A Stochastic Quasi-Newton Optimizer for TensorFlow

Jason Chen
Department of Computer Science
Stony Brook University
Stony Brook, NY 11790
hungrchen@cs.stonybrook.edu

David Kraemer
Department of Applied Mathematics
Stony Brook University
Stony Brook, NY 11790
davidkraemer@stonybrook.edu

Abstract

We do some shit and it doesn't work.

1 Introduction

1. Introduce problem that SdLBFGS solves.
2. TensorFlow current state of the art.
3. Problems with the TensorFlow state.
4. Overview of paper.

2 Review of theory

1. Review main results of the Wang, Ma, Goldfarb, Liu paper [1]
2. State the SdLBFGS algortihm.
3. Mention some details about how the algorithm translates to code.

The SdLBFGS algorithm is proposed by Wang, Ma, Goldfarb, and Liu [1] for solving nonconvex stochastic optimization problems. In particular, it solves

$$\underset{x \in \mathbb{R}^n}{\text{minimize}} \quad f(x) = \mathbb{E}[F(x, \xi)] \tag{1}$$

for $F \in C^1(\mathbb{R}^n \times \mathbb{R}^d)$ and $\xi \in \mathbb{R}^d$ is a random variable with cumulative distribution function P. Here $\mathbb{E}[\cdot]$ denotes the mathematical expectation. In general F need not be convex.

The function F is readily interpreted in the context of supervised statistical learning. Given a corpus Ξ of data, we put a distribution on Ξ which reflects a sampling regime for a training set. The object of the learning process is, then, to the model parameter $x \in \mathbb{R}^n$ which minimizes F *in expectation* with respect to the training set. Provided that the sampling maintains some resemblance to the overall corpus, this gives a useful approximate parameter for the whole corpus.

The SdLBFGS algorithm emerges from two concurrent optimization techniques. First, it follows in the tradition of approximate second-order algorithms which seek to perform a version of Newton's method while avoiding the computation of the Hessian $\nabla^2 f(x)$ explicitly. The classical BFGS approach is the canoncal quasi-Newton method. Second, it follows the development of stochastic algorithms which seek to randomly sample the components of the gradient $\nabla f(x)$ so that the essential convergence properties hold in expectation. In this sense it succeeds Stochastic Gradient Descent, among other canonical methods of this class. The limited memory constraint has antecedents in both streams of optimization techniques.

Wang et al. [1] assume that Problem (1) satisfies the following assumptions:

Preprint. Work in progress.

1. There is a real lower bound for f. That is, $\inf_{x \in \mathbb{R}^n} f(x) > -\infty$.
2. The gradient ∇f is Lipschitz continuous. That is, there exists $L > 0$ such that

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

for all $x, y \in \mathbb{R}^n$. Moreover, $\nabla_{xx} F(x, \xi)$ exists and is continuous and there exists $\kappa > 0$ with $\|\nabla^2_{xx} F(x, \varepsilon)\| \leq \kappa$ for any x, ξ.

3. The sampling regime is unbiased. That is, for the kth iterate the following hold:

$$\mathbb{E}[g(x_k, \xi_k)]_{\xi_k} = \nabla f(x_k),$$
$$\mathbb{E}[\|g(x_k, \xi_k) - \nabla f(x_k)\|^2]_{\xi_k} \leq \sigma^2,$$

where $g(x, \xi_k) = \nabla F(x, \xi_k)$ and σ^2 is the noise level of estimating $g(x, \xi_k)$.

For a convex function, $\nabla^2 f(x)$ is alwayas positive definite, but this need not be the case in for nonconvex functions. The work in Wang et al. [1] shows how the BFGS approximate Hessian may be tweaked to ensure that positive definiteness is preserved even in nonconvex cases. The overall stochastic quasi-Newton method is described in Algorithm 2, while the detailed implementation of SdLBFGS is given in Algorithm 2.

Algorithm 1 The high level stochastic quasi-Newton minimization algorithm. Given an initial point $x_1 \in \mathbb{R}^n$, batch sizes $\langle m_k \rangle$, step sizes $\langle \alpha_k \rangle$, and a maximum memory capacity p, perform BFGS with the stochastic dampened direction update.

function SQN($x_1, \langle m_k \rangle_{k=1}^{\infty}, \langle \alpha_k \rangle_{k=1}^{\infty}, p$)
  data $\leftarrow \emptyset$
  for $k = 1, 2, \ldots$ **do**
    $g_k \leftarrow \frac{1}{m_k} \sum_{i=1}^{m_k} g(x_k, \varepsilon_{k,i})$
    $\Delta x_k, s_{k-1}, \bar{y}_{k-1}, \rho_{k-1} \leftarrow$ SdLBFGS($g_k, {}^*(\text{data}^\top)$) ▷ Tuple unpacking
    $x_{k+1} \leftarrow x_k - \alpha_k \Delta x_k$
    append(data, $(s_{k-1}, \bar{y}_{k-1}, \rho_{k-1})$)
    if $k > p$ **then**
      pop(data)
    end if
  end for
end function

3 Implementation

1. Describe how TensorFlow algorithms are usually implemented.
2. Contrast with our implementation of SdLBFGS.
3. Explain some implementation issues.

4 Empirical results

1. Results on simple optimization problems.
2. Results on potentially more difficult optimization problems.
3. Lack of results on MNIST with TensorFlow.

5 Discussion

1. Explaining why the basic project — implementing a useful practical TensorFlow optimzier — failed.
2. Takeaways from the results.
3. Future work.

Algorithm 2 The SdLBFGS update step. The resulting output is $H_k g_k = v_p$, where H_k is the approximation of the kth iterate Hessian and g_k is the approximation of the kth iterate gradient. Note that the computation of H_k is implicit, preventing additional storage requirements.

function SDLBFGS($g_{k-1}, \langle s_j \rangle_{j=k-p}^{k-2}, \langle \bar{y}_j \rangle_{j=k-p}^{k-2}, \langle \rho_j \rangle_{j=k-p}^{k-2}$)
  mem $\leftarrow$ *min*$(p, k-1)$
  $s_{k-1} \leftarrow x_k - x_{k-1}$
  $y_{k-1} \leftarrow \frac{1}{m_{k-1}} \sum_{i=1}^{m_{k-1}} [g(x_k, \xi_{k-1,i}) - g(x_{k-1}, \xi_{k-1,i})]$
  $\gamma_k \leftarrow max\left(\frac{{y_{k-1}}^\top y_{k-1}}{s_{k-1}^\top y_{k-1}}, \delta\right) \geq \delta$
  $\theta \leftarrow max\left(\frac{3}{4} \frac{{\gamma_k}^{-1} {s_{k-1}}^\top s_{k-1}}{{\gamma_k}^{-1} {s_{k-1}}^\top s_{k-1} - {s_{k-1}}^\top y_{k-1}}, 1\right)$ ▷ θ preserves positive definiteness.
  $\bar{y}_{k-1} \leftarrow \theta y_{k-1} + (1 - \theta {\gamma_k}^{-1} s_{k-1})$
  $\rho_{k-1} \leftarrow ({s_{k-1}}^\top \bar{y}_{k-1})^{-1}$
  for $i = 0, \ldots,$ mem $- 1$ **do**
    $\mu_i \leftarrow \rho_{k-i-1} {u_i}^\top s_{k-i-1}$
    $u_{i+1} \leftarrow u_i - \mu_i \bar{y}_{k-i-1}$
  end for
  $v_0 \leftarrow {\gamma_k}^{-1} u_{\text{mem}}$
  for $i = 0, \ldots,$ mem $- 1$ **do**
    $\nu_i \leftarrow \rho_{k-\text{mem}+i} {v_i}^\top \bar{y}_{k-\text{mem}+i}$
    $v_{i+1} \leftarrow v_i + (\mu_{\text{mem}-i-1} - \nu_i) s_{k-\text{mem}+i}$
  end for
  return $v_p, s_{k-1}, \bar{y}_{k-1}, \rho_{k-1}$
end function

References

1. Wang, X., Ma, S., Goldfarb, D. & Liu, W. Stochastic Quasi-Newton Methods for Nonconvex Stochastic Optimization. *SIAM Journal on Optimization* **27,** 927–956 (2017).