OCO Project

F. Genans Boiteux

May 25, 2023

Contents

1 Preliminary

Note on the dataset:

The '0' column contained no information, all values were equal to 0 so we removed this column.

The '5' column contained the label of each digit, from 0 to 9,we renamed it 'label'.

Rescaling of the data so that the pixel values are in [0,1].

Addition of an 'intercept' column (column with only 1s).

1.1 Theoretical context:

We want to classify handwritten digits using a Linear Support Vector Machine (SVM), dealing with two categories: $\{-1, 1\}$ where 1 represents the digit 0 and -1 all the other digits.

Whe have 28x28 pixels images that we represent as vectors of $\mathbb{R}^{784}$. We note $a_i \in \mathbb{R}^{785}$ an image and it's intercept, b_i its category.

Lets recall that by analogy with our lectures note we want to find $x \in \mathbb{R}^{785}$ that minimizes the following soft margin problem:

Given n images and labels $(a_i, b_i)_{1 \leq i \leq n}$:

$$\min_{x \in \mathbb{R}^{785}} f(x) := \left\{ \frac{1}{n} \sum_{1 \leq i \leq n} l_{a_i, b_i}(x) + \frac{\lambda}{2} ||x||^2 \right\}$$

where $l_{a,b}(x) = \text{hinge}(b \cdot x^T a) = \max\{0, 1 - b \cdot x^T a\}$.

Remark: For the basic gradient descent, we set $\lambda = 0$ and the function to minimize is convex. For $\lambda > 0$, this problem is strictly convex thanks to the l_2-norm regularization term.

In order to implement gradient descent we need to compute the gradient of f:

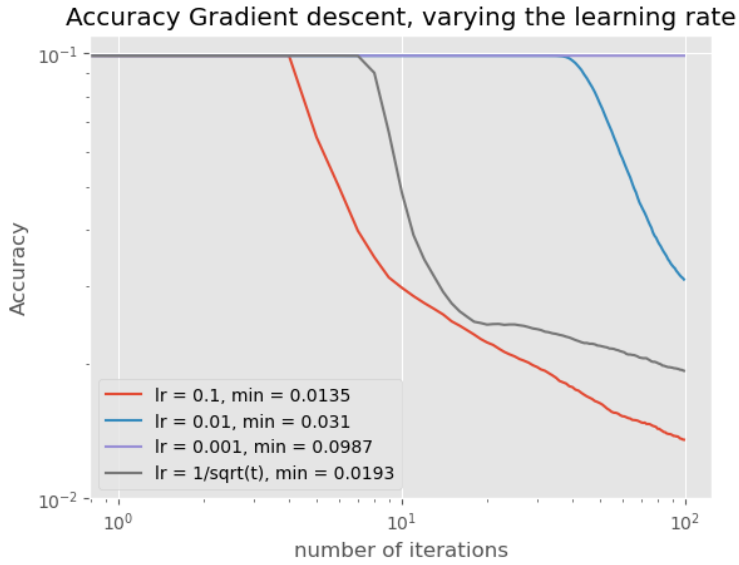
$$\nabla f(x) = \lambda x + \frac{1}{n} \sum_{1 \leq i \leq n} \nabla l_{a_i, b_i}(x)$$

where: $\nabla l_{a_i, b_i} = \begin{cases} -b_i \cdot a_i & \text{if } b_i \cdot x^T a_i \leq 1 \\ 0 & \text{otherwise} \end{cases}$ (1)

Remark: For all our algorithms, we usually took the gradient step in accordance with the theoretical results of the course.

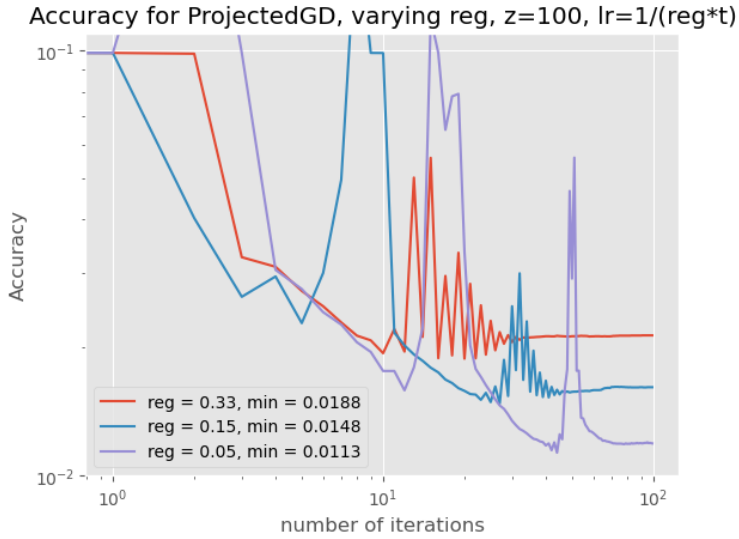
2 Gradient Descent

2.1 Unconstrained GD

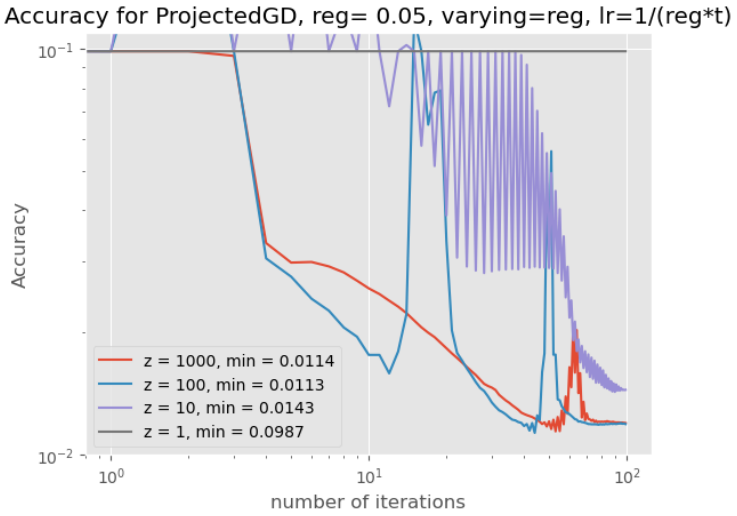


We observe that a too low learning rate tends to affect negatively the model and that despite having an optimal theoretical learning rate of $1/\sqrt{(t)}$, this learning rate seems too slow.

2.2 Projected GD

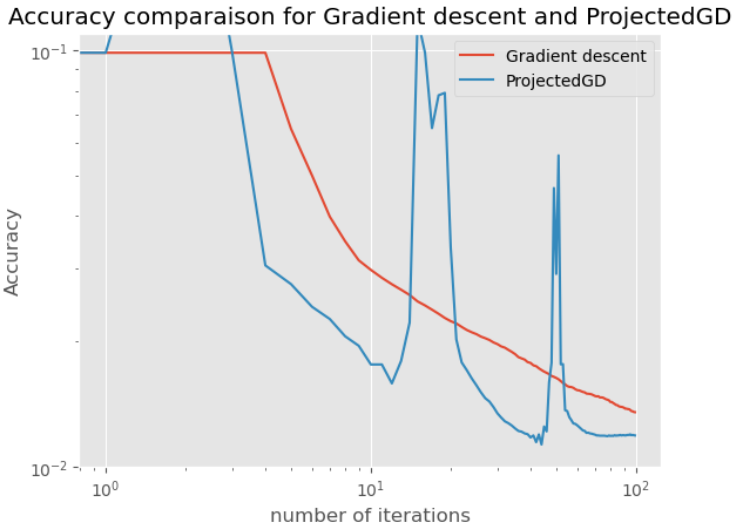


We set the hyperparameter z (the radius of the ball l_1) to 100 and took as learning rate $1/\lambda t$.
We can notice that the convergence is noisy, and that the lower the regularization parameter λ is, the better the result is.



We always take the learning rate $1/reg * \lambda t$, but we vary z the the diameter of the ball l_1. We notice that it's a bit tricky to find the best z, since a too big z would result in a useless projection but if z too small, the projection will put too much noise. The value $z = 100$ seems to be the best compromise between the best minimum and the quickest convergence.

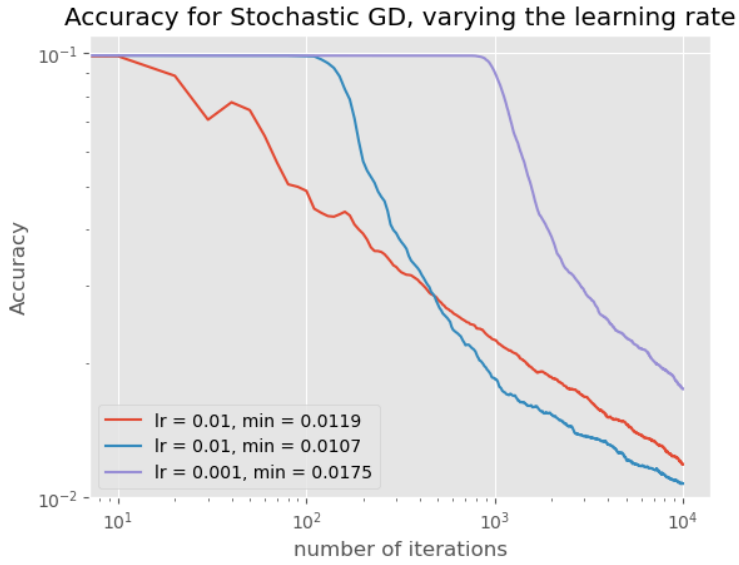
2.3 Comparison



The curve of ProjectedGD has three peaks, but its convergence is better.

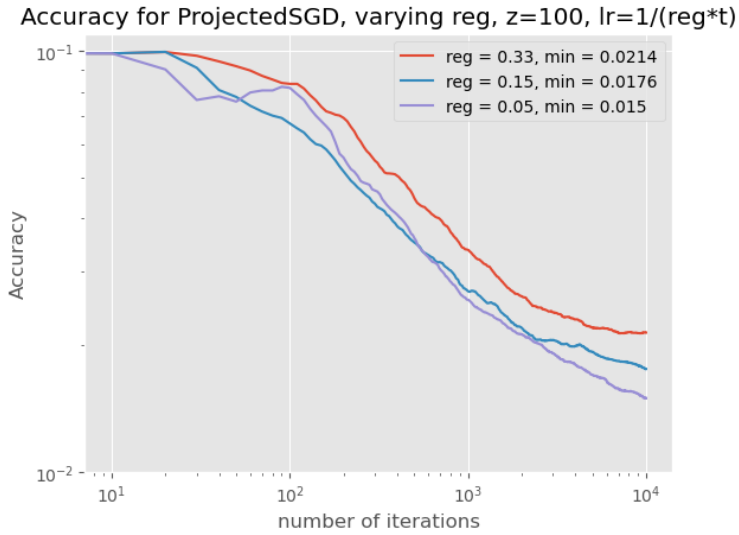
3 Stochastic Gradient Descent

3.1 Unconstrained SGD



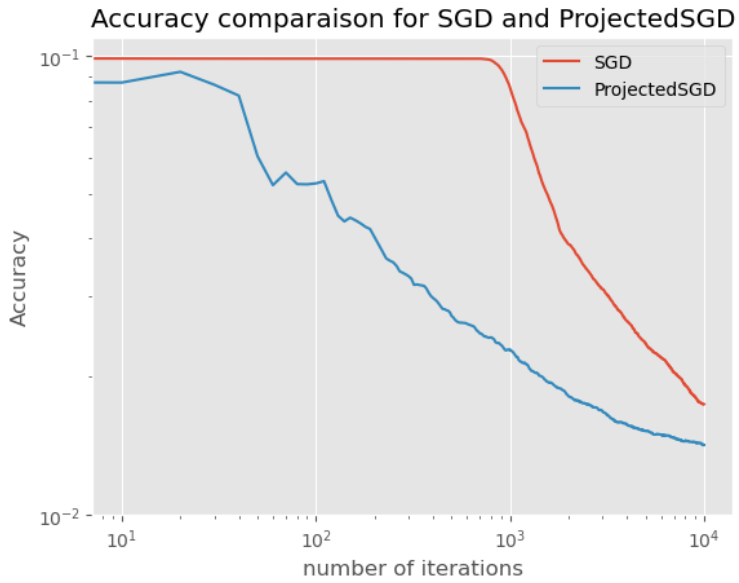
By varying the learning rate, we see that the convergence is better when it is larger.

3.2 Projected SGD



We set the hyperparameter z to 100 and took as learning rate $1/\lambda t$, as we did with Projected GD.
We can notice that the convergence is much less noisy with Projected SGD.
The lower the regularization parameter, the better the result, still.

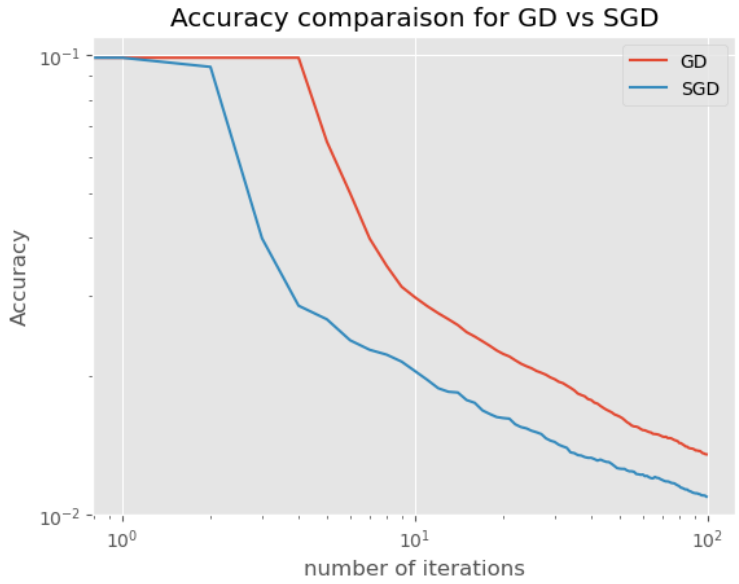
3.3 Comparisons



In accordance to what we have seen for GD, the convergence is quicker for Projected SGD.

```
Projected GD --- 190.45396399497986 seconds ---
Projected Stochastic GD --- 0.08901095390319824 seconds ---
Projected SGD is faster than Projected GD by a factor of  2140
```

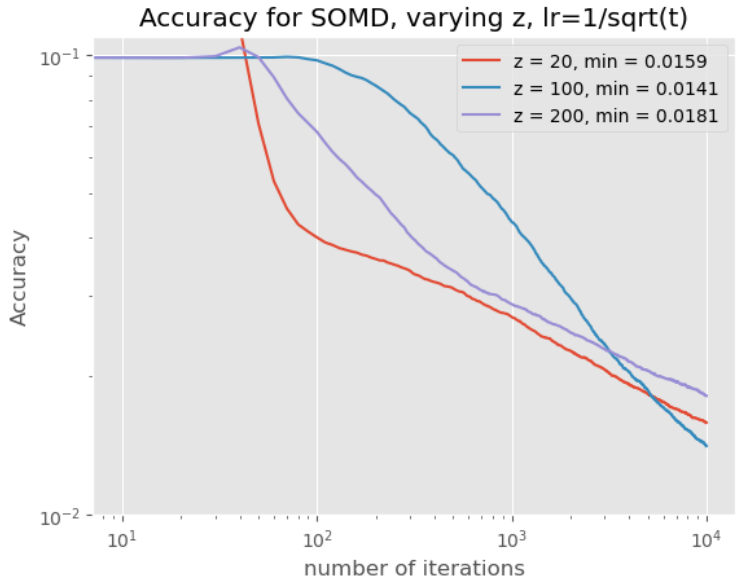
Since Stochastic GD is 1000 times fastter, we can have 1000 times more iterations to make our comparison.
We took our best candidates for each algorithm, based on our previous plots.



For the same time, we can see that Stochastic Gradient Descent is way more effective than Vanilla Gradient Descent.

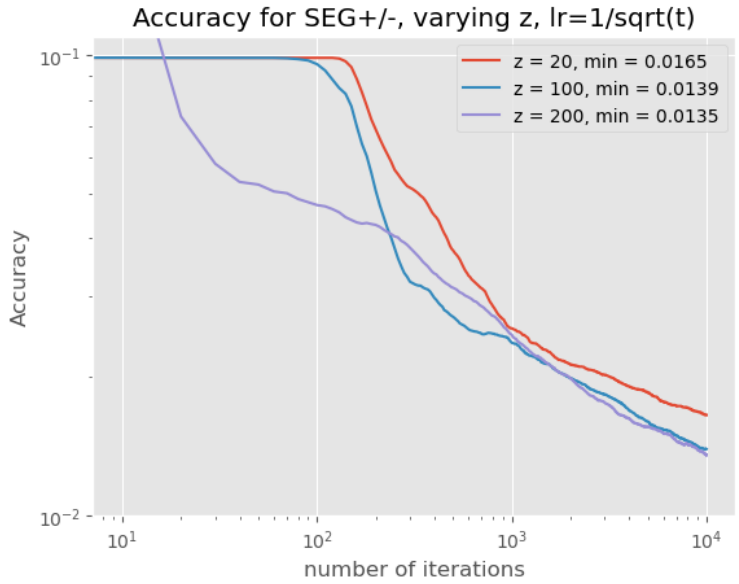
4 Regularized Follow the Leader

4.1 Stochastic Mirror Descent

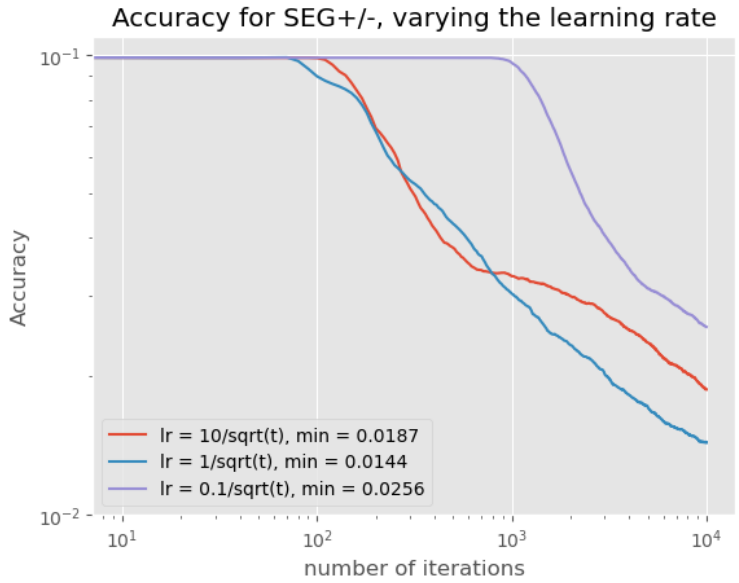


We take as learning rate $1/\sqrt{t}$, and we vary z.
As for ProjectedSGD and ProjectedGD, we have our best result with $z = 100$ even though the accuracy decreases way quicker at the beginning for $z = 20$.

4.2 Stochastic Exponentiated Gradient

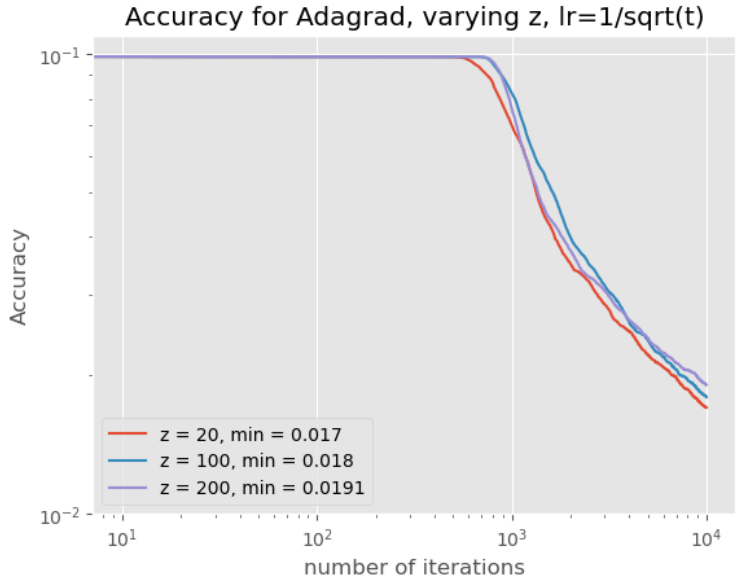


When we vary the hyperparameter z and fix the learning rate $1/\sqrt{t}$, similarly to the SOMG algorithm, the best convergence result is obtained for z=100.



By taking different values of the learning rate we get the best result for $1/\sqrt{t}$!

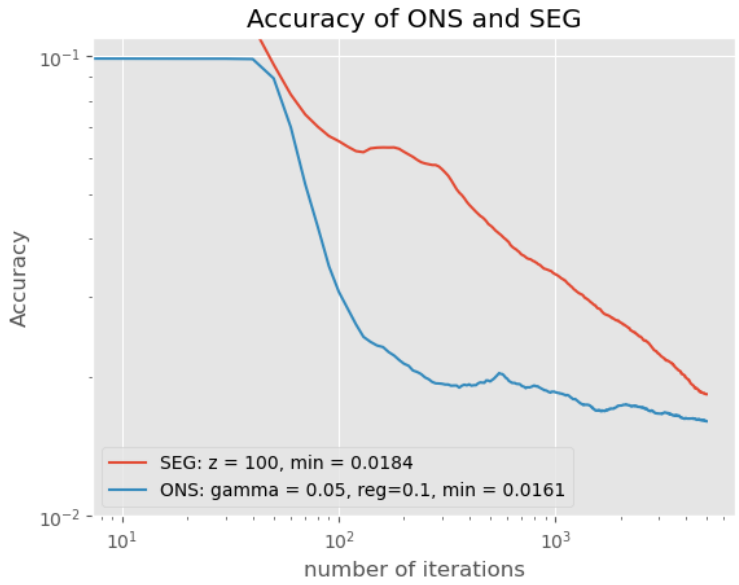
4.3 Stochastic AdaGrad



For the Adagrad algorithm, with the learning rate $1/\sqrt{t}$, the radius of ball l_1 does not seem to strongly affect the accuracy, at least when we compute it on the test set.

5 Online Newton Step

5.1 Accuracy comparison ONS - SEG+/-



We compare ONS with $\gamma = 0.05$ *and* $\lambda = 0.1$, to SEG with z=100.
We can see that for the same number of iterations, ONS converges way quicker at the beginning but but starts to plateau after 1000 iterations while the accuracy of SEG continues to decrease after 5 000 iterations.

```
ONS time for 100 iterations --- 39.85885500907898 seconds ---
SEG time for 100 iterations --- 0.1383500099182129 seconds ---
SEG is faster than ONS by a factor of  288
```

We notice that the ONS algorithm is much slower. This can be explained by the fact that we have a lot of matrix calculations and that they are not optimized. On another computer the speed difference was 5 times less. In any case, this may be due to a non-optimal implementation of the algorithm; either at the level of the projection on the simplex, or at the level of the matrix calculations themselves.

6 Research paper: Accelerating SGD using Predictive Variance Reduction

6.1 Introduction

The article we decided to investigate is the following: Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In Advances in Neural Information Processing Systems, pp. 315-323. 2013.

Motivations - The randomness of stochastic gradient descent (SGD) leads to variance in the gradients, and forces us to choose a learning rate η_t that decays to zero in order to insure convergence. This choice leads to a sub-linear convergence in $O(\frac{1}{t})$. The idea behind this modified version of SGD, called SVRG, is to reduce the variance of our gradient estimator and thus allow a higher learning rate.

Setting - Lets consider the following minimization problem:

$$(P) : \min_{x\in\mathbb{R}} \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

Theoretical guarantees: - Linear convergence $O\big((\frac{1-\alpha}{\beta})^t\big)$ if f is α-stricly convex and the f_i are convex and β-smooth, compared to $O(1/t)$ for SGD. - Convergence rate of $O(1/T)$ compared to $O(\frac{1}{\sqrt{T}})$ with SGD for a non-strongly convex problem.

6.2 Variance reduction

6.2.1 Principle

If X and Y are independent, then $\mathrm{var}(X+Y) = \mathrm{var}(X) + \mathrm{var}(Y)$. If they are not independent the covariance enters. Letting μ_X, μ_Y denote the means of X and Y, we have

$$\begin{aligned}\mathrm{var}(X+Y) &= E\left[(X+Y)^2\right] - (\mu_X+\mu_Y)^2 \\ &= E\left[X^2\right] - \mu_X^2 + E\left[Y^2\right] - \mu_Y^2 + 2\left(E[XY] - \mu_X\mu_Y\right) \\ &= \mathrm{var}(X) + \mathrm{var}(Y) + 2\,\mathrm{cov}(X,Y)\end{aligned}$$

The covariance is $\mathrm{cov}(X,Y) = \rho_{X,Y}\sigma_X\sigma_Y$ and ρ lies between -1 and 1 . If ρ is negative the variance of $X+Y$ is smaller than the sum of their variances.

6.2.2 Intuition to reduce the gradient variance

Stochastic gradient estimator - In SGD, at each iteration, we compute the unbiased estimator of $\nabla f(x_t)$: $\nabla f_I(x_t)$

Where $I \sim \mathcal{U}(\{1,\cdots,n\})$.

Let $z \in \mathbb{R}$.

We define: $g(x_t) = \nabla f_I(x) - \big(\nabla f_I(z) - \nabla f(z)\big)$.

We have:

$$E_I\big[g(x_t)\big] = E_I\big[\nabla f_I(x_t)] - E_I\big[\nabla f_I(z)\big] - \nabla f(z) = \nabla f(x_t)$$

so $g(x_t)$ is aso an unbiased estimator of $\nabla f(x_t)$.

6.2.3 SVRG Algorithm

Procedure SVRG

Parameters update frequency m and learning rate η
Initialize $\tilde{w}_0$
Iterate: for $s = 1, 2, \ldots$
  $\tilde{w} = \tilde{w}_{s-1}$
  $\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} \nabla \psi_i(\tilde{w})$
  $w_0 = \tilde{w}$
  Iterate: for $t = 1, 2, \ldots, m$
    Randomly pick $i_t \in \{1, \ldots, n\}$ and update weight
      $w_t = w_{t-1} - \eta(\nabla \psi_{i_t}(w_{t-1}) - \nabla \psi_{i_t}(\tilde{w}) + \tilde{\mu})$
  end
  option I: set $\tilde{w}_s = w_m$
  option II: set $\tilde{w}_s = w_t$ for randomly chosen $t \in \{0, \ldots, m-1\}$
end

The picture was taken from the article. To fit our notations, one should replace the ψ_i functions by f_i.

Theorem: Consider SVRG with option II. if f is α-stricly convex and the f_i are convex and β-smooth. Let $w_* = \arg\min_w f(w)$. Assume that m is sufficiently large so that

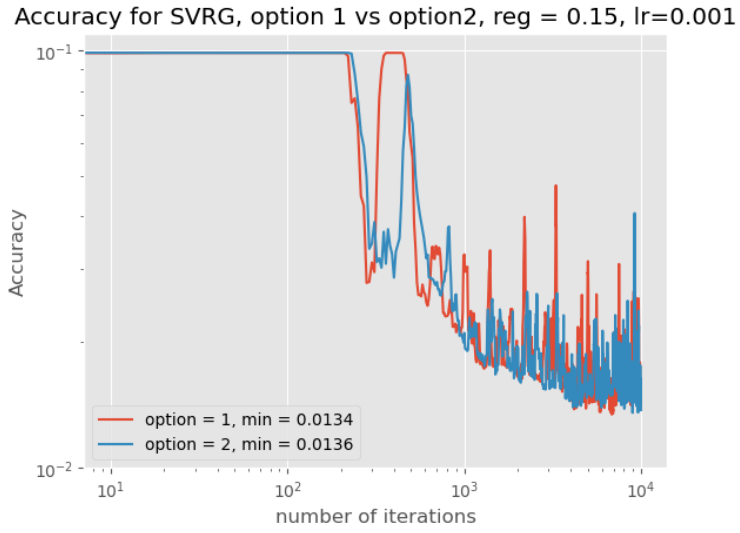
$$\lambda = \frac{1}{\alpha\eta(1 - 2\beta\eta)m} + \frac{2\alpha\eta}{1 - 2\alpha\eta} < 1,$$

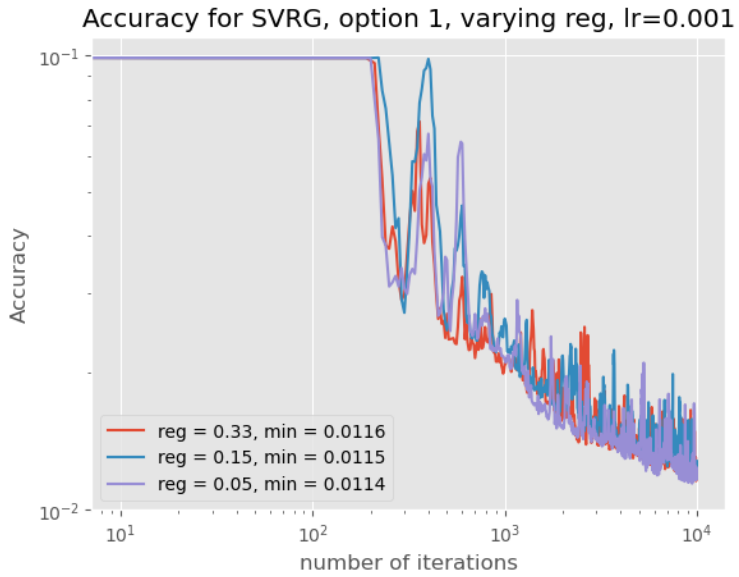
then we have geometric convergence in expectation for SVRG:

$$\mathbb{E} f\left(\tilde{w}_s\right) \leq \mathbb{E} f\left(w_*\right) + \lambda^s \left[f\left(\tilde{w}_0\right) - P\left(w_*\right)\right]$$

Remark: Our notation differs from that introduced in the article, in order to be more faithful to the one seen in class.

6.3 Implementation and comparison with other algorithms

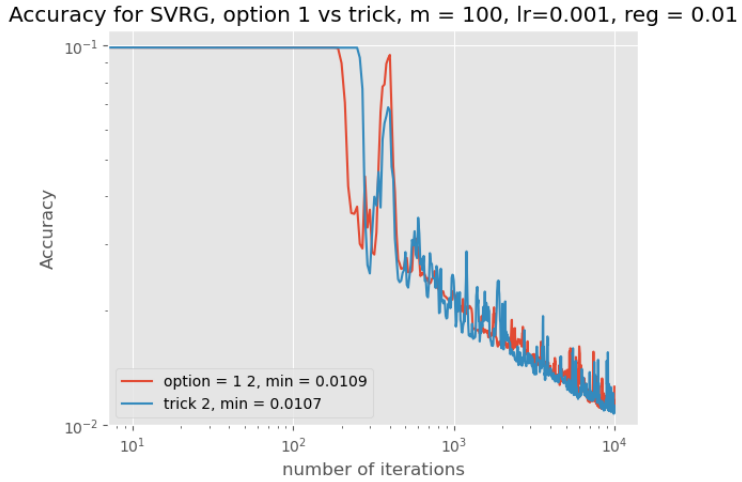




6.3.1 A little trick for option 1

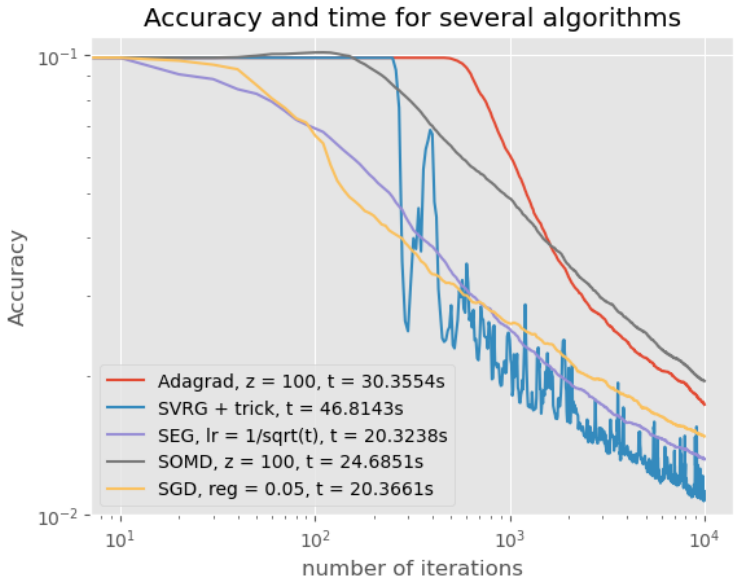
In the SVRG algorithm with option 1, despite computing the full gradient $\nabla f(w_m)$ every m steps, we don't use it to update x_t when $x_t = w_m$. Instead of that, we should use the information of the full gradient for the best uptade possible. For the algorithm pseudo-code, our idea is translated by:

if $t == 1$, **update** $w_t = w_{t-1} - \eta\mu$



On our exemple, the "trick" version seems to decrease the acuracy a bit quicker and hat a slightly better minimum. This little change can maybe help when we want the computation of the full gradient more often or if we want to increase the learning rate.

6.4 Accuracy and time comparison



Typo: On the plot, the curve labeled "SGD" is in fact ProjectedSGD with reg=0.05 and z=100.

Finally, we can see that SVRG may be a good optimization algorithm, but the addition of a new parameter to compute a full gradient makes it more difficult to configure. On the MNIST dataset, this approach was the most effective according to the minimum achieved, despite being noisy. Nevertheless, the result/time ratio is better for other algorithms such as Projected SGD or SEG+/-.

Since the main theorem in the article concerns β-smooth f_i and the hinge loss is not smooth, it is difficult to conclude whether the results from the article are true or false in practice even though we achieved really good results.

An idea developed in the paper was to first run a faster algorithm like SGD to quickly approach a local minimum, and then use SVRG. This technique is motivated by the desire to take advantage of the variance reduction that occurs when the difference between two gradient values is smaller. We did not test this approach, as it seems somewhat tedious to configure.