Adaptive Sampling for SGD by Exploiting Side Information

Siddharth Gopal SIDDHARTHG@GOOGLE.COM
Google Inc, 1600 Amphitheatre Parkway

Abstract

This paper proposes a new mechanism for sampling training instances for stochastic gradient descent (SGD) methods by exploiting any side-information associated with the instances (for e.g. class-labels) to improve convergence. Previous methods have either relied on sampling from a distribution defined over training instances or from a static distribution that fixed before training. This results in two problems a) any distribution that is set apriori is independent of how the optimization progresses and b) maintaining a distribution over individual instances could be infeasible in large-scale scenarios. In this paper, we exploit the side information associated with the instances to tackle both problems. More specifically, we maintain a distribution over classes (instead of individual instances) that is adaptively estimated during the course of optimization to give the maximum reduction in the variance of the gradient. Intuitively, we sample more from those regions in space that have a *larger* gradient contribution. Our experiments on highly multiclass datasets show that our proposal converge significantly faster than existing techniques.

1. Introduction

Stochastic Gradient Descent (SGD) based methods (Robbins & Monro, 1951) are among the choice methods for optimizing the average of a large number of functions. We consider the typical setup where we want to solve,

$$\arg\min_{\theta} F(\theta) = \frac{1}{N}\sum_{i=1}^{N} f(i;\theta) \tag{1}$$

Typically, each $f(i;\theta)$ is convex and denotes the misfit between the i^{th} training instance and its corresponding target output. Throughout this paper, we will assume that there N training instances and associated target outputs $\{x_i, y_i\}_{i=1}^{N}$, where $x_i \in \mathcal{R}^d$ and the target y_i could be a label or real value. Moreover we assume that each training instance x_i is associated with some side information c_i where $c_i \in \mathcal{C} \equiv \{1, 2, \dots K\}$. The side information could represent a class-label or time-stamp or any tag associated with x_i. The side information c_i is essentially used as a way to partition the N instances into K bins. Note that c_i is not the same as y_i although it could be. To give a concrete example, lets assume each x_i is associated with one of 100 different class-labels (numbered from 1 to 100) and we are learning a binary logistic regression to distinguish between class-labels $\{1-50\}$ vs $\{51-100\}$. In this case, the side information c_i is the class-label from 1 to 100 and the target $y_i \in \{+1, -1\}$. The function $f(i;\theta)$ is

$$f(i;\theta) = \log\left(1 + \exp\left(-y_i\theta^{\top}x_i\right)\right)$$

Similar situations arise when we take the popular one-versus-rest approach for performing multiclass classification (Rifkin & Klautau, 2004).

To solve such optimization problems, in the standard SGD setting (Bottou, 2010), the t^{th} iteration involves picking an instance i^t *uniformly sampled* over all instances and performing the following update,

$$d^t = \nabla f(i^t;\theta^{t-1}) \tag{2}$$
$$\theta^t = \theta^{t-1} - \gamma^t d^t \tag{3}$$

where γ^t is the step size at the t'th iteration. In the presence of any additional penalty term $r(\theta)$ like L1 regularizer, the following proximal update (Beck & Teboulle, 2009) is preferred instead of (3),

$$\theta^t = \text{Prox}_{\gamma^t r}(\theta^{t-1} - \gamma^t d^t) \tag{4}$$

Despite working with only one instance an iteration, this procedure still converges to the optima as long as γ^t is appropriately set (Bottou, 2010; Robbins & Monro, 1951). This is because, the descent direction d^t at each iteration is an unbiased estimate of the true gradient,

$$E_{i_t}\left[d^t|\theta^{t-1}\right] = \frac{1}{N}\sum_{i=1}^{N}\nabla f(i;\theta^{t-1}) = \nabla F(\theta)$$

Proceedings of the 33^rd International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48. Copyright 2016 by the author(s).

However, the downside of SGD methods is that the oracle rate of convergence is known to be $O(\frac{1}{T})$ for strongly-convex and $O(\frac{1}{\sqrt{T}})$ for convex objectives respectively; which are slower than the corresponding rates for standard gradient descent (Shamir & Zhang, 2012). One of the main reasons for this suboptimality is the variance introduced by the descent direction - instead of averaging over all the N instances we descend in a direction determined by a single training instance. Nevertheless, these are the best possible rates of convergence when $f(i;\theta)$ can be any arbitrary convex function.

However, in practice, $f(i;\theta)$ are not arbitrary and often have some latent structure. For e.g., in the same binary classification task $\{1-50\}$ vs $\{51-100\}$, it might be the case that the decision boundary is largely determined by classes 50 and 51 and that all the other classes play little or no role. This means that faster convergence can be achieved by concentrating more near the decision boundary, i.e. sampling more instances from classes 50, 51 than from other classes. These observations lead to two important questions,

1. Can we automatically infer such latent structure ?

2. Since variance is the limiting factor, can we exploit such side information c_i to improve convergence ?

In this paper, we propose to use the side information c_i to reduce the variance of the descent direction. More specifically, we learn an optimal sampling distribution over the bins i.e. class-labels (instead of individual training instances) that gives the maximum reduction in the variance of the descent direction. We show that reducing the variance automatically uncovers such latent structures in the data. In fact the optimal sampling distribution corresponds to sampling more instances from those bins which have a larger gradient contribution in the objective. Intuitively, instances with larger gradients correspond to the more *confusable* instances w.r.t the current decision boundary. Unlike other work (Zhao & Zhang, 2014b;a; Needell et al., 2013; Zhang & Xiao, 2014) where the sampling distribution is fixed apriori, we relearn the distribution once in several iterations. Overall, this has two important consequences (a) maintaining a distribution over bins instead of instances reduces the memory complexity from $O(N)$ to $O(K)$ (using the efficient walker-alias sampling method (Walker, 1977)) and (b) the distribution is adaptive to how the optimization progresses.

Although our proposal does not improve the theoretical convergence rate for arbitrary convex functions f, it exploits the structure in the problem and in practice exhibits faster convergence. Our preliminary experiments show promising results and are of practical importance,

1. Additional side information about the instance for e.g. class-labels are almost always available - but this has never been exploited in improving the convergence.

2. To date, most of the proposed schemes to reduce variance do not directly adapt to how the optimization progresses for e.g. do not actively learn which subset of training instances to sample.

2. Related Work

Despite the suboptimal convergence of SGD, they have been extensively used to train both convex (Bottou, 2010) and non-convex objectives (Bottou, 1991). Recently, there has been many work that improve the convergence of SGD in practice or theory by resorting to one or a combination of the following techniques,

1. **Averaging the parameters**: In these methods, the optimal solution is represented as an average of the iterates $\theta^1, \theta^2, \ldots$ (Polyak & Juditsky, 1992). Averaging introduces a notion of stability of the solution. A partial averaging scheme was also proposed in (Rakhlin et al., 2011) that achieves the oracle $O(\frac{1}{T})$ for strongly convex objectives.

2. **Tweaking the learning rate**: The learning rate is a one parameter approximation to the inverse of the Hessian (Bousquet & Bottou, 2008). Setting the learning rate to decay as $\gamma^t = \gamma^0(1+\lambda\gamma^0 t)^{-1}$ has been empirically found to work well (Xu, 2011). Using dimension-specific learning rates based on first order gradient information was proposed in (Duchi et al., 2011) and further improved in (Zeiler, 2012). Other work on setting the learning rate based on the local curvature and gradients include (Amari et al., 2000; Schaul et al., 2012; Roux & Fitzgibbon, 2010).

3. **Averaging the gradients**: One way to reduce the variance of the descent direction is to average gradients from multiple training instances. The simplest is to average the gradients over all instances which corresponds to the standard gradient descent. An alternative is to average over a mini batch of instances, which can be implemented efficiently and also improves the convergence for certain mini batch sizes(Li et al., 2014; Cotter et al., 2011).

4. **Momentum methods** Recently, there have been a slew of methods (Schmidt et al., 2013), (Konečný & Richtárik, 2013; Defazio et al., 2014; Johnson & Zhang, 2013; Konečný et al., 2014) where the descent direction uses some momentum from past gradients. For example, in one of the variants (Johnson & Zhang, 2013), the descent direction at the t'th iteration is de-

fined as,

$$d^t = \nabla f(i^t;\theta^{t-1}) - \nabla f(i^t;\tilde{\theta}) + \frac{1}{N}\sum_i \nabla f(i;\tilde{\theta}) \quad (5)$$

where i^t represents the instance chosen at the t'th step, and $\tilde{\theta}$ represents some historical snapshot of θ updated periodically. In this variant d^t is an unbiased estimate of the gradient but with a lower variance than in (2). On finite datasets, non asymptotic analysis yields exponential and $O(\frac{1}{T})$ convergence of strongly convex and convex objectives.

5. **Selective sampling**: These methods are most closely related to our proposed techniques. The key idea behind these methods is to replace the uniform distribution used for sampling i^t with a weighted distribution instead. (Zhao & Zhang, 2014b) and (Needell et al., 2013) propose picking a training instance with a probability proportional to the lipschitz constant of $f(i;\theta)$ or $\nabla f(i;\theta)$ respectively. This not only reduces the variance of the descent direction (Zhao & Zhang, 2014b) but improves the rate of convergence as well (Needell et al., 2013). In (Zhao & Zhang, 2014a), a variant of (Zhao & Zhang, 2014b) was proposed where the dataset was partitioned into clusters and a minibatch of instances was sampled from each cluster. However, the fundamental limitation of all these methods are that - (a) all the proposed distributions are statically set before optimization (b) maintaining a distribution (i.e. storing as well as updating) over training instances is infeasible in large-scale scenarios and (c) the additional side information c_i is not exploited.
During the publication of this work, a very similar idea was pursued in (Alain et al., 2015), where the sampling weights are proportional to the L2 norm of the gradient (similar to our work) but the distribution is maintained at an instance level (thereby requiring multiple machines to update/search and sample from an approximation to this distribution). In our work the distribution is maintained at a class-level making a) updating the distribution more tractable (i.e. O(K) instead of O(N)) and b) better estimates for the gradient norm due to pooling of information. Moreover, we also show an analysis of why this improves convergence in the convex case (section 3.1).

In this paper we propose a non-uniform adaptive sampling strategy as an alternative to (Zhao & Zhang, 2014b) or (Needell et al., 2013). Our work is not an alternative to other work based on averaging, using momentum or tweaking the learning rates; but should be considered complimentary and if the application allows used in conjunction (see Experiments).

3. Adaptive Sampling

Let $\mathcal{C}_k$ denote that set of all training instances which have the side-information set to k, that is, $\mathcal{C}_k = \{i : c_i = k\}$. First, we rewrite (1) as an equivalent minimization of weighted training instances,

$$\arg\min_\theta \sum_{k\in\mathcal{C}}\sum_{i\in\mathcal{C}_k} P(i)f(i;\theta), P(i) = p_k\cdot\frac{1}{|\mathcal{C}_k|},\ \ p_k = \frac{|\mathcal{C}_k|}{N} \quad (6)$$

Here, $P(i)$ is the probability of picking the i^{th} training instance. The above reformulation allows to reinterpret the uniform sampling as a two stage process - first sample a bin with probability p_k proportional to the number of instances in the bin, and then uniformly sample an instance from the selected bin. Similar to (2), the descent direction is given by

$$d^t = \frac{1}{N}\frac{1}{P(i^t)}\nabla f(i^t;\theta^{t-1}) \quad (7)$$

It is easy to verify that the expected value of the descent direction $\mathbb{E}[d^t]$ matches true gradient of the objective (1) for any distribution p_k. Instead of setting p_k to a static value of $\frac{|\mathcal{C}_k|}{N}$, we propose to set p_k to enable faster convergence. More specifically, we set $\mathbf{p} = \{p_1, p_2, \dots p_k\}$ so as to reduce the total variance of the descent direction,

$$\min_{p_1,p_2,\dots p_K} \mathbb{V}(d^t) = \mathbb{E}[{d^t}^\top d^t] - \mathbb{E}[d^t]^\top\mathbb{E}[d^t]$$

The optimal distribution $\mathbf{p}$ at the t^{th} iteration is given by (see Appendix for the details),

$$p_k \propto \frac{|\mathcal{C}_k|}{N}\sqrt{\frac{1}{|\mathcal{C}_k|}\sum_{i\in\mathcal{C}_k}||\nabla f(i;\theta^{t-1})||^2} \quad (8)$$

The probability of picking a bin is proportional to the square-root of the sum of squared gradient norms of all the instances in that bin. Intuitively, we set p_k such that we choose bins which have a larger gradient contribution more and ignore bins which have lower gradients. For example, in a one-versus-rest setting where each bin is a class, this roughly corresponds to drawing training instances from those classes which tend to get confused with the training class-label and ignoring those instances which have been classified correctly. The same result can be seamlessly extended to the fully stochastic case where we minimize an expectation $\arg\min_\theta \mathbb{E}_i[f(i;\theta)]$,

$$p_k \propto n_k\sqrt{s_k}$$
$$\text{where } n_k = \frac{\mathcal{C}_k}{N},\quad s_k = \mathbb{E}_{i\sim\mathcal{C}_k}\left[\|\nabla f(i;\theta^{t-1})\|^2\right] \quad (9)$$

Algorithm 1 Adaptive Sampling

```
1: Input: $T$, $\delta$, $J$, $\{x_i, y_i\}_{i=1}^N$, where $x_i \in \mathcal{R}^d$, $y_i \in \{+1, -1\}$
2: Let $ssg[1 \dots K] \leftarrow 0, count[1 \dots K] \leftarrow 0$.
3: for $t = 1, 2, \dots T$ do
4:     Sample bin $k^t$ from $\mathbf{p} = \{p_1, p_2, \dots p_K\}$.
5:     Sample instance $i^t$ from bin $k^t$ uniformly at random.
6:     Compute descent direction $d^t$ using (7).
7:     Compute learning rate $\gamma^t$.
8:     Update $\theta^t$ using (3) or (4) accordingly.
9:     $ssg[k^t] \leftarrow ssg[k^t] + \|\nabla f(i^t; \theta^t)\|^2$
10:    $count[k^t] \leftarrow count[k^t] + 1$
11:    if $t$ mod $J = 0$ then
12:        Set $p_k \propto \frac{\mathcal{C}_k}{N}\sqrt{\frac{ssg[k]}{count[k]}}$, i.e. (8)    $\forall k$
13:        Smooth $\mathbf{p}$ with the original distribution, i.e, $p_k = \delta p_k + (1-\delta)\frac{\mathcal{C}_k}{N}$    $\forall k$
14:        $ssg[1 \dots K] \leftarrow 0, count[1 \dots K] \leftarrow 0$    $\forall k$
15:    end if
16: end for
```

3.1. Analysis

As such, all the typical convergence guarantees of vanilla SGD (Robbins & Monro, 1951) follow even when using any arbitrary sampling distribution $P(i)$, as long as the gradient estimates are unbiased and $P(i)$ is a strictly positive distribution (Zhao & Zhang, 2014b).

We additionally provide some basic analysis to show where and when such adaptive sampling helps. Following (Robbins & Monro, 1951), for any arbitrary convex function $f(i;\theta)$, step sizes γ^t such that $\sum_{i=1}^{T} \gamma^t \to \infty$ and $\sum_{i=1}^{T} (\gamma^t)^2 < \infty$, and $D = ||\theta^1 - \theta^*||$, the following holds for the stepize-averaged iterate $\bar{\theta}^t$

$$\mathbb{E}[F(\bar{\theta}^t) - F(\theta^*)] \leq \left(4D^2 + \sum_{i=1}^{T}(\gamma^t)^2 v^t\right)\left(2\sum_{i=1}^{T}\gamma^t\right)^{-1}$$

Here, $v^t = \mathbb{E}[d_t^\top d_t]$, is the expectation of the squared norm of the descent direction at the t^{th} step. By using adaptive sampling, we reduce the magnitude of v^t even further. To see this, we compare v^t under a uniform sampling distribution $\mathbf{u}$ vs v^t under the optimal distribution $\mathbf{p}$,

$$v_{\mathbf{u}}^t = \sum_{k \in \mathcal{C}} n_k s_k, \qquad v_{\mathbf{p}}^t = \left(\sum_{k \in \mathcal{C}} n_k \sqrt{s_k}\right)^2 \tag{10}$$

Note that $v_{\mathbf{p}}^t$ (i.e. the optimal value of (13))is an improvement over $v_{\mathbf{u}}^t$, i.e. $v_{\mathbf{p}}^t \leq v_{\mathbf{u}}^t \;\; \forall t$, which can be seen using the Engel form of Cauchy-Schwarz inequality. The convergence is particularly improved if,

$$\frac{v_{\mathbf{p}}^t}{v_{\mathbf{u}}^t} = \frac{\sum_{k \in \mathcal{C}} n_k^2 s_k + \sum_{i,j \in \mathcal{C}, j \neq i} n_j n_k \sqrt{s_j}\sqrt{s_j}}{\sum_{k \in \mathcal{C}} n_k \sqrt{s_k}\sqrt{s_k}} \tag{11}$$

is lower. Intuitively, adaptive sampling is more beneficial when the gradients are less correlated between different classes (numerator) and more correlated within the same class (denominator). In other words, convergence depends on how well separated the classes are.

3.2. Computational Cost

In practice, it is impossible to reset or estimate the optimal sampling distribution after each iteration. Estimating p_k requires making a full pass through all the data which is infeasible. Therefore we propose the following two computationally feasible ways of approximating p_k,

1. Stop every J iterations (say $J \sim N/4$ iterations), sample a few training instances from each bin (say 10) and estimate $\mathbb{E}_{i \sim \mathcal{C}_k}\left[\|\nabla f(i; \theta^{t-1})\|^2\right]$. The computational cost dramatically reduces to processing only an additional $O(10)$ instances per bin every J iterations. The downside is that there are now two more tweaking parameters - J and the number of instances to pick from each bin. However, in some cases, processing even $O(10)$ instances per bin could be fairly expensive if the instances have a large feature space.
2. Another alternative is to maintain a running sum of the squared gradient norms across the K bins during the optimization. The gradient needs to be computed anyway, for updating θ, and can simply be reused for updating the sums. Now, every J iterations ($J \sim N/4$), we estimate $\mathbb{E}_{i \sim \mathcal{C}_k}\left[\|\nabla f(i; \theta^{t-1})\|^2\right]$ from the sums and reset the distribution $\mathbf{p}$ according to (9) (see algorithm (1)). The computational cost is only $O(K \log K)$ every J iterations. The potential downside of this approach is that the gradient information could be stale and the estimated $\mathbf{p}$ might not be optimal w.r.t the current θ^t.

In our experiments, approximation (b) worked better as it had no noticeable increase in training time and also avoided a tuning parameter. However, one issue with approximation (b) is that at some point in the optimization it is possible that p_k could be set to zero causing the bin k to be never visited again. Clearly this is suboptimal; to avoid such issues as well as for other reasons (see Appendix) we smooth $\mathbf{p}$ with the original distribution to ensure that all p_k are strictly positive at all times. If δ is the smoothing parameter, the smoothed distribution is given by

$$p_k^{smooth} = \delta p_k + (1-\delta)\frac{\mathcal{C}_k}{N} \tag{12}$$

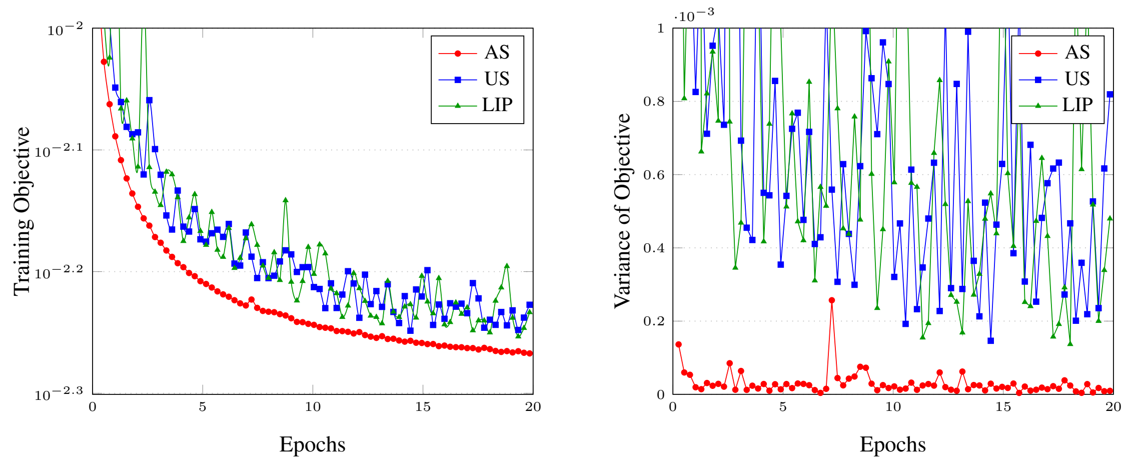


Figure 1. Binary logistic regression with L1 penalty on ALOI dataset - training objective averaged over 10 runs (left) and its variance (right).

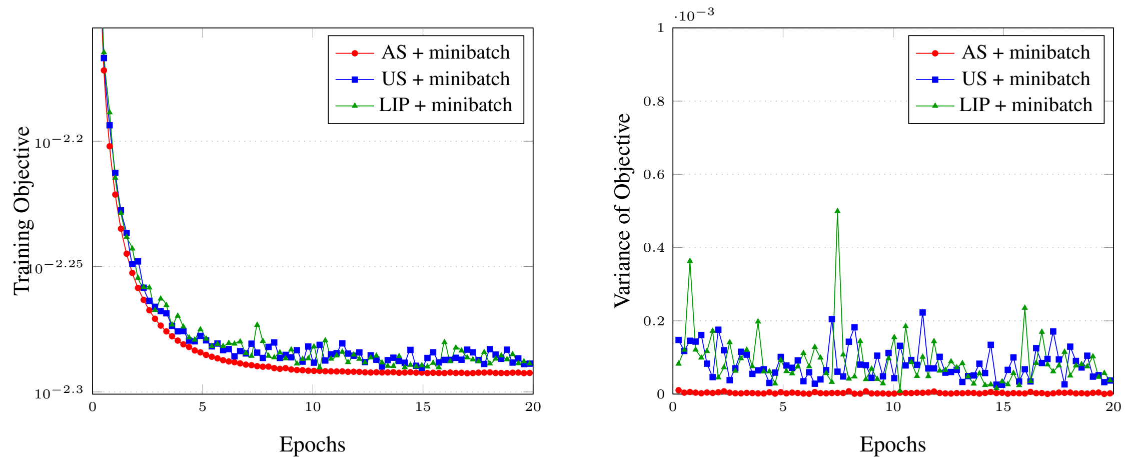


Figure 2. Binary logistic regression with L1 penalty on ALOI dataset with a minibatch size of 50- training objective averaged over 10 runs (left) and its variance (right).

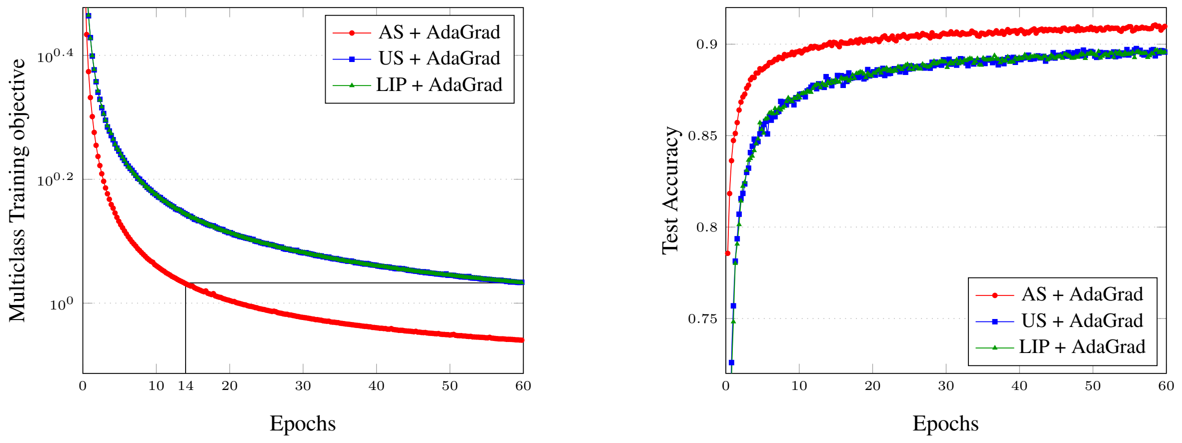


Figure 3. Multiple one-versus-rest logistic regression with L1 penalty on ALOI dataset using AdaGrad learning rates - training objective (left) and accuracy (right).

The complete pseudocode is given in Algorithm 1.

4. Experiments

We used three datasets for experimentation - ALOI, CIFAR100 and IPC,

1. **ALOI** [1] : An image database containing 1000 objects in various illuminations and poses. The features are 128 dimensional color histograms. There are a total of 97,020 training instances and 10,080 test instances.
2. **CIFAR100**: A collection of 60,000 32x32 tiny images of objects labeled across 100 different classes. 50,000 of them were used for training and the rest for testing. We used the patch level features generated using vector quantization (Coates & Ng, 2012) leading to a 6400 dimensional feature space.
3. **IPC** [2] : A set of 75,250 patents labeled into one of 451 classes - 46,324 of were used for training and the rest for testing. We used the top 500 principal components as features.

For all the experiments, we define c_i to be the class-label associated with the instance. We generically set $J = \frac{N}{4}$ and $\delta = 0.5$. The choice for J was made so as to ensure no noticeable increase in the computational cost and δ was set to a midpoint value between the two distributions. Unless otherwise noted, all learning rates were carefully tuned (using the scheme in (Bottou, 2010)) to achieve the lowest objective at the cutoff point and the regularization was set using a 20% validation set.

In the first experiment, we show that using adaptive sampling converges faster than non-adaptive sampling as well as reduces the variance of the gradient (and thereby the objective). We use the ALOI dataset and train a binary logistic regressor (with L1 penalty to introduce non-smoothness) to distinguish between classes $\{1, 2\}$ vs $\{3, 4 \dots 1000\}$. We compare our proposed adaptive sampling (**AS**) outlined in algorithm 1 against two other baselines,

- uniformly sampling a training instance (**US**).
- sampling an instance with a probability proportional to the lipschitz constant of its gradient (**LIP**) (Needell et al., 2013). This worked better than using the lipschitz constant of the function (Zhao & Zhang, 2014b). Note that both proposals reduce to uniform sampling when the instances are unit normed.

We also tried simpler baselines like setting p_k to a uniform distribution, or even binarizing p_k (i.e. binning examples into positive/negative class). We do not report the results as they were no better than uniformly sampling instances (and also to prevent clutter in figures).

Figure 1 plots the results over 10 different runs. AS clearly converges faster than both the baselines (Figure 1, left). Figure 1 top right, plots the variance of the objective (y axis) at iteration t for various t (x axis). The variance is significantly lesser using AS than US or LIP. Note that there was no difference in training time between the three methods - the additional $O(K \log K)$ computation every J iterations for AS was insignificant compared to the total computation.

In the second experiment (Figure 2), we study the effect of using minibatches. Minibatching is known to effectively reduce the variance of the gradient compared to standard SGD(Li et al., 2014), is there any further benefit in using AS? We repeat the same analysis on the ALOI dataset using a minibatch of 50 instances **at each iteration** (roughly $\sim .5\sqrt{N}$) instead of a single instance. The results provide similar conclusions and show that AS adds an additional layer of variance reduction on top of minibatching. We got similar conclusion when using other minibatch sizes.

In the third experiment, we do a fuller-scale analysis on the ALOI dataset and show that AS gives measurable benefit in the convergence of both the training objective as well as the accuracy on the test set. We use the popular one-versus-rest approach for multiclass classification and train 1000 binary logistic regressors with L1 penalty and AdaGrad (Duchi et al., 2011) learning rates (similar conclusions was reached even if AdaGrad was not used). Figure 3 plots the convergence in the training objective and the convergence in accuracy on the test set. For a thorough comparison, we also present the results on using ProxSVRG method (Xiao & Zhang, 2014) instead of standard SGD (Figure 4). In both cases, AS performs significantly better. For instance, in Figure 3 left, AS achieves the best training objective reached by US in just the 14th iteration yielding a 4X speedup. Note that the unusual jumps using ProxSVRG is caused when the historical snapshot $\tilde{\theta}$ is updated every $2N$ iterations (see (Xiao & Zhang, 2014), pg 13) - this update incurs an additional cost of making a full pass over the data.

In the fourth experiment, we repeat the full-scale analyis on two other datasets and confirm that our findings are generalizable. We trained 100 binary logistic regressors on CIFAR100 and 451 binary classifiers using squared hingeloss (squared hingeloss worked better than the logistic) on the IPC dataset. We additionally included a L1 penalty term and used AdaGrad learning rates. On both of the datasets sampling using AS converges significantly faster than US (Figure 5). We omit the results for LIP as it is was much worse and could not be illustrated well in the same graph.

[1] http://www.csie.ntu.edu.tw/c̃jlin/libsvmtools/datasets/multiclass.html

[2] http://gcdart.blogspot.com/2012/08/datasets_929.html

4.1. Failure/NOP cases

In general, there is no one-fit all solution and not all techniques work on all kinds of data/scenarios. The intuition behind AS is to concentrate more on those regions/bins of training instances which have larger gradient contribution. However this intuition breaks down when the assignment of training instances to bins is not informative. For example, in the covertype dataset [3] and the 20newsgroup dataset [4], the ratio of bins to instances is too small causing the bins to be partitioned very coarsely. AS did not give any measurable benefit in both the datasets, there was some mild improvement in news20 and mildly worse convergence in covertype dataset. Another extreme case happens when our hypothesis that $f(i;\theta)$ has some latent structure itself breaks down. For example, if we choose c_i uniformly at random from $\{1, 2, \dots K\}$, it is unlikely that AS would give any benefit as it can learn only spurious correlations with uniform distribution.

5. Conclusion

In this work, we proposed an adaptive sampling scheme for drawing training instances for SGD based methods. Our proposal estimates an optimal distribution over *bins* of instances to achieve the maximum reduction in the variance of the gradient. Our experiments establish that AS exhibits significantly faster convergence than existing schemes on multiple datasets.

Acknowledgements

The author would like to thank Ian Goodfellow and Samy Bengio for several useful discussions.

Appendix

5.1. Proof of equation (8)

We rewrite equation (8) as,

$$\min_{p_1, p_2, \dots p_K} \mathbb{E}[{d^t}^\top d^t] - \mathbb{E}[d^t]^\top \mathbb{E}[d^t]$$

$$\text{where} \quad \mathbb{E}[{d^t}^\top d^t] = \sum_{i=1}^{N} P(i) \frac{1}{N^2} \frac{1}{P(i)^2} \|\nabla f(i;\theta^{t-1})\|^2$$

$$\mathbb{E}[d^t] = \sum_{i=1}^{N} P(i) \left[\frac{1}{N} \frac{1}{P(i)} \nabla f(i;\theta^{t-1})\right] = \frac{1}{N} \sum_{i=1}^{N} \nabla f(i;\theta^{t-1})$$

Note that $\mathbb{E}[d^t]$ is independent of the $p_1, p_2, \dots p_k$ and can be dropped. Using (6), the optimization problem can be rewritten as,

$$\min_{p_1, p_2, \dots p_K} \sum_{k \in \mathcal{C}} \frac{\alpha_k}{p_k} \tag{13}$$

$$s.t$$

$$\sum_{k \in \mathcal{C}} p_k = 1 \quad \text{and } p_k > 0 \quad \forall \; k = 1, 2, \dots K$$

$$\text{where} \quad \alpha_k = \frac{|\mathcal{C}_k|}{N^2} \sum_{i \in \mathcal{C}_k} \|\nabla f(i;\theta^{t-1})\|^2$$

Since all $\alpha_k > 0$, we can safely neglect the inequality constraint (i.e, the optimal solution of the un-inequality-constrained problem satisfies the inequality constraints anyway). Introducing the lagrangian variable λ for the equal constraint,

$$L(\mathbf{p}, \lambda) = \frac{\alpha_k}{p_k} + \lambda \left(\sum_{k \in \mathcal{C}} p_k - 1\right)$$

and setting $\frac{\partial L(\mathbf{p}, \lambda)}{\partial p_k} = 0$, we get $p_k = \frac{\sqrt{\alpha_k}}{\sqrt{\lambda}}$. Applying the equality constraint, the optimal $\mathbf{p}$ is

$$p_k^* = \sqrt{\alpha_k} \left(\sum_{k \in \mathcal{C}} \sqrt{\alpha_k}\right)^{-1}$$

It is obvious that as along $\alpha_k > 0$, the resulting p_k^* satisfies the inequality constraints. To ensure that all $\alpha_k > 0$, we add a small ϵ to all the α_ks.

[3] https://archive.ics.uci.edu/ml/datasets/Covertype

[4] http://qwone.com/ jason/20Newsgroups/

References

Alain, Guillaume, Lamb, Alex, Sankar, Chinnadhurai, Courville, Aaron, and Bengio, Yoshua. Variance reduction in sgd by distributed importance sampling. *arXiv preprint arXiv:1511.06481*, 2015.

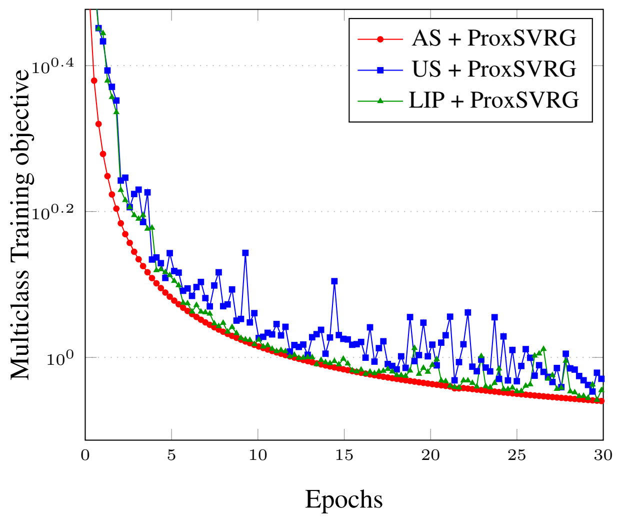
Amari, Shun-Ichi, Park, Hyeyoung, and Fukumizu, Kenji. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, 12(6): 1399–1409, 2000.

Beck, Amir and Teboulle, Marc. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Bottou, Léon. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes*, 91(8), 1991.

Bottou, Léon. Large-scale machine learning with stochastic gradient descent. In *COMPSTAT*, pp. 177–186. Springer, 2010.

Bousquet, Olivier and Bottou, Léon. The tradeoffs of large scale learning. In *NIPS*, pp. 161–168, 2008.



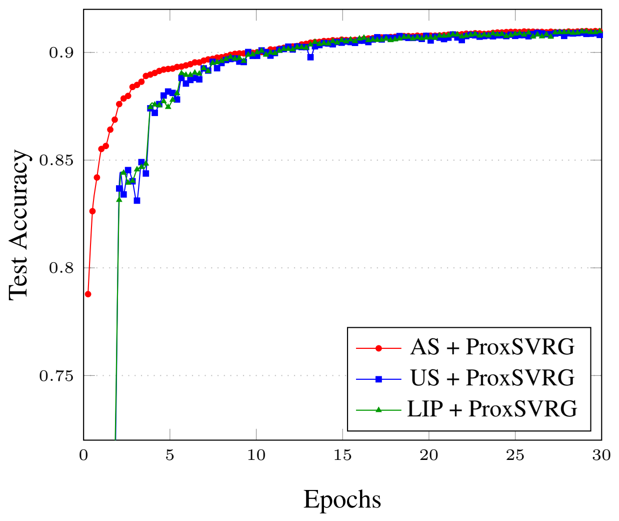
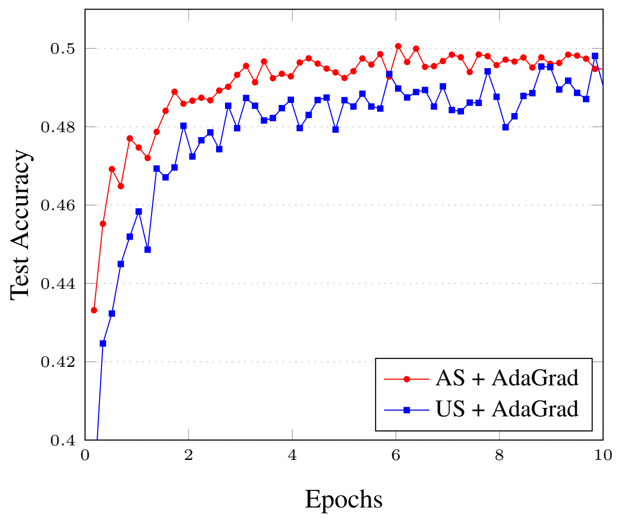


Figure 4. Multiple one-versus-rest logistic regression with L1 penalty on ALOI dataset using Proximal SVRG - training objective (left) and test accuracy (right).



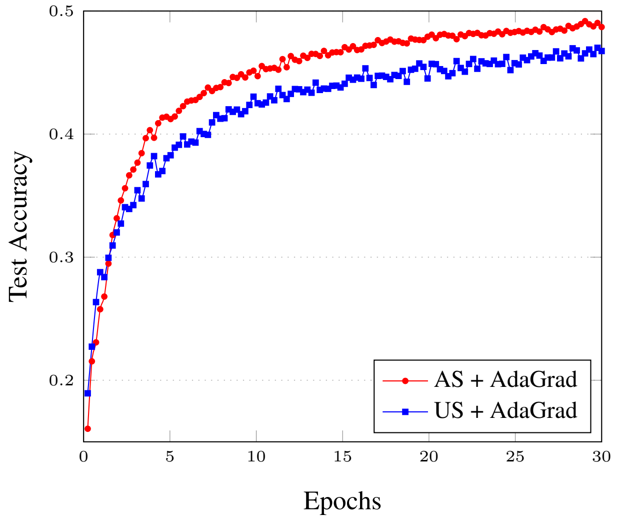


Figure 5. Convergence of Test Accuracy on IPC (left) and CIFAR100 dataset (right).

Coates, Adam and Ng, Andrew Y. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, pp. 561–580. Springer, 2012.

Cotter, Andrew, Shamir, Ohad, Srebro, Nati, and Sridharan, Karthik. Better mini-batch algorithms via accelerated gradient methods. In *NIPS*, pp. 1647–1655, 2011.

Defazio, Aaron, Bach, Francis, and Lacoste-Julien, Simon. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *NIPS*, pp. 1646–1654, 2014.

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

Johnson, Rie and Zhang, Tong. Accelerating stochastic gradient descent using predictive variance reduction. In *NIPS*, pp. 315–323, 2013.

Konečný, Jakub and Richtárik, Peter. Semi-stochastic gradient descent methods. *arXiv preprint arXiv:1312.1666*, 2013.

Konečný, Jakub, Qu, Zheng, and Richtárik, Peter. Semi-stochastic coordinate descent. *arXiv preprint arXiv:1412.6293*, 2014.

Li, Mu, Zhang, Tong, Chen, Yuqiang, and Smola, Alexander J. Efficient mini-batch training for stochastic optimization. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 661–670. ACM, 2014.

Needell, Deanna, Srebro, Nathan, and Ward, Rachel. Stochastic gradient descent and the randomized kaczmarz algorithm. *arXiv preprint arXiv:1310.5715*, 2013.

Polyak, Boris T and Juditsky, Anatoli B. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Rakhlin, Alexander, Shamir, Ohad, and Sridharan, Karthik. Making gradient descent optimal for strongly convex stochastic optimization. *arXiv preprint arXiv:1109.5647*, 2011.

Rifkin, Ryan and Klautau, Aldebaro. In defense of one-vs-all classification. *The Journal of Machine Learning Research*, 5:101–141, 2004.

Robbins, Herbert and Monro, Sutton. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Roux, Nicolas L and Fitzgibbon, Andrew W. A fast natural newton method. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 623–630, 2010.

Schaul, Tom, Zhang, Sixin, and LeCun, Yann. No more pesky learning rates. *arXiv preprint arXiv:1206.1106*, 2012.

Schmidt, Mark, Roux, Nicolas Le, and Bach, Francis. Minimizing finite sums with the stochastic average gradient. *arXiv preprint arXiv:1309.2388*, 2013.

Shamir, Ohad and Zhang, Tong. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. *arXiv preprint arXiv:1212.1824*, 2012.

Walker, Alastair J. An efficient method for generating discrete random variables with general distributions. *ACM Transactions on Mathematical Software (TOMS)*, 3(3): 253–256, 1977.

Xiao, Lin and Zhang, Tong. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, 2014.

Xu, Wei. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv:1107.2490*, 2011.

Zeiler, Matthew D. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zhang, Yuchen and Xiao, Lin. Stochastic primal-dual coordinate method for regularized empirical risk minimization. *arXiv preprint arXiv:1409.3257*, 2014.

Zhao, Peilin and Zhang, Tong. Accelerating minibatch stochastic gradient descent using stratified sampling. *arXiv preprint arXiv:1405.3080*, 2014a.

Zhao, Peilin and Zhang, Tong. Stochastic optimization with importance sampling. *arXiv preprint arXiv:1401.2753*, 2014b.