MLRG 2022: Adaptive Gradient Descent without Descent

Curtis Fox

UBC Department of Computer Science

August 2, 2024

Rough Outline

- Definitions
- Background/Motivation
- Algorithms
- Experimental Results
- Future Work

Problem Formulation

Our main goal is to solve a problem of the form:

$$\min_{x \in \mathbb{R}} f(x)$$

where $f : \mathbb{R}^d \to \mathbb{R}$ and f is differentiable.

Definitions

- In order to solve the problem above, we can use a technique called gradient descent. The algorithm involves the following iteration:

$$x_{k+1} = x_k - \lambda \nabla f(x_k), k \geq 0, \lambda > 0$$

- For the duration of the talk, we will assume that f has a minimum, and will denote f^* as the minimum value of f.

Definitions

- Convexity:

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle \quad \forall x, y$$

  - The function f is bounded below by its linear approximations.

- Smoothness:

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2 \quad \forall x, y$$

  - The function f can be bounded above by a quadratic at every point.
  - We call L the Lipschitz constant.
  - Also implies that:

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

Motivation

Some issues that can arise with GD:

- Many functions do not satisfy smoothness globally.
- Figuring out the best stepsize λ can be challenging, requiring guessing many stepsizes.
- Robustness issues: Picking too large of a stepsize can lead to divergence.
- GD is too slow: Even if L is finite, this may not be a good representation of local smoothness, and so we pick too small of stepsize.

Background

Ways that previous work get around some of these issues, among many:

- Do a line search, which involves picking a stepsize λ_k so that a condition such as the following holds:

$$f(x_k - \lambda_k \nabla f(x_k)) \leq f(x_k) - c\lambda_k \|\nabla f(x_k)\|^2$$

  - If f is costly to compute, this won't be very efficient.
- Use an adaptive Polyak's stepsize:

$$\lambda_k = \frac{f(x_k) - f^*}{\|\nabla f(x_k)\|^2}$$

  - This requires knowing f^*, which isn't always possible.

Key Ideas

The paper states two key ideas to effectively automate gradient descent:

- Don't increase the step size too quickly.
- Don't overstep the local curvature.

In particular:

- Pick the stepsize to be an estimate of the inverse **local** Lipschitz constant.
- Why? The global Lipschitz constant may not be a good estimate of the local curvature of the function being minimized.

Stepsize Selection

The main idea of the paper is to use a stepsize λ_k so that the following two inequalities hold:

$$\lambda_k \leq \lambda_{k-1}\sqrt{(1+\theta_{k-1})}$$

$$\lambda_k \leq \frac{\|x_k - x_{k-1}\|}{2\|\nabla f(x_k) - \nabla f(x_{k-1})\|}$$

where $\theta_k = \frac{\lambda_k}{\lambda_{k-1}}$.

- The stepsize picked in the current iteration uses the stepsize, iterates, and gradients from the previous step.
- The idea here is using information from both the current and previous step can allow one to estimate the local curvature.

Adaptive GD Algorithm

Algorithm 1 Adaptive gradient descent

1: **Input:** $x^0 \in \mathbb{R}^d$, $\lambda_0 > 0$, $\theta_0 = +\infty$
2: $x^1 = x^0 - \lambda_0 \nabla f(x^0)$
3: **for** $k = 1, 2, \dots$ **do**
4: $\quad \lambda_k = \min\left\{\sqrt{1+\theta_{k-1}}\lambda_{k-1}, \frac{\|x^k - x^{k-1}\|}{2\|\nabla f(x^k) - \nabla f(x^{k-1})\|}\right\}$
5: $\quad x^{k+1} = x^k - \lambda_k \nabla f(x^k)$
6: $\quad \theta_k = \frac{\lambda_k}{\lambda_{k-1}}$
7: **end for**

- The same as GD with a more elaborate stepsize scheme.
- Line 4 is where the estimation of the local Lipschitz constant occurs.

Definition

- Strong Convexity:

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\mu}{2}\|y - x\|^2 \quad \forall x, y$$

  - The function f can be bounded below by a quadratic at every point (the same as the smoothness definition but a lower bound instead).

Adaptive Accelerated GD Algorithm

Algorithm 2 Adaptive accelerated gradient descent

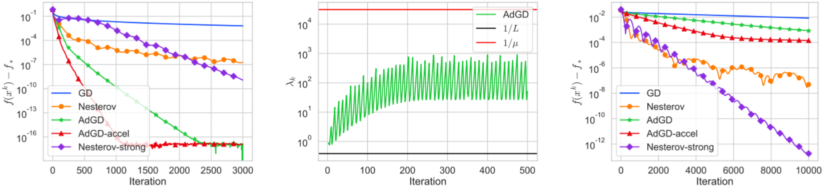
1: **Input:** $x^0 \in \mathbb{R}^d$, $\lambda_0 > 0$, $\Lambda_0 > 0$, $\theta_0 = \Theta_0 = +\infty$
2: $y^1 = x^1 = x^0 - \lambda_0 \nabla f(x^0)$
3: **for** $k = 1, 2, \dots$ **do**
4: $\quad \lambda_k = \min\left\{\sqrt{1 + \frac{\theta_{k-1}}{2}}\lambda_{k-1}, \frac{\|x^k - x^{k-1}\|}{2\|\nabla f(x^k) - \nabla f(x^{k-1})\|}\right\}$
5: $\quad \Lambda_k = \min\left\{\sqrt{1 + \frac{\Theta_{k-1}}{2}}\Lambda_{k-1}, \frac{\|\nabla f(x^k) - \nabla f(x^{k-1})\|}{2\|x^k - x^{k-1}\|}\right\}$
6: $\quad \beta_k = \frac{\sqrt{1/\lambda_k} - \sqrt{\Lambda_k}}{\sqrt{1/\lambda_k} + \sqrt{\Lambda_k}}$
7: $\quad y^{k+1} = x^k - \lambda_k \nabla f(x^k)$
8: $\quad x^{k+1} = y^{k+1} + \beta_k (y^{k+1} - y^k)$
9: $\quad \theta_k = \frac{\lambda_k}{\lambda_{k-1}}$, $\Theta_k = \frac{\Lambda_k}{\Lambda_{k-1}}$
10: **end for**

- More steps involved in the accelerated variant, but line 4 is almost the same as in the standard variant.
- Can think of this variant as using momentum for faster convergence.
- Line 5 attempts to estimate the local strong convexity constant.

Adaptive SGD Algorithm

- The paper goes further and discusses a stochastic variant of their idea.
- Missing theory for this variant.

Experimental Results



(a) Mushrooms dataset, objective (b) Mushrooms dataset, stepsize (c) Covtype dataset, objective

Figure 1: Results for the logistic regression problem.

- We see that the adaptive method performs better for the dataset used in the left, but not the right for the logistic regression problem.

Experimental Results

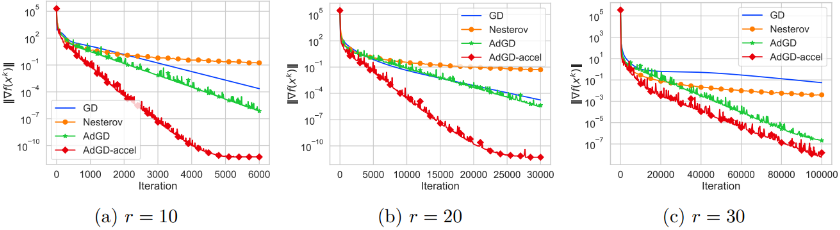
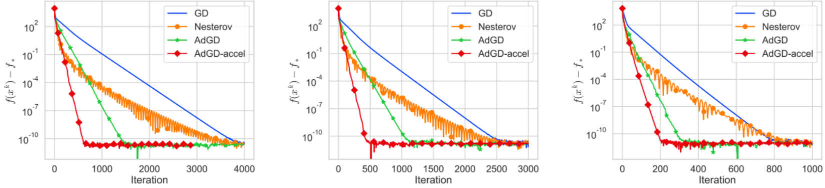


Figure 2: Results for matrix factorization. The objective is neither convex nor smooth.

- The adaptive method works better for the matrix factorization problem (for varying ranks). The accelerated variant is even faster, as expected.

Experimental Results

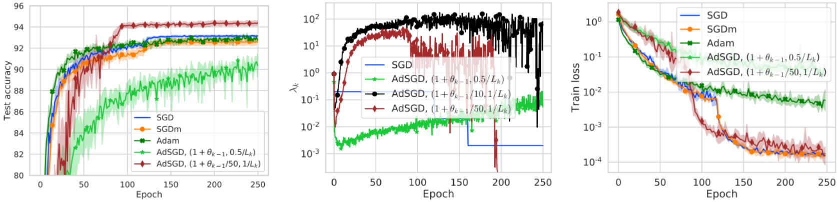


(a) $M = 10$ (b) $M = 20$ (c) $M = 100$

Figure 3: Results for the non-smooth subproblem from cubic regularization.

- Again the adaptive method performs the best, and this is the case for varying levels of regularization.

Experimental Results

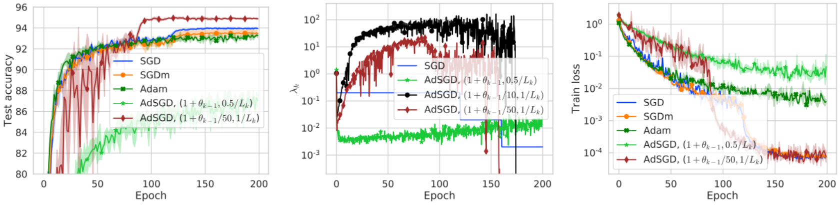


(a) Test accuracy (b) Stepsize (c) Train loss

Figure 5: Results for training ResNet-18 on Cifar10. Labels for AdGD correspond to how λ_k was estimated.

- The adaptive SGD method gives the best test loss, and comparable training loss to the other methods.

Experimental Results



(a) Test accuracy (b) Stepsize (c) Train loss

Figure 6: Results for training DenseNet-121 on Cifar10.

- Same as before, the adaptive SGD method gives the best test loss, and comparable training loss to the other methods.

Future Research Directions

Some future directions to extend this paper:

- Handing the case where f is non-convex.
  - This paper assumes convexity in their proofs. It's not clear how to extend their methods beyond this case.
- Their theoretical rates are the same as GD (up to constants). However, experimentally they show that their method often performs better than GD, and it's unclear why.
- Better theoretical results for both the accelerated and stochastic variants of their algorithm.

Finale

Thank you for listening!

Questions?

References

Malitsky, Y. and Mishchenko, K. (2019).
Adaptive gradient descent without descent.