Speed Maintained SVRG

November 12, 2017

Speed Maintained SVRG

In this section we describe a novel algorithm: Speed Maintained SVRG (SMSVRG), which can set the appropriate iteration number in each epoch automatically and has superior convergence properties in our experiments.

Algorithm 1 just requires two parameters: learning rate η, mini-epoch size m_0. There are two loops in SMSVRG. In the outer loop(we call each outer iteration as an mini-epoch), m_0 SGD iterations are computed. And then we compute the inequality

$$\frac{\|\tilde{\omega}_{s+1} - \tilde{\omega}_s\|}{\|\tilde{\omega}_s - \tilde{\omega}_{s-1}\|} < \frac{\|\tilde{\omega}_s - \tilde{\omega}_{s-1}\|}{\|\tilde{\omega}_{s-1} - \tilde{\omega}_{s-2}\|} \tag{1}$$

If the inequality holds, we compute a snapshot of the full gradient and step into a new mini-epoch. Otherwise, a new mini-epoch begins directly. It is apparent that we cannot compute the inequality until we have finished two mini-epochs. Note mini-epoch is different from epoch in SVRG, thus we define one epoch as the group of all mini-epochs between two full gradient computations.

In practical experiments, we modify the inequality as follows:

$$\frac{\|\tilde{\omega}_{s+1} - \tilde{\omega}_s\|}{\|\tilde{\omega}_s - \tilde{\omega}_{s-1}\|} < \left(\frac{\|\tilde{\omega}_s - \tilde{\omega}_{s-1}\|}{\|\tilde{\omega}_{s-k} - \tilde{\omega}_{s-k-1}\|}\right)^{\frac{1}{k}} \tag{2}$$

which can reduce the errors incurred by variance and have a better performance than the former.

Numerical Experiments

Experimental settings

In this section, we conduct some experiments to illustrate the efficiency of our proposed algorithm, i.e. SMSVRG. We evaluate our algorithm on eight training datasets, which are public on the LIBSVM website[1]. In our experiments, SBSBRG is applied for two standard machine learning tasks: $l2$-regularized logistic regression and $l2$-regularized ridge regression.

The $l2$-regularized logistic regression task is conducted on the four datasets: ijcnn1, a9a, mushrooms, w8a. Since the label of each instance in these datasets is set to be 1 or -1, the loss function of $l2$-regularized logistic regression task is:

$$\min_{\omega} \frac{1}{n} \sum_{i=1}^{n} \log(1 + e^{-y_i \omega^{\mathrm{T}} x_i}) + \lambda \parallel \omega \parallel^2 . \tag{3}$$

The $l2$-regularized ridge regression task is conducted on the four datasets: abalone, cadata, cpusmall, space_ga. The loss function of $l2$-regularized ridge regression task is:

$$\min_{\omega} \frac{1}{n} \sum_{i=1}^{n} \left(\omega^{\mathrm{T}} x_i - y_i\right)^2 + \lambda \parallel \omega \parallel^2 . \tag{4}$$

We scale the value of all features to $[-1, 1]$ and set the weighting parameter λ to 10^{-4} for all evaluations.

[1] http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

Algorithm 1 Speed Maintained SVRG

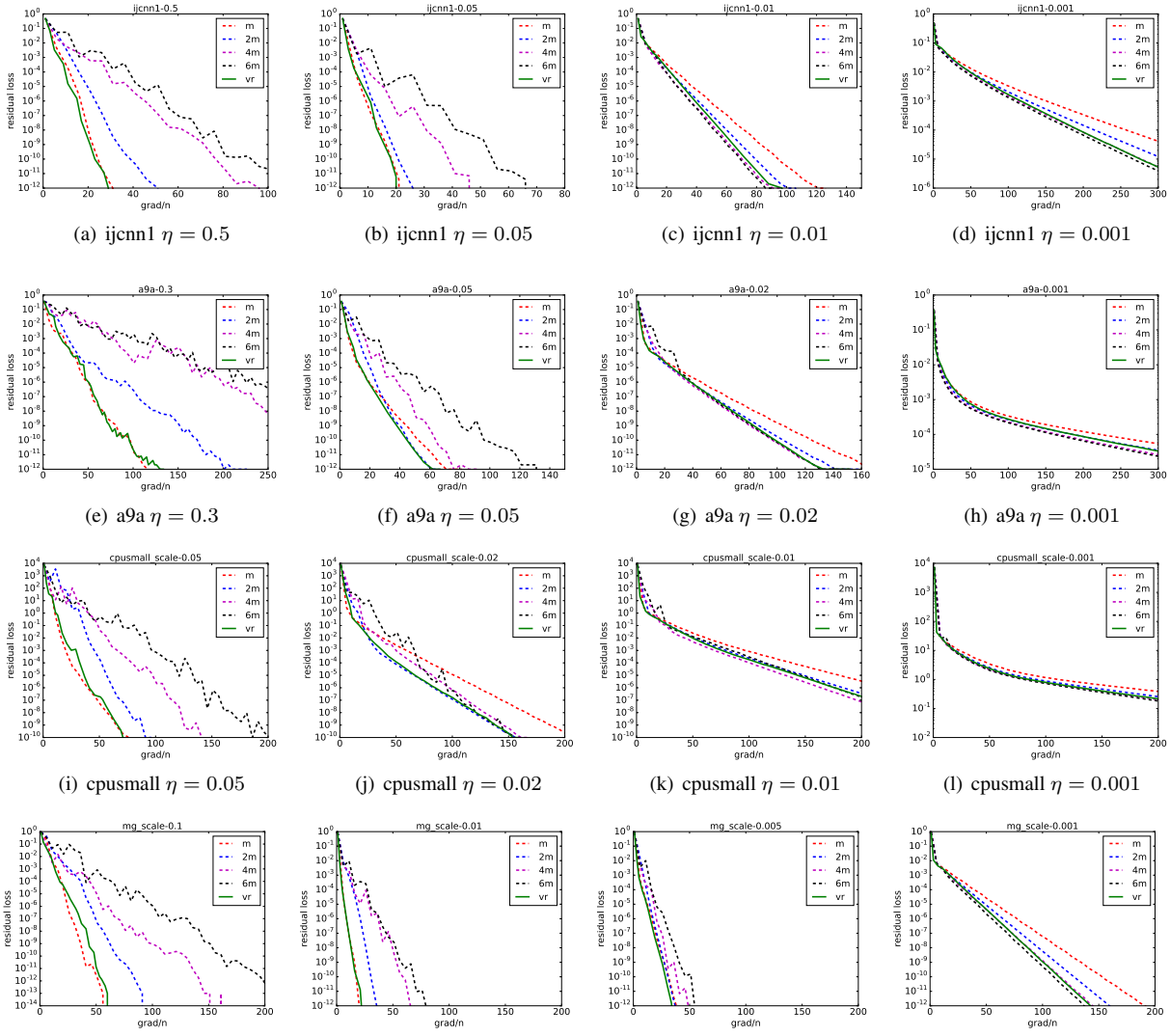
Require: learning rate η, minimal epoch size m_0
Initialize: $\tilde{\omega}_0 = \mathbf{0}$, k=0

$\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega}_0)$
for $s = 0, 1, ...$ **do**
  $\omega_0 = \tilde{\omega}_s$
  for $t = 0, 1, ..., m_0$ **do**
    Randomly pick $i_t \in \{1, 2, ..., n\}$
    $\omega_t = \omega_{t-1} - \eta(\nabla f_{i_t} - \nabla f_{i_t}(\tilde{\omega}_s) + \tilde{\mu})$
  end for
  $\tilde{\omega}_{s+1} = \omega_{\frac{n}{2}}$
  if $s > 1$ and $\frac{\|\tilde{\omega}_{s+1} - \tilde{\omega}_s\|}{\|\tilde{\omega}_s - \tilde{\omega}_{s-1}\|} < \frac{\|\tilde{\omega}_s - \tilde{\omega}_{s-1}\|}{\|\tilde{\omega}_{s-1} - \tilde{\omega}_{s-2}\|}$ **then**
    $\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(\tilde{\omega}_{s+1})$
  end if
end for
return $\tilde{\omega}_{s+1}$

Numerical Results of SMSVRG

We compare SMSVRG with SVRG for optimize (3) and (4). For SMSVRG, we set m_0 to be $n/2$, while n represents the size of datasets. For SVRG, we set epoch size m as four different values: $n, 2n, 4n, 6n$. Our experiments on SVRG show that epoch sizes under n perform always worse than that equals n, so we set the lower bound to be n. Besides, according to the experiments we found epoch sizes bigger than $6n$ perform almost the same, so we set the upper bound of m to be $6n$. And then it is natural to choose $m = 2n$ and $m = 4n$ in this range. For each dataset, we experiment for different learning rates, i.e. η to confirm the theoretical results and insights. In all figures, the x-axis denotes the computational cost, which is measured by the number of gradient computation divided by the size of training data, i.e. n. The y-axis denotes training loss residual, i.e. $F(\tilde{\omega}_s) - F(\omega^*)$. Note that the optimum ω^* is estimated by running the gradient descent for a long time. In all figures, the dashed lines correspond to SVRG with fixed epoch size given in the legends of the figures, while the green solid lines correspond to SMSVRG

It can be seen from Figures 1(a) to 1(p) that SMSVRG can always have the similar performance as SVRG with most suitable epoch size. We observe that when η is big, setting m to be a small value, i.e. n, can achieve better performance. The main reason is that when η is big, the variance becomes big simultaneously, so m must be set small to constrain the variance. As η diminishes, the optimal value of m increases, which means that the algorithm can tolerate more variance induced by extra iterations. As illustrated in Figures, our method is comparable to and sometimes even better than SVRG with best-tuned epoch sizes when learning rate is large or medium. However, if η is set to be too small, SMSVRG performs slightly inferior to SVRG with large epoch sizes, but outperforms SVRG with recommended epoch sizes, i.e. n and $2n$. It is noting that setting η to be too small is not a practical approach when using SVRG or its variants, because the convergence rate will be extremely low. Therefore, the sub-optimal performance of SMSVRG with very small η is acceptable.



(a) ijcnn1 $\eta = 0.5$ (b) ijcnn1 $\eta = 0.05$ (c) ijcnn1 $\eta = 0.01$ (d) ijcnn1 $\eta = 0.001$

(e) a9a $\eta = 0.3$ (f) a9a $\eta = 0.05$ (g) a9a $\eta = 0.02$ (h) a9a $\eta = 0.001$

(i) cpusmall $\eta = 0.05$ (j) cpusmall $\eta = 0.02$ (k) cpusmall $\eta = 0.01$ (l) cpusmall $\eta = 0.001$

(m) mg $\eta = 0.1$ (n) mg $\eta = 0.01$ (o) mg $\eta = 0.005$ (p) mg $\eta = 0.001$

Figure 1: SCSVRG can automatically set a appropriate m for different learning rates