frac{\max_{\theta' \in B_2(\epsilon,\theta)} \big(L(\theta') - L(\theta)\big)}{1 + L(\theta)} \quad \approx \quad \frac{\big\|\big|(\nabla^2 L)(\theta)\big|\big\|_2 \epsilon^2}{2\big(1 + L(\theta)\big)}$$

- But is this sharpness measure really related to generalization?

[1] Dinh et al. “Sharp Minima Can Generalize For Deep Nets”. March 2017

26

The new paper agrees that the sharpness of the loss at a critical minimum is related to the eigenvalues of a Hessian. And this paper goes a step further to approximates the formula to be the spectral norm of the Hessian at this critical minimum.

However, if this metric really relates the the generalization is still questionable? Because one can easily give an opposite example and increase the norm of hessian without changing the behaviour of the function.

Discussion

- Alpha scale transformation:

$$T_\alpha : (\theta_1, \theta_2) \mapsto (\alpha\theta_1, \alpha^{-1}\theta_2)$$

$$L(\theta_1, \theta_2) = L(\alpha\theta_1, \alpha^{-1}\theta_2)$$

- By picking alpha:

$$\left\| \left| (\nabla^2 L)\big(T_\alpha(\theta)\big) \right| \right\|_2 \geq M$$

[1] Dinh et al. “Sharp Minima Can Generalize For Deep Nets”. March 2017

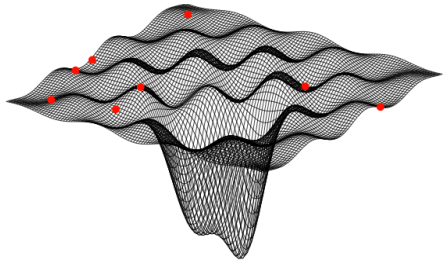
27

The paper gives an example, for a rectified two layer MLP, theta1 and theta2 are the params of the two layers.

By picking alpha, we can get arbitrarily large spectral norm. which indicates arbitrarily sharp minimum. And the function behave exactly the same as before due to the alpha transformation.

Discussion

- The problem remains.



credit:https://www.quora.com/Why-is-the-error-surface-convex-for-a-neural-network-that-uses-a-monotonic-activation-function

28

From two papers of sharpness and generalization, we can see that this seemingly basic question of (why LB ..) is actually a deep question.

Although the correlation theoretically proves to be wrong, the experiments result in the LB paper is still valuable as it consistently suggests something is different when using LB training and the problem of LB training remains. We just need to find a better metric to describe it.

If we find a metric that better describes it, we are closer to explain why LB gives poor results.