

Optimization

Vladimir Feinberg

July 17, 2017

This document contains notes on optimization of ANNs; i.e., the ERM task for a prescribed family of neural networks. Content is mostly from [Goodfellow's Deep Learning Book](#). In the optimization chapter Dr. Goodfellow warns that over the past 30 years of machine learning advances in training speed have mostly been made from making a net that's easier to optimize, not by focusing on the optimization algorithm.

1 Setting

To cover a broad NN setting (both FFNNs, RNNs, and unsupervised networks), we assume that we wish to minimize empirical risk:

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m L(z^{(i)}, \boldsymbol{\theta}) + \Omega(\boldsymbol{\theta})$$

Above, L is a fixed, differentiable loss function, as is the regularizer Ω , which is also assumed to be one of several canonical forms. L is dependent on an ANN, f , a real-valued circuit. Altogether this and backpropagation implies that we have efficient evaluation of f and ∇f through backprop, as well as some second-order information if desired through Hessian-vector products $(\nabla^2 f) \mathbf{v} = \nabla(\mathbf{v}^\top \nabla f)$ ([Christianson 1992](#)). Loss is dependent on examples $z^{(i)}$, whatever they may be in our supervised or unsupervised use case.

2 Stochastic Gradients

Since J above decomposes easily, we frequently use a stochastic gradient with $B \ll m$ examples,

$$\frac{1}{B} \sum_{i=1}^B \nabla_{\boldsymbol{\theta}} L(z^{(\sigma_i)}, \boldsymbol{\theta}),$$

for random samples σ_i uniformly taken from the dataset. Using this for stochastic gradient descent (SGD) as opposed to full gradient descent (GD) enables more efficient derivatives.

Frequently, this random sampling is performed without replacement, by shuffling the dataset once and then performing sequential reads on the data. This is not the same as an independent sample with replacement from the dataset, as is typically assumed. For analysis see [Shamir 2016](#) and [Wright 2015](#).

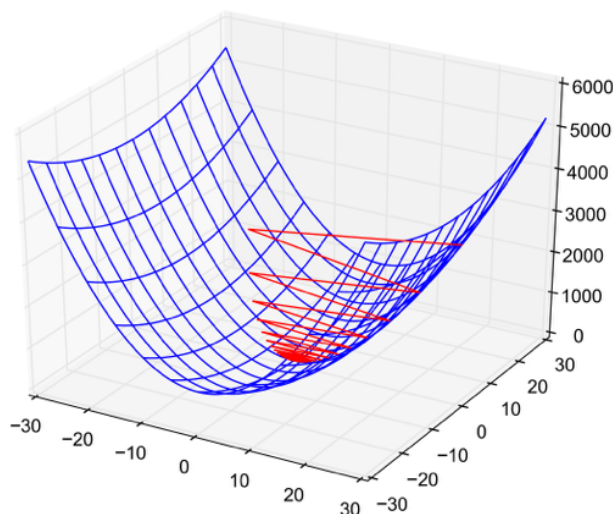
3 Conditioning

An ill-conditioned Hessian can result in poor optimization (Fig. 1). At any given step a twice-differentiable loss will have the following improvement with local Hessian H and gradient \mathbf{g} assuming step size ϵ :

$$\epsilon^2 \mathbf{g}^\top H \mathbf{g} - \epsilon \mathbf{g}^\top \mathbf{g}$$

Monitoring both of the terms above (which can be done efficiently) during training can help with identifying training issues.

Poor conditioning



(Goodfellow 2015)

Figure 1: From Dr. Goodfellow’s DL book lecture notes, Chapter 8.

4 Non-convexity

J is almost always non-convex in the NN case. This implies that local minima (LM) reached are not necessarily global. However, a global minimum (GM) is not necessary: just a LM close enough to it.

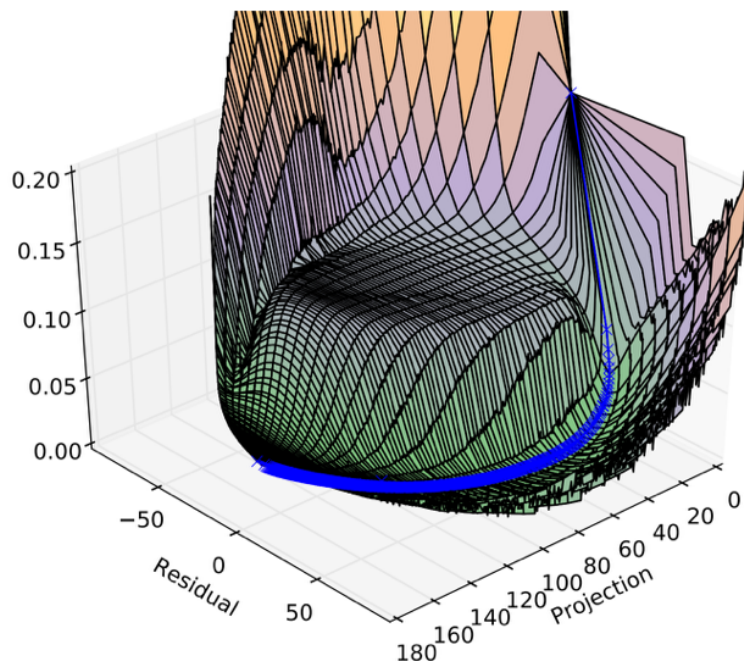
LMs, however, may not always be the root of the problem. Many LMs are induced by symmetry (highly connected nets have combinatorially many symmetries). Recent research shows that LMs concentrate around the GM ([Choromanska et al 2014](#)), so many of these LMs are equivalent to a good solution anyway. Goodfellow claims that most of the time (thanks to early stopping, for instance), most ANN optimizations don’t arrive at a critical point of any kind.

A more pertinent problem is that of saddle points (which tend to be much more populous than LMs): locations where the gradient vanishes but the Hessian has both positive and negative eigenvalues ([Dauphin et al 2014](#)). The issue is that along one direction (the linear subspace of positive eigenvectors), the saddle appears to be a LM, and it can take many steps to get close enough to the bottom of the saddle to locate the direction of negative eigenvalue in the Hessian (Fig. 2).

Wide, flat objective regions, “plateaus,” are another threat: one that is difficult to circumvent altogether. Momentum is an approach to solve some versions of this problem.

Cliffs may cause a misstep (onto the cliff) to cause a far overshoot, sending the parameters careening very far away (Figure 3). This is resolved with gradient clipping.

3-D Visualization of MP-DBM



(Goodfellow 2015)

Figure 2: From Dr. Goodfellow's DL book lecture notes, Chapter 8.

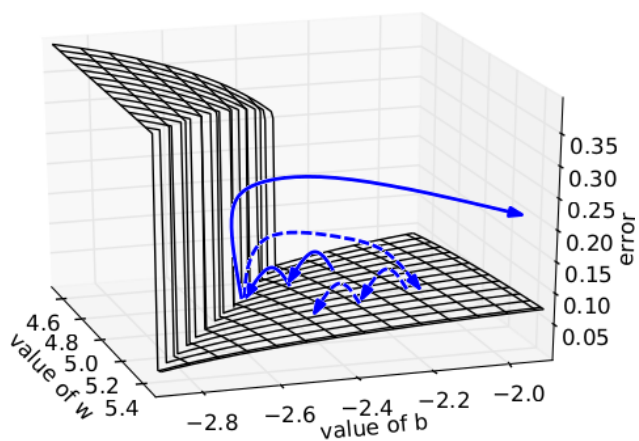


Figure 3: Image from Figure 6 of [Pascanu et al 2013](#) demonstrating how cliffs or valleys in optimization landscapes may cause missteps corrected by gradient clipping.

5 Optimization Methods

See my [blog post](#).

6 Techniques

Many techniques have been developed to make ANN optimization easier. Many of these are based on finding

6.1 Batch Size

([Li et al 2014](#)). Note that variance in the gradient is reduced sublinearly in batch size (from the standard error of the mean formula). However, very small batches may damage gradient accuracy to the point where noise dominates gradient signal at the end of training (though, with early stopping, this may not be a large problem)—a common strategy is to increase batch size during training. Second-order methods may require larger batches since the inverse Hessian is sensitive to fluctuation.

6.2 Batch Normalization (BN)

([Ioffe and Szegedy](#), [Cooijmans et al 2017](#)). Goodfellow draws BN inspiration from mitigation of higher-order affects: suppose we have a d -layer-deep network with 1 input neuron and 1 hidden layer per unit with no activation, so that $y = x \prod_{i=1}^d w_i$. A k -th order partial derivative on the loss between variables, say, $\{w_i\}_{i=1}^k$ will have, for a step size η , an effect size proportional to $\eta^k \prod_{i=k+1}^d |w_i|$ according to Taylor series. Then for small k and $w_i > 1$ we may find large noise in SGD steps due to an exponential dependence on d . To reduce this noise, we can standardize the activations of each layer according to the minibatch mean and variance for each hidden unit activation. At test time, a whole-training averaged mean and variance vector for every BN layer is used. Somewhat counterintuitively, a learned layer mean shift and variance shift can be added (see [backprop notes](#) for BN formulation). This approach is fairly general and has seen use in RNN setting as well.

6.3 FitNets

([Chen et al 2016](#), [Romero et al 2015](#)). FitNets and network re-use techniques can enable faster training by teaching a larger network to replicate the outputs of the layers in a smaller network. This kind of bootstrapping lets larger networks find useful representations for prediction. The smaller network should be short and wide, whereas the large one should be deep and narrow. In this way we can force some kind of compression of the knowledge of the shallow network.

6.4 Polyak Averaging

([Xu 2011](#)). For a sequence of iterates during training $\{\theta_i\}_{i=1}^t$, Polyak averaging can be viewed as a regularization method inspired by convex optimization. While in the convex case an arithmetic average is preferred, Goodfellow recommends an exponential average for the convex case $\bar{\theta}_i = \alpha \bar{\theta}_{i-1} + (1 - \alpha) \theta_i$.

6.5 Curriculum learning

([Zaremba and Sutskever 2014](#)). A stochastic curriculum of training examples generally ordered from easy to hard but with some mixing has proven essentially in some hard-to-learn RNNs, as compared to minibatch SGD over the entire dataset).

6.6 Weight Normalization (WN)

([Salimans and Kingma 2016](#)). Inspired by BN’s reparameterization approach to solving the problem of pathological curvature, weight normalization suggests parameterizing each neuron’s weight \mathbf{w} as $\mathbf{w} = \frac{g}{\|\mathbf{v}\|} \mathbf{v}$ where g, \mathbf{v} are free parameters. This decoupling forces gradient updates to \mathbf{v} to be orthogonal to the current value of $\mathbf{w} \parallel \mathbf{v}$. The orthogonality increases the norm of \mathbf{v} , and this effect is exacerbated by noisy gradients, which creates a self-stabilizing effect since then \mathbf{v} will be more resistant to change. Moreover, the authors find empirically that the noise in the gradient is frequently parallel to \mathbf{w} , so projecting it away helps improve learning robustness. WN coupled with mean-only BN improves on a variety of FFNN and RNN tasks. Moreover, WN is a much lighter addition than BN.

[Reddit overview of normalization techniques](#).