Parallel training of DNNs with Natural Gradient and Parameter Averaging

Daniel Povey, Xiaohui Zhang & Sanjeev Khudanpur

Johns Hopkins University

Overview

- Setting: large-scale DNN training (for speech recognition)
- Two key ideas:
  - Efficient factored natural gradient version of SGD
  - Data parallelization via periodic (~2 mins) model averaging
- Presenting them together because empirically the model-averaging doesn't work without NG-SGD.
- Our system is highly competitive (e.g. best single system for IARPA's ASpIRE challenge on reverberant speech).
- We have been using these methods for ~2 years.

Natural Gradient

In Natural Gradient SGD (NG-SGD), the SGD update

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta_t \mathbf{g}_t$$

becomes

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \eta_t \mathbf{F}_t^{-1} \mathbf{g}_t$$

where $\mathbf{F}_t^{-1}$ is the inverse Fisher matrix estimated given the parameters on time t ($\mathbf{F}_t$ is just the scatter of gradients of training samples).

- Naïvely implemented, extremely slow per step.
- Our implementation only 10-25% slower per step than regular SGD.
- We use a factorized Fisher matrix (one factor for the input-space and one for the output-space of each weight matrix).
- Each factor is unit-matrix-plus-low-rank (typical rank: 40).
- We experiment with two methods of estimating the factors:
  - "Simple" method estimates each factor from the other members of the current minibatch
  - "Online" method (preferred) keeps low-rank estimates up-to-date using a forgetting factor.
- Previous work in efficient natural gradient includes TONGA (low-rank block-diagonal factorization). No experimental comparison with that here.

Data-parallel computation

- Our data parallelization method works on general-purpose hardware.
- Different nodes access different data
- **Periodically average parameters** across the nodes.
- We generally do this every 400 000 samples
  - This is a couple of minutes, if using GPUs.
- **Works surprisingly well** even though problem is not convex.
  - But only works well if using our NG-SGD method
  - We believe NG-SGD stops the parameters "moving too fast" in certain "problem" directions within the parameter space.
- If using n machines (e.g. $n = 4$) we need n times the learning rate to get the same effective learning rate.
- This eventually limits how many machines we can use (would eventually get instability).

Experimental setup

- We wanted to demonstrate large-scale parallelization, so used the Fisher database (1600 hours of speech). Testing is on a subset of Fisher data.
- That is not a recognized test set, to we offer the following comparison to confirm that our system is state of the art:
  - Andrew Ng's recent "DeepSpeech" paper reports 12.5% WER on the Switchboard subset of eval2000, training on Fisher and Switchboard (was at the time the best published number for that setup).
  - Our current Kaldi system on that training and test set gets 11.5% WER
  - This number does not even include all the best and latest stuff, e.g. sequence-discriminative training and silence modeling.
- Our model has 4 hidden layers and 10 million parameters
- For this paper we used our own nonlinearity (p-norm) but since then we've started switching to ReLU.
- In this poster we're just showing convergence plots
  - WERs track these closely since training data is huge.

Results