ADAPTIVE BACKTRACKING FOR FAST OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Backtracking line search is foundational in numerical optimization. The basic idea is to adjust the step size of an algorithm by a *constant* factor until some chosen criterion (e.g. Armijo, Goldstein, Descent Lemma) is satisfied. We propose a novel way to adjust step sizes, replacing the constant factor used in regular backtracking with one that takes into account the degree to which the chosen criterion is violated, with no additional computational burden. This light-weight adjustment leads to significantly faster optimization, which we confirm by performing a variety of experiments on over fifteen real world datasets. For convex problems, we prove adaptive backtracking requires fewer adjustments to produce a feasible step size than regular backtracking does for two popular line search criteria: the Armijo condition and the descent lemma. For nonconvex smooth problems, we prove adaptive backtracking enjoys the same guarantees of regular backtracking.

1 INTRODUCTION

We consider learning settings that can be posed as the unconstrained optimization problem

$$\arg\min_{x \in \mathbb{R}^d} F(x). \tag{1}$$

Typically, algorithms solve (1) iteratively, refining the current iterate x_k by taking a step $\alpha_k d_k$:

$$x_k + \alpha_k d_k. \tag{2}$$

Here, α_k is the size of the step taken in the direction d_k. Examples of iteration algorithms include gradient descent (GD), Newton's method, quasi-Newton methods (Moré & Sorensen, 1982; Nocedal & Wright, 2006), Nesterov's accelerated gradient method (AGD) (Nesterov, 1983), adaptive gradient methods (Ruder, 2016) and their stochastic and coordinate-update variants (Boyd et al., 2011). To find an appropriate step size, iterative algorithms typically call a *line search* (LS) subroutine, which adopts some criterion and adjusts a tentative step size until this criterion is satisfied. For many popular criteria, if the direction d_k selected by the base algorithm is somewhat aligned with the gradient of F, then a feasible step size can be produced in a finite number of updates by successively adjusting an initial tentative step size until the criteria are satisfied. The standard practice for this process, known as *backtracking*, is to multiply the tentative step size by a *predefined constant factor* to update it. We propose a simple alternative to standard practice:

to adjust the step size by an ***online variable*** *factor that depends on the line search criterion violation.*

While in principle this idea can be applied to many criteria, this paper focuses on illustrating it in the context of two line search criteria: the Armijo condition (Armijo, 1966), arguably the most popular example of such criteria, and the descent lemma (Bertsekas, 1999, proposition A.24) in the context of composite objectives (Beck & Teboulle, 2009). After motivating our choices of online adaptive factors, we show that they enjoy the best theoretical guarantees one can hope for. To conclude, we present numerical experiments on several real-world problems confirming that using online adaptive factors in line search subroutines can *produce higher-quality step sizes* and *significantly reduce* the total number of function evaluations standard backtracking subroutines require.

Contributions. Our contributions can be summarized as follows.

- In Section 2, we propose a template for adaptive backtracking procedures with broad applicability.

- In Section 2.3, we apply the template to enforce the Armijo condition and, in Section 3, present experiments on real-world problems showcasing that the adaptive subroutine outperforms regular backtracking and improves the performance of standard baseline optimization algorithms.
- In Section 2.4, we apply the template on proximal-based algorithms to satisfy the descent lemma and present more real-world problems in Section 3 illustrating that the adaptive subroutine improves the performance of FISTA.
- For both subroutines, in Section 4 we prove that for convex problems, adaptive backtracking takes no more function evaluations to terminate than regular backtracking in any single iteration. We also give global theoretical guarantees for adaptive backtracking in nonconvex smooth problems, which match those of regular backtracking.

2 Adaptive backtracking

2.1 Line search: criteria and search procedures

Every line search subroutine can be decomposed into the *criteria* that it enforces and the *procedure* it uses to return a feasible step size. We now briefly discuss each component and provide examples.

Criteria. The most popular of line search criteria is the Armijo condition (Armijo, 1966), which requires that the objective function sufficiently decrease along consecutive iterates. Other popular sets of criteria are the weak and strong Wolfe conditions (Wolfe, 1969), which comprise the Armijo condition and an additional curvature condition that prevents excessively small step sizes and induces step sizes for which the objective function decreases even more. In contrast, nonmonotone criteria (Grippo et al., 1986; Zhang & Hager, 2004) only require that some aggregate metric of the objective function values (e.g., an exponential moving average) decrease along consecutive iterates.

Search procedures. The second component of a line search subroutine is the procedure that finds a step size satisfying the target criteria. For example, Wolfe line search is often implemented by bracketing procedures based on polynomial interpolation (Nocedal & Wright, 2006, pp. 60–61). In contrast, several criteria consisting in a single condition such as Armijo and nonmonotone, are provably satisfied by sufficiently small step sizes. For them, the standard procedure to compute step sizes fixes an initial tentative step size and then consecutively multiplies it by a constant $\rho \in (0, 1)$ until the criteria are satisfied. This subroutine is generally known as *backtracking line search* (BLS).

2.2 Adaptive backtracking

BLS often enforces an inequality that is *affine* in the step size. In this case, BLS can be reformulated as computing $v(\alpha_k)$, which is less than 1 when the line search criterion evaluated at the tentative step size α_k is violated, and then scaling α_k by a factor until $v(\alpha_k)$ is greater than 1. Regular BLS (Algorithm 1) employs a fixed factor $\rho \in (0, 1)$. We propose a simple modification of this subroutine:

to replace ρ with an adaptive factor $\hat{\rho}(v(\alpha_k))$ chosen as a nontrivial function of the violation $v(\alpha_k)$.

Algorithm 1 Backtracking Line Search

Input: $\alpha_0 > 0, v\colon \mathbb{R}_+ \to \mathbb{R}, \rho \in (0, 1)$
Output: α_k

1: $\alpha_k \leftarrow \alpha_0$
2: **while** $v(\alpha_k) < 1$ **do**
3: $\quad \alpha_k \leftarrow \rho \cdot \alpha_k$
4: **end while**

Algorithm 2 Adaptive Backtracking Line Search

Input: $\alpha_0 > 0, v\colon \mathbb{R}_+ \to \mathbb{R}, \hat{\rho}\colon \mathbb{R} \to (0, 1)$
Output: α_k

1: $\alpha_k \leftarrow \alpha_0$
2: **while** $v(\alpha_k) < 1$ **do**
3: $\quad \alpha_k \leftarrow \hat{\rho}(v(\alpha_k)) \cdot \alpha_k$
4: **end while**

We now present two case studies demonstrating how adaptive factors can be combined with BLS, and then compare the performance of regular and adaptive BLS (ABLS) on benchmark problems.

2.3 CASE STUDY: ARMIJO CONDITION

The most popular criterion used in line search is the *Armijo condition* (Armijo, 1966), which is specified by a hyperparameter $c \in (0,1)$ and requires sufficient decrease in the objective function:

$$F(x_k + \alpha_k d_k) - F(x_k) \leq c \cdot \alpha_k \langle \nabla F(x_k), d_k \rangle. \tag{3}$$

For the Armijo condition, the direction d_k is usually assumed to be a descent direction:

Assumption 1 (descent direction)**.** *The direction* d_k *satisfies* $\langle \nabla F(x_k), d_k \rangle < 0$*.*

We define the *violation* of (3) as

$$v(\alpha_k) := \frac{F(x_k + \alpha_k d_k) - F(x_k)}{c \cdot \alpha_k \langle \nabla F(x_k), d_k \rangle}. \tag{4a}$$

Under Assumption 1, (3) can be written as $v(\alpha_k) \geq 1$. To account for the information conveyed by (3) when violated, we choose the corresponding adaptive geometric factor $\hat{\rho}(v(\alpha_k))$ as

$$\boxed{\hat{\rho}(v(\alpha_k)) := \max\left(\epsilon, \rho^{\frac{1-c}{1-c\cdot v(\alpha_k)}}\right),} \tag{4b}$$

where $\epsilon > 0$ is a small factor that prevents occasional numerical errors in $v(\alpha_k)$ from spreading to $\hat{\rho}(v(\alpha_k))$. Although (4b) is parameterized by ϵ and ρ, for each method we fix their values on all experiments, effectively making Algorithm 2 parameter free. We use our adaptive BLS procedure to find suitable step sizes for three standard base methods: gradient descent (GD), Nesterov's accelerated gradient descent (AGD) (Nesterov, 1983) and Adagrad (Duchi et al., 2011). The standard implementations that we use for these algorithms are given in Appendix C. Incorporating line search into GD and Adagrad is straightforward, but the case of AGD merits further comment.

Backtracking and AGD. Unlike GD, AGD is not necessarily a monotone method in the sense that $F(x_k + \alpha_k d_k) \leq F(x_k)$ need not hold. But AGD is a *multistep* method, one being a GD step, for which line search can help to compute a step size or, equivalently, to estimate the Lipschitz constant L. If the estimate of L satisfies (3) with $c = 1/2$ and is increasingly multiplied by a lower bounded positive geometric factor, then AGD with line search enjoys essentially the same theoretical guarantees of AGD tuned with constant parameters. We also consider AGD with memoryless line search with fixed predetermined initial step sizes. Then, unless some variant such as Scheinberg et al. (2014) is used, the theoretical guarantees are not necessarily preserved when AGD is combined with memoryless line search. For some values of ρ, however, we find empirically that not only does the resulting method converge, but it does so much faster than the monotone line search variant, which in turn typically converges faster than AGD tuned with a pre-computed estimate $\bar{L}$ (see Appendix D.1.)

2.4 CASE STUDY: DESCENT LEMMA

A standard assumption in the analysis and design of several optimization algorithms is that gradients are Lipschitz-smooth, which implies (Nesterov, 2018, Thm. 2.1.5.) that there is some $L > 0$ such that

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \tfrac{L}{2}\|y - x\|^2, \qquad \forall x, y. \tag{5}$$

Inequality (5) is commonly known as the *descent lemma* (Bertsekas, 1999). In particular, it is commonly assumed (5) holds for algorithms that solve problems with composite objective functions $F := f + \psi$, where f is Lipschitz-smooth convex and ψ is continuous, possibly nonsmooth, convex. A prototypical example of such an algorithm is FISTA (Beck & Teboulle, 2009), which is an extension of Nesterov's AGD to composite problems (for details see Appendix C.) FISTA assumes that it produces points y_k and $p_{\alpha_k}(y_k)$ satisfying (5) applied to F with $x = y_k$ and $y = p_{\alpha_k}(y_k)$, where p_α denotes the proximal operator (Parikh & Boyd, 2014) parameterized by $\alpha > 0$ and defined by

$$p_\alpha(y) := \arg\min_x \left\{ \psi(x) + \frac{1}{2\alpha} \left\| x - \big(y - \alpha \nabla f(y)\big) \right\|^2 \right\}. \tag{6}$$

In practice, $\alpha = 1/L$ is seldom known for a given f, and FISTA estimates it with some α_k by checking

$$F(p_{\alpha_k}(y_k)) \leq f(y_k) + \langle \nabla f(y_k), p_{\alpha_k}(y_k) - y_k \rangle + \tfrac{1}{2\alpha_k}\|p_{\alpha_k}(y_k) - y_k\|^2 + \psi(p_{\alpha_k}(y_k)). \tag{7}$$

Since (7) holds for any $\alpha_k \leq 1/L$, an estimate α_k can be precomputed from analytical upper bounds on L for particular cases of f, but these bounds tend to be overly conservative and can lead to poor performance. A better alternative, adopted by FISTA and many methods (Nesterov, 2013; Scheinberg et al., 2014), is to backtrack: reduce α_k by some constant factor $\rho < 1$ until (7) holds.

We define the *violation* of Eq. (7) as

$$v(\alpha_k) := \tfrac{1}{2\alpha_k}\|p_{\alpha_k}(y_k) - y_k\|^2 \Big/ \Big(f(p_{\alpha_k}(y_k)) - f(y_k) - \langle \nabla f(y_k), p_{\alpha_k}(y_k) - y_k\rangle\Big), \tag{8a}$$

and the corresponding adaptive factor as:

$$\boxed{\hat{\rho}(v(\alpha_k)) := \rho v(\alpha_k).} \tag{8b}$$

In experiments below, we use (8b) to find suitable step sizes for FISTA, with a fixed $\rho < 1$ value.

3 Empirical Performance

We present four experiments illustrating different ways and scenarios in which our adaptive backtracking subroutine (Algorithm 2) can outperform regular backtracking (Algorithm 1).

3.1 Convex objective: logistic regression + Armijo

First, we consider the logistic regression objective with L_2 regularization, defined by

$$F(x) = -\frac{1}{n}\sum_{i=1}^{n}\Big(y_i \log(\sigma(a_i^\top x)) + (1 - y_i)\log(1 - \sigma(a_i^\top x))\Big) + \frac{\gamma}{2}\|x\|^2, \tag{9}$$

where $\gamma > 0$ and $(A_i, b_i) \in \mathbb{R}^d \times \{0, 1\}$ are n observations pairs from a given dataset and $\sigma(z) = 1/(1 + \exp(-z))$ is the sigmoid function. For each dataset, we compute $\bar{L} = \lambda_{\max}(A^\top A)/(4n)$, which is a standard upper bound on the true Lipschitz parameter of the first term in (9), then set the regularization constant to $\gamma = \bar{L}/(10n)$ and the step size of gradient descent to $1/(\bar{L} + \gamma)$. In all experiments, the initial point x_0 is the origin as is standard.

Result Summary. A succinct summary of our results contained in Table 1 and Appendix D is that

across datasets and step size initializations considered, adaptive backtracking is more robust than regular backtracking and often leads to significant improvements with respect to base methods.

Datasets and methods. We take observations from seven datasets: A9A[1], GISETTE_SCALE (G_SCALE), MUSHROOMS, PHISHING and WEB-1[2] from LIBSVM (Chang & Lin, 2011), PROTEIN from KDD Cup 2004 (Caruana et al., 2004) and MNIST (LeCun et al., 1998). The number of data observations and dimensions for each dataset are listed on Table 4, in Appendix D. We consider GD, AGD, and Adagrad . For each method we run a standard implementation, found in Appendix C, with regular BLS for $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ and our adaptive BLS with a pre-set ρ.

Initialization. We fix $\epsilon = 0.01$ in (4b) on all experiments. We also fix ρ, but it changes according to the base method. For more details, see Appendix D.

Evaluation Details. We run all base method and their variants for long enough to produce solutions with designated precision; Then, we account for the number of gradient and function evaluations and elapsed time each variant takes to produce that solution. Finally, for each BLS variant we average those numbers over the four initial step sizes that we considered. All methods compute exactly one gradient per iteration. To account for elapsed time, we record wall clock time after every iteration. Although somewhat imprecise, elapsed time reflects the relative computational cost of gradient and function evaluations and, especially in larger problems, is a reasonable metric to compare performance.

Remarks. Table 1 shows that adaptive BLS variants significantly outperform regular BLS variants. In the case of GD and Adagrad, adaptive BLS variants outperforms regular BLS for almost every

[1] A preprocessed version of the ADULT dataset (Becker & R, 1996).
[2] A subsample of the WEB dataset (Platt, 1998).

		Backtracking Line Search (BLS)						Adaptive BLS			
			$\rho = 0.2$			$\rho = 0.3$			$\rho = 0.3$		
Method	Dataset	$\#f$	$\#\nabla f$	ET [s]	$\#f$	$\#\nabla f$	ET [s]	$\#f$	$\#\nabla f$	ET [s]	**gain**
GD	ADULT	148597.2	32582.0	1319.0	74258.5	18749.0	**694.7**	37296.0	13583.8	**370.0**	**46.7%**
	G_SCALE	58488.5	18252.2	**3050.4**	65429.2	17111.5	3059.7	25917.2	11700.5	**1607.0**	**47.3%**
	MNIST	148469.8	41726.2	**10986.4**	207786.8	46475.5	13474.8	52616.5	22385.8	**4841.9**	**55.9%**
	MUSHROOMS	14170.5	6507.2	**31.7**	14800.8	6628.5	33.9	7611.0	3693.8	**17.3**	**45.5%**
	PHISHING	33388.2	8059.8	63.0	34193.5	7938.5	**62.4**	16543.2	6434.5	**26.4**	**57.6%**
	PROTEIN	28011.0	13260.5	**4481.6**	35733.2	14868.8	5282.0	27656.0	13207.2	**4137.4**	**7.7%**
	WEB-1	6721.5	3139.0	9.0	6156.8	2972.8	**8.4**	6192.2	3024.8	**7.9**	**(5.4%)***
			$\rho = 0.5$			$\rho = 0.6$			$\rho = 0.9$		
Method	Dataset	$\#f$	$\#\nabla f$	ET [s]	$\#f$	$\#\nabla f$	ET [s]	$\#f$	$\#\nabla f$	ET [s]	**gain**
AGD	ADULT	17288.8	2014.2	116.6	49331.2	3806.2	247.0	7999.0	2275.0	**49.2**	**52.5%**
	G_SCALE	3580.2	592.5	**114.9**	18530.8	1964.0	512.9	2204.5	728.5	**84.1**	**26.8%**
	MNIST	8934.2	1524.5	**452.3**	14365.8	1846.5	643.7	4943.2	1666.2	**283.0**	**37.4%**
	MUSHROOMS	1100.5	372.2	1.7	1146.8	367.8	**1.6**	850.0	376.2	**1.4**	**15.5%**
	PHISHING	6763.2	944.2	9.1	8344.2	944.8	**8.6**	3699.0	1058.0	**4.0**	**53.6%**
	PROTEIN	2865.0	1232.2	396.3	3397.5	1272.5	**346.5**	2743.2	1257.0	**291.6**	**15.8%**
	WEB-1	651.8	208.5	0.6	699.8	202.2	**0.5**	519.2	217.8	**0.4**	**3.3%**
			$\rho = 0.2$			$\rho = 0.3$			$\rho = 0.3$		
Method	Dataset	$\#f$	$\#\nabla f$	ET [s]	$\#f$	$\#\nabla f$	ET [s]	$\#f$	$\#\nabla f$	ET [s]	**gain**
Adagrad	ADULT	124102.0	20001.0	**699.5**	145159.8	19789.8	746.5	27179.0	8000.5	**178.8**	**74.4%**
	G_SCALE	274023.2	33071.5	**7396.5**	361852.0	34933.8	9740.9	84201.0	17176.2	**2425.4**	**67.2%**
	MNIST	86240.5	13843.8	**3921.2**	99021.8	14760.8	4377.2	12366.2	3521.8	**679.5**	**82.7%**
	MUSHROOMS	7794.8	1967.8	**9.0**	7239.5	1693.0	9.3	3751.5	1446.0	**6.0**	**33.2%**
	PHISHING	74737.5	15833.5	**68.9**	117564.0	20001.0	96.4	18053.0	6375.0	**19.4**	**71.8%**
	PROTEIN	6103.0	809.2	420.6	4040.2	429.0	**257.9**	1845.8	446.5	**164.4**	**36.2%**
	WEB-1	4384.2	1027.0	2.9	3857.8	745.2	**2.5**	1726.5	568.8	**1.3**	**50.1%**

Table 1: Backtracking for logistic regression. $\#f$ and $\#\nabla f$ denote the number of function and gradient evaluations and ET refers to elapsed time in seconds. The **gain** is given by $1 - (\text{ET of ABLS})/(\text{ET of BLS})$ with the best ET for BLS across ρ in each experiment, which is bolded. We ran each BLS experiment with a grid of four ρ's and present the best two in the table. The gain for GD on WEB-1 is colored orange because although ABLS terminated before the best performing BLS variant, it required more function and gradient evaluations. This anomaly can be attributed to the relatively small ET for this problem.

combination of ρ and α_0 by saving function evaluations and returning better step sizes, which speed up convergence. Fig. 1 illustrates this point by showing how the suboptimality gap evolves with time for the baseline GD, its adaptive BLS variant and regular BLS variants for two choices of initial step size. In particular, increasing the initial step size helps regular BLS in some datasets but hurts it in others. Fig. 2 shows a similar trend for the case of AGD. In general, adaptive BLS variants are more robust to the choice of initial step size and that is the main reason why the adaptive BLS AGD variant outperforms the regular BLS AGD variants. In Appendix D.2, Figs. 12 and 13 show the evolution of the corresponding step sizes for the methods shown in Figs. 1 and 2, respectively. Initially, adaptive backtracking returns smaller GD step sizes than regular backtracking, but this trend quickly reverses. A plausibly explanation is that regular backtracking returns the largest step sizes that satisfy (3), within a factor of ρ. If the step sizes are excessively large initially, they can lead to worse optimization paths (e.g., more zig-zagging). For AGD, the step sizes follow a similar trend initially, but then the adaptive step size seems to converge while the regular step sizes not always do. This can be indicative of another shortcoming of regular backtracking, namely that it can only return step sizes that are powers of ρ times the initial step size, in contrast with adaptive backtracking.

3.2 Convex objective: linear inverse problems + descent Lemma

The goal of a linear inverse problem is to recover the sparse signal x from a noisy measurement model $y = Ax + \epsilon$, where $A \in \mathbb{R}^{n \times d}$ and $y \in \mathbb{R}^n$ are known, and ϵ is unknown noise. The problem of estimating x is typically posed as a Lasso objective (Santosa & Symes, 1986; Tibshirani, 1996)

$$F(x) = \frac{1}{2}\|Ax - y\|^2 + \lambda\|x\|_1.$$

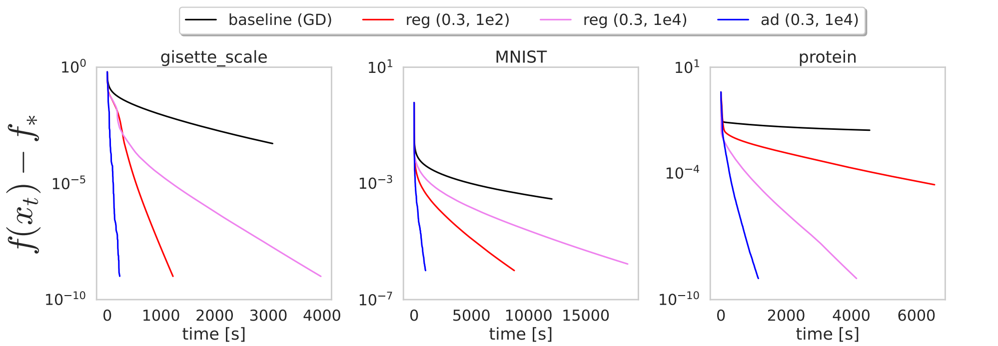


Figure 1: Baseline: GD with constant $\alpha_k = 1/\bar{L}$; reg (ρ, β) and ad (ρ, β): GD with, respectively, regular and adaptive memoryless BLS parameterized by ρ and $\alpha_0 = \beta/\bar{L}$.

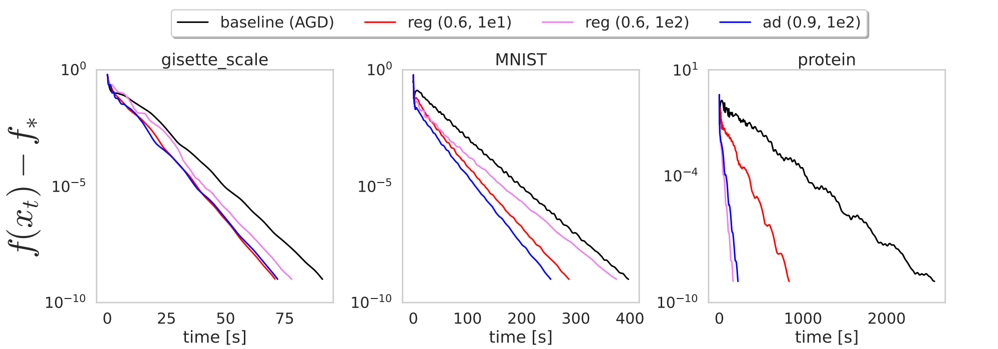


Figure 2: Baseline: AGD with constant $\alpha_k = 1/\bar{L}$; reg (ρ, β) and ad (ρ, β): AGD with, respectively, regular and adaptive memoryless BLS parameterized by ρ and $\alpha_0 = \beta/\bar{L}$.

Datasets. We take observations A from eight datasets: IRIS, DIGITS, WINE, OLIVETTI_FACES and LFW_PAIRS from scikit-learn (Pedregosa et al., 2011), SPEAR3 and SPEAR10 (Lorenz et al., 2014) and SPARCO (van den Berg et al., 2007). For multi-class datasets, the first two are considered. The number of datapoints and dimensions of each dataset can be found on Table 5, in Appendix E.

Methods. We consider FISTA (Beck & Teboulle, 2009) (Algorithm 6) and BLS variants. For regular BLS, $\rho = \{1/2, 1/3, 1/5\}$ and for adaptive BLS $\rho = 1/1.1 \approx .9$ mirroring the choice for AGD above. All BLS methods start with the same initial Lipschitz constant estimate increase it accordingly.

Initialization. For each dataset, we empirically find values of $\alpha_0 = 1/L_0$ around which backtracking becomes active, and then increase them successively (values reported on Appendix E.)

Results Summary. The adaptive BLS FISTA variant outperforms the regular BLS variants across all datasets tested. Moreover, the best performing value of ρ used in the regular variant changes from one dataset to the other. Since the Lipschitz constant estimate is monotone, function evaluations vary little across method and the overall impact on performance is small. Nevertheless, the adaptive variant requires fewer function evaluations for all datasets.

3.3 NONCONVEX OBJECTIVE: ROSENBROCK + ARMIJO

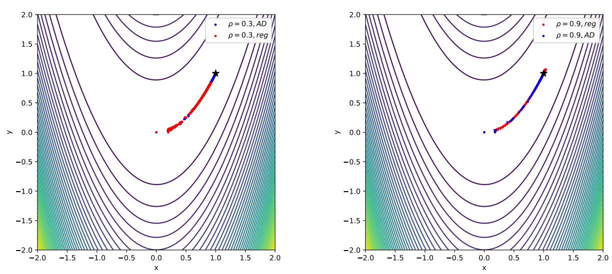
We consider the classical nonconvex problem given by the Rosenbrock objective function

$$F(u, v) = 100(u - v^2)^2 + (1 - v)^2.$$

We use the origin as the initial point and 0.1 as the initial step size. Fig. 3 shows the optimization paths for regular and adaptive memoryless BLS variants of GD and AGD after 1000 iterations, using $\rho = 0.3$ and $\rho = 0.9$, respectively. We see that the adaptive BLS variants achieve better losses, requiring far fewer function evaluations and less time to do so.

		Backtracking Line Search (BLS)				Adaptive BLS		
		$\rho = 0.5$		$\rho = 0.3$		$\rho = 0.9$		
Method	Dataset	$\Delta\#f$	$\#\nabla f$	$\Delta\#f$	$\#\nabla f$	$\Delta\#f$	$\#\nabla f$	**gain**
FISTA	DIGITS	15.5	**28282.25**	10	34730.75	4.75	**16756**	**40.8%**
	IRIS	10.75	**726.25**	7	816.5	4	**710**	**2.2%**
	OLIVETTI	10	**242709.5**	6.25	246827.75	2	**212930.75**	**12.3%**
	LFW*	26.25	49093.75	17	**49014.5**	2	**45070.75**	**8.0%**
	SPEAR3	13.25	**328328.75**	9	506308.5	2	**255417.5**	**22.2%**
	SPEAR10	44.75	**18691**	29.75	19992.75	8	**15128**	**19.1%**
	SPARCO3	27.75	**266.25**	18.5	276.75	3.25	**251**	**5.7%**
	WINE	48	**529333.25**	27.75	564293.75	8.5	**472527**	**10.7%**

Table 2: Backtracking for FISTA. $\#\nabla f$ and $\Delta\#f$ denote the number of gradient and excess function evaluations (total function evaluations minus two times total iterations). The **gain** is given by $1 - (\#\nabla f \text{ of ABLS})/(\#\nabla f \text{ of BLS})$ with the best ET for BLS across ρ in each dataset (bolded.) We run each BLS experiment with three ρ's and present the best two in the table. ABLS reached the desired precision in all testpoints while the asterisk on LFW* indicates BLS did not in at least one testpoint.



(a) GD (b) AGD

Method	$\#f$	loss	ET [s]
GD+BLS	4992	7.30e-03	1.04e-1
GD+ABLS	2754	7.21e-12	**9.98e-2**
AGD+BLS	42263	9.25e-11	5.59e-1
AGD+ABLS	2991	4.01e-13	**6.40e-2**

(c)

Figure 3: Performance of GD and AGD regular (red) and adaptive (blue) BLS variants on Rosenbrock. "loss" refers to the final loss after 1000 iterations.

3.4 Nonconvex objective: matrix factorization + Armijo

Lastly, we consider the nonconvex problem of matrix factorization, defined by the objective

$$F(U, V) = \frac{1}{2}\|UV^\top - A\|_F^2, \tag{10}$$

where $A \in \mathbb{R}^{m\times n}$, $U \in \mathbb{R}^{m\times r}$, $V \in \mathbb{R}^{n\times r}$ and $r < \min\{m, n\}$. We take A from the MovieLens 100K dataset (Harper & Konstan, 2015) and consider three rank values $r \in \{10, 20, 30\}$ (see Appendix F for further details and full plots.)

For this experiment we replicate the initialization and evaluation methodology of Section 3.1, except we restrict the base methods to GD and AGD, and pick different values for initial step sizes, $\{0.05, 0.5, 5, 50\}$, and ρ. Namely, we let $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ for the regular BLS variants, but fix $\rho = 0.3$ and $\rho = 0.9$ for the adaptive BLS GD and AGD variants, respectively. Table 3 summarizes the results for adaptive BLS and the top two regular BLS variants. Once again, the best value of ρ for regular BLS is inconsistent: $\rho = 0.3$ for ranks 10 and 20 but $\rho = 0.2$ for rank 30. Adaptive BLS requires significantly fewer gradient and function evaluations than the top regular BLS variant does, which leads to considerable gains in time to achieve the desired precision.

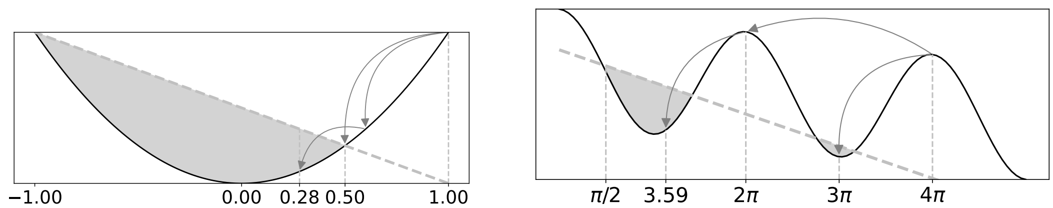
4 Motivation and theoretical results

In this section, we motivate our choices of adaptive factors and characterize them theoretically.

The particular choices of adaptive factors were made with two goals in mind: *generate more aggressive backtracking factors to save function evaluations* and *guarantee reasonably large step sizes to achieve fast convergence*. To meet our first goal, $\hat{\rho}(v(\alpha_k)) \in (0, \rho)$ must hold. Indeed, if (3) is violated, then

Method	Rank	Backtracking Line Search						Adaptive BLS			
		$\rho = 0.2$			$\rho = 0.3$			$\rho = 0.3$			
		$\#f$	$\#\nabla f$	ET [s]	$\#f$	$\#\nabla f$	ET [s]	$\#f$	$\#\nabla f$	ET [s]	**gain**
GD	10	39960.0	5683.5	1034.5	33531.8	4897.8	**999.5**	52075.8	1631.8	**64.8**	**93.5%**
	20	127480.2	18734.5	1153.2	111322.0	16109.2	**1010.9**	377111.8	5133.0	**170.9**	**83.1%**
	30	231170.5	35639.8	**2035.5**	394472.2	50001.0	4195.1	681952.0	14802.8	**669.2**	**67.1%**
		$\rho = 0.3$			$\rho = 0.5$			$\rho = 0.9$			
AGD	10	362029.7	21909.3	2377.0	44873.0	6198.3	**379.4**	22707.7	6663.7	**172.0**	**54.7%**
	20	479432.0	35672.0	**2987.9**	491753.3	33523.3	3588.6	80720.3	24588.7	**738.0**	**75.3%**
	30	357885.7	39234.3	**2187.8**	478084.3	42985.3	5157.0	95040.0	27454.0	**855.1**	**60.9%**

Table 3: Backtracking for matrix factorization. $\#f$ and $\#\nabla f$ denote the number of function and gradient evaluations and ET refers to elapsed time in seconds. The **gain** is given by $1 - (\text{ET of ABLS})/(\text{ET of BLS})$ with the best ET for BLS across ρ in each experiment, which is bolded.



(a) Reducing ρ can lead to greater step sizes.

(b) Reducing ρ can lead to more step size adjustments.

Figure 4: Backtracking line search convex and nonconvex examples. The dashed negative slopes represent the threshold to satisfy the Armijo condition and the shaded regions indicate feasible iterates.

$v(\alpha_k) < 1$ because d_k is assumed a descent direction, where v is defined by (4a). So, if in addition $\epsilon \in (0, \rho)$ in (4b), then $\hat{\rho}(v(\alpha_k)) < \rho$. To meet the second goal, the returned step size must be non-trivially lower bounded. To this end, note that if (3) is violated, then $1 - c \cdot v(\alpha_k) > 0$, since $c \in (0, 1)$ by assumption. Thus, $\hat{\rho}(v(\alpha_k))$ is bounded away from zero. Furthermore, in Section 4.2 we show that if the objective function is Lipschitz-smooth, then ABLS returns a step size on par with the greatest step size that is guaranteed to satisfy (3). Moreover, that same bound applies to regular BLS. Similar conclusions can be reached if the BLS criterion is (7) and $\hat{\rho}$ and v are chosen as (8b) and (8a).

4.1 The scope of theoretical guarantees for a backtracking subroutine

There are limitations to the scope of the theoretical guarantees one can expect of a backtracking subroutine and the extent to which two subroutines can be compared. We delineate these limits with two simple examples (see Appendix B.1 for details) that establish three facts: 1) the step size returned by backtracking is not monotone in ρ, *even for convex problems*; 2) For nonconvex problems, given step sizes α' and α with $\alpha' < \alpha$, α being feasible does not imply α' is too. 3) For nonconvex problems, decreasing ρ may increase the number of criteria evaluations required to compute a feasible step size.

The first example consists in a simple scalar quadratic objective given by $F(x) = x^2$, a base iterate $x_k = -1$ and an initial step size α_k such that $x_k + \alpha_k d_k = 1$. Fig. 4a shows the plot of F and a dashed line representing $F(x_k) + c\alpha_k \langle \nabla F(x_k), d_k \rangle$. The tentative iterate is not feasible, therefore the step size must be adjusted. If $\rho = 0.75$, then the adjusted iterate is feasible, but not if $\rho = 0.8$, in which case another adjustment is required to produce a feasible point. This establishes the first fact.

The second example consists in an objective function with oscillatory downward slope and a base iterate $\pi/2$. Fig. 4b shows there are two (shaded) intervals of feasible step sizes. In particular, α_k is feasible if $x_k + \alpha_k d_k = 3\pi$, but not if $x_k + \alpha_k d_k = 2\pi$. That is, reducing a feasible α_k can turn it infeasible, establishing the second fact. An analogous rationale establishes the third fact. But, in Appendix B.2, we show that only the first fact is true for convex problems.

Overall, the above examples show that *even for a single backtracking call* only somewhat modest theoretical guarantees can be given. Moreover, different backtracking subroutines induce different optimization paths, which further limits the extent to which subroutines can be compared theoretically.

4.2 THEORETICAL RESULTS

In this subsection, we present theoretical results regarding regular and adaptive backtracking. Several convergence results and full proofs can be found in Appendix B.

Convex problems. We show that only the first of the three facts above holds for convex problems, which expands the extent to which two backtracking subroutines can be compared.

Proposition 1. *Let F be convex differentiable. Given a point x_k, a direction d_k and a step size $\alpha_k > 0$ satisfying (3) for some c, then x_k, d_k and α'_k also satisfy (3) for any $\alpha'_k \in (0, \alpha_k)$.*

The following proposition refers to "compatible inputs." By this we mean:

Definition 1 (Compatibility)**.** *The inputs to Algorithms 1 and 2 are said to be* compatible *if α_0, c, v coincide and the input $\hat{\rho}$ to Algorithm 2 is parameterized by the same ρ that Algorithm 1 takes as input.*

Proposition 2. *Let F be convex differentiable. Fixing all other inputs, the number of function evaluations that Algorithm 1 and Algorithm 2 take to return a feasible step size is nondecreasing in the input ρ. Moreover, given compatible inputs with a descent direction and $\epsilon < \rho$, Algorithm 2 takes no more function evaluations to return a feasible step size than Algorithm 1 does.*

Nonconvex problems. For convex problems, we were able to compare the number of times regular and adaptive backtracking must evaluate their criteria in order to return a single feasible step size. But what really matters is the total number of criteria evaluations up to a given iteration. We bound this number for general nonconvex problems, hinging on the following properties.[3]

Definition 2 (Smoothness)**.** *A function F is said to be* Lipschitz-smooth *if (5) holds for some $L > 0$.*

Definition 3 (Gradient related)**.** *The directions d_k are said to be* gradient related *if there are $c_1 > 0$ and $c_2 > 0$ such that $\langle \nabla F(x_k), -d_k \rangle \geq c_1 \|\nabla F(x_k)\|^2$ and $\|d_k\| \leq c_2 \|\nabla F(x_k)\|$, for all $k \geq 0$.*

Assumption 2. *We assume F is Lipschitz-smooth and d_k are gradient related.*

Gradient relatedness ensures that d_k is not "too large" or "too small" with respect to $\nabla F(x_k)$ and that the angle between d_k and $\nabla F(x_k)$ is not "too close" to being perpendicular (Bertsekas, 1999, p. 41). Together with c_1 and c_2, the Lipschitz constant L and Armijo constant c define a step size threshold $\bar{\alpha} = 2c_1(1-c)/Lc_2^2$ below which (3) holds. This quantity is central in the following result.

Informal Theorem (Armijo)**.** *Let F be Lipschitz-smooth and d_k gradient related. Given compatible inputs, if $\epsilon < \rho$ and $v, \hat{\rho}$ are chosen as (4), then Algorithms 1 and 2 share the same bounds on the total number of backtracking criteria evaluations up to any iteration. If α_k is received as the initial step size input at iteration $k+1$ for all $k \geq 0$, then they evaluate (3) at most $\lfloor \log_\rho(\bar{\alpha}/\alpha_0) \rfloor + 1 + k$ times up to iteration k. If, on the other hand, α_0 is received as the initial step size input at every iteration, then they evaluate (3) at most $k(\lfloor \log_\rho(\bar{\alpha}/\alpha_0) \rfloor + 1)$ times up to iteration k. Moreover, Algorithms 1 and 2 always return a step size α_k such that $\alpha_k \geq \min(\alpha_0, \rho\bar{\alpha})$.*

Informal Theorem (Descent lemma)**.** *Let f be Lipschitz-smooth convex and let ψ be continuous convex. Also, suppose v and $\hat{\rho}$ are chosen as (8a) and (8b). If $\alpha_k \in (0, 1/L)$, then (7) holds for all y_k. If Algorithms 1 and 2 receive α_k as the initial step size input in iteration $k+1$, then they evaluate (7) at most $\lfloor \log_\rho(1/L\alpha_0) \rfloor + 1 + k$ times up to iteration k. If, on the other hand, Algorithms 1 and 2 receive α_0 as the initial step size input in every iteration, then they evaluate (7) at most $k(\lfloor \log_\rho(1/L\alpha_0) \rfloor + 1)$ times up to iteration k. Moreover, they return a feasible step size α_k such that $\alpha_k \geq \min\{\alpha_0, \rho/L\}$.*

5 FUTURE WORK: FURTHER APPLICATIONS, EXTENSIONS AND LIMITATIONS

Adaptive backtracking is a general idea that can be broadly applied in a variety of settings. Our goal in this paper was to rigorously validate it in classical machine learning and optimization problems. This section outlines several promising directions for future work and speculate about potential limitations.

[3]Instead of the usual condition that $\|\nabla F(x) - \nabla F(y)\| \leq L\|x - y\|$ hold for all x, y, we adopt the equivalent (Nesterov, 2018, Thm. 2.1.5.) condition (5) as the definition of Lipschitz-smoothness for the sake of convenience.

5.1 Further applications and extensions

Stochastic line search. In machine learning, models such as over-parameterized neural networks are sufficiently expressive to *interpolate* immense datasets (Zhang et al., 2016; Ma et al., 2018). Interpolation provides theoretical foundation for the stochastic line search (SLS) proposed by Vaswani et al. (2019b), which enforces the Armijo condition on the training mini-batches. In the same vein, Galli et al. (2023) replaced the Armijo condition in SLS with a nonmonotone criterion and used Polyak's step size to devise an initial step size heuristic, obtaining the Polyak Nonmonotone Stochastic (PoNoS) method. Below, we reproduce experiment 1 from (Galli et al., 2023) to demonstrate the potential of applying ABLS in combination with stochastic line search in the interpolating regime. Fig. 5 shows that combining ABLS with PoNoS leads to good test accuracy in fewer epochs (details in Appendix A.) We defer fully developing this application to future work.

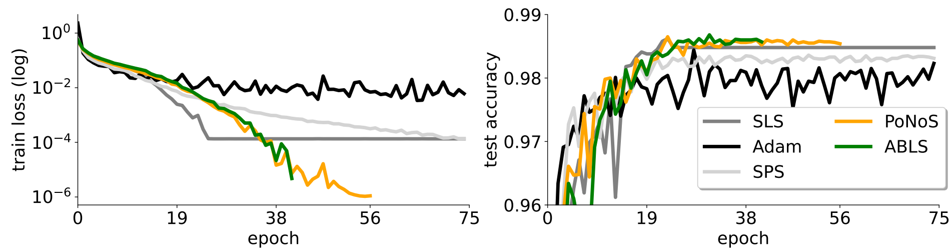


Figure 5: MLP trained on MNIST with different algorithms.

Additional line search criteria. Adaptive backtracking may be useful to enforce other conditions that are affine in the step size. A prominent candidate are the Goldstein conditions (Goldstein & Price, 1967), which comprise two inequalities that are affine in the step size. Nonmonotone line search criteria (Grippo et al., 1986; Zhang & Hager, 2004) also offer several candidates.

Additional algorithms and increasing step sizes. In this paper, we experimented with increasing step sizes through memoryless line search, where the initial step size is fixed for every iteration, but other schemes are possible. For example, adaptive adjustments can also be used to increase step sizes. In addition, adaptive backtracking can replace regular backtracking in line search methods that increase the current step size and then use it as the initial step size, such as the two-way method (Truong & Nguyen, 2021) and FISTA variants (Scheinberg et al., 2014; Calatroni & Chambolle, 2019; Rebegoldi & Calatroni, 2022). Also, it would be interesting to see how adaptive backtracking works together with schemes that handle problems where the strong convexity constant is unknown, namely restarting schemes Becker et al. (2011); O'Donoghue & Candès (2015); Aujol et al. (2024). Finally, we note that the violation of a condition can also be used indirectly to adjust step sizes, for example to pick the degree to which a fixed ρ is exponentiated, saving backtracking cycles.

5.2 Limitations

The weak and strong Wolfe conditions (Wolfe, 1969) are not affine in the step size and are not satisfied by arbitrarily small step sizes. Hence, Wolfe conditions are not enforced by backtracking (e.g., Nocedal & Wright (2006, pp. 60–61)) and it is unclear how to find analogous adaptive schemes. In turn, quasi-Newton methods, which often must enforce Wolfe conditions to guarantee global convergence, may not be suitable candidates for adaptive line search subroutines. In reality, the role of line search for these methods is to guarantee they converge globally rather than finding the "right" step size, since they work with unit step size locally. Hence, only few adjustments may be necessary. The same applies to Newton's method and the Barzilai–Borwein method (Barzilai & Borwein, 1988).

It is also unclear if adaptive adjustments can be useful for more general stochastic line search methods that do not rely on the interpolation property (Cartis & Scheinberg, 2017; Paquette & Scheinberg, 2020). Instead, they resample function and gradient mini-batches in every loop, whether a sufficient descent condition is violated or not. But the information conveyed by the violation for one sample need not be relevant to satisfy the same condition with a different sample, which poses a potential limitation.

REFERENCES

L. Armijo. Minimization of functions having lipschitz continuous first partial derivatives. *Pac. J. Math.*, 16(1):1–3, 1966.

J.-F. Aujol, L. Calatroni, C. Dossal, H. Labarrière, and A. Rondepierre. Parameter-free fista by adaptive restart and backtracking. *SIAM Journal on Optimization*, 34(4):3259–3285, 2024.

J. Barzilai and J. Borwein. Two-point step size gradient methods. *IMA J. Numer. Anal.*, 8(1):141–148, 1988.

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

B. Becker and Kohavi R. Adult. UCI Machine Learning Repository, 1996.

S. R. Becker, E. J. Candès, and M. C. Grant. Templates for convex cone problems with applications to sparse signal recovery. *Mathematical Programming Computation*, 3(2):165–218, 2011.

D. P. Bertsekas. *Nonlinear Programming*. Athena Scientific, 1999.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning*, 3(1):1–122, 2011.

L. Calatroni and A. Chambolle. Backtracking strategies for accelerated descent methods with smooth composite objectives. *SIAM Journal on Optimization*, 29(3):1772–1798, 2019.

C. Cartis and K. Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Math. Program.*, 169:337–375, 2017.

R. Caruana, T. Joachims, and L. Backstrom. Kdd-cup 2004: Results and analysis. *SIGKDD Explorations Newsletter*, 6(2):95–108, 2004.

C.-C. Chang and C.-J. Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(27):1–27, 2011.

J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

L. Galli, H. Rauhut, and M. Schmidt. Dont be so monotone: Relaxing stochastic line search in over-parameterized models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34752–34764. Curran Associates, Inc., 2023.

A. Goldstein and J. Price. An effective algorithm for minimization. *Numer. Math.*, 10:184–189, 1967.

Luigi Grippo, Francesco Lampariello, and Stefano Lucidi. A nonmonotone line search technique for newton's method. *SIAM journal on Numerical Analysis*, 23(4):707–716, 1986.

M.F. Harper and J.A. Konstan. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 5(4), 2015.

D.P. Kingma and J.L. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICML)*, 2015.

Y. LeCun, C. Cortes, and C.J.C. Burges. The MNIST database of handwritten digits. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

N. Loizou, S. Vaswani, I. H. Laradji, and S. Lacoste-Julien. Stochastic polyak step-size for sgd: An adaptive learning rate for fast convergence. In *International Conference on Artificial Intelligence and Statistics*, pp. 1306–1314, 2021.

D.A. Lorenz, M.E. Pfetsch, A.M. Tillmann, and C. Kruschel. SPEAR: Sparse exact and approximate recovery. https://wwwopt.mathematik.tu-darmstadt.de/spear/, 2014.

S. Ma, R. Bassily, and M. Belkin. The power of interpolation: Understanding the effectiveness of sgd in modern over-parametrized learning. In *International Conference on Machine Learning (ICML)*, 2018.

J. J. Moré and D. C. Sorensen. Newton's method. Technical report, Argonne National Lab., IL (USA), 1982.

Y. Nesterov. A method for unconstrained convex minimization problem with the rate of convergence o(1/k2). *Doklady AN SSSR*, 269:543–547, 1983.

Y. Nesterov. Gradient methods for minimizing composite functions. *Math. Program. Ser. B*, 140: 125–161, 2013.

Y. Nesterov. *Lectures on convex optimization*, volume 137. Springer, 2018.

J. Nocedal and S. Wright. *Numerical Optimization*. Springer Science & Business Media, 2006.

B. O'Donoghue and E. J. Candès. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3):715–732, 2015.

C. Paquette and K. Scheinberg. A stochastic line search method with expected complexity analysis. *SIAM Journal on Optimization*, 30(1):349–376, 2020.

N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python. *JMLR*, pp. 2825–2830, 2011.

J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*. MIT Press, January 1998.

S. Rebegoldi and L. Calatroni. Scaled, inexact, and adaptive generalized fista for strongly convex optimization. *SIAM Journal on Optimization*, 32(3):2428–2459, 2022.

Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv:1609.04747*, 2016.

F. Santosa and W.W. Symes. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing*, 7(4):1307–1330, 1986.

K. Scheinberg, D. Goldfarb, and X. Bai. Fast first-order methods for composite convex optimization with backtracking. *Foundations of Computational Mathematics*, 14:389–417, 2014.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 58(1):267–288, 1996.

T. T. Truong and T. H. Nguyen. Backtracking gradient descent method for general c1 functions, with applications to deep learning. *Appl. Math. Optm.*, 84:2557–2586, 2021.

E. van den Berg, G. Friedlander, M.P.and Hennenfent, F. Herrmann, R. Saab, and Ö. Yilmaz. Sparco: a testing framework for sparse reconstruction. Technical Report Tech. Report TR-2007-20, University of British Columbia, Vancouver, Dept. Computer Science, 2007.

S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. In *NeurIPS*, 2019a.

Sharan Vaswani, Aaron Mishkin, Issam Laradji, Mark Schmidt, Gauthier Gidel, and Simon Lacoste-Julien. Painless stochastic gradient: Interpolation, line-search, and convergence rates. *Advances in neural information processing systems*, 32, 2019b.

P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv*, 2016.

Hongchao Zhang and William W Hager. A nonmonotone line search technique and its application to unconstrained optimization. *SIAM journal on Optimization*, 14(4):1043–1056, 2004.

A STOCHASTIC LINE SEARCH EXAMPLE

The results presented in Fig. 5 correspond to experiment 1 from (Galli et al., 2023) without any modifications, and are run using the base code from the same paper, which can be found at:

https://github.com/leonardogalli91/PoNoS.

In this experiment, a multilayer perceptron (MLP) with a single layer, width 1000 and 535818 parameters is trained on the MNIST dataset (LeCun et al., 1998) until the training loss becomes less than 10^{-6}. This is the same stopping criterion adopted in (Galli et al., 2023). To train the MLP, the following methods are used:

1. Adam: adaptive moment estimation method (Kingma & Ba, 2015).
2. SLS: stochastic line search (Vaswani et al., 2019a).
3. SPS: stochastic Polyak step size method (Loizou et al., 2021).
4. PoNoS: Polyak nonmonotone stochastic method (Galli et al., 2023).
5. ABLS: an adaptive backtracking variant of PoNoS, detailed below.

As Fig. 5 shows, only PoNoS and ABLS terminate within 75 epochs. PoNoS does so after 56 epochs and 583 seconds, while ABLS finishes in 41 epochs and 464 seconds.

For all the above methods, we preserve the parameters recommended in (Galli et al., 2023) unaltered from the source code. The ABLS method combines our adaptive backtracking procedure with a simplified version of PoNoS. Namely, PoNoS generates initial step sizes with

$$\eta_k = \eta_{k,0}\delta^{\bar{l}_k + l_k}, \tag{11}$$

where $\delta \in (0, 1)$ and l_k is the amount of backtracks in iteration $k - 1$, which are accounted for in

$$\bar{l}_k = \max\{\bar{l}_{k-1} + l_{k-1} - 1, 0\}.$$

That is, previous backtracks are used to discount an initial step size given by

$$\eta_{k,0} = \min\{\tilde{\eta}_{k,0}, \eta^{\max}\},$$

which in turn is based on the Polyak initial step size

$$\tilde{\eta}_{k,0} = \frac{f_{i_k}(w_k) - f^*_{i_k}}{c_p\|\nabla f_{i_k}(w_k)\|^2}, \tag{12}$$

where $c_p \in (0, 1)$ is a hyperparameter, i_k denotes the mini-batch sampled in iteration k, w_k denotes the MLP parameters in iteration k and $f^*_{i_k}$ refers to the minimum of the mini-batch training loss:

$$f_{i_k} = \frac{1}{|i_k|}\sum_{j \in i_k} f_j.$$

Instead, we simply use (12) as the initial step size, with $c_p = 1/2$, the same value proposed in (Galli et al., 2023). Then, we apply ABLS to enforce

$$f_{i_k}(w_k - \eta_k \nabla f_{i_k}(w_k)) \le C_k - c\eta_k\|\nabla f_{i_k}\|^2, \qquad c \in (0, 1), \tag{13}$$

where C_k denotes an exponential moving average of losses given by

$$C_k = \max\{\tilde{C}_k, f_{i_k}(w_k)\}, \qquad \tilde{C}_k = \frac{\xi Q_k C_{k-1} + f_{i_k}(w_k)}{Q_{k+1}}, \qquad Q_{k+1} = \xi Q_k + 1$$

with $\xi \in (0, 1)$. The inequality (13) is a stochastic variant enforced by PoNoS of the deterministic criterion proposed by Zhang & Hager (2004). This inequality can be seen as a generalization of the Armijo condition with the current loss replaced with an average. As such, it preserves the structure of (3), which allows us to seamlessly apply (4b), with $\rho = 0.9$ and $\epsilon = 0.5$.

B Examples, theorems and proofs

In this section, we detail the examples from Section 4 and prove the results stated therein.

B.1 Examples

In this subsection, we provide the details of the two examples presented in Section 4.1. The examples illustrate the effects of changing ρ in regular backtracking for two objective functions, one is convex and the other is not.

Example 1. *Let F be defined by $F(x) = x^2$ and fix $x_k = -1$, $d_k = -F'(x_k) = -2x_k = 2$. Then, the Armijo condition with $c = 1/4$ is satisfied if and only if*

$$(-1 + 2\alpha_k)^2 \leq 1 - \alpha_k.$$

To find the critical step size values for which this inequality is satisfied, we simply solve a second-order equation, which gives the positive value of $\alpha_k^* = 0.75$ with corresponding iterate $x_k + \alpha_k^* d_k = 0.5$. Thus, if the initial tentative step size is $\alpha_k = 1$, which produces the tentative iterate is 1, then the Armijo condition is not satisfied and the step size must be adjusted. If $\rho = 0.75$, then the adjusted step size 0.75 produces the tentative iterate 0.5 and the Armijo condition is satisfied, therefore the step size requires no further adjustments. On the other hand, if $\rho = 0.8$, then the adjusted step size 0.8 produces the tentative iterate 0.6 and the Armijo condition is not satisfied. Adjusting the step size once more produces a step size of $0.64 < \alpha_k^*$ and an corresponding iterate $x_k + \alpha_k d_k = -1 + 0.64 \cdot 2 = 0.28$, satisfying the Armijo condition. Therefore, increasing $\rho = 0.75$ to $\rho = 0.8$ decreases the step size that backtracking returns.

Example 2. *Let F be defined by $F(x) = \cos x - ax$, where $a = \frac{1}{5\pi}$, and also fix $x_k = \frac{\pi}{2}$ and*

$$d_k = -F'(x_k) = \sin x_k + a = 1 + a.$$

Given the above choices, the Armijo condition parameterized by $c = \frac{1}{2\pi}$ is satisfied if and only if

$$\cos\Big(\frac{\pi}{2} + (1+a)\alpha_k\Big) - a\Big(\frac{\pi}{2} + (1+a)\alpha_k\Big) \leq \cos\Big(\frac{\pi}{2}\Big) - \frac{a\pi}{2} - (1+a)^2\frac{\alpha_k}{2\pi},$$

or, equivalently, if and only if

$$\cos\Big(\frac{\pi}{2} + (1+a)\alpha_k\Big) \leq a(1+a)\alpha_k - (1+a)^2\frac{\alpha_k}{2\pi}.$$

If the initial tentative step size is picked as $\frac{7\pi}{2(1+a)}$, then $(1+a)\alpha_k = \frac{7\pi}{2}$, so that

$$\cos\Big(\frac{\pi}{2} + 7\frac{\pi}{2}\Big) = 1 \geq -1.16 \approx 7/10 - 7\frac{5\pi+1}{20\pi}.$$

That is, the Armijo condition is not satisfied, therefore the step size must be adjusted. If $\rho = \frac{5}{7}$, then the step size is adjusted to $\frac{5\pi}{2(1+a)}$, so that $(1+a)\alpha_k = \frac{5\pi}{2}$ and the Armijo condition is satisfied, since

$$\cos\Big(\frac{\pi}{2} + (1+a)\alpha_k\Big) = \cos(3\pi) = -1 \leq -0.83 \approx \frac{1}{2} - 5\frac{5\pi+1}{20\pi}.$$

On the other hand, if $\rho = \frac{3}{7}$, then $(1+a)\alpha_k = \frac{3\pi}{2}$ and the Armijo condition is not satisfied, since

$$\cos\Big(\frac{\pi}{2} + (1+a)\alpha_k\Big) = 1 \geq -0.5 \approx 3/10 - 3\frac{5\pi+1}{20\pi}.$$

Adjusting the step size once more by $\rho = \frac{3}{7}$ produces the step size $\frac{9\pi}{14(1+a)}$ and, in turn, the iterate $x_k + \alpha_k d_k = \frac{\pi}{2} + \frac{9\pi}{14} = \frac{8\pi}{7} \approx 3.59$, which is feasible since

$$\cos\Big(\frac{8\pi}{7}\Big) \approx -1.13 \leq -0.21 \approx \frac{9}{70} - 9\frac{5\pi+1}{140\pi}.$$

In this example, the step size $\frac{3\pi}{2(1+a)}$ is not feasible although it is smaller than $\frac{5\pi}{2(1+a)}$, which is feasible. More generally, this establishes that feasibility is not monotone in the step size for nonconvex functions. Moreover, by reducing $\rho = \frac{5}{7}$ to $\rho = \frac{3}{7}$, backtracking must adjust the initial step size one additional time. Therefore, reducing ρ might increase the number of criteria evaluations to return a feasible step size.

B.2 Convex problems

In this subsection, we present and proof the results for convex problems stated in Section 4.

Proposition 3. *Let F be convex differentiable. Given a point x_k, a direction d_k and a step size $\alpha_k > 0$ satisfying (3) for some c, then x_k, d_k and α_k' also satisfy (3) for any $\alpha_k' \in (0, \alpha_k)$.*

Proof. Let $\beta := \alpha_k'/\alpha_k \in (0, 1)$. Then, expressing $x_k + \alpha_k' d_k$ as $\beta(x_k + \alpha_k d_k) + (1-\beta)x_k$, we obtain

$$\begin{aligned} F(x_k + \alpha_k' d_k) = F(\beta(x_k + \alpha_k d_k) + (1-\beta)x_k) &\leq \beta F(x_k + \alpha_k d_k) + (1-\beta)F(x_k) \\ &\leq \beta(F(x_k) + c\alpha_k\langle\nabla F(x_k), d_k\rangle) + (1-\beta)F(x_k) \\ &= c\alpha_k'\langle\nabla F(x_k), d_k\rangle + F(x_k), \end{aligned}$$

where the first and second follow from F being convex and x_k, d_k and α_k satisfying (3), respectively. □

Proposition 4. *Let F be convex differentiable. Fixing all other inputs, the number of backtracking criteria evaluations that Algorithm 1 takes to return a feasible step size is nondecreasing in the input ρ.*

Proof. Consider the inputs α_0, c and v to Algorithm 1 fixed. Then, let $0 < \rho_1 < \rho_2 < 1$ and let N_1 and N_2 denote the number of adjustments Algorithm 1 takes to compute a feasible step size when it receives respectively ρ_1 and ρ_2 as inputs. If $\rho_i^{N_i}\alpha_0$ is a feasible step size and $N_i' > N_i$ for some $i \in \{1, 2\}$, then so is $\rho_i^{N_i'}\alpha_0$, in view of the fact that $\rho_i^{N_i} < \rho_i^{N_i'}$ and of Proposition 3. Moreover, Algorithm 1 must test if the step size $\rho_i^{N_i}$ is feasible before testing the step size $\rho_1^{N_1'}$ and therefore cannot return $\rho_1^{N_1'}\alpha_0$. Inductively, we conclude that N_1 and N_2 are the least nonnegative integers such that $\rho_i^{N_i}$ are feasible. Now, since $\rho_2^{N_2}$ is feasible, if $N_1 > N_2$, then so is $\rho_1^{N_2} < \rho_2^{N_2}$, in view of the assumption that $\rho_1 < \rho_2$ and of Proposition 3. That is, N_1 is not the least nonnegative integer such that $\rho_1^{N_1}$ is feasible, a contradiction. Moreover, each adjustment requires evaluating the objective function once, so the total number of function evaluations Algorithm 1 takes to return a feasible step size is $N_i + 2$. Therefore, if Algorithm 1 receives ρ_1 as input, then it takes **no more function evaluations** to return a feasible step size than if receives ρ_2 as input.

□

Definition 4. *The inputs to Algorithms 1 and 2 are said to be* compatible *if α_0, c, v coincide and the input $\hat{\rho}$ to Algorithm 2 is parameterized by the same ρ that Algorithm 1 takes as input.*

Proposition 5. *Let F be convex differentiable. Given compatible inputs with a descent direction d_k and $\epsilon < \rho$, Algorithm 2 takes no more function evaluations to return a feasible step size than Algorithm 1 does.*

Proof. Suppose Algorithms 1 and 2 receive compatible inputs. If (3) is violated for some tentative step size α_k, then $v(\alpha_k) < 1$ which together with $c \in (-0, 1)$ imply $1 - c \cdot v(\alpha_k) > 1 - c > 0$. In turn, $\hat{\rho}(v(\alpha_k)) < \rho$ because $\epsilon < \rho$, by assumption. The result follows by repeating the arguments used to prove Proposition 4 above. □

B.3 Nonconvex problems

B.3.1 Armijo condition

Proposition 6 (Armijo feasibility for C^2 functions)**.** *Let F be twice continuously differentiable. Given a base point x_k, a descent direction d_k, an initial step size α_0 and a constant $c \in (0, 1)$ for the Armijo condition (3), there is some $\bar{\alpha} = \bar{\alpha}(x_k, d_k, c) \leq \alpha_0$ such that $x_k + \alpha_k d_k$ satisfies (3) for all $\alpha_k \in (0, \bar{\alpha})$.*

Proof. Assuming F twice continuously differentiable, then by Taylor's theorem (Nocedal & Wright, 2006, p. 14), there exists some $t = t(x_k, d_k, \alpha_k) \in (0, 1)$ such that

$$F(x_k + \alpha_k d_k) = F(x_k) + \alpha_k\langle\nabla F(x_k), d_k\rangle + \alpha_k^2\tfrac{1}{2}\langle d_k, \nabla^2 F(x_k + t\alpha_k d_k)d_k\rangle. \tag{14}$$

Moreover, the eigenvalues of $\nabla^2 F$ are continuous and the line segment $\{x_k + \alpha_k d_k : \alpha_k \in [0, \alpha_0]\}$ is compact, therefore there is some $\lambda > 0$ such that for all $\alpha_k \in (0, \alpha_0)$ and $t \in (0, 1)$

$$\left|d_k^\top \nabla^2 F(x_k + t\alpha_k d_k) d_k\right| \leq \lambda \|d_k\|^2. \tag{15}$$

So, let $\bar{\alpha} = \bar{\alpha}(x_k, d_k, c) := 2(1-c)\langle \nabla F(x_k), -d_k\rangle / (\lambda \|d_k\|^2) > 0$, which is positive since d_k is a descent direction, by assumption. Combining (14) with (15), it follows that if $\alpha_k \in (0, \bar{\alpha})$, then (3) holds. □

For the sake of convenience, we now restate some definition from Section 4.

Definition 5 (Smoothness)**.** *A function F is said to be* Lipschitz-smooth *if (5) holds for some $L > 0$.*

Definition 6 (Gradient related)**.** *The directions d_k are said to be* gradient related *if there are $c_1 > 0$ and $c_2 > 0$ such that $\langle \nabla F(x_k), -d_k\rangle \geq c_1 \|\nabla F(x_k)\|^2$ and $\|d_k\| \leq c_2 \|\nabla F(x_k)\|$, for all $k \geq 0$.*

Given a Lipschitz-smooth function F, we are particularly interested in applying the Descent Lemma (5) with $x = x_k$ and $y = x_k + \alpha_k d_k$, which gives

$$F(x_k + \alpha_k d_k) \leq F(x_k) + \alpha_k \langle \nabla F(x_k), d_k\rangle + \alpha_k^2 \frac{L}{2} \|d_k\|^2. \tag{16}$$

Proposition 7 (Armijo feasibility for Lipschitz-smooth functions)**.** *Let F be Lipschitz-smooth. Given a base point x_k, a descent direction d_k, an initial step size α_0 and a constant $c \in (0, 1)$ for the Armijo condition (3), there is some $\bar{\alpha} = \bar{\alpha}(x_k, d_k, c) \leq \alpha_0$ such that (3) holds for all $\alpha_k \in (0, \bar{\alpha})$. If, in addition, d_k are gradient related, then (3) holds for all $\alpha_k \in (0, \frac{2(1-c)c_1}{Lc_2^2}]$, independent of x_k and d_k.*

Proof. To guarantee (3) holds, we impose that the right-hand side of (16) is less than the right-hand side of (3):

$$F(x_k) + \alpha_k \langle \nabla F(x_k), d_k\rangle + \alpha_k^2 \frac{L}{2} \|d_k\|^2 \leq F(x_k) + c\alpha_k \langle \nabla F(x_k), d_k\rangle.$$

In turn, simplifying the above inequality, it follows that if

$$\alpha_k \leq \frac{2(1-c)\langle \nabla F(x_k), -d_k\rangle}{L\|d_k\|^2}, \tag{17}$$

then (3) holds, where we note that (17) is positive, since d_k is assumed a descent direction.

Now, suppose that $\langle \nabla F(x_k), -d_k\rangle \geq c_1 \|\nabla F(x_k)\|^2$ and $\|d_k\| \leq c_2 \|\nabla F(x_k)\|$ for some $c_1 > 0$ and $c_2 > 0$. Then, for all α_k such that $\alpha_k \leq 2(1-c)c_1 / Lc_2^2$, we have that

$$\alpha_k \leq \frac{2(1-c)}{L} \frac{c_1 \|\nabla F(x_k)\|^2}{c_2^2 \|\nabla F(x_k)\|^2} \leq \frac{2(1-c)\langle \nabla F(x_k), -d_k\rangle}{L\|d_k\|^2}.$$

That is, (17) holds. Therefore, (3) also holds. □

Proposition 8. *Let F be Lipschitz-smooth, $\epsilon < \rho$ and assume $v, \hat{\rho}$ are given by (4). Also, suppose d_k are gradient related. If Algorithms 1 and 2 receive α_k as the initial step size input at iteration $k+1$ for all $k \geq 0$, then they evaluate (3) at most $\lfloor \log_\rho(\bar{\alpha}/\alpha_0) \rfloor + 1 + k$ times up to iteration k, where $\bar{\alpha} := 2(1-c)c_1 / Lc_2^2$. If, on the other hand, Algorithms 1 and 2 receive α_0 as the initial step size input at every iteration, then they evaluate (3) at most $k(\lfloor \log_\rho(\bar{\alpha}/\alpha_0) \rfloor + 1)$ times up to iteration k.*

Proof. Suppose that Algorithm 2 evaluates (3) and it does not hold for a given tentative step size α_k. Then,

$$F(x_k + \alpha_k d_k) - F(x_k) > c\alpha_k \langle \nabla F(x_k), d_k\rangle.$$

Dividing both sides above by $c\alpha_k \langle \nabla F(x_k), d_k\rangle < 0$ gives $v(\alpha_k) < 1$. In turn, since $c \in (0, 1)$, it follows that $1 - c > 1 - cv(\alpha_k) > 0$ and $(1-c)/(1 - c \cdot v(\alpha_k)) < 1$. Plugging this inequality into (4b), we obtain

$$\hat{\rho}(\alpha_k) = \max(\epsilon, \rho(1-c)/(1 - c \cdot v(\alpha_k)) < \rho,$$

since by assumption $\epsilon < \rho$. Therefore, if (3) does not hold for a given tentative step size, then Algorithms 1 and 2 multiply it by a factor of at most ρ to adjust it.

Moreover, by Proposition 7, (3) is satisfied for all $\alpha_k \in (0, \bar{\alpha})$, independently of x_k and d_k.

Hence, if Algorithms 1 and 2 use α_0 as the initial step size for the first iteration and α_k at iteration $k+1$ for $k \geq 0$, then at most $\lfloor \log_\rho(\bar{\alpha}/\alpha_0) \rfloor + 1$ adjustments are necessary until a step size that is uniformly feasible is found. Each adjustment entails evaluating (3) once. In addition, (3) must be evaluated once every iteration. Therefore, (3) is evaluated at most $\lfloor \log_\rho(\bar{\alpha}/\alpha_0) \rfloor + 1 + k$ times up to iteration k.

Now, suppose Algorithms 1 and 2 use α_0 as the initial step size in every iteration. Then, at most $\lfloor \log_\rho(\bar{\alpha}/\alpha_0) \rfloor + 1$ adjustments are necessary in every iteration until a feasible step size is found. As before, each adjustment entails evaluating (3) once, in addition to the first evaluation. Therefore, (3) is evaluated at most $k(\lfloor \log_\rho(\bar{\alpha}/\alpha_0) \rfloor + 1)$ times up to iteration k. □

Proposition 9 (step size lower bounds)**.** *Let F be Lipschitz-smooth. Also, suppose d_k are gradient related. Given appropriate inputs, Algorithm 2 with the choices specified by (4) and Algorithm 1 return a step size α_k such that*

$$\alpha_k \geq \min\left\{\alpha_0, \rho\frac{2(1-c)c_1}{Lc_2^2}\right\} > 0.$$

Proof. Since d_k is a descent direction, dividing both sides of (16) by $\langle \nabla F(x_k), -d_k \rangle > 0$ yields

$$-cv(\alpha_k) = -\frac{F(x_k + \alpha_k d_k) - F(x_k)}{\alpha_k \langle \nabla F(x_k), d_k \rangle} \leq -1 - \frac{\alpha_k L \|d_k\|^2}{2\langle \nabla F(x_k), d_k \rangle}.$$

Hence, if $\hat{\rho}$ is chosen as (4b) and (3) does not hold, then step sizes α_k returned by Algorithm 2 satisfy

$$\hat{\rho}(v(\alpha_k))\alpha_k \geq \rho\frac{1-c}{1-cv(\alpha_k)}\alpha_k \geq \rho\frac{2(1-c)\langle \nabla F(x_k), -d_k \rangle}{L\|d_k\|^2} \geq \rho\frac{2(1-c)c_1}{Lc_2^2} > 0.$$

Moreover, by Proposition 7, the greatest step size for which (3) is guaranteed to hold is $2(1-c)c_1/Lc_2^2$. If $\alpha_0 \geq 2(1-c)c_1/Lc_2^2$, then Algorithm 1 returns a step size at least within a ρ factor of $2(1-c)c_1/Lc_2^2$. □

B.3.2 Descent Lemma

First, note the proximal operator p_{α_k} given by (6) is well-defined. Indeed, given a continuous convex function g, a point y_k and some $\alpha_k > 0$, the map $x \mapsto g(x) + (1/2\alpha_k)\left\|x - (y - \alpha_k \nabla f(y))\right\|^2$ is continuous strongly convex and therefore admits a unique minimum.

Proposition 10 (Lipschitz step size feasibility)**.** *Let f be Lipschitz-smooth convex and let g be continuous convex. Also, suppose v and $\hat{\rho}$ are chosen as (8a) and (8b). If $\alpha_k \in (0, 1/L)$, then (7) holds for all y_k. If Algorithms 1 and 2 receive α_k as the initial step size input in iteration $k+1$, then they evaluate (7) at most $\lfloor \log_\rho(1/L\alpha_0) \rfloor + 1 + k$ times up to iteration k. If, on the other hand, Algorithms 1 and 2 receive α_0 as the initial step size input in every iteration, then they evaluate (7) at most $k(\lfloor \log_\rho(1/L\alpha_0) \rfloor + 1)$ times up to iteration k.*

Proof. Given any y_k, if $\alpha_k \in (0, 1/L)$, then applying (5) with $x = p_{\alpha_k}(y_k)$ and $y = y_k$, we get

$$\begin{aligned} f(p_{\alpha_k}(y_k)) &\leq f(y_k) + \langle \nabla f(y_k), p_{\alpha_k}(y_k) - y_k \rangle + (L/2)\|p_{\alpha_k}(y_k) - y_k\|^2 \\ &\leq f(y_k) + \langle \nabla f(y_k), p_{\alpha_k}(y_k) - y_k \rangle + (1/2\alpha_k)\|p_{\alpha_k}(y_k) - y_k\|^2. \end{aligned}$$

Adding $\psi(p_{\alpha_k}(y_k))$ to both sides, we recover (7). Thus, if $\alpha_k \in (0, \bar{\alpha})$, then (7) holds for all y_k.

Given an initial step size α_k and the points y_k and $p_{\alpha_k}(y_k)$, Algorithm 1 checks if (7) holds. If it does hold, then Algorithm 1 returns α_k, otherwise Algorithm 1 adjusts α_k by ρ, recomputes $p_{\alpha_k}(y_k)$, checks if (7) and repeats. Since (7) is guaranteed to hold for $\alpha_k \in (0, 1/L)$, given an initial step size α_0, Algorithm 1 computes a feasible step size after adjusting α_k at most $\lfloor \log_\rho(1/L\alpha_0) \rfloor + 1$ times. Each time Algorithm 1 adjusts α_k, Algorithm 1 evaluates (7). In addition, Algorithm 1 evaluates Eq. (7) once every time it is called to check if the initial step size is feasible. Hence, if

Algorithm 1 receives α_k as the initial step size input at iteration $k+1$, then it evaluates Eq. (7) at most $\lfloor \log_\rho(1/L\alpha_0) \rfloor + 1 + k$ times up to iteration k. On the other hand, if Algorithm 1 receives the same α_0 as initial step size input at every iteration, then it might have to adjust α_k up to $\lfloor \log_\rho(1/L\alpha_0) \rfloor + 1$ in every iteration, therefore Algorithm 1 evaluates (7) at most $k(\lfloor \log_\rho(1/L\alpha_0) \rfloor + 1)$ times up to iteration k.

Now, consider Algorithm 2, with v and $\hat{\rho}$ chosen as (8a) and (8b). Given an initial step size α_k and the points y_k and $p_{\alpha_k}(y_k)$, Algorithm 2 checks if (7) holds. Suppose (7) does not hold. Then, moving the terms $f(p_{\alpha_k}(y_k))$ and $\langle \nabla f(y_k), p_{\alpha_k}(y_k) - y_k \rangle$ to the left-hand side and cancelling $\psi(p_{\alpha_k}(y_k))$ on both sides, we obtain

$$f(p_{\alpha_k}(y_k)) - f(y_k) - \langle \nabla f(y_k), p_{\alpha_k}(y_k) - y_k \rangle > (1/2\alpha_k)\|p_{\alpha_k}(y_k) - y_k\|^2. \tag{18}$$

Since $\|\cdot\| \geq 0$, the left-hand side must be positive. So, dividing both sides by the left-hand side and using (8a), it follows that $v(\alpha_k) < 1$. Hence, Algorithm 2 adjusts α_k to $\hat{\rho}(\alpha_k)\alpha_k < \rho\alpha_k$. That is, the factor by which Algorithm 2 adjusts α_k is smaller than the factor by which Algorithm 1 adjusts α_k. Therefore, Algorithm 2 evaluates Eq. (7) at most as many times as (1) does. □

Proposition 11. *Let f be Lipschitz-smooth convex and let g be continuous convex. Also, suppose v and $\hat{\rho}$ are chosen as (8a) and (8b). If Algorithms 1 and 2 receive an initial step size $\alpha_0 > 0$, then they return a feasible step size α_k such that $\alpha_k \geq \min\{\alpha_0, \rho/L\}$.*

Proof. By Proposition 10, every step size $\alpha_k \in (0, 1/L)$ is feasible. Hence, if α_0 is not feasible, then since Algorithm 1 adjusts step sizes by ρ, it must return a feasible step size within a ρ factor of $1/L$.

Now, consider Algorithm 2, with v and $\hat{\rho}$ chosen as (8a) and (8b). Algorithm 2 only adjusts α_k when (7) does not hold, so suppose that is the case. Applying (5) with $y = y_k$ and $x_k = p_{\alpha_k}(y_k)$ yields

$$f(p_{\alpha_k}(y_k)) - f(x_k) - \langle \nabla f(y_k), p_{\alpha_k}(y_k) - y_k \rangle \leq (L/2)\|p_{\alpha_k}(y_k) - y_k\|^2.$$

Dividing both sides by $(L/\rho)(f(p_{\alpha_k}(y_k)) - f(x_k) - \langle \nabla f(y_k), p_{\alpha_k}(y_k) - y_k \rangle)$, which is positive by (18), we obtain

$$\frac{\rho}{L} \leq \rho \frac{\frac{1}{2}\|p_{\alpha_k}(y_k) - y_k\|^2}{f(p_{\alpha_k}(y_k)) - f(x_k) - \langle \nabla f(y_k), p_{\alpha_k}(y_k) - y_k \rangle} = \rho v(\alpha_k)\alpha_k,$$

where the identity follows from (8a). Hence, Algorithm 2 adjusts α_k to $\hat{\rho}(v(\alpha_k))\alpha_k \geq \rho/L$. □

B.4 Convergence Results

B.4.1 A general convergence result for adaptive backtracking

Under mild conditions, we now show that $\lim_{k\to+\infty} \|\nabla f(x_k)\|^2 = 0$ for iterates x_k in the form (2) with gradient related d_k and step sizes generated by adaptive backtracking. We emphasize that the following results make no further assumptions on how the descent directions are generated and that (Nocedal & Wright, 2006, p. 40):

For line search methods of the general form (2), the limit $\lim_{k\to+\infty} \|\nabla f(x_k)\|^2 = 0$ is the strongest global convergence result that can be obtained: We cannot guarantee that the method converges to a minimizer, but only that it is attracted by stationary points. Only by making additional requirements on the search direction d_k—by introducing negative curvature information from the Hessian $\nabla^2 f(x_k)$, for example—can we strengthen these results to include convergence to a local minimum

Proposition 12. *Let f be bounded below and Lipschitz-smooth on an open set containing the level set $\{x : f(x) \leq f(x_0)\}$, where x_0 is the initial point of iterates (2) where d_k are gradient related and α_k are generated by adaptive backtracking (Algorithm 2) with some $\alpha_0 > 0$ and using $\hat{\rho}$ and v given by (4). Then, $\lim_{k\to+\infty} \|\nabla f(x_k)\|^2 = 0$.*

Proof. Under the above assumptions, we have that $\alpha_k \geq \min\{\alpha_0, \rho\overline{\alpha}\}$, where $\overline{\alpha} = 2(1-c)c_1/(Lc_2^2)$, by Proposition 9. Moreover, we have that $\langle \nabla f(x_k), d_k \rangle \leq -c_1\|\nabla f(x_k)\|^2$, because d_k are gradient related. Hence, since adaptive backtracking enforces the Armijo condition, (3), it follows that

$$f(x_{k+1}) - f(x_k) \leq -\alpha_k c\|\nabla f(x_k)\|^2 \leq -c\min\{\alpha_0, \rho 2(1-c)c_1/(Lc_2^2)\}\|\nabla f(x_k)\|^2.$$

Telescoping the above difference, we get

$$f(x_{k+1}) - f(x_0) = \sum_{t=1}^{k}(f(x_{t+1}) - f(x_t)) \leq -c\min\Big\{\alpha_0, \rho\frac{2(1-c)c_1}{Lc_2^2}\Big\}\sum_{t=1}^{k}\|\nabla f(x_t)\|^2.$$

Rearranging the above inequality and using the assumption that f is lower bounded, we obtain

$$c\min\Big\{\alpha_0, \rho\frac{2(1-c)c_1}{Lc_2^2}\Big\}\sum_{t=1}^{k}\|\nabla f(x_t)\|^2 \leq f(x_0) - f(x_{k+1}) < +\infty.$$

That is, $\|\nabla f(x_k)\|^2$ are square-summable. Therefore, it follows that

$$\lim_{k\to+\infty}\|\nabla f(x_k)\|^2 = 0.$$

□

B.4.2 Convergence results for gradient descent

We show that the standard convergence results for gradient descent are preserved if step sizes are generated by adaptive backtracking. We address smooth and then smooth strongly convex objectives.

Proposition 13. *Let f be convex, Lipschitz-smooth and suppose $\nabla f(x^*) = 0$ for some x^*. If the step sizes α_k of gradient descent (Algorithm 3) are chosen by adaptive backtracking (Algorithm 2) using $\hat{\rho}$ and v given by (4) with $c \in [1/2, 1)$ and $\alpha_0 > 0$, then $\alpha_k \geq \min\{\alpha_0, \rho\overline{\alpha}\}$, where $\overline{\alpha} = 2(1-c)/L$, and*

$$f(x_k) - f(x^*) \leq \frac{\|x_0 - x^*\|^2}{2\min\{\alpha_0, \rho\overline{\alpha}\}k}.$$

Proof. Under the above assumptions, all iterates of gradient descent satisfy the Armijo condition, (3). Moreover, since f is convex, we have that

$$f(x_k) \leq f(x^*) + \langle\nabla f(x_k), x_k - x^*\rangle.$$

Hence, combining the above inequality with (3), it follows that

$$f(x_{k+1}) \leq f(x_k) - c\alpha_k\|\nabla f(x_k)\|^2 \leq f(x^*) + \langle\nabla f(x_k), x_k - x^*\rangle - c\alpha_k\|\nabla f(x_k)\|^2.$$

In turn, since $c \geq 1/2$, rearranging the above inequality and completing a square, we get

$$\begin{aligned}
f(x_{k+1}) - f(x^*) &\leq \frac{1}{2\alpha_k}(2\alpha_k\langle\nabla f(x_k), x_k - x^*\rangle - \alpha_k^2\|\nabla f(x_k)\|^2) \\
&= \frac{1}{2\alpha_k}(2\alpha_k\langle\nabla f(x_k), x_k - x^*\rangle - \alpha_k^2\|\nabla f(x_k)\|^2 \pm \|x_k - x^*\|^2) \\
&= \frac{1}{2\alpha_k}(\|x_k - \alpha_k\nabla f(x_k) - x^*\|^2 - \|x_k - x^*\|^2) \\
&= \frac{1}{2\alpha_k}(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2).
\end{aligned}$$

Now, since gradient descent sets $d_k = -\nabla f(x_k)$, then d_k are gradient related with $c_1 = c_2 = 1$. Moreover, since f is Lipschitz-smooth, then $\alpha_k \geq \min\{\alpha_0, \rho\overline{\alpha}\}$, where $\overline{\alpha} = 2(1-c)/L$, by Proposition 9. Plugging this lower bound into the above inequality, it follows that

$$f(x_{k+1}) - f(x^*) \leq \frac{1}{2\min\{\alpha_0, \rho\overline{\alpha}\}}(\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2).$$

Telescoping the above, we get

$$\begin{aligned}
\sum_{t=1}^{k}(f(x_{t+1}) - f(x^*)) &\leq \frac{1}{2\min\{\alpha_0, \rho\overline{\alpha}\}}\sum_{t=1}^{k}(\|x_t - x^*\|^2 - \|x_{t+1} - x^*\|^2) \\
&\leq \frac{\|x_0 - x^*\|^2 - \|x_{k+1} - x^*\|^2}{2\min\{\alpha_0, \rho\overline{\alpha}\}} \\
&\leq \frac{\|x_0 - x^*\|^2}{2\min\{\alpha_0, \rho\overline{\alpha}\}}.
\end{aligned}$$

Since $\nabla f(x^*) = 0$ and f is convex, we have that $f(x_{k+1}) - f(x^*) \geq 0$. Moreover, $f(x_k)$ are decreasing because the Armijo condition holds in every iteration. Therefore

$$f(x_{k+1}) - f(x^*) \leq \frac{\|x_0 - x^*\|^2}{2\min\{\alpha_0, \rho\overline{\alpha}\}k}.$$

□

Next, we show that adaptive backtracking also preserves the convergence rate of gradient descent on strongly convex objectives, which we define below.

Definition 7 (Strong convexity)**.** *A continuously differentiable function f is said to be strongly convex if there exists some $m > 0$ such that for every x and y*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle + \frac{m}{2}\|y - x\|^2. \tag{19}$$

Proposition 14. *Let f be Lipschitz-smooth and strongly convex. If the step sizes α_k of gradient descent (Algorithm 3) are chosen by adaptive backtracking (Algorithm 2) using $\hat{\rho}$ and v given by (4) with $c \in [1/2, 1)$ and $\alpha_0 \in (0, 1/m)$, then*

$$f(x_k) - f(x^*) \leq (1 - m\min\{\alpha_0, \rho\overline{\alpha}\})^k \frac{L+m}{2}\|x_0 - x^*\|^2.$$

In particular, if $c = 1/2$ and $\alpha_0 > \rho/L$, then

$$f(x_k) - f(x^*) \leq (1 - \rho q)^k \frac{L+m}{2}\|x_0 - x^*\|^2,$$

where $q = m/L$ is the reciprocal of the condition number of f.

Proof. Let L and m denote the Lipschitz-smoothness and strong convexity constants of f. The assumption that f is strongly convex implies the existence of a unique global minimizer x^* for f. We then use x^* to define a Lyapunov function V, given by

$$V(x_k) = f(x_k) - f(x^*) + \frac{m}{2}\|x_k - x^*\|^2,$$

which is positive for $x_k \neq x^*$. To prove the result, we show that $(1+\delta_k)V(x_{k+1}) - V(x_k) \leq 0$, where

$$\delta_k = \frac{1}{Q_k - 1}, \qquad Q_k = \frac{L_k}{m} \qquad \text{and} \qquad L_k = \frac{1}{\alpha_k}.$$

And we note that the assumption that $\alpha_k \leq \alpha_0 < 1/m$ implies $L_k > m$, thus δ_k are well-defined.

By assumption, the iterates of gradient descent satisfy (3) with $c \in [1/2, 1)$, hence

$$f(x_{k+1}) - f(x_k) \leq -c\alpha_k\|\nabla f(x_k)\|^2 \leq -\frac{\alpha_k}{2}\|\nabla f(x_k)\|^2.$$

Moreover, by strong convexity, we have that

$$f(x_k) - f(x^*) \leq \langle \nabla f(x_k), x_k - x^*\rangle - \frac{m}{2}\|x_k - x^*\|^2.$$

Next, expanding quadratic terms, it follows that

$$\begin{aligned}(1+\delta_k)\|x_{k+1} - x^*\|^2 - \|x_k - x^*\|^2 =& (1+\delta_k)(\alpha_k^2\|\nabla f(x_t)\|^2 - 2\alpha_k\langle \nabla f(x_k), x_k - x^*\rangle) \\ &+ \delta_k\|x_k - x^*\|^2.\end{aligned}$$

Now, from the definition of δ_k, we obtain

$$(1+\delta_k)(1 - m\alpha_k) = \frac{Q_k}{Q_k - 1}\frac{Q_k - 1}{Q_k} = 1 \qquad \text{and} \qquad (1+\delta_k)m\alpha_k = \frac{Q_k}{Q_k - 1}\frac{1}{Q_k} = \delta_k.$$

Then, we put everything together to get

$$\begin{aligned}(1+\delta_k)V(x_{k+1}) - V_k(x_k) \leq& -(1+\delta_k)(1 - m\alpha_k)\frac{\alpha_k}{2}\|\nabla f(x_k)\|^2 \\ &- (\delta_k - (1+\delta_k)m\alpha_k)\langle \nabla f(x_k), x_k - x^*\rangle \\ \leq& -\frac{\alpha_k}{2}\|\nabla f(x_k)\|^2.\end{aligned}$$

Applying the above inequality inductively, it follows that

$$V(x_{k+1}) \leq V(x_0) \prod_{t=1}^{k} \frac{1}{1+\delta_t}.$$

Moreover, applying (5) with $y = x_0$ and $x = x^*$, and noticing that $\nabla f(x^*) = 0$, we obtain

$$V(x_0) = f(x_0) - f(x^*) + \frac{m}{2}\|x_0 - x^*\|^2 \leq \frac{L+m}{2}\|x_0 - x^*\|^2.$$

Furthermore, under the above assumptions, we have that $\alpha_k \geq \min\{\alpha_0, \rho\overline{\alpha}\}$, where $\overline{\alpha} = 2(1-c)/L$, which implies that

$$1 + \delta_k = \frac{Q_k}{Q_k - 1} = \frac{1}{1 - m\alpha_k} \geq \frac{1}{1 - m\min\{\alpha_0, \rho\overline{\alpha}\}}.$$

Finally, we put everything together and obtain

$$\begin{aligned} f(x_k) - f(x^*) \leq V(x_{k+1}) \leq & \frac{L+m}{2}\|x_0 - x^*\|^2 \prod_{t=1}^{k} \frac{1}{1+\delta_t} \\ \leq & (1 - m\min\{\alpha_0, \rho\overline{\alpha}\})^k \frac{L+m}{2}\|x_0 - x^*\|^2. \end{aligned}$$

□

B.4.3 A CONVERGENCE RESULT FOR ACCELERATED GRADIENT DESCENT

To establish that adaptive backtracking preserves the convergence rate of accelerated gradient descent, we employ a Lyapunov argument based on the function V_k defined by

$$V_t(x_k, y_k) = f(y_k) - f(x^*) + \frac{m}{2}\|z_t - x^*\|^2, \tag{20}$$

where the point $z_t = z_t(x_k, y_k)$ is defined as

$$z_t = x_k + \sqrt{Q_{t-1}}(x_k - y_k), \tag{21}$$

and the estimated condition number Q_t and estimated Lipschitz constant are given by

$$Q_t = \begin{cases} L_0/m, & t < 0, \\ L_t/m, & t \geq 0, \end{cases} \qquad \text{and} \qquad L_t = \frac{1}{\alpha_t}. \tag{22}$$

Note that the index t of z_t follows that of V_t but is independent of the indices of x_k and y_k, which allows us to split the Lyapunov analysis in two auxiliary lemmas. First, we show that for a fixed index $k+1$, the Lyapunov function V_{k+1} decreases along consecutive AGD iterates at an accelerated rate. Second, we bound by how much V_{k+1} can increase with respect to V_k for the same AGD iterate.

Lemma 1. *Let f be Lipschitz-smooth and strongly convex. If the Lipschitz constant estimates L_k of accelerated gradient descent (Algorithm 4) are generated by adaptive backtracking (Algorithm 2) using $\hat{\rho}$ and v given by (4) with $c \in [1/2, 1)$ and $L_0 > m$, then for $k \geq 0$*

$$(1 + \delta_{k+1})V_{k+1}(y_{k+1}, x_{k+1}) - V_{k+1}(y_k, x_k) \leq 0,$$

where $\delta_{k+1} = 1/(\sqrt{Q_k} - 1)$.

Proof. We start by splitting $(1 + \delta_{k+1})(f(y_{k+1}) - f(x^*))$ into three further differences:

$$\begin{aligned} &(1 + \delta_{k+1})(f(y_{k+1}) - f(x^*)) - (f(y_k) - f(x^*) \\ =&(1 + \delta_{k+1})(f(y_{k+1}) - f(x_k)) + \delta_{k+1}(f(x_k) - f(x^*)) + (f(x_k) - f(y_k)). \end{aligned}$$

Since $c \in [1/2, 1)$, then adaptive backtracking generates L_k such that

$$(1 + \delta_{k+1})(f(y_{k+1}) - f(x_k)) \leq -(1 + \delta_{k+1})\frac{1}{2L_k}\|\nabla f(x_k)\|^2. \tag{23}$$

Moreover, applying (19) with $x = x_k$ and $y = x^*$ and using that f is convex, we get

$$\delta_{k+1}(f(x_k) - f(x^*)) \leq \delta_{k+1} \langle \nabla f(x_k), x_k - x^* \rangle - \delta_{k+1} \frac{m}{2} \|x_k - x^*\|^2, \tag{24}$$

$$f(x_k) - f(y_k) \leq \langle \nabla f(x_k), x_k - y_k \rangle . \tag{25}$$

Next, we express the difference $z_{k+1} - x^*$ as

$$\begin{aligned} z_{k+1} - x^* &= x_{k+1} + \sqrt{Q_k}(x_{k+1} - y_{k+1}) - x^* \\ &= y_{k+1} + \beta_k(y_{k+1} - y_k) + \sqrt{Q_k}\beta_k(y_{k+1} - y_k) - x^* \\ &= -\frac{1}{L_k}(1 + \beta_k(1 + \sqrt{Q_k}))\nabla f(x_k) + \beta_k(1 + \sqrt{Q_k})(x_k - y_k) + x_k - x^* \\ &= -\frac{1}{L_k}\sqrt{Q_k}\nabla f(x_k) + (\sqrt{Q_k} - 1)(x_k - y_k) + x_k - x^*, \end{aligned}$$

where we used the identities

$$1 + \beta_k(1 + \sqrt{Q_k}) = \sqrt{Q_t} \qquad \text{and} \qquad \beta_k(1 + \sqrt{Q_k}) = \sqrt{Q_k} - 1.$$

In the same vein, when expanding the 2-norm term $\|z_{k+1} - x^*\|^2$ below, we use the following identities after colons to simplify the coefficients of terms before colons:

$$\begin{aligned} \|\nabla f(x_k)\|^2 :& \quad (Q_k/L_k^2)(m/2) = 1/2L_k, \\ \langle \nabla f(x_k), x_k - y_k \rangle :& \quad m(1 + \delta_{k+1})\sqrt{Q_k}(\sqrt{Q_k} - 1)/L_k = 1, \\ \langle \nabla f(x_k), x_k - x^* \rangle :& \quad m(1 + \delta_{k+1})\sqrt{Q_k}/L_k = \delta_k, \\ \|x_k - y_k\|^2 :& \quad (1 + \delta_{k+1})(\sqrt{Q_k} - 1)^2 = \sqrt{Q_k}(\sqrt{Q_k} - 1), \\ \langle x_k - y_k, x_k - x^* \rangle :& \quad (1 + \delta_{k+1})(\sqrt{Q_k} - 1) = \sqrt{Q_k}. \end{aligned}$$

As a result, the 2-norm difference in $(1 + \delta_{k+1})V_{k+1}(y_{k+1}, x_{k+1}) - V_{k+1}(y_k, x_k)$ becomes

$$\begin{aligned} &(1 + \delta_{k+1})\frac{m}{2}\|z_{k+1} - x^*\|^2 - \frac{m}{2}\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 \\ =& \frac{1 + \delta_{k+1}}{2L_k}\|\nabla f(x_k)\|^2 - \langle \nabla f(x_k), x_k - y_k \rangle - \delta_k \langle \nabla f(x_k), x_k - x^* \rangle \\ & \frac{m}{2}\sqrt{Q_k}(\sqrt{Q_k} - 1)\|x_k - y_k\|^2 + \frac{m}{2}(2\sqrt{Q_k}\langle x_k - y_k, x_k - x^* \rangle + (1 + \delta_{k+1})\|x_k - x^*\|^2) \\ & - \frac{m}{2}(Q_k\|x_k - y_k\|^2 + 2\sqrt{Q_k}\langle x_k - y_k, x_k - x^* \rangle + \|x_k - x^*\|^2) \\ =& \frac{1 + \delta_{k+1}}{2L_k}\|\nabla f(x_k)\|^2 - \langle \nabla f(x_k), x_k - y_k \rangle - \delta_k \langle \nabla f(x_k), x_k - x^* \rangle \\ & - \frac{m}{2}\sqrt{Q_k}\|x_k - y_k\|^2 + \delta_k \frac{m}{2}\|x_k - x^*\|^2. \end{aligned} \tag{26}$$

Finally, combining (23) to (26) and then canceling several terms, we obtain

$$(1 + \delta_{k+1})V_{k+1}(y_{k+1}, x_{k+1}) - V_{k+1}(y_k, x_k) \leq -\frac{m}{2}\sqrt{Q_k}\|x_k - y_k\|^2 \leq 0.$$

□

Lemma 2. *Let f be Lipschitz-smooth strongly convex. Given initial points $x_0 = y_0$, if the estimates L_k of the Lipschitz constant in accelerated gradient descent (Algorithm 4) are generated monotonically by adaptive backtracking (Algorithm 2 with L_k serving as the initial estimate for L_{k+1}) using $\hat{\rho}$ and v given by (4) with $c \in [1/2, 1)$ and $L_0 > m$, then for $k \geq 0$*

$$V_{k+1}(y_k, x_k) \leq \frac{Q_k^2}{Q_{k-1}^2} V_k(y_k, x_k).$$

Proof. We argue by induction. If x_0 and y_0 match, then

$$z_1(y_0, x_0) = x_0 + Q_0(x_0 - y_0) = x_0 = x_0 + Q_{-1}(x_0 - y_0) = z_0(y_0, x_0).$$

Moreover, $Q_{-1} = Q_0$, by definition. Therefore, we have that

$$\begin{aligned} V_1(y_0, x_0) &= f(y_0) - f(x^*) + \frac{m}{2}\|z_1(y_0, x_0) - x^*\|^2 \\ &= \frac{Q_0^2}{Q_{-1}^2}(f(y_0) - f(x^*) + \frac{m}{2}\|z_0(y_0, x_0) - x^*\|^2) \\ &= \frac{Q_0^2}{Q_{-1}^2} V_0(y_0, x_0), \end{aligned}$$

which establishes the base case. To prove the inductive step, we divide the analysis in two cases, each representing a possible sign of $\langle x_k - y_k, x_k - x^*\rangle$. For each case, we bound

$$\begin{aligned} &\|x_k - x^* + \sqrt{Q_k} x_k - y_k\|^2 - \|z_k - x^*\|^2 \\ &= 2(\sqrt{Q_k} - \sqrt{Q_{k-1}})\langle x_k - x^*, x_k - y_k\rangle + (Q_k - Q_{k-1})\|x_k - y_k\|^2. \end{aligned} \tag{27}$$

In turn, bounds on (27) translate into bounds on $V_{k+1}(y_k, x_k) - V_k(y_k, x_k)$, since

$$V_{k+1}(y_k, x_k) - V_k(y_k, x_k) = \frac{m}{2}(\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 - \|z_k - x^*\|^2). \tag{28}$$

Then, to prove the inductive step, we express bounds on (28) in terms of V_{k+1} and V_k.

First, suppose $\langle x_k - y_k, x_k - x^*\rangle \geq 0$. Also assuming $L_k \geq L_{k-1}$, then $\sqrt{Q_{k-1}}/\sqrt{Q_k} \leq 1$, so that

$$\sqrt{Q_k} - \sqrt{Q_{k-1}} \leq \frac{Q_k}{\sqrt{Q_k}} - \sqrt{Q_{k-1}}\frac{\sqrt{Q_{k-1}}}{\sqrt{Q_k}} = \frac{Q_k - Q_{k-1}}{\sqrt{Q_k}}.$$

Hence, applying the above inequality to (27) and then adding a nonnegative $\|x_k - x^*\|^2$ term to it, we get

$$\begin{aligned} &\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 - \|z_k - x^*\|^2 \\ &\leq 2\frac{Q_k - Q_{k-1}}{\sqrt{Q_k}}\langle x_k - x^*, x_k - y_k\rangle + (Q_k - Q_{k-1})\|x_k - y_k\|^2 + \frac{Q_k - Q_{k-1}}{Q_k}\|x_k - x^*\|^2 \\ &= \frac{Q_k - Q_{k-1}}{Q_k}\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2. \end{aligned} \tag{29}$$

Plugging (29) back into (28) yields

$$\begin{aligned} V_{k+1}(y_k, x_k) - V_k(y_k, x_k) &\leq \frac{Q_k - Q_{k-1}}{Q_k}\frac{m}{2}\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 \\ &\leq \frac{Q_k - Q_{k-1}}{Q_k} V_{k+1}(y_k, x_k), \end{aligned} \tag{30}$$

where the last inequality follows from the definition of V_k, as $f(y_k) - f(x^*) \geq 0$ implies

$$V_{k+1}(y_k, x_k) \geq \frac{m}{2}\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2. \tag{31}$$

Thus, rearranging terms in (30) and then multiplying both sides by Q_k/Q_{k-1}, we obtain

$$V_{k+1}(y_k, x_k) \leq \frac{Q_k}{Q_{k-1}} V_k(y_k, x_k) \leq \frac{Q_k^2}{Q_{k-1}^2} V_k(y_k, x_k),$$

where the second inequality holds because $Q_k/Q_{k-1} \geq 1$.

Now, suppose $\langle x_k - y_k, x_k - x^*\rangle < 0$. As in the previous case, we start by bounding the gap (27). But given the negative sign of $\langle x_k - y_k, x_k - x^*\rangle$ term, we bound the $\|x_k - y_k\|^2$ term instead. To this end, we first invoke the assumption that $\langle x_k - y_k, x_k - x^*\rangle < 0$ to establish that

$$\begin{aligned} \|y_k - x^*\|^2 &= \|x_k - x^* - (x_k - y_k)\|^2 \\ &= \|x_k - x^*\|^2 - 2\langle x_k - x^*, x_k - y_k\rangle + \|x_k - y_k\|^2 \\ &\geq \|x_k - x^*\|^2. \end{aligned} \tag{32}$$

To use the above inequality on (27), first we rewrite it more conveniently as

$$\begin{aligned}
&\|x_k - x^* + \sqrt{Q_k} x_k - y_k\|^2 - \|z_k - x^*\|^2 \\
=&2\frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\langle x_k - x^*, \sqrt{Q_k}(x_k - y_k)\rangle + \sqrt{Q_k}(\sqrt{Q_k} - \sqrt{Q_{k-1}})\|x_k - y_k\|^2 \\
&+ \sqrt{Q_{k-1}}(\sqrt{Q_k} - \sqrt{Q_{k-1}})\|x_k - y_k\|^2 \pm \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\|x_k - x^*\|^2 \\
=& \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 + \sqrt{Q_{k-1}}(\sqrt{Q_k} - \sqrt{Q_{k-1}})\|x_k - y_k\|^2 \\
&- \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\|x_k - x^*\|^2. \qquad (33)
\end{aligned}$$

Next, we use the following elementary inequality, which is a consequence of $\|a/c + bc\|^2 \geq 0$:

$$\|a - b\|^2 = \|a\|^2 - 2\langle a, b\rangle + \|b\|^2 \leq (1 + 1/c^2)\|a\|^2 + (1 + c^2)\|b\|^2.$$

Namely, we apply the above inequality with $a = z_k - x^*$, $b = x_k - x^*$ and $c^2 = \sqrt{Q_{k-1}}/\sqrt{Q_k}$ to bound the $\|x_k - y_k\|^2$ term on (33) and obtain

$$\begin{aligned}
&\sqrt{Q_{k-1}}(\sqrt{Q_k} - \sqrt{Q_{k-1}})\|x_k - y_k\|^2 \\
&= \sqrt{Q_{k-1}}(\sqrt{Q_k} - \sqrt{Q_{k-1}})\|x_k - y_k \pm (x_k - x^*)/\sqrt{Q_{k-1}}\|^2 \\
&= \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}}\|z_k - x^* - (x_k - x^*)\|^2 \\
&\leq \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}}\Big(1 + \frac{\sqrt{Q_k}}{\sqrt{Q_{k-1}}}\Big)\|z_k - x^*\|^2 + \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}}\Big(1 + \frac{\sqrt{Q_{k-1}}}{\sqrt{Q_k}}\Big)\|x_k - x^*\|^2 \\
&= \frac{Q_k - Q_{k-1}}{Q_{k-1}}\|z_k - x^*\|^2 + \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\frac{\sqrt{Q_k} + \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}}\|x_k - x^*\|^2. \qquad (34)
\end{aligned}$$

Plugging (34) back into (33) and then using (32), we get

$$\begin{aligned}
&\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 - \|z_k - x^*\|^2 \\
&\leq \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 + \frac{Q_k - Q_{k-1}}{Q_{k-1}}\|z_k - x^*\|^2 \\
&\quad + \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\Big(\frac{\sqrt{Q_k} + \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}} - 1\Big)\|x_k - x^*\|^2 \\
&\leq \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 + \frac{Q_k - Q_{k-1}}{Q_{k-1}}\|z_k - x^*\|^2 \\
&\quad + \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}}\|y_k - x^*\|^2. \qquad (35)
\end{aligned}$$

In turn, plugging (35) back into (28) and then using the assumptions that $m \geq m$ and $m \leq m$ yields

$$\begin{aligned}
&V_{k+1}(y_k, x_k) - V_k(y_k, x_k) \\
&\leq \frac{m}{2}\frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 + \frac{m}{2}\frac{Q_k - Q_{k-1}}{Q_{k-1}}\|z_k - x^*\|^2 \\
&\quad + \frac{m}{2}\frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}}\|y_k - x^*\|^2 \\
&\leq \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_k}}\frac{m}{2}\|x_k - x^* + \sqrt{Q_k}(x_k - y_k)\|^2 + \frac{Q_k - Q_{k-1}}{Q_{k-1}}\frac{m}{2}\|z_k - x^*\|^2 \\
&\quad + \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}}\frac{m}{2}\|y_k - x^*\|^2. \qquad (36)
\end{aligned}$$

Now, as in (31), the fact that $f(y_k) - f(x^*) \geq 0$ implies

$$V_k(y_k, x_k) = f(y_k) - f(x^*) + \frac{m}{2}\|z_k - x^*\|^2 \geq \frac{m}{2}\|z_k - x^*\|^2. \tag{37}$$

In the same vein, applying (19) with $x = x^*$ and $y = y_k$ to the definition of V_k, we obtain

$$V_k(y_k, x_k) = f(y_k) - f(x^*) + \frac{m}{2}\|z_k - x^*\|^2 \geq \frac{m}{2}\|y_k - x^*\|^2. \tag{38}$$

Plugging in (31), (37) and (38) back into (36), and then moving all $V_{k+1}^{acc}(y_k, x_k)$ terms to the left-hand side and all $V_k(y_k, x_k)$ to the right-hand side, we obtain

$$\frac{\sqrt{Q_{k-1}}}{\sqrt{Q_k}} V_{k+1}(y_k, x_k) \leq \Big(\frac{Q_k}{Q_{k-1}} + \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}}\Big) V_k(y_k, x_k) \tag{39}$$

Multiplying both sides of (39) by $\sqrt{Q_k}/\sqrt{Q_{k-1}}$, and then using the fact that $\sqrt{Q_k} \geq \sqrt{Q_{k-1}}$ yields

$$V_{k+1}(y_k, x_k) \leq \frac{\sqrt{Q_k}}{\sqrt{Q_{k-1}}}\Big(\frac{Q_k}{Q_{k-1}} + \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}}\Big) V_k(y_k, x_k) \leq \frac{Q_k^2}{Q_{k-1}^2} V_k(y_k, x_k),$$

where the last inequality above holds because $Q_k \geq Q_{k-1}$ implies the following equivalences hold:

$$\begin{aligned} \frac{Q_k}{Q_{k-1}} + \frac{\sqrt{Q_k} - \sqrt{Q_{k-1}}}{\sqrt{Q_{k-1}}} \leq \frac{Q_k^{3/2}}{Q_{k-1}^{3/2}} &\iff \sqrt{Q_{k-1}} Q_k + Q_{k-1}(\sqrt{Q_k} - \sqrt{Q_{k-1}}) \leq Q_k^{3/2}, \\ &\iff Q_{k-1}(\sqrt{Q_k} - \sqrt{Q_{k-1}}) \leq Q_k(\sqrt{Q_k} - \sqrt{Q_{k-1}}). \end{aligned}$$

Therefore, both when $\langle x_k - x^*, x_k - y_k \rangle \geq 0$ and when $\langle x_k - x^*, x_k - y_k \rangle < 0$, the inequality

$$V_{k+1}(y_k, x_k) \leq \frac{Q_k^2}{Q_{k-1}^2} V_k(y_k, x_k)$$

holds generically for all y_k, x_k, proving the lemma. □

Proposition 15. *Let f be Lipschitz-smooth strongly convex. Given initial points $x_0 = y_0$, if the estimates L_k of the Lipschitz constant in accelerated gradient descent (Algorithm 4) are generated monotonically by adaptive backtracking (Algorithm 2 with L_k serving as the initial estimate for L_{k+1}) using $\hat{\rho}$ and v given by (4) with $c \in [1/2, 1)$ and $L_0 > m$, then for $k \geq 0$*

$$f(y_{k+1}) - f(x^*) \leq \left(\frac{\sqrt{Q} - \sqrt{2(1-c)\rho}}{\sqrt{Q}}\right)^k \frac{Q^2}{4(1-c)^2\rho^2} \frac{L+m}{2}\|x_0 - x^*\|^2.$$

Proof. Combining Lemmas 1 and 2, we have that for every $k \geq 0$

$$V_{k+1}(y_{k+1}, x_{k+1}) \leq \frac{1}{1+\delta_k} V_{k+1}(y_k, x_k) \leq \frac{1}{1+\delta_k}\frac{Q_k^2}{Q_{k-1}^2} V_k(y_k, x_k).$$

Moreover, from Proposition 9 and the assumption that $L_0 > m$, it follows that

$$L_k \leq \max\{L_0, L/(2(1-c)\rho)\} \leq \max\{m, L/(2(1-c)\rho)\}$$

and, in turn, we obtain

$$\frac{1}{1+\delta_k} \leq \frac{\sqrt{Q_k} - 1}{\sqrt{Q_k}} \leq \frac{\sqrt{Q} - \sqrt{2(1-c)\rho}}{\sqrt{Q}} \qquad \text{where} \qquad Q = \frac{L}{m}. \tag{40}$$

Furthermore, assuming $y_0 = x_0$, we have that

$$V_0(y_0, x_0) = f(y_0) - f(x^*) + \frac{m}{2}\|z_0 - x^*\|^2 \leq \frac{L+m}{2}\|x_0 - x^*\|^2.$$

Arguing inductively, all but Q_k^2 and $Q_{-1}^2 = Q_0^2$ cancel and, since $L_0 > m$, we get

$$\begin{aligned} f(y_{k+1}) - f(x^*) &\leq V_{k+1}(y_{k+1}, x_{k+1}) \\ &\leq \left(\frac{\sqrt{Q} - \sqrt{2(1-c)\rho}}{\sqrt{Q}}\right)^k \frac{L_k^2}{L_0^2} V_0(y_0, x_0) \\ &\leq \left(\frac{\sqrt{Q} - \sqrt{2(1-c)\rho}}{\sqrt{Q}}\right)^k \frac{Q^2}{4(1-c)^2\rho^2} \frac{L+m}{2}\|x_0 - x^*\|^2. \end{aligned}$$

□

C METHODS

In this section, we briefly state standard implementations of the base methods that we use in the paper. For the sake of simplicity, we only state a single iteration of the corresponding method.

Algorithms 3 and 4 summarize gradient descent and Nesterov's accelerated gradient descent in the formulation with constant momentum coefficient (Nesterov, 2018, 2.2.22). Algorithm 5 summarizes Adagrad (Duchi et al., 2011).

To state the last base method that we consider in this paper, we must introduce an auxiliary operator. Given a convex Lipschitz-smooth function f with Lipschitz constant L and a continuous convex function, the proximal operator p_L is defined by

$$p_L(y) \coloneqq \arg\min_x \left\{ g(x) + \frac{L}{2} \left\| x - \left(y - \frac{1}{L} \nabla f(x) \right) \right\|^2 \right\}. \tag{41}$$

With the above definition, we can state Algorithm 6, which summarizes FISTA (Beck & Teboulle, 2009).

Algorithm 3 Gradient Descent.

Input: $x_k, \nabla f(x_k), \alpha_k > 0$
Output: x_{k+1}
1: $x_{k+1} \leftarrow x_k - \alpha_k \nabla f(x_k)$

Algorithm 4 Nesterov's accelerated gradient descent (Nesterov, 2018, 2.2.22).

Input: $x_k, y_k, \nabla f(x_k), L_k > m > 0$
Output: x_{k+1}, y_{k+1}
1: $y_{k+1} \leftarrow x_k - (1/L_k) \nabla f(x_k)$
2: $\beta_k \leftarrow \frac{\sqrt{L_k} - \sqrt{m}}{\sqrt{L_k} + \sqrt{m}}$
3: $x_{k+1} \leftarrow (1 + \beta_k) y_{k+1} - \beta_k y_k$

Algorithm 5 Adagrad (Duchi et al., 2011). Superscript i means the i-th entry of the vector.

Input: $x_k, \nabla f(x_k), y_k, x_k \alpha_k > 0$
Output: x_{k+1}, s_{k+1}
1: $s^i_{k+1} = y_k, x^i_k + (\nabla f(x_k)^i)^2$
2: $x^i_{k+1} \leftarrow x^i_k - \frac{\alpha_k}{\sqrt{s^i_{k+1}}} \nabla f(x^i_k)$

Algorithm 6 FISTA (Beck & Teboulle, 2009).

Input: $x_k, x_{k-1}, y_k, t_k, \nabla f(x_k)$
Output: $x_{k+1}, y_{k+1}, t_{k+1}$
1: $x_{k+1} \leftarrow p_L(y_k)$
2: $t_{k+1} \leftarrow \frac{1 + \sqrt{1 + 4t_k^2}}{2}$
3: $y_{k+1} \leftarrow x_k + \frac{t_k - 1}{t_{k+1}} (x_k - x_{k-1})$

D LOGISTIC REGRESSION EXPERIMENTS

In this section, we provide further details and present full plots of all runs for the logistic regression experiments.

Table 4: Details of datasets and method precisions used in the logistic regression problem.

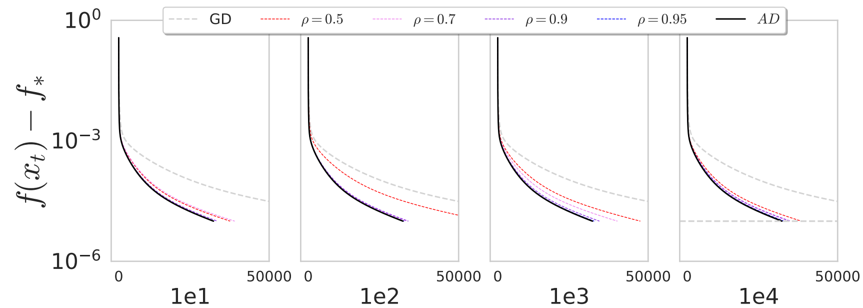
dataset	datapoints	dimensions	AGD	GD	GD (monotone)	Adagrad
a9a	32561	123	10^{-9}	10^{-6}	10^{-5}	10^{-6}
gisette_scale	6000	5000	10^{-9}	10^{-9}	10^{-5}	10^{-9}
MNIST	60000	784	10^{-9}	10^{-6}	10^{-3}	10^{-9}
mushrooms	8124	112	10^{-9}	10^{-9}	10^{-5}	10^{-9}
phishing	11055	68	10^{-9}	10^{-9}	10^{-6}	10^{-6}
protein	102025	75	10^{-9}	10^{-9}	10^{-5}	10^{-9}
web-1	2477	300	10^{-9}	10^{-9}	10^{-8}	10^{-9}

Initialization details. For Lipschitz-smooth problems, a step size of $1/\bar{L}$ is guaranteed to satisfy the Armijo condition (with $c = 1/2$) if $\bar{L} \geq L$. Accordingly, we consider four choices of initial step sizes, $\alpha = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$, which capture the transition from initial step sizes that do not require adjustments to satisfy the Armijo condition to step sizes that do. In practice, L is unknown and the transition would occur as one attempted an arbitrary initial step size and adjusted it correspondingly until the line search was activated. Hence, using $\bar{L}$ to anchor the choice of initial step sizes is merely an educated guess of the transition values that would be found in practice. We adopt the standard choice $c = 10^{-4}$ (Nocedal & Wright, 2006, p. 62) in (3) for BLS used with GD and Adagrad but, motivated by both theory and practice, we choose $c = 1/2$ in the case of AGD. Also, we use the regularization parameter γ as the strong convexity parameter input for AGD.

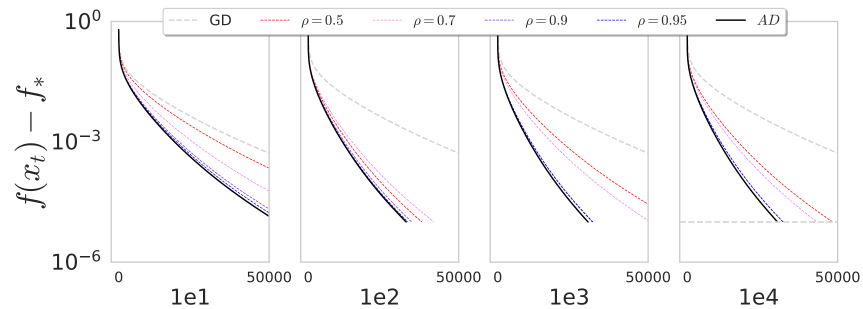
Additional comments. We make the following additional remarks and observations:

- We considered two ways to initialize the step size for line search at each iteration: (1) using the step size from the previous iteration and (2) using the same fixed step size at every iteration. We refer to the corresponding line search subroutines as *monotone* and *memoryless*. The monotone variants are robust to every choice of ρ while some values of ρ may turn the memoryless variants of AGD unstable or unacceptably slow. When the memoryless variants work, however, they generally work much better than the corresponding monotone variants and the baseline methods.
- Monotone backtracking is not as appealing as memoryless backtracking because although both variants take fewer iterations than the baseline method does to reach a given precision, the savings in iterations generated by the monotone variants are not enough to outweigh the additional computational cost of function evaluations that the same variants accrue. Therefore, we only report results for the memoryless variant in the main text and defer results for the monotone variant to Appendix D.1.
- The initial step sizes greatly impact performance. For some starting step sizes, vanilla backtracking is better suited for finding the optimal solution than our adaptive method. However, we find that there tends to be more variance in the performance of vanilla backtracking.
- When $\bar{L}$ is a good estimate of the true Lipschitz constant, the computational cost of function evaluations may outweigh the savings in gradient evaluation and even memoryless backtracking might not improve on the baseline method. This is the case for the COVTYPE dataset from LIBSVM (Chang & Lin, 2011), as shown by Fig. 17b, in Appendix D.2.
- The corresponding stable values of ρ for the adaptive counterpart of AGD lie in the upper interval $(0.7, 1)$ and usually greater values of ρ make the adaptive variant more stable but also more computationally expensive. AGD with regular memoryless backtracking fails to consistently converge for values of ρ outside the interval $(0.3, 0.5)$. In fact, on COVTYPE, for at least one of the initial step sizes, AGD with regular memoryless backtracking line search fails to converge. On the other hand, as shown in Fig. 17b in Appendix D.2, the adaptive variant converges for $\rho = 0.9$ and even for $\rho = 0.7$, the more unstable end of feasible ρ values.

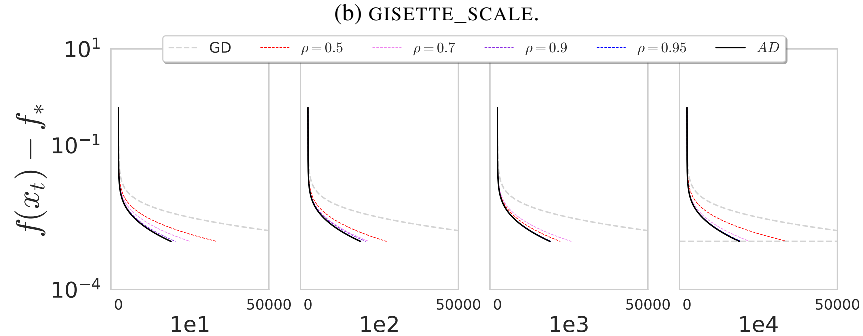
D.1 MONOTONE BACKTRACKING LINE SEARCH



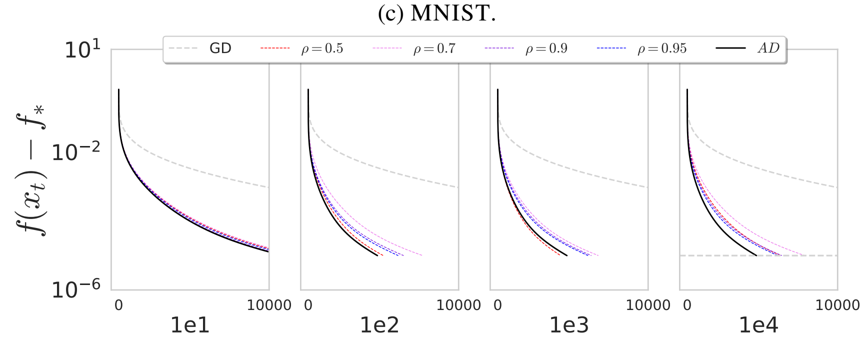
(a) ADULT.



(b) GISETTE_SCALE.

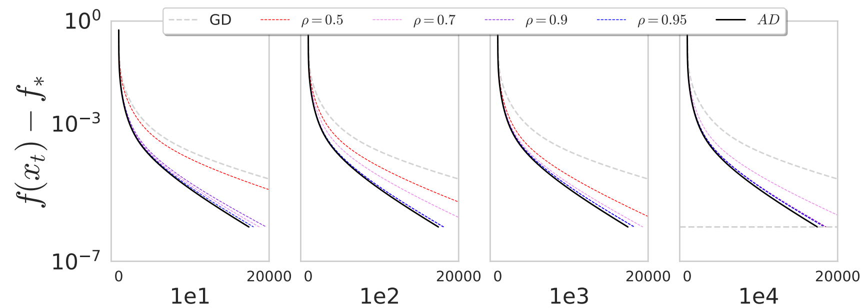


(c) MNIST.

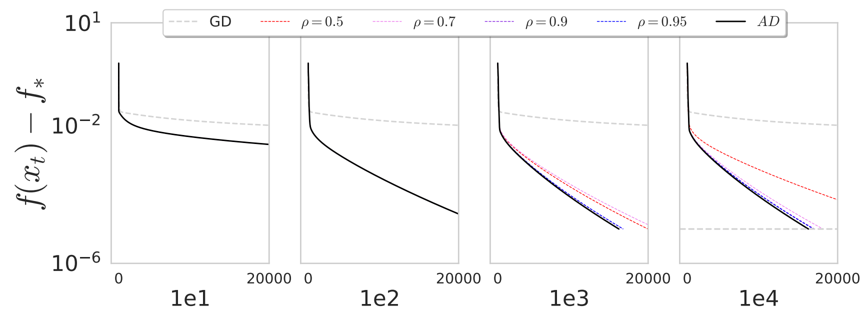


(d) MUSHROOMS.

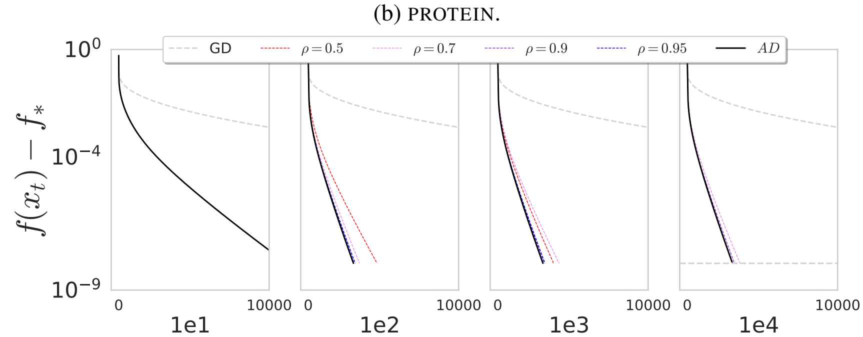
Figure 6: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for GD, GD with standard backtracking line search using $\rho \in \{0.5, 0.7, 0.9, 0.95\}$ and GD with adaptive memoryless backtracking line search using $\rho = 0.9$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.



(a) PHISHING.

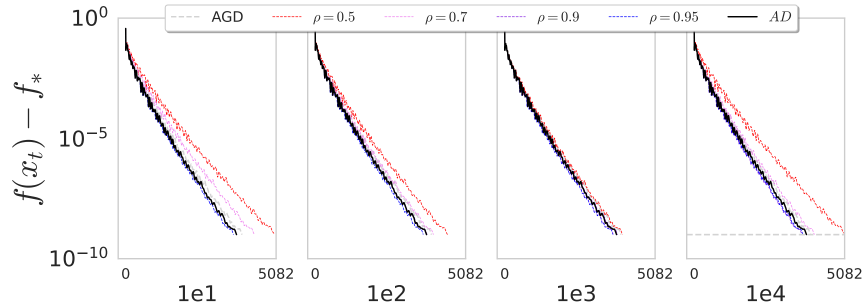


(b) PROTEIN.

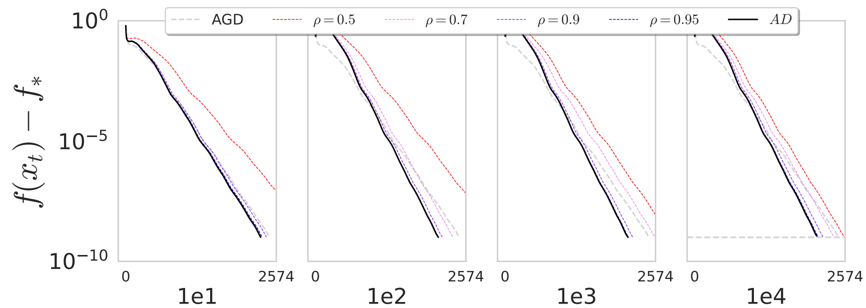


(c) WEB-1.

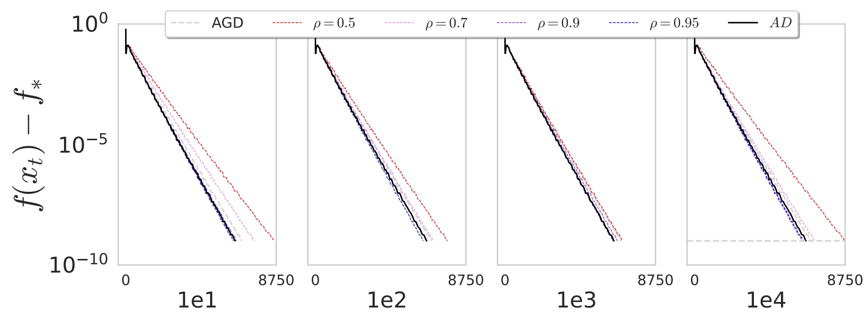
Figure 7: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for GD, GD with standard backtracking line search using $\rho \in \{0.5, 0.7, 0.9, 0.95\}$ and GD with adaptive memoryless backtracking line search using $\rho = 0.9$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.



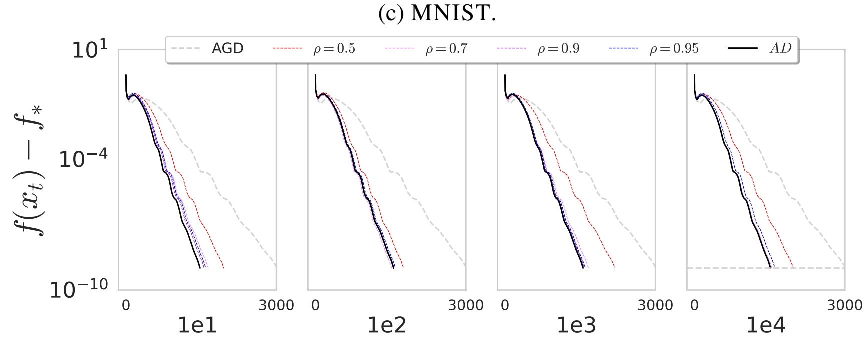
(a) ADULT.



(b) GISETTE_SCALE.

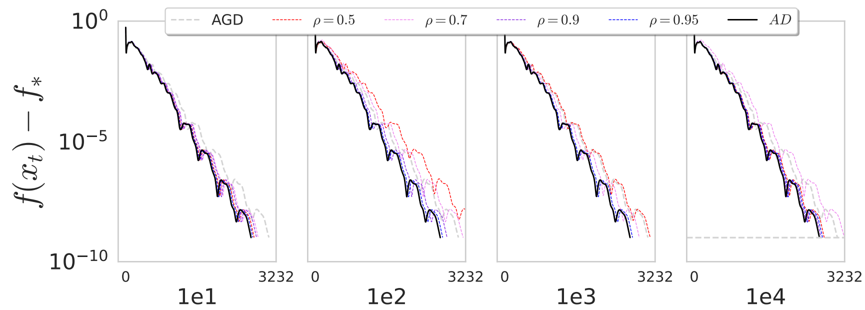


(c) MNIST.

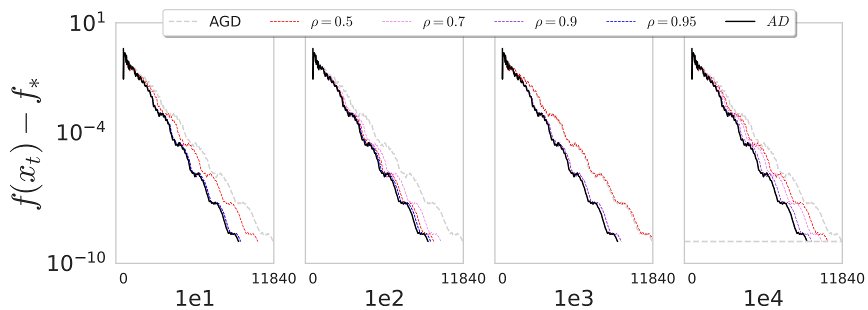


(d) MUSHROOMS.

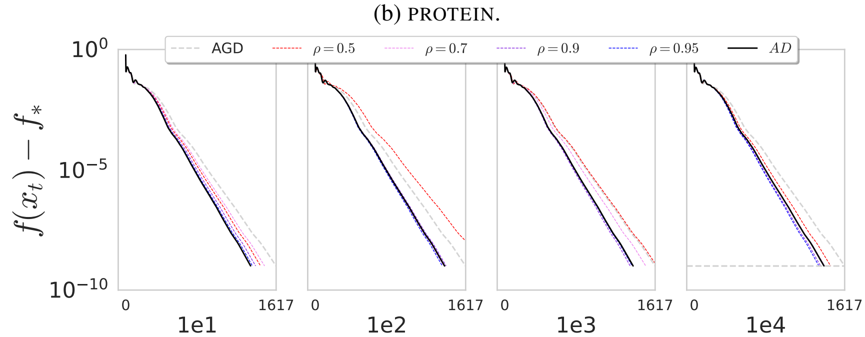
Figure 8: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for AGD, AGD with standard backtracking line search using $\rho \in \{0.5, 0.7, 0.9, 0.95\}$ and AGD with adaptive memoryless backtracking line search using $\rho = 0.9$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.



(a) PHISHING.

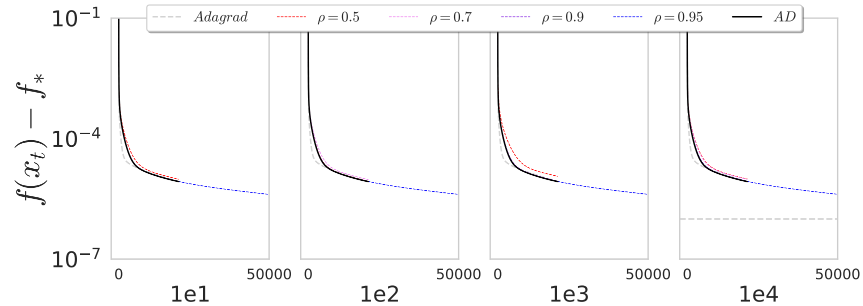


(b) PROTEIN.

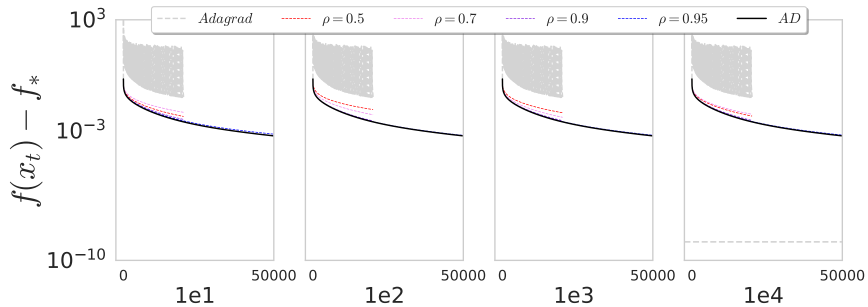


(c) WEB-1.

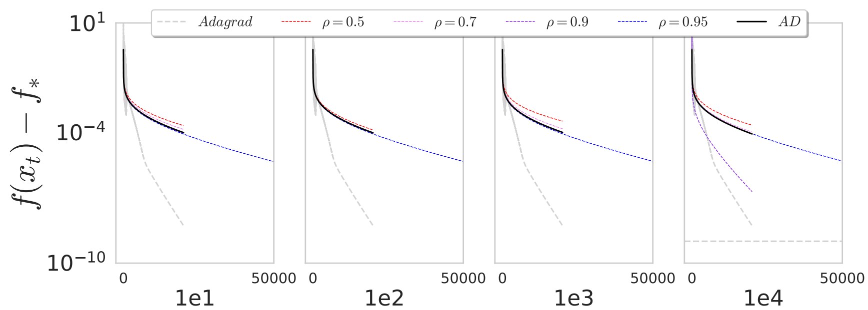
Figure 9: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for AGD, AGD with standard backtracking line search using $\rho \in \{0.5, 0.7, 0.9, 0.95\}$ and AGD with adaptive memoryless backtracking line search using $\rho = 0.9$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.



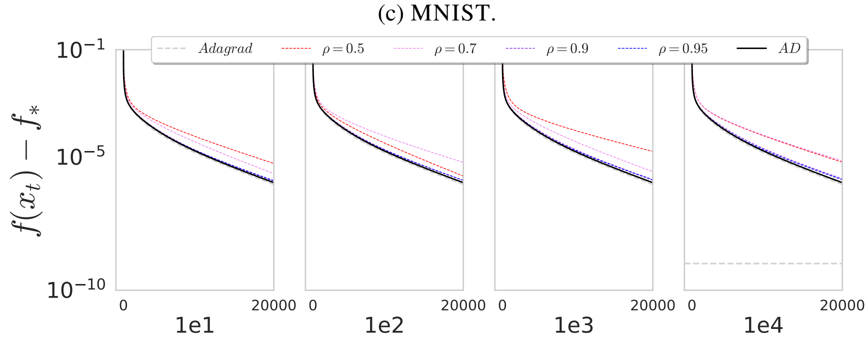
(a) ADULT.



(b) GISETTE_SCALE.

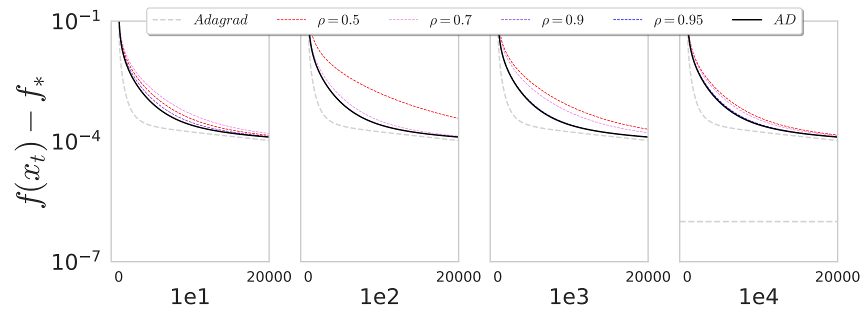


(c) MNIST.

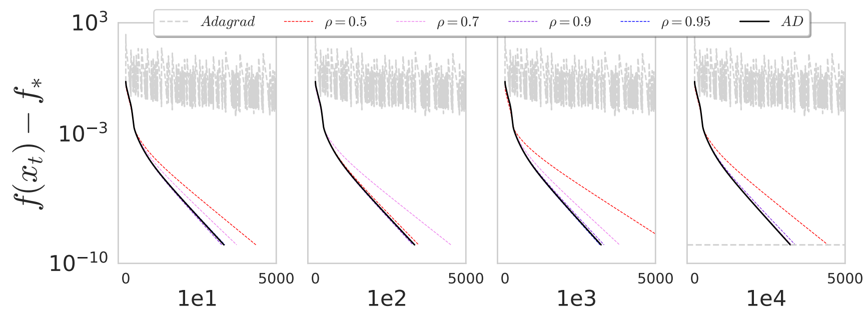


(d) MUSHROOMS.

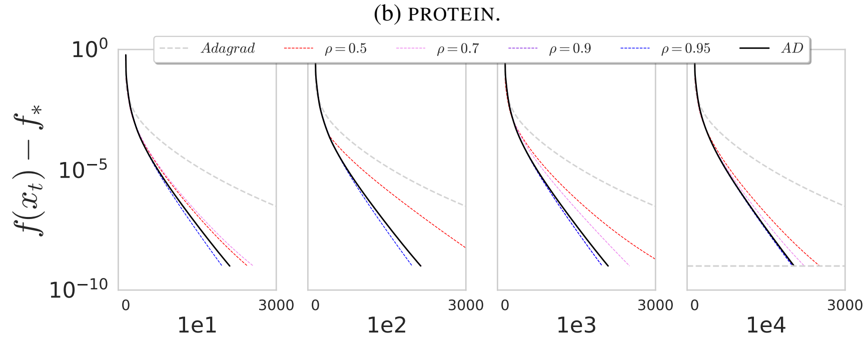
Figure 10: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for Adagrad, Adagrad with standard backtracking line search using $\rho \in \{0.5, 0.7, 0.9, 0.95\}$ and Adagrad with adaptive memoryless backtracking line search using $\rho = 0.9$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.



(a) PHISHING.



(b) PROTEIN.



(c) WEB-1.

Figure 11: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for Adagrad, Adagrad with standard backtracking line search using $\rho \in \{0.5, 0.7, 0.9, 0.95\}$ and Adagrad with adaptive memoryless backtracking line search using $\rho = 0.9$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.

D.2 Memoryless Backtracking Line Search

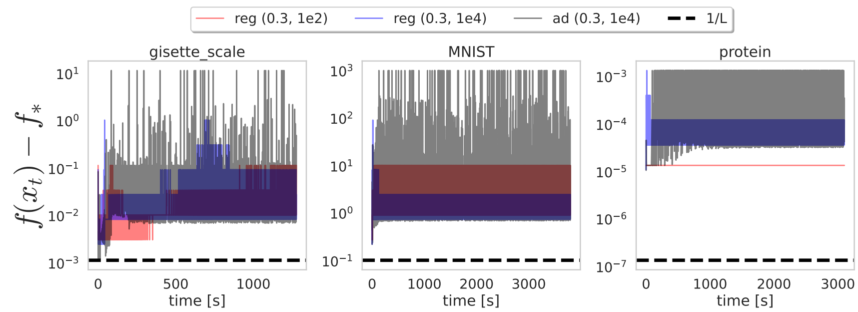


Figure 12: Step sizes for experiments shown in Fig. 1. Baseline: GD with constant $\alpha_k = 1/\bar{L}$; reg (ρ, β) and ad (ρ, β): GD with, respectively, regular and adaptive memoryless BLS parameterized by ρ and $\alpha_0 = \beta/\bar{L}$. The thick black dashed line denotes $1/\bar{L}$, where $\bar{L} = \lambda_{\max}(A^\top A)/4n$.

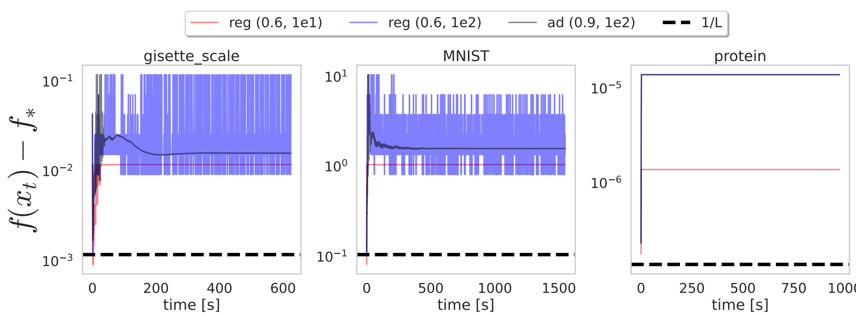
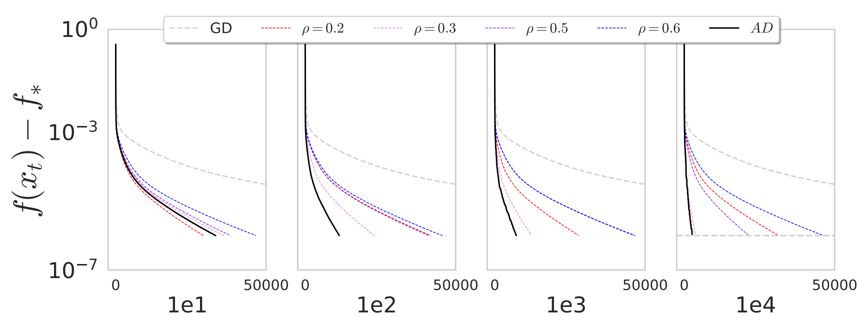
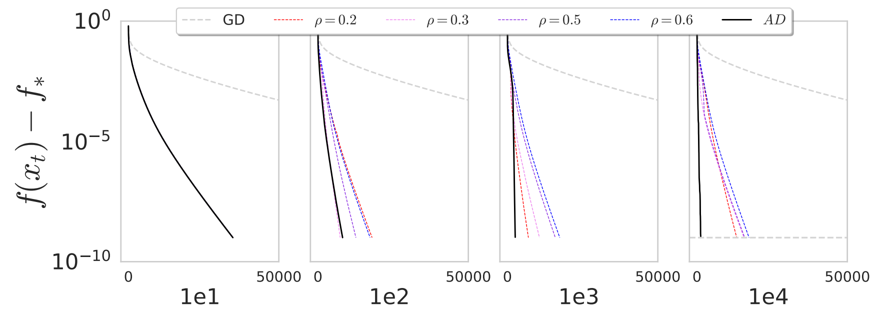


Figure 13: Step sizes for experiments shown in Fig. 2. Baseline: AGD with constant $\alpha_k = 1/\bar{L}$; reg (ρ, β) and ad (ρ, β): AGD with, respectively, regular and adaptive memoryless BLS parameterized by ρ and $\alpha_0 = \beta/\bar{L}$. The thick black dashed line denotes $1/\bar{L}$, where $\bar{L} = \lambda_{\max}(A^\top A)/4n$.

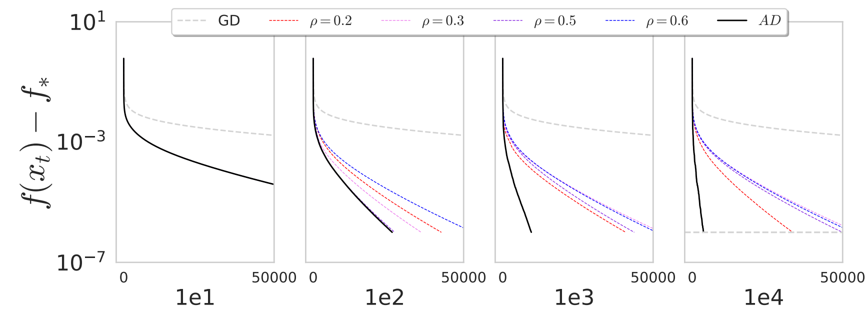


(a) ADULT.

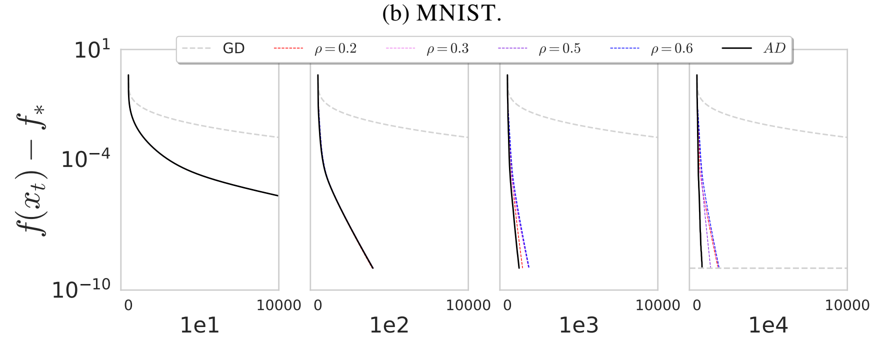
Figure 14: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for GD, GD with standard memoryless backtracking line search using $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ and GD with adaptive memoryless backtracking line search using $\rho = 0.3$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.



(a) GISETTE_SCALE.

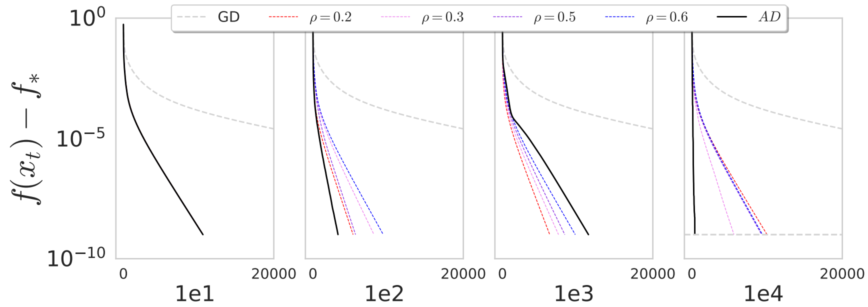


(b) MNIST.

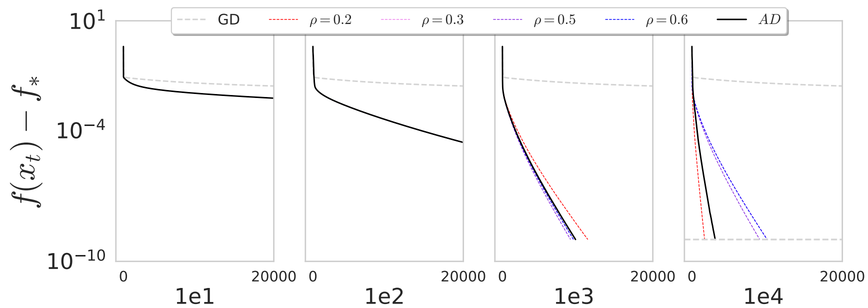


(c) MUSHROOMS.

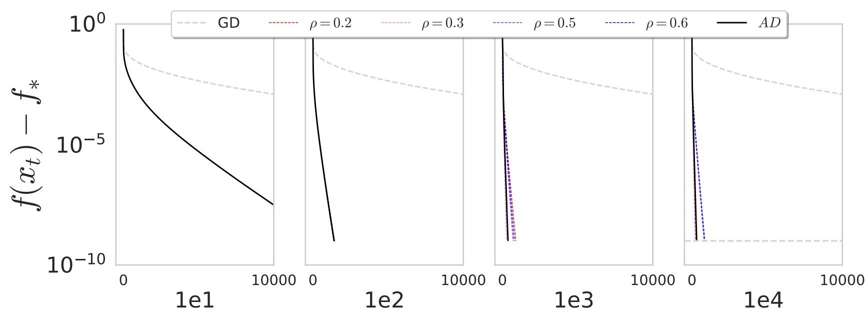
Figure 15: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for GD, GD with standard memoryless backtracking line search using $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ and GD with adaptive memoryless backtracking line search using $\rho = 0.3$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.



(a) PHISHING.

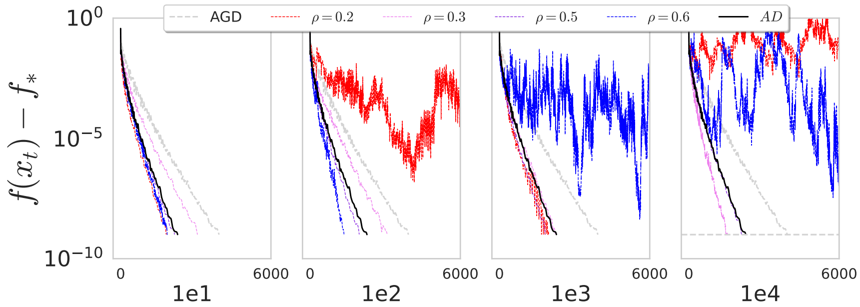


(b) PROTEIN.

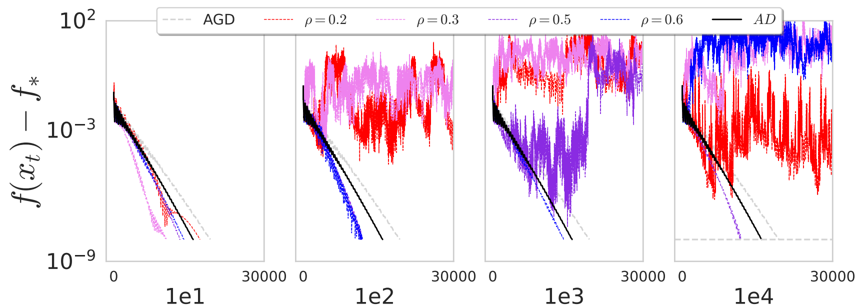


(c) WEB-1.

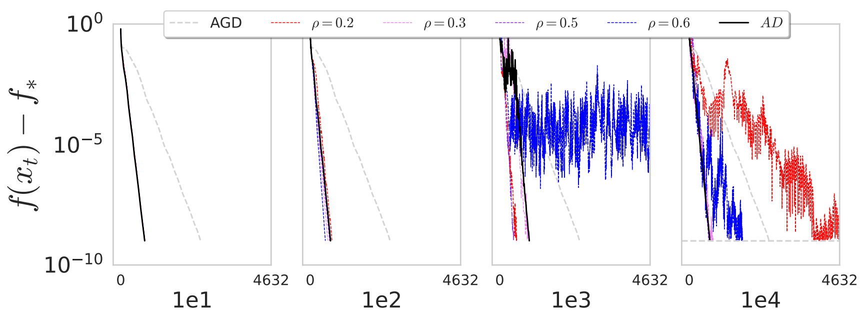
Figure 16: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for GD, GD with standard memoryless backtracking line search using $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ and GD with adaptive memoryless backtracking line search using $\rho = 0.3$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.



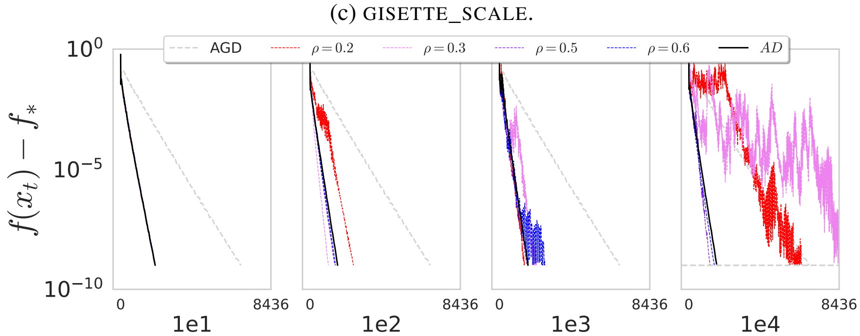
(a) ADULT.



(b) COVTYPE.

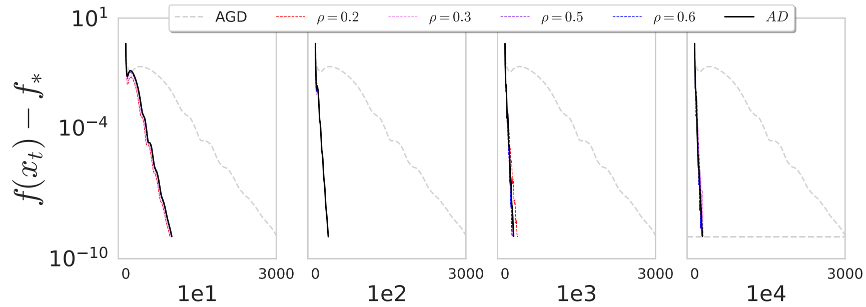


(c) GISETTE_SCALE.

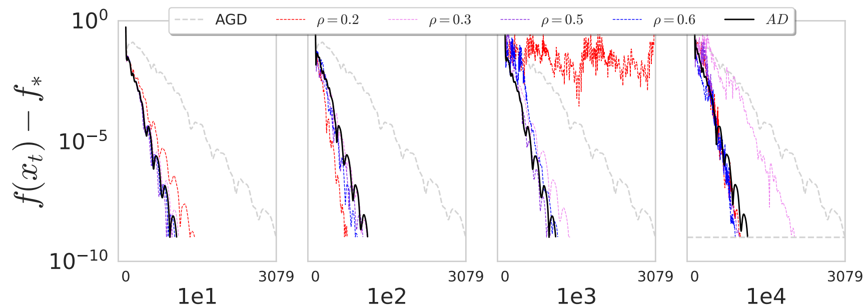


(d) MNIST.

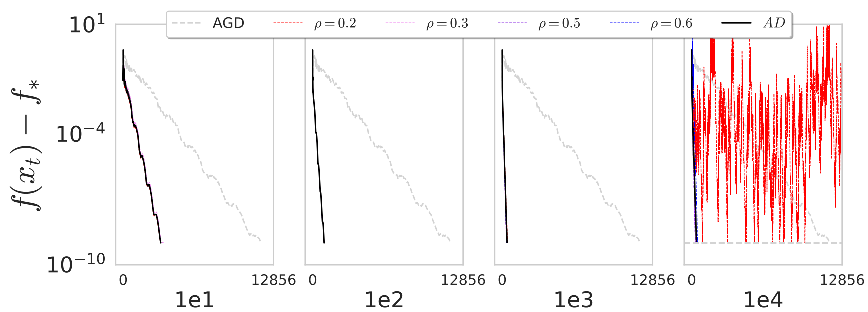
Figure 17: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for AGD, AGD with standard memoryless backtracking line search using $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ and AGD with adaptive memoryless backtracking line search using $\rho = 0.9$. The light gray horizontal dashed line shows the precision used to compute performance for all methods, 10^{-9}.



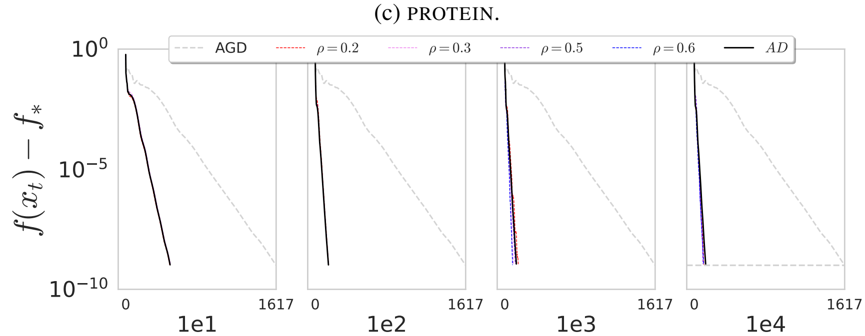
(a) MUSHROOMS.



(b) PHISHING.

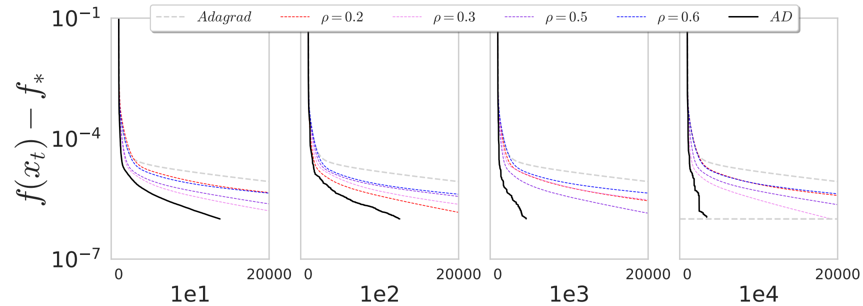


(c) PROTEIN.

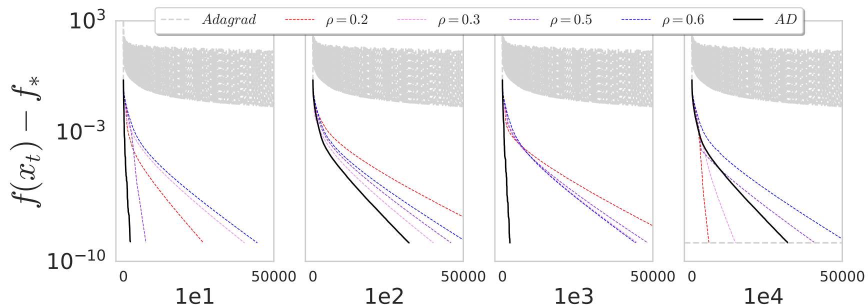


(d) WEB-1.

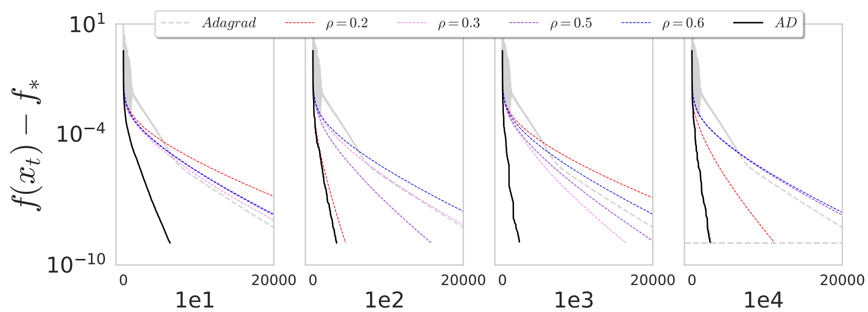
Figure 18: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for AGD, AGD with standard memoryless backtracking line search using $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ and AGD with adaptive memoryless backtracking line search using $\rho = 0.9$. The light gray horizontal dashed line shows the precision used to compute performance for all methods, 10^{-9}.



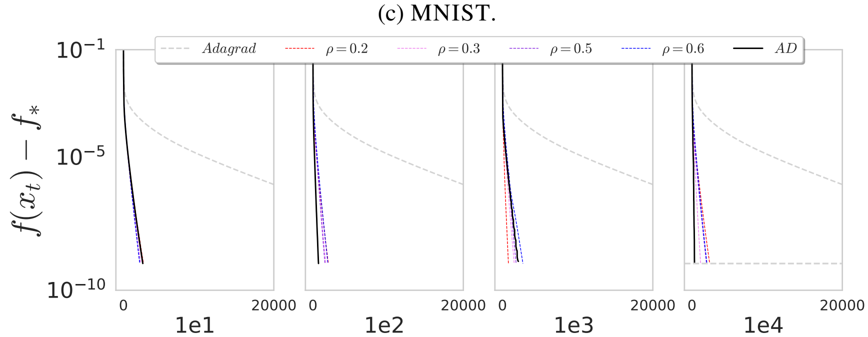
(a) ADULT.



(b) GISETTE_SCALE.

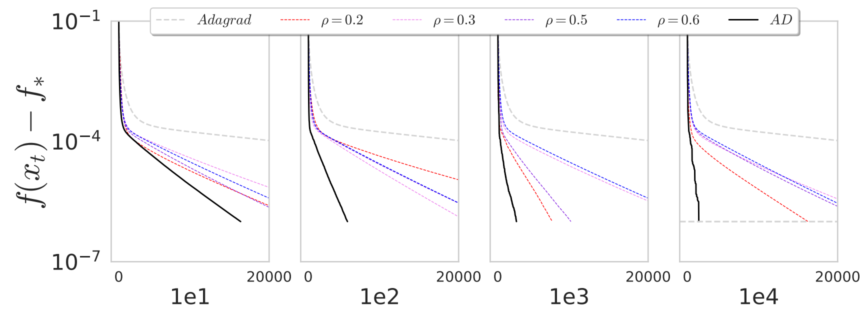


(c) MNIST.

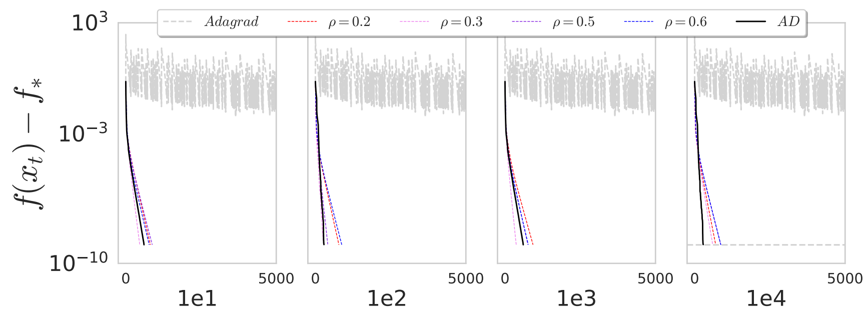


(d) MUSHROOMS.

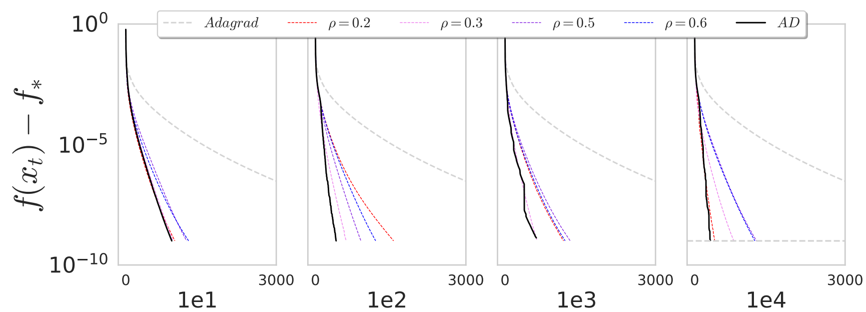
Figure 19: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for Adagrad, Adagrad with standard memoryless backtracking line search using $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ and Adagrad with adaptive memoryless backtracking line search using $\rho = 0.3$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.



(a) PHISHING.



(b) PROTEIN.



(c) WEB-1.

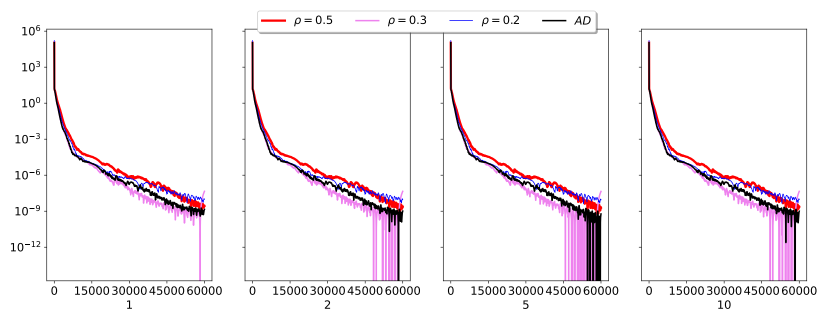
Figure 20: Logistic regression on four different datasets and four initial step sizes $\alpha_0 = \{10^1, 10^2, 10^3, 10^4\}/\bar{L}$: suboptimality gap for Adagrad, Adagrad with standard memoryless backtracking line search using $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ and Adagrad with adaptive memoryless backtracking line search using $\rho = 0.3$. The light gray horizontal dashed line shows the precision used to compute performance for each dataset.

E LINEAR INVERSE PROBLEMS

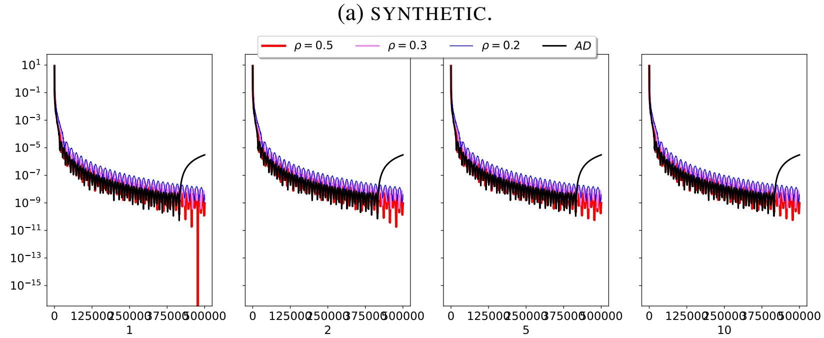
Details of the FISTA experiments are presented on Table 5.

Table 5: Details of FISTA experiments.

dataset	**datapoints**	**dimensions**	λ	L_0
digits	360	64	10^{-1}	$1, 10^1, 10^2, 10^3$
iris	100	4	10^{-2}	$10^{-1}, 1, 10^1, 10^2$
lfw_pairs	2200	5828	1	$10^{-3}, 10^{-2}, 10^{-1}, 1$
olivetti_faces	20	4096	10^{-2}	$1, 10^1, 10^2, 10^3$
Spear3	512	1024	10^{-1}	$10^{-3}, 10^{-2}, 10^{-1}, 1$
Spear10	512	1024	10^{-2}	$10^{-3}, 10^{-2}, 10^{-1}, 1$
Sparco3	1024	2048	10^{-2}	$10^{-3}, 10^{-2}, 10^{-1}, 1$
wine	130	13	10^{-2}	$1, 10^1, 10^2, 10^3$

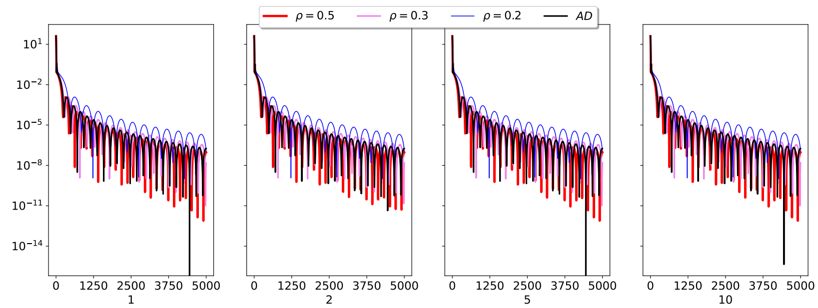


(a) SYNTHETIC.

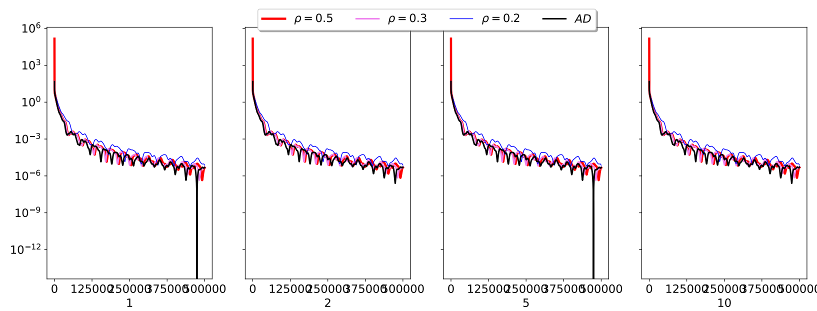


(b) OLIVETTI_FACES.

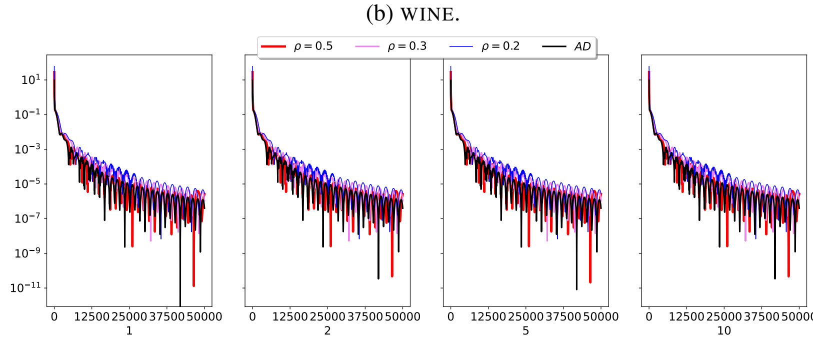
Figure 21



(a) IRIS.



(b) WINE.

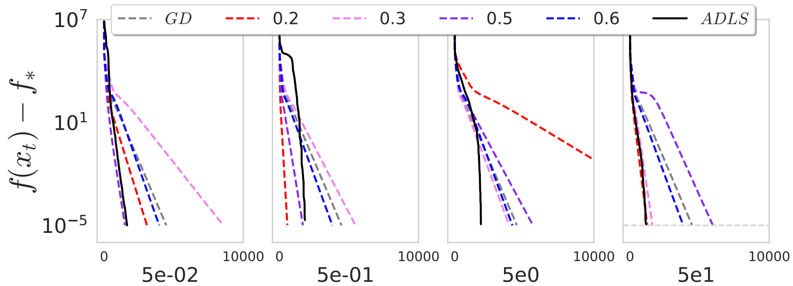


(c) DIGITS.

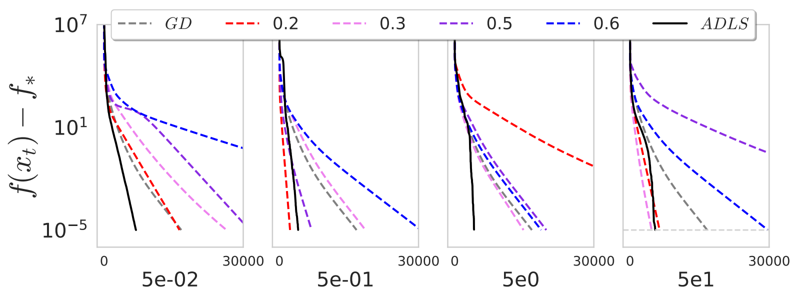
Figure 22

F Matrix Factorization Experiments

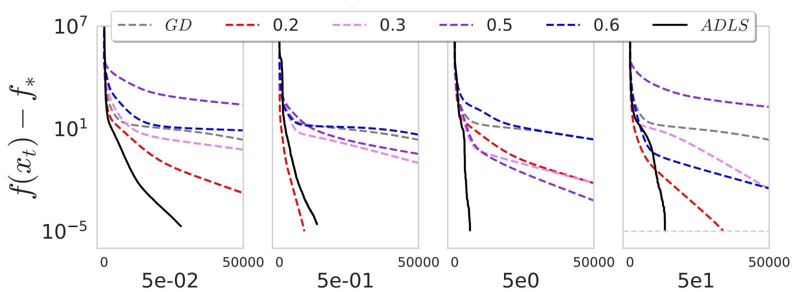
We sample A from the file "u.data", part of the MovieLens 100K dataset (grouplens.org/datasets/movielens/100k/). Moreover, we choose the precision representing a reduction to 10^{-12} in the suboptimality gap, which corresponds to a lower bound of 10^{-5} as the initial objective values typically hover around 10^7.



(a) Rank 10.



(b) Rank 20.



(c) Rank 30.

Figure 23: Matrix factorization and three values of rank: suboptimality gap for gradient descent, gradient descent with standard backtracking line search using $\rho \in \{0.2, 0.3, 0.5, 0.6\}$ and gradient descent with adaptive backtracking line search using $\rho = 0.3$. The light gray horizontal dashed line shows the precision used to compute performance for all methods, 10^{-5}.