Variance Reduction with Neighbours for Adaptive Optimization

Jose Gallego GALLEGOJ@MILA.QUEBEC
Mila and DIRO, Université de Montréal

Sanae Lotfi SANAE.LOTFI@UMONTREAL.CA
CERC, Polytechnique Montréal and CentraleSupélec

Abstract

Recent works in optimization have exhibited the impact of variance reduction on accelerating the convergence of stochastic gradient methods. However, it is still an open research question how to effectively apply variance reduction techniques on deep learning problems. In this work we propose to combine variance reduction with neighborhood-based adaptive moment estimation optimization algorithms. We achieve on-par performance as state-of-the-art techniques on convex and non-convex classification on MNIST, while achieving better levels of variance reduction, with a memory overhead that scales efficiently with the number of datapoints.

1. Introduction

Emprical risk minimization (ERM) problems play a paramount role in modern machine learning. We are given a dataset of examples $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, and we aim to minimize the function

$$f(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$
$$f_i(w) = l(w, (x_i, y_i)) + \lambda\Omega(w),$$

with respect to the model parameters w. Here Ω is a μ-strongly convex regularizer and the loss function l is convex and L-smooth.

First order optimization methods have been widely used to solve this problem, since the estimation of higher order derivatives can be both time and memory-consuming. Among this class of algorithms, we can find stochastic gradient methods (Robbins & Monro, 1951; Bottou & LeCun, 2004). These methods are highly used in the context of deep learning, where we often have a large training data set and where the loss is not necessarily convex.

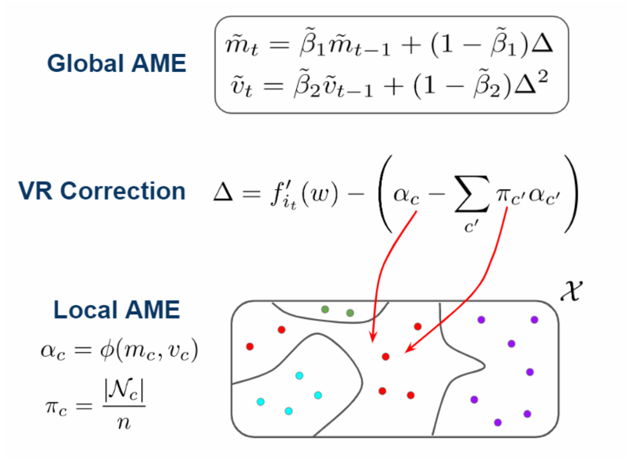
Global AME
$\tilde{m}_t = \tilde{\beta}_1\tilde{m}_{t-1} + (1-\tilde{\beta}_1)\Delta$
$\tilde{v}_t = \tilde{\beta}_2\tilde{v}_{t-1} + (1-\tilde{\beta}_2)\Delta^2$
VR Correction
$\Delta = f'_{i_t}(w) - \left(\alpha_c - \sum_{c'} \pi_{c'}\alpha_{c'}\right)$
Local AME
$\alpha_c = \phi(m_c, v_c)$
$\pi_c = \frac{|\mathcal{N}_c|}{n}$
$\mathcal{X}$

Figure 1. Graphical representation of our proposed algorithm. See Section 3 for details.

Recent works (Roux et al., 2012; Defazio et al., 2014; Hofmann et al., 2015; Johnson & Zhang, 2013) have been able to speed up stochastic gradient methods, obtaining linear convergence rate for finite sum problems. These works improve the convergence speed by reducing the variance of the estimation of the gradient by means of corrections applied to the raw gradients.

Moreover, (Brock et al., 2018) noticed that increasing the batch size in deep learning problems improved the obtained results efficiently. They hypothesize that this is due to the variance reduction introduced by the use of large batch size. Paradoxically, (Defazio & Bottou, 2018) show that these algorithms do not improve upon the performance of popular algorithms for deep learning optimization like Adam and AMSGrad (Kingma & Ba, 2015; Reddi et al., 2018) that are based on adaptive moment estimation (AME).

Our contribution: We propose to combine variance reduction with algorithms based on adaptive moment estimation by leveraging the structure of the data on the input

Final Project. *IFT6132 - Structured Prediction and Optimization*. Université de Montéal, 2019.

space. We expect this to be beneficial in the deep learning setting since it allows us to obtain low-variance updates (similar to the large-batch regime), without having the computational overhead. Moreover, compared to other variance reduction techniques, our approach can scale to datasets with large numbers of datapoints since we do not store control variates per datapoint.

Our work includes:

- Understanding and implementing two AME algorithms: Adam and AMSGrad.
- Understanding and using Stochastic Variance Reduced Gradient (SVRG) (Johnson & Zhang, 2013) as a baseline variance reduction based algorithm.
- Implementing our algorithm and conducting experiments in the convex (logistic Regression) and non-convex (multi-layer perception) settings.
- Analyzing the performance of our algorithm in comparison to AMSGrad and SVRG.

Our code is available at `github.com/jgalle29/vrame`.

2. Related work

The last decade has witnessed major achievements regarding the improvement of the convergence rate of first-order stochastic gradient-based methods. Two of the most fruitful directions are variance reduction techniques and adaptive methods based on estimates of the moments of the stochastic gradient.

In the context of optimizing a finite sum objective, where the sum is strongly convex, SAG (Roux et al., 2012) achieves linear convergence rate. For non-strongly convex problems, SAGA (Defazio et al., 2014) allows for a linear convergence rate as well and . However, these incremental gradient algorithms have big memory requirements as they store the gradient estimates for each data points, which is especially not practical in the deep learning setting. In the same context, the adaptation of SDCA (Shalev-Shwartz & Zhang, 2013) to deep learning is not practical because it requires the storage of all dual variables. SVRG (Johnson & Zhang, 2013) from the other side is memory-friendly and enjoys the same fast convergence rate as those of SAG. Nevertheless, recent works (Defazio & Bottou, 2018; Chavdarova et al., 2019) point out the ineffectiveness of the naive application of SVRG and other variance reduction techniques in the context of non-convex optimization problems, that characterize the training of deep learning models.

A different line of research has to do with algorithms relying on estimates of the moments of the stochastic gradients, which have proven to work well for the training of deep networks tasks and are currently the community default in deep learning. We refer to this family of methods as AMEs. Among these, two remarkable examples are Adam (Kingma & Ba, 2015) and AMSGrad (Reddi et al., 2018). These methods keep estimates for the first and second moments of the gradient in order to perform a coordinate-wise tuning of the learning rate. (Chavdarova et al., 2019) remark on the dilemma that arises when trying to combine VR techniques with AMEs. Since VR performs corrections on the estimates of the stochastic gradient so as to decrease its variance, a naive combination of this can result in an arbitrary growth in the step size, due to the division by small estimates of the second moment of the gradient.

(Hofmann et al., 2015) provides a unified analysis technique for *uniform memorization algorithms* (UMA) which encompass SAGA and SVRG. A generic UMA aims at minimizing a finite sum objective and performs the following updates:

$$w^+ = w - \gamma g_i(w), \qquad g_i(w) = f_i'(w) - \bar{\alpha}_i \tag{1}$$

$$\bar{\alpha}_i = \alpha_i - \frac{1}{n}\sum_{j=1}^{n} \alpha_j \qquad \alpha_j^+ = \begin{cases} f_j'(w) & \text{if } j \in J \\ \alpha_j & \text{else,} \end{cases} \tag{2}$$

where J is an arbitrary subset of $\{1, ..., n\}$, which might depend on j.

Within this framework, they introduce $\mathcal{N}$-SAGA as a way to share gradient information between similar datapoints which trades off between computation time and exactness. $\mathcal{N}$-SAGA takes advantage of additional structure in the form of *neighborhoods* for each datapoint $\mathcal{N}_i$, which determine which training points are considered similar to example i. This is a proxy for comparing how similar the corresponding functions f_i are under the assumption of a well behaved loss l. $\mathcal{N}$-SAGA corresponds to an *approximate* UMA which replaces the memory α_i with an approximation . However, thid approximation does not necessarily contain the gradients $f_i'(\tilde{w})$ (for some past iterate $\tilde{w}$), but rather the most recent gradient $f_j'(w)$ for the latest sampled point j such that $i \in \mathcal{N}_j$.

3. Method

As mentioned previously, exploiting the finite sum structure of certain optimization problems allowed for the development of algorithms based on stochastic gradients with linear convergence rates. We propose to take one step further and make use of extra structure present in ERM problems. Namely, we are trying to optimize a specific type of finite sum: one in which all the terms are the same loss function $l(w, (x, y))$ evaluated at different training datapoints $\{(x_i, y_i)\}_{i=1}^{n}$.

Consider the variance of the stochastic gradient ∇f. If our goal is to obtain an estimate of ∇f with low variance, a sensible way to achieve this is by trying to get a sample of functions f_i which is as diverse as possible. Here is where can take advantage of the fact that the f_i functions share their internal structure.

Let $z_i = (x_i, y_i)$. Assuming that the loss function l is sufficiently well-behaved (for instance, having L-(locally) Lipschitz gradients for each w), we can compare the gradients of the loss at different training instances:

$$d(\nabla_w l(w, z_i), \nabla_w l(w, z_j)) \leq L\, d(z_i, z_j) \tag{3}$$

This inequality implies that, provided that two training points are close enough, the gradient we observe at one point is a decent approximation of the gradient at the other.

Formally, we make use of a collection of neighborhoods $\mathcal{N} = \{\mathcal{N}_c\}_{c=1}^{C}$ which partitions the training dataset. For each of the neighborhoods, we define their population ratio $\pi_c = \frac{|\mathcal{N}_c|}{n}$. Such a choice of ratios ensures that we have unbiased gradient updates.

This neighborhood structure can be thought of as a hyperparameter in our method. In the experiments presented below, we construct the neighborhoods based only on the class labels. However, more challenging datasets (including regression problems) might require the used of more sophisticated techniques, like clustering algorithms.

Let us now explain in detail how our proposed algorithm operates. These steps are summarized in Algorithm 1 and illustrated in Figure 1.

Algorithm 1 Inner loop of the proposed algorithm

```
Sample community $\mathcal{N}_c \sim \pi := (\frac{|\mathcal{N}_{c'}|}{n})_{c' \in [1:C]}$
Sample $(x_{i_t}, y_{i_t})$ uniformly within $\mathcal{N}_c$
AMSGrad update on local $m_c$ and $v_c$ using $f'_{i_t}(w)$
$\alpha_c \leftarrow m_c \frac{|m_c|}{\sqrt{V_c}}$
$\Delta \leftarrow f'_{i_t}(w) - (\alpha_c - \sum_{c'} \pi_{c'} \alpha_{c'})$
AMSGrad update on global $\tilde{m}$ and $\tilde{v}$ using $\Delta$
$w \leftarrow w - \gamma \frac{\tilde{m}}{\sqrt{\tilde{v}}}$
```

During training, at time t, we perform hierarchical sampling: first sample a neighborhood $\mathcal{N}_c$ according to the ratios π and then sample a batch of examples i_t uniformly, with replacement, within the community $\mathcal{N}_c$. We then compute the gradient of the loss with respect to the sampled minibatch, $f'_{i_t}(w_t)$, and perform a local-AME update[1]:

$$m_c = \beta_1 m_c + (1 - \beta_1) f'_{i_t}(w_t) \tag{4}$$
$$v_c = \beta_2 v_c + (1 - \beta_2) f'_{i_t}(w_t)^2 \tag{5}$$
$$\alpha_c = m_c \frac{|m_c|}{\sqrt{v_c}} \frac{\sqrt{1 - \tilde{\beta}_2^{t+1}}}{1 - \tilde{\beta}_1^{t+1}} \tag{6}$$

We made the choice of using local AMEs as a way to stabilize the training and reduce the variance of the control variates. This is achieved since the moving-average nature of our local AME estimates is more robust to noise in the choice of a datapoint in the community than the full refresh update proposed by $\mathcal{N}$-SAGA.

We then use the directions provided by each of the local AME optimizers as control variates to apply a correction on the minibatch estimate of the stochastic gradient:

$$\Delta = f'_{i_t}(w_t) - \left(\alpha_c - \sum_{c'} \pi_{c'} \alpha_{c'} \right) \tag{7}$$

This corrected gradient is then passed to the global AME, which uses it to update its estimates for the first and second moments and then perform a model update:

$$\tilde{m} = \tilde{\beta}_1 \tilde{m} + (1 - \tilde{\beta}_1) \Delta \tag{8}$$
$$\tilde{v} = \tilde{\beta}_2 \tilde{v} + (1 - \tilde{\beta}_2) \Delta^2 \tag{9}$$
$$w_{t+1} = w_t - \gamma \frac{\tilde{m}}{\sqrt{\tilde{v}}} \frac{\sqrt{1 - \tilde{\beta}_2^{t+1}}}{1 - \tilde{\beta}_1^{t+1}} \tag{10}$$

In contrast to $\mathcal{N}$-SAGA, rather than having per-data point memories which get fully updated once a neighbor is sampled, we propose to use per-neighborhood memories which are updated updated following an AME scheme. These per-neighborhood statistics are then used as control variates to perform variance reduction. This is closely related to the idea of local SGD (see (Lin et al., 2018) for a recent application). However, in our proposal we use the local estimates of the gradient as control variates rather than applying them directly in the update of the iterates.

Note that when the number of communities $C = 1$, we recover the behavior of an AME; while the SAGA algorithm corresponds to setting $C = n$ and the parameters of the local and global AMEs to only keep the most recent gradient and constant identity for the second moment estimates.

We highlight that the use of the additional neighborhood structure (proposed by $\mathcal{N}$-SAGA) allows us to interpolate between the behaviors of an AME and SAGA, while keeping the application of VR feasible for DL settings, in which the memory cost of standard SAGA is prohibitive. This is because, in practice, the number of neighborhoods grows considerably slower than the number of points in the dataset.

[1]We actually employ the same maximum v_t book-keeping as AMSgrad.

4. Experiments

Convergence speed

We conducted experiments on the MNIST dataset for the digits classification problem to measure the performance of our algorithm, compared to SVRG and AMSGrad, in both the convex and non-convex settings using logistic regression and multi-layer perceptron (MLP) classifiers.

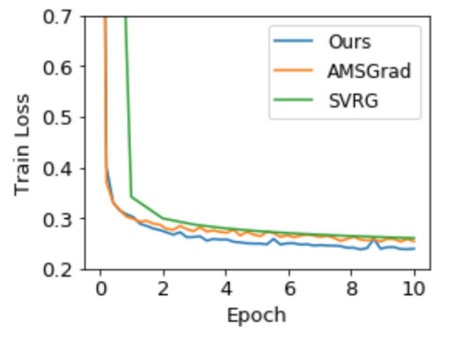

Figure 2. **Convex setting:** evolution of the training loss for logistic regression on MNIST.

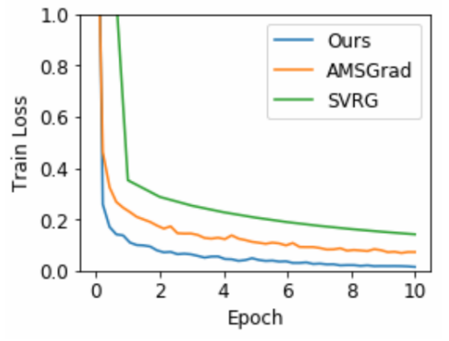

Figure 3. **Non-convex setting:** evolution of the training loss for MNIST classification with a 1-layer MLP with 100 tanh units.

Figures 2 and 3 show the evolution of the training loss for the three algorithms. We see that our method is marginally better than AMSGrad and SVRG for both tasks, and the loss has less fluctiations than AMSGrad. We remark that although our method obtained better performance here, we did not perform extensive experimentation on the parameters of AMSGrad and SVRG with our implementations.

Evolution of moment estimates

Consider the updates proposed by Adam and AMSGrad. As the value of the gradients goes to zero during training, the estimates of the second moment need to converge to zero faster than those of the second moment in order to avoid having large effective learning rates, as mentioned by (Chavdarova et al., 2018). We performed experiments on both Adam and AMSGrad to observe the evolution of m_t, v_t and α_t during the training.

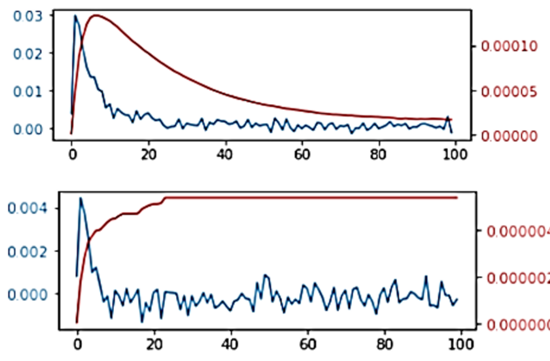

Figure 4. Evolution of the biased first (blue) and second (red) moment estimates using Adam (top) and AMSgrad (bottom) for a given parameter for logistic regression on MNIST.

Figure 4 shows the behavior observed for Adam and AMSGrad. We see that the maximum book-keeping on the second moment estimates occurs effectively in our implementation, and we use this book-keeping in both our local and global AMEs in order to avoid exploiding step sized as a consequence of variance reduction.

Variance reduction

In this section we examine whether our method effectively reduces the variance of the updates compared to standard AMEs. To evaluate the amount of variance reduction that we obtain using a given algorithm, we sample a large batch of examples at a given iteration and we calculate the the mean and standard deviation of the updates that would have been executed by the algorithm if it only had seen each one of the elements of said minibatch.

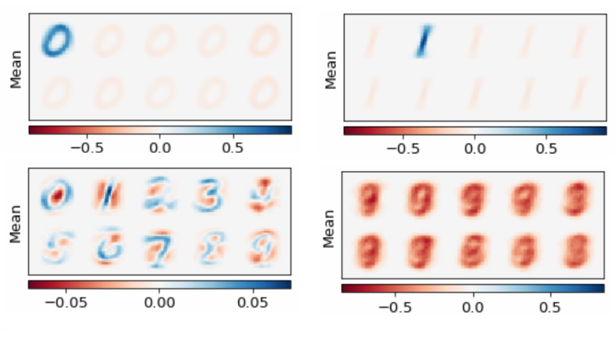

Figure 5. Qualitative analysis of the first update of our algorithm. Top: first-step control variates for classes 0 and 1. Bottom left: first-step average control variate. Bottom right: global optimizer first step.

At the beginning of training we perform a warm-start on each of the local AMEs by computing a first update on their moving averages. The value of the steps for the control variates of classes 0 and 1, as well as the average control variate and effective first update are displayed in Figure 5. Note that the clean gradient contained in each control variate comes from the separation into classes from the neighborhood structure. Moreover, the average control-variate shows a clean aggregation of the information coming from all of the communities. We remark that the first step performed by the global optimizer in our algorithm is equivalent to that of Adam and AMSGrad (see Section 5 for more details).

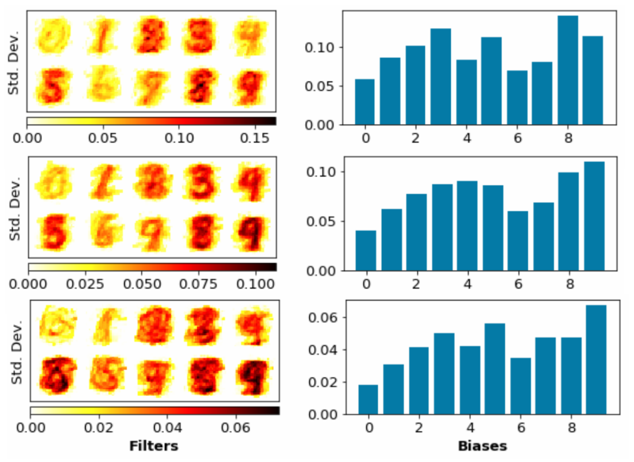

Figure 6. Standard deviation of updates around convergence for Adam (top), AMSGrad (middle) and our method (bottom).

Figure 6 shows the standard deviation of the updates for several methods close to convergence on the logistic regression task. We see that our method provides the a considerably smaller variance in the updates than the other two alternatives, indicating that our control variate technique was effective. We note that AMSGrad has a better behavior in terms of the variance of the updates compared to Adam, which is another reason to prefer AMSGrad over Adam, besides the theoretical arguments regarding its convergence.

5. Discussion

In our algorithm, we suggest a local AME update which is different from the standard update used by Adam and AMSGrad by a factor of $|m_t|$.

In fact, if one uses the standard AMSGrad update, we notice that the norm of our first update is very large. This is because we initialize $v_0 = 0$ and $m_0 = 0$, thus at the first iteration, (after bias correction) we have:

$$m_1 = (1-\beta_1) f'_{i_1}(w_0) \tag{11}$$

$$v_1 = (1-\beta_2) f'_{i_1}(w_0)^2 \tag{12}$$

$$\alpha_1 = \frac{\hat{m}_1}{\sqrt{\hat{v}_1}} = \frac{f'_{i_1}(w_0)}{|f'_{i_1}(w_0)|} = \text{sign}(f'_{i_1}(w_0)) \tag{13}$$

The first step in Adam (and AMSGrad) only takes into account the sign of the components of the gradient without keeping the value of the gradient. The gradient is generally not sparse (even for the first step using logistic regression), thus the norm of the first update is large which might result in a sudden increase of the loss. Adam deals with this issue by taking a very small initial step size. For next steps, the value of the norm of the is generally more controlled because m_t and v_t change in a way that the above simplification in equation 13 is no longer possible. However, our algorithm adds the value of α_1 to the average control variate $\sum_{c'} \pi_{c'} \alpha_{c'}$ and we keep having a large update norm, even after the first iteration when using the same learning rate as Adam.

To fix this issue, we use the new variant of the AMEs update, which has been suggested by (Chavdarova et al., 2019), in turn inspired by (Schaul et al., 2013). In contrast with (Chavdarova et al., 2019), we don't use this solution for the vanishing v_t problem, since -as previously mentioned in the experiments- it can be solved by using AMSGrad instead of Adam. Instead, we use it to correct the large first update in Adam and to make it coincide with the gradient in the first step. This also means that one might only use this alternative for the first update and continue the training using the standard scheme.

Further experiments that we run to evaluate variance reduction showed that Adam and AMSGrad do actually reduce the variance of their parameter updates during the training, and that this variance reduction is correlated with the values of β_1 and β_2. In fact, if we sample a batch of points and calculate the standard deviation of the parameter update, the contribution of their gradient to the direction update would increase as the values of β_1 and β_2 decrease. In practice, Adam and AMSGrad use high values of β_1 and β_2 (default values are $\beta_1 = 0.9$ and $\beta_2 = 0.999$) and these values would allow for some variance reduction. Our experiments show that AMSGrad introduces more variance reduction than Adam. They also show that our algorithm has the opposite behaviour and that the variance of the updates decreases as the values of β_1 and β_2 decrease. This can be explained by the fact that $f'_{i_t}(w_t)$ and $(\alpha_c - \sum_{c'} \pi_{c'} \alpha_{c'})$ are more correlated for small values of β_1 and β_2, thus allow for more variance reduction.

The purpose of our work was to have an algorithm that achieves variance reduction. However, (Tishby & Zaslavsky, 2015) proposes a two phases in the training of deep models using stochastic gradient methods: an initial *drift* phase in which the gradient provides a clear training signal, and a *diffusion* stage which perform a sort of random walk in parameter space and might be useful for generalization. We suspect that VR techniques might have a larger impact if applied only to this second stage of training.

6. Conclusions

In this work, we combined variance reduction with neighborhood-based adaptive moment estimation optimization algorithms. We showed that our algorithm achieves on-par performance as AMSGrad and SVRG in both convex and non-convex classification on MNIST. We achieve better levels of variance reduction compared to both algorithms. The number of neighborhoods that we use grows much slower than the number of data points, which provides better scalability compared to $\mathcal{N}$-Saga that leverages the input data structure as well. Our work allowed us to understand better the implicit variance reduction of parameter update in both Adam and AMSGrad, and to notice that AMSGrad achieves more variance reduction than Adam. We were also exposed to the issue of the combination of variance reduction with Adam, and offered some understanding to the issue linked to the first update in Adam. Future work would include experiments on datasets that have more challenging structure and where the neighborhood structure might play a larger role, as well as offering a better direction update formula, that would solve the problems exposed in the previous section.

References

Bottou, Léon and LeCun, Yann. Large scale online learning. In *Advances in Neural Information Processing Systems*, pp. 217–224, 2004.

Brock, Andrew, Donahue, Jeff, and Simonyan, Karen. Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Chavdarova, Tatjana, Stich, Sebastian, Jaggi, Martin, and Fleuret, François. Stochastic variance reduced gradient optimization of generative adversarial networks. In *Workshop on Theoretical Foundations and Applications of Deep Generative Models at International Conference on Machine Learning*, 2018.

Chavdarova, Tatjana, Stich, Sebastian, Jaggi, Martin, and Fleuret, Franois. Reducing noise in gan training with variance reduced extragradient. Preprint, to appear in arXiv., 2019.

Defazio, Aaron and Bottou, Léon. On the ineffectiveness of variance reduced optimization for deep learning. *arXiv preprint arXiv:1812.04529*, 2018.

Defazio, Aaron, Bach, Francis, and Lacoste-Julien, Simon. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

Hofmann, Thomas, Lucchi, Aurelien, Lacoste-Julien, Simon, and McWilliams, Brian. Variance Reduced Stochastic Gradient Descent with Neighbors. In *Advances in Neural Information Processing Systems*, pp. 2305–2313, 2015. URL `http://arxiv.org/abs/1506.03662`.

Johnson, Rie and Zhang, Tong. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in neural information processing systems*, pp. 315–323, 2013.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

Lin, Tao, Stich, Sebastian U., and Jaggi, Martin. Don't use large mini-batches, use local SGD. *CoRR*, abs/1808.07217, 2018. URL `http://arxiv.org/abs/1808.07217`.

Reddi, Sashank J, Kale, Satyen, and Kumar, Sanjiv. On the Convergence of Adam and Beyond. In *International Conference on Learning Representations*, 2018. ISBN 9781538610329. doi: 10.1134/S0001434607010294. URL `https://openreview.net/pdf?id=ryQu7f-RZ`.

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

Roux, Nicolas L, Schmidt, Mark, and Bach, Francis R. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems*, pp. 2663–2671, 2012.

Schaul, Tom, Zhang, Sixin, and LeCun, Yann. No more pesky learning rates. In *International Conference on Machine Learning*, pp. 343–351, 2013.

Shalev-Shwartz, Shai and Zhang, Tong. Stochastic Dual Coordinate Ascent methods for regularized loss minimization. *Journal of Machine Learning Research*, 14 (Feb):567–599, 2013.

Tishby, Naftali and Zaslavsky, Noga. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5. IEEE, 2015.