An Empirical Investigation of Second-order methods for Neural Network Optimization

Mike Kayser

April 3, 2015

1 Introduction

Neural network optimization is a well-studied topic. It is widely believed that simple first-order methods (e.g., stochastic gradient descent or minibatch gradient descent, or variants such as AdaGrad [?] or momentum methods [?]) do at least as well as more powerful second-order methods. This is in contrast to many other optimization regimes, where gradient descent is considered a poor technique.

There are many reasons why neural networks might indeed be a "special case." One big difference is that for typical nonlinear regression problems, neural networks represent a *stochastic optimization problem*, one in which the objective function is a simple average of many (perhaps thousands or millions) of noisy sub-objectives. In particular, the loss which we are attempting to minimize has the least-squares form:

$$\begin{aligned} L(\theta, X, y) &= \frac{1}{n}\sum_{i=1}^{n} \ell(\theta, X^{(i)}, y^{(i)}) \\ &= \frac{1}{n}\sum_{i=1}^{n} (f(\theta, X^{(i)}) - y^{(i)})^2 \end{aligned}$$

where $X = \{X^{(i)} | 1 \leq i \leq n\}$ is the set of *inputs*, $y = \{y^{(i)} | 1 \leq i \leq n\}$ are the corresponding *outputs*, and $f(\theta, x)$ represents the output of the neural network with parameters θ, when taking input x.

This stochastic regime is notable for two reasons. First, *noisy gradients can be cheaply computed on small subsets of the data.* This implies that progress can be made without computing gradients on the full objective. Second, and relatedly, *the use of noisy gradients provides a possible way to escape stationary points.* This is because a, at a point x_k where the full gradient g_k is close to zero, there may well be one or several sub-objectives for which the noisy estimated gradient $g_k^{(i)}$ is not nearly zero. Note that this reasoning does not imply that stochastic gradient descent can easily avoid the problem of long valleys of low curvature, although momentum methods try to address this second problem.

In practice, the first of these considerations means that the humble *stochastic gradient method* (SGD) is in fact very very hard to beat, despite its theoretical shortcomings. In this work we implement the recently proposed *Hessian-free* optimization technique in an attempt to beat SGD.

2 Problem statement

The CHiME challenge [?] is a competition affiliated with a workshop at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). We use the noisy speech recognition dataset from Track 1 of the 2013 CHiME challenge to train and evaluate an audio-denoising convolutional network. The CHiME dataset consists of clean and corresponding artificially mixed noisy audio tracks. The competition organizers claim that the noise-mixing techniques used in data preparation are highly representative of typical noise encountered in a real application. The training data consists of 17,000 very short utterances that make use of a closed-class and extremely limited vocabulary. The development and test data consist of 600 similarly structured utterances. Each test utterance is mixed with various levels of background noise, so that one can measure the average performance of a system at various noise levels.

The ultimate task of CHiME is to maximize speech recognition accuracy at various levels of noise. To allow participants to focus on the task of input denoising, the organizers have made Hidden Markov Model Toolkit (HTK) [**?**], a baseline trainable recognition system, available. Thus, more concretely the task is to supply *processed input* to the speech recognizer that best allows it to correctly recognize the words of the utterance. The final task metric is word recognition accuracy.

3 Technical Approach

We process each utterance into its spectrogram representation. Note that if the spectrogram is appropriately discretized, the result is grayscale image heat map of energies at each frequency at every time slice (see figure **??**).

We make the denoising task concrete as follows. First, we define our goal as being to provide *cleaned Mel Frequency Cepstral Coefficient (MFCC) features* to an HMM-based speech recognition system. The input to our CNN is thus a noisy spectrogram. In one variation, the output of the CNN is a cleaned spectrogram, and a deterministic procedure converts this into MFCC's for use in the recognizer. Alternatively, the CNN could be responsible for directly generating MFCC's, e.g. by adding a multilayer perceptron to the final layers of the network. These two approaches can be seen in figures **??** and **??** below.

At training time, in method 1 we learn to translate noisy spectrogram patches via a CNN into cleaned spectrogram patches, using the corrupted and clean versions of the training data. In method 2 we learn to translate noisy spectrogram slices (e.g. patches whose height is the full spectrogram height) into cleaned MFCC features.

We aim to answer experimental questions such as the following:

- Are CNN's a good modelling fit for the task of audio denoising? As mentioned, we are not aware of any published work applying CNN's in this way. If CNN's are effective at this task, it would offer the hope that noise-robust speech recognition could be possible without the significant effort of hand-engineered transforms.

- What filter sizes are best for audio de-noising? For example, when using CNN's for image deblurring, Xu et al. found that using long and thin filters followed by tall and skinny filters offered good deconvolution performance while not requiring too many parameters[**?**]. One could imagine, in particular, that when deconvolving a reverberation one wants filters that are long in the time axis, and short in the frequency axis (since reverberation causes significant autocorrelation in the signal but does not cause significant frequency interaction). However, deconvolution is only one of the kinds of noise present. Another is well-formed but independent sources of noise, e.g. from a washing machine. One might expect that more standard, e.g. 3x3, convolutional filters would do better here.

- Is method 1 or method 2 (as described above) better? It is not obvious that a reasonably sized MLP can accurately convert convolved spectral representations into MFCC features as needed in method 2. On the other hand, low reconstruction error on the spectrogram (as in method 1) does not necessarily correspond to low reconstruction error on the MFCC features. Due to time constraints, we may only have time to thoroughly explore one of these methods, but given enough time we would like to perform a fair comparison of both techniques.

4 Preliminary Results

We have put significant effort into setting up the pipeline surrounding the CNN above. In particular, we have experimented with a variety of MFCC extraction libraries, in the process learning about many subtleties of the MFCC extraction algorithm. We have identified a library which appears to emulate HTK's MFCC extraction closely. This is important because (1) HTK is the ultimate consumer of the MFCC features, so it is reasonable to expect that we should not deviate much from its behavior, and (2) we cannot simply use HTK's extraction directly, as the code is not easily factorable. We have also identified a library which can convert HTK's packed binary MFCC format to and from a simple text format. Surprisingly, this was not straightforward to do using HTK itself. Finally, we have extracted spectrogram representations of all data (training, development, and test, noisy and clean).

Overall, we have done the initial legwork necessary to begin experimenting with CNN's.