

Optimization

Vladimir Feinberg

July 1, 2017

This document contains notes on optimization of DNNs; 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, which, in addition to deep neural networks (DNN) includes recurrent neural networks (RNN), we assume that we wish to minimize empirical risk:

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m L\left(f\left(\mathbf{x}^{(i)}, \boldsymbol{\theta}\right), y^{(i)}\right) + \Omega(\boldsymbol{\theta})$$

Above, L is a fixed, differentiable loss function, as is the regularizer Ω , which is also assumed to be one of several canonical forms. Next, f is a neural network, and therefore 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](#)).

2 Stochastic Gradients

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

$$\frac{1}{B} \sum_{i=1}^B \nabla_{\boldsymbol{\theta}} L\left(f\left(\mathbf{x}^{(\sigma_i)}, \boldsymbol{\theta}\right), y^{(i)}\right),$$

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.

3 Conditioning

An ill-conditioned Hessian can result in poor optimization. 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.

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 DNN 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.

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 [TODO figure 6 from [Pascanu et al 2013](#)]. This is resolved with gradient clipping.

5 Initialization

Parameters should not be initialized uniformly so as to break symmetry between hidden units; otherwise, the parameters will remain the same during training. This motivates random initialization. Biases, however, are typically constants, heuristically chosen.

Weights are typically Gaussian or uniform, but scale is important. Large scale, which breaks symmetry strongly, must be balanced with stability of the learning algorithm. Heuristics do exist:

1. ReLU activations, since they're linear, suggest that the scale of the output for a basic ReLU MLP with $\|y\| \sim \|\mathbf{x}\| \prod_i \|W^{(i)}\|$. In general, the scale of weights should be treated as a hyperparameter. Importantly, loss (such as softmax) not scale-invariant; keep an eye out for initialization.
2. ([Glorot and Bengio 2010](#)). Normed initialization suggests that a fully connected layer initializes an $n \times m$ weight matrix entrywise with Uniform $(-u, u)$ where $u = \sqrt{\frac{6}{m+n}}$. This approach intends to keep gradient variance the same across layers.
3. ([Saxe et al 2013](#), [Susillo 2014](#)). Orthogonal matrix initialization with a scaling factor dependent on layer activation is motivated by making hidden units track a varied set of functions, resulting in training time independent of depth (if nonlinearities are the identity).
4. Output layer bias should be initialized to the marginal statistics of observations.
5. Certain initialization schemes may be compatible with positive constant bias initialization, which lets gradients start on the linear part of a ReLU.
6. ([Jozefowicz et al 2015](#)). If a unit acts as a control gate for letting other information pass (like a forget gate in an LSTM), then bias should be initialized as positive there.

6 Optimization Methods

See my [blog post](#).

7 Techniques

Batch Size. 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. [TODO 8.7 DL Book]

batch normalization vs initialization ([Mishkin and Matas 2015](#))

Polyak averaging (<https://arxiv.org/abs/1107.2490>)

skip coordinate descent

network reuse (“Net2Net: Accelerating Learning via Knowledge Transfer.”)

Greedy pre-training

simultaneous full-depth training (find better name): attaching MLP heads in the middle of an architecture

Curriculum learning (*skip* continuation)

8 Cost Function Shape

TODO: link to goodfellow SGD slides + 2015 paper (copy them in) “Qualitatively Characterizing Neural Network Optimization Problems,” Goodfellow, Vinyals and Saxe, ICLR 2015