Train longer, generalize better: closing the generalization gap in large batch training of neural networks

The problems

- what is the **generalization gap**?
  there is a persistent degradation in generalization performance when using large batch size.
- generalizaiton gap stems from **the relatively small number of updates rather than the batch size**.

Theoretical Analysis

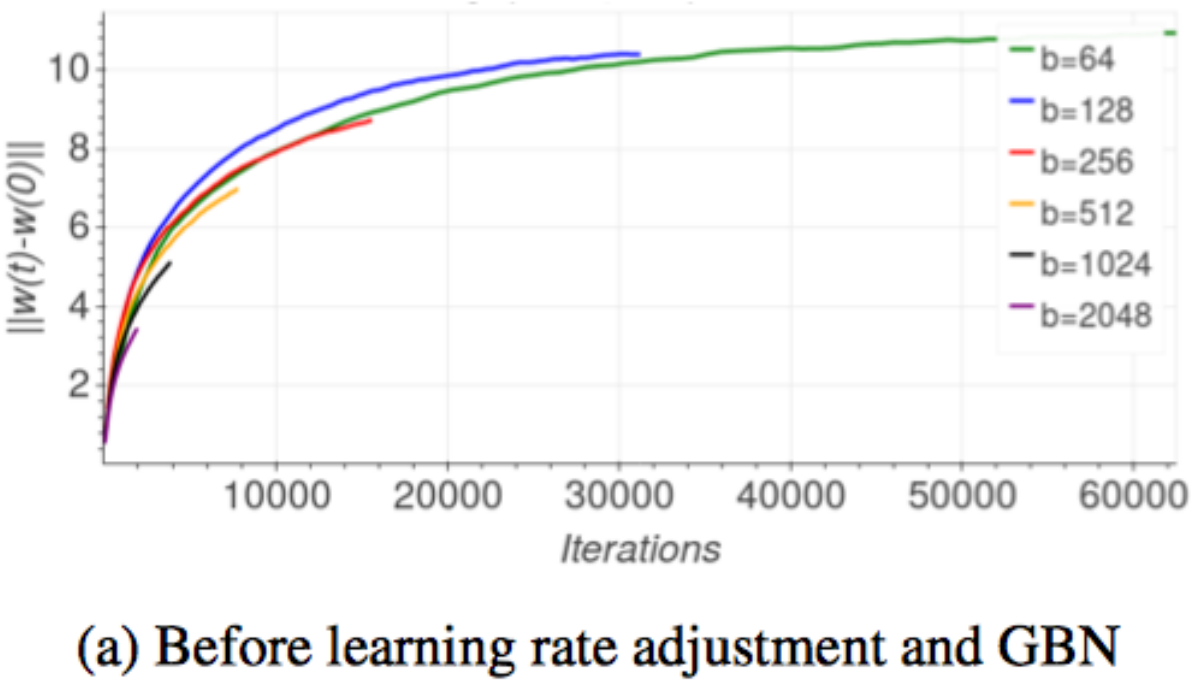
- For physical intuition, one can think of the *weight vector* $\mathbf{w}_t$ as: **a particle performing a random walk on the loss ("potential") landscape** $L(\mathbf{w_t})$.
- **ultra-slow diffusion**: the asymptotic behavior of the auto-covariance of the random potential: $\mathbb{E}(L(\mathbf{w}_1)L(\mathbf{w}_2)) \sim \| \mathbf{w}_1 - \mathbf{w}_2 \|^2$, determines the asymptotic behavior of the random walker:

$$\mathbb{E} \| \mathbf{w}_t - \mathbf{w_0} \|^2 \sim (\log \mathrm{t}^{\frac{4}{\alpha}})$$

- typically: $d \triangleq \| \mathbf{w}_t - \mathbf{w}_0 \| \sim (\log t)^{\frac{2}{\alpha}}$, where $\| \cdot \|$ is the Euclidean distance of two vector.
  - to climb (or go around) each barrier takes exponentially long time in the heigh of the barrier: $t \sim \exp(d^{\frac{\alpha}{2}})$

Empirical results and implications

The below figure examines $\| \mathbf{w}_t - \mathbf{w}_0 \|$ during the initial training phrase:



(a) Before learning rate adjustment and GBN

Figure 2(a) from the paper.

> In Figure 2 (in the paper), all the models are trained for a constant number of epochs, smaller batch sizes entail more training iteration in total.

Foundings in the paper:

- Weight distance from initial training phrase increases logarithmically with the number of training iterations (weight updates): $\| \mathbf{w}_t - \mathbf{w}_0 \| \sim \log(t)$.
- A very similar logarithmic graph is observed for all batch sizes, however:
  1. Each batch size seems to have a different slop, indicating a somewhat **differnet different diffusion rate for different batch sizes**.
  2. Weight distance reached at the end of the initial learning phrase are significantly different for different batch sizes.

The findings lead to the flowing informal argument:

- To reach a minima of "width" d the weight vector $\mathbf{w}_t$ has to travel at least a distance d.
- Thus, to reach wide ("flat") minima, we need to have the highest possible diffusion rates and a large number of training iterations.

Solutions given in this paper

- Learning rate tuning (LB+LR): Using a large batch, while adapting the learning rate to be larger so that $\eta_L = \sqrt{\frac{|B_L|}{|B_S|}}\eta_S$ where η_S is the original learning rate used for small batch, η_L is the adapted learning rate and $|B_L|, |B_S|$ are the large and small batch sizes, respectively.
- Ghost batch norm (LB+LR+GBN): Additionally using the "Ghost batch normalization" method in our training procedure. The "ghost batch size" used is 128.
- Regime adaptation: Using the tuned learning rate as well as ghost batch-norm, but with an adapted training regime. The training regime is modified to have the same number of iterations for each batch size used - effectively multiplying the number of epochs by the relative size of the large batch.

Solutions given in this paper.

Does "Regime adaption" means: to achieve the same learning performance when using a large batch size, a longer time is required for training. If so, what is the advantage of large batch training.

Conclusions in this paper

- Even though small batch updates still posses a desirable property of convergence to solutions with good generalizaiton, there is no fundamental issues with big batch updates.
- A certain amount of SGD updates is required to reach a good solution. By contrast, less computations are required when the number of samples is small in each batch.
- Good generalization can result from extensive amount of gradient updates in which there is no apparent validation error change and training error continues to drop.

Move our attentions from the "generalization gap" to a more modest "computational gap" where we desire that big-batches will improve optimization time without incurring large computional overhead.