OPT2019: 11th Annual Workshop on Optimization for Machine Learning

Escaping Saddle Points with Inequality Constraints via Noisy Sticky Projected Gradient Descent

Dmitrii Avdiukhin DAVDYUKH@IU.EDU *
Indiana University, Bloomington, IN 47405

Chi Jin CHIJ@PRINCETON.EDU
Princeton University, Princeton, NJ 08544

Grigory Yaroslavtsev GRIGORY@GRIGORY.US *
Indiana University, Bloomington, IN 47405

Abstract

We give the first analysis of convergence of a first-order method in smooth non-convex optimization under a bounded number of inequality constraints. Our algorithm uses $\tilde{O}(\frac{1}{\varepsilon^2} + \frac{k}{\sqrt{\varepsilon}})$ gradient evaluations for k linear inequality constraints and converges to a relaxed definition of an ϵ-second-order stationary point (for which the gradient and the Hessian are evaluated with respect to active constraints).

1. Introduction

Escaping from saddle points is one of the most intriguing questions in high-dimensional non-convex optimization and applications to machine learning (see [3–5, 9, 10], etc). In this paper we make progress towards the central question in breakthrough work of [3] who showed that a simple first-order method, Noisy Gradient Descent (Noisy GD), can escape from saddle points quickly. As argued in [3], Noisy GD works for both unconstrained and equality-constrained optimization. The main open questions left after this work is whether the same can be said for inequality-constrained problems. In general, the answer to this question is known to be negative due to NP-hardness of the copositivity problem [6, 9], which conditionally rules out the possibility of verifying whether a given point is a local minimum with any polynomial-time algorithm. However, this hardness result only applies when the number of constraints is equal to the dimension of the domain of the optimized function.

It has been observed empirically in [1] that for multi-dimensional balanced graph partitioning problem, which can be reduced to optimizing a simple quadratic function under linear inequality constraints, a certain variant of Noisy Projected Gradient Descent (Noisy PGD), which we refer to as Noisy Sticky Projected Gradient Descent (Noisy SPGD, Algorithm 1), converges quickly to a high-quality solution. In this paper we make a step towards quantifying the behavior of Noisy SPGD and give bounds on its convergence in terms of the number of inequality constraints in the general smooth non-convex optimization setting.

In the unconstrained case, [4] showed convergence of Noisy GD to an ε-second order stationary point in $\tilde{O}(\frac{1}{\varepsilon^2})$ iterations. For Riemannian manifolds, [10] showed convergence in $\tilde{O}(\frac{1}{\varepsilon^2})$ iterations. For inequality constraints due to NP-hardness of copositivity results are only known for second-order

* Supported by NSF grant 1657477

methods. [5] and [9] show convergence in $\tilde{O}(\frac{1}{\varepsilon^2})$ iterations. However, their algorithms require access to an oracle for finding an approximate solution for constrained quadratic problems, making it computationally expensive for large-scale problems.

1.1. Our Contributions

To get provable guaranties we relax the definition of a second order stationary point. In our paper we consider a notion of an *stationary point with respect to active constraints*: both gradient and Hessian are evaluated in the linear space defined by active constraints. We present Noisy SPGD (Algorithm 1), a variation of Noisy PGD. The key idea of the algorithm is that after encountering a new active constraint, the algorithm "sticks" to this constraint, meaning that this constraint will always be active. In this work, we focus on the case of linear inequality constraints. Our main result is:

Theorem 1 (Informal) *Let $f\colon \mathbb{R}^d \to \mathbb{R}$ be a Δ-bounded function with L-Lipschitz gradient and ρ-Lipschitz Hessian. If $\mathcal{M}$ is defined by k linear inequality constraints then Noisy SPGD w.h.p. finds an ε-second order stationary point $\mathbf{x} \in \mathcal{M}$ w.r.t. active constraints in $\tilde{O}_{L,\rho,\Delta}(\frac{1}{\varepsilon^2} + \frac{k}{\sqrt{\varepsilon}})$ iterations.*

2. Preliminaries

Table 1 introduces the notations used in the paper. We consider the following constrained minimization problem:

$$\text{minimize } f(\mathbf{x}) \quad \text{s.t. } \mathbf{x} \in \mathcal{M},$$

where $\mathcal{M}$ is defined by k linear inequality constraints:

$$\mathcal{M} = \{\, \mathbf{x} \in \mathbb{R}^d \mid \mathbf{B}\mathbf{x} \geq \mathbf{c} \,\},\ \mathbf{B} \in \mathbb{R}^{k\times d},\ \mathbf{c} \in \mathbb{R}^k$$

In the smooth non-convex optimization setting f is Δ-bounded: $|f(x)| \leq \Delta$, has L-Lipschitz gradient: $\|\nabla f(x) - \nabla f(y)\| < L\|x-y\|$, and ρ-Lipschitz Hessian: $\|\nabla^2 f(x) - \nabla^2 f(y)\| < \rho\|x-y\|$.

Since finding the global minimum is typically NP-hard, a common optimization objective is to find a local minimum. In order to guarantee convergence, this is typically further relaxed to finding an *ε-second-order stationary point* $\mathbf{x}$, such that $\|\nabla f(\mathbf{x})\| < \varepsilon$ and $\lambda_{\min}(\nabla^2 f(\mathbf{x})) > -\sqrt{\rho\varepsilon}$ ([4, 7]). In case of equality constraints, [3] formulate a similar condition using a Lagrangian. In case of inequality constraints, one has to use the more general KKT conditions instead of the Lagrangian. As a result, even checking that the point is a constrained local minimum is generally NP-hard ([6]). Therefore, we relax this definition by considering stationary points with respect to active constraints.

Let $\mathcal{A}(\mathbf{x})$ be the set of active constraints at $\mathbf{x}$: $\mathcal{A}(\mathbf{x}) = \bigcap_{i:\ \mathbf{B}_i\mathbf{x}=c_i} \{\, \mathbf{x} \mid \mathbf{B}_i\mathbf{x} = c_i \,\}$. Since all constraints are linear, active constraints form an affine subspace. Consider a matrix $Z(\mathbf{x}) \in \mathbb{R}^{d\times \dim(\mathcal{A}(\mathbf{x}))}$ defining a basis of $\mathcal{A}(\mathbf{x})$ ([8], Section 12.5). W.l.o.g. we choose $Z(\mathbf{x})$ to have orthonormal columns. Using $Z(\mathbf{x})$ we can introduce a natural parametrization of the constrained space: projected gradient in the constrained space can be represented as $Z^\top(\mathbf{x})\nabla f(\mathbf{x})$ and projected Hessian as $Z(\mathbf{x})^\top \nabla^2 f(\mathbf{x}) Z(\mathbf{x})$. The main advantage of such parametrization is that it allows us to leverage unconstrained optimization techniques.

Definition 2 *We call $\mathbf{x}$ an ε-*Second-Order Stationary Point with respect to Active Constraints *(ε-SOSPAC) if $\|Z^\top(\mathbf{x})\nabla f(\mathbf{x})\| < \varepsilon$ and $\lambda_{\min}(Z^\top(\mathbf{x})\nabla^2 f(\mathbf{x})Z(\mathbf{x})) > -\sqrt{\rho\varepsilon}$.*

Notation	Value	Explanation
Properties of the objective function		
d		Dimension
f		Objective function
L		Gradient Lipschitz constant
ρ		Hessian Lipschitz constant
Notation for constraints		
$\mathcal{M} = \{\, \mathbf{x} \mid \mathbf{B}\mathbf{x} \geq \mathbf{c} \,\}$		Feasible set
k		Number of constraints
$\mathcal{A}(\mathbf{x})$	$\bigcap_{i:\, \mathbf{B}_i\mathbf{x}=c_i} \{\, \mathbf{x} \mid \mathbf{B}_i\mathbf{x} = c_i \,\}$	Active constraints at a point $\mathbf{x}$
$\mathbf{x}^{(t)}$		Point at t-th iteration
Δ_t	$f(\mathbf{x}^{(t)}) - f(\mathbf{x}^{(t+1)})$	Objective change at t-th iteration
$\mathcal{P}_{\mathcal{S}}(\mathbf{x})$	$\operatorname{argmin}_{\mathbf{y}\in\mathcal{S}} \|\mathbf{y} - \mathbf{x}\|$	Projection of a point $\mathbf{x}$ on a set $\mathcal{S}$
$P_{\mathcal{A}(\mathbf{x})}\nabla f(\mathbf{x})$		Gradient at $\mathbf{x}$ projected on the linear subspace corresponding to $\mathcal{A}(\mathbf{x})$
Algorithm parameters		
Δ	$f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)$	Difference between the initial and the optimal objective
ε		Gradient threshold: if $\|\nabla f(\mathbf{x})\| > \varepsilon$, $\mathbf{x}$ is a saddle point
δ		Error probability
$\tilde{O}(\cdot)$		$O(\cdot)$, hiding polylogarithmic dependence on $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$ and d and dependence on L, ρ and Δ
c	10^{-3}	Small enough constant
χ	$3\max(\log(\frac{dL\Delta}{c\delta\varepsilon^2}), 4) = \tilde{O}(1)$	A parameter
η	$\frac{c}{L} = \tilde{O}(1)$	Step size
r	$\frac{\sqrt{c}\varepsilon}{\chi^2 L} = \tilde{O}(\varepsilon)$	Maximum L_2-norm of a noise
g_θ	$\frac{\sqrt{c}\varepsilon}{\chi^2} = \tilde{O}(\varepsilon)$	Gradient threshold: if $\|\nabla f(\mathbf{x})\| < g_\theta$, add noise and try to escape from a saddle point
f_θ	$\frac{c\varepsilon^{3/2}}{\chi^3\sqrt{\rho}} = \tilde{O}(\varepsilon^{3/2})$	Lower bound on how much the objective decreases in case of successful escape from a saddle point
t_θ	$\frac{\chi L}{c^2\sqrt{\rho\varepsilon}} = \tilde{O}(\frac{1}{\sqrt{\varepsilon}})$	Number of iterations used to escape from a saddle point

Table 1: Notations used in the paper

In other words, we require that $\mathbf{x}$ is a second-order stationary point when the objective function is restricted to $\mathcal{A}(\mathbf{x})$. Note that for specific active constraints there exist multiple possible choices of $Z(\mathbf{x})$. However, as long as $Z(\mathbf{x})$ represents an orthonormal basis, all such definitions are equivalent: all orthonormal bases can be obtained from each other by rotations, and both vector norm and eigenvalues are rotation-invariant.

3. Algorithm

Our Algorithm 1 is a projected gradient descent approach based on [4]. Each iteration is a gradient descent step followed by projection: $\mathcal{P}_{\mathcal{M}\cap\mathcal{A}(\mathbf{x}^{(t)})}(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in\mathcal{M}\cap\mathcal{A}(\mathbf{x}^{(t)})} \|\mathbf{y}-\mathbf{x}\|$.

First, consider the case when the set of active constraints doesn't change. We define the projected gradient at $\mathbf{x}$ as $P_{\mathcal{A}(\mathbf{x})}\nabla f(\mathbf{x})$, where $P_{\mathcal{A}(\mathbf{x})}$ is the linear projection operator on the linear subspace corresponding to the active constraints (after removing the shift). If the projected gradient is large, in Lemma 1 we show that the objective will sufficiently decrease after an ordinary projected gradient descent step. Otherwise, as shown in Algorithm 2, we add a noise by uniformly sampling a point from $\mathcal{A}(\mathbf{x})\cap B(\mathbf{x},r)$, where $B(\mathbf{x},r)$ is the Euclidean ball of radius r centered at $\mathbf{x}$. If function value does not sufficiently decrease after some number of iterations, by Lemma 3 we conclude that the algorithm found an ε-SOSPAC.

The key idea behind our algorithm is the way we handle constraints. Active constraints form an affine subspace, and if they don't change, restricting the objective function to the subspace makes unconstrained optimization techniques applicable. If the set of active constraints changes, we "stick" to the new constraints: all new active constraints are required to remain active throughout the rest of the algorithm.

Algorithm 1: Noisy Sticky Projected Gradient Descent

parameters: body $\mathcal{M}$, starting point $\mathbf{x}^{(0)}$; constant c; Δ: upper bound on $f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)$; error probability δ

$c = 10^{-3}$, $\chi = 3\max(\log(\frac{dL\Delta}{c\delta\varepsilon^2}), 4)$, $\eta = \frac{c}{L}$, $f_\theta = \frac{c\varepsilon^{3/2}}{\chi^3\sqrt{\rho}}$, $g_\theta = \frac{\sqrt{x}\varepsilon}{\chi^2}$;

for $t = 0, 1, 2, \ldots$ **do**

  if $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}\nabla f(\mathbf{x}^{(t)})\| < g_\theta$ **then**

    $\mathbf{x}^{(t+1)} = \texttt{EscapeSaddlePoint}(\mathbf{x}^{(t)})$

    if $\mathbf{x}^{(t+1)} = \perp$ **then return** $\mathbf{x}^{(t)}$;

  else

    $\mathbf{x}^{(t+1)} = \mathcal{P}_{\mathcal{M}\cap\mathcal{A}(\mathbf{x}^{(t)})}(\mathbf{x}^{(t)} - \eta\nabla f(\mathbf{x}^{(t)}))$; // SPGD step

  end

end

4. Main Result

Since Noisy SPGD enforces in future iterations all active constraints it has previously encountered, each new active constraint can be encountered at most once. By bounding how much these events increase the objective, we bound the overall increase in the number of iterations. Iterations in

Algorithm 2: `EscapeSaddlePoint`$(\mathbf{x})$

$c = 10^{-3}, \quad \chi = 3\max(\log(\frac{dL\Delta}{c\delta\varepsilon^2}), 4), \quad \eta = \frac{c}{L}, \quad r = \frac{\sqrt{c}\varepsilon}{\chi^2 L}, \quad f_\theta = \frac{c\varepsilon^{3/2}}{\chi^3\sqrt{\rho}}, \quad t_\theta = \frac{\chi L}{c^2\sqrt{\rho\varepsilon}};$

$\tilde{\mathbf{x}}^{(0)} = \texttt{Unif}(B(\mathbf{x}, r) \cap \mathcal{A}(\mathbf{x}));$

for $t = 0 \dots t_\theta$ **do**

  $\tilde{\mathbf{x}}^{(t+1)} = \mathcal{P}_{\mathcal{M}\cap\mathcal{A}(\mathbf{x})}(\tilde{\mathbf{x}}^{(t)} - \eta\nabla f(\mathbf{x}^{(t)}));$ `// PGD step`

  if $\mathcal{A}(\mathbf{x}) \neq \mathcal{A}(\tilde{\mathbf{x}}^{(t)})$ **then return** $\tilde{\mathbf{x}}^{(t+1)}$; `// New active constraint`

end

if $f(\mathbf{x}) - f(\tilde{\mathbf{x}}^{(t_\theta)}) > f_\theta$ **then return** $\tilde{\mathbf{x}}^{(t_\theta)}$ **else return** $\perp$;

Algorithm 1 and Algorithm 2 correspond to two different phases in the algorithm: a first-order phase, when the projected gradient norm is greater than g_θ, and an escaping phase, when the algorithm enforces escaping from a saddle point. Each phase behaves differently depending on whether new constraints become active. We analyze these cases separately in Lemmas 1-4 and then combine them in Theorem 5. The proofs are provided in Appendix A.

We introduce the following notation: $\Delta_t = f(\mathbf{x}^{(t)}) - f(\mathbf{x}^{(t+1)})$. In the following statements $\tilde{O}(\cdot)$ and $\tilde{\Omega}(\cdot)$ hide polylogarithmic dependence on $\frac{1}{\varepsilon}$, $\frac{1}{\delta}$ and d and dependence on L, ρ and Δ.

Lemma 1 *For each iteration t of Algorithm 1, if $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}\nabla f(\mathbf{x}^{(t)})\| \geq g_\theta$ and $\mathcal{A}(\mathbf{x}^{(t)}) = \mathcal{A}(\mathbf{x}^{(t+1)})$, then $\Delta_t = \tilde{\Omega}(\varepsilon^2)$.*

Lemma 2 *For any iteration t of Algorithm 1, if $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}\nabla f(\mathbf{x}^{(t)})\| \geq g_\theta$ and $\mathcal{A}(\mathbf{x}^{(t)}) \neq \mathcal{A}(\mathbf{x}^{(t+1)})$, then $\Delta_t \geq 0$.*

Lemma 3 *For any iteration t of Algorithm 1, if $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}\nabla f(\mathbf{x}^{(t)})\| < g_\theta$ and $\mathbf{x}^{(t)}$ is not an ε-SOSPAC, then w.h.p. $\mathbf{x}^{(t+1)} \neq \perp$. If $\mathbf{x}^{(t+1)} \neq \perp$ and $\mathcal{A}(\mathbf{x}^{(t)}) = \mathcal{A}(\mathbf{x}^{(t+1)})$, then $\Delta_t = \tilde{\Omega}(\varepsilon^{3/2})$.*

If no new active constraint appears, w.h.p. the objective decreases by $\tilde{\Omega}(\varepsilon^2)$ on average per iteration.

Lemma 4 *For any iteration of Algorithm 1, if $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}\nabla f(\mathbf{x}^{(t)})\| < g_\theta$, $\mathbf{x}^{(t+1)} \neq \perp$ and $\mathcal{A}(\mathbf{x}^{(t)}) \neq \mathcal{A}(\mathbf{x}^{(t+1)})$, then $|\Delta_t| = \tilde{O}(\varepsilon^2)$.*

If the set of active constraints changes, the objective can increase; however, it can increase by at most $\tilde{O}(k\varepsilon^2)$ over algorithm execution, which allows us to bound the number of additional iterations.

Theorem 5 *Algorithm 1 converges to an ε-SOSPAC w.h.p. and uses $\tilde{O}\left(\frac{1}{\varepsilon^2} + \frac{k}{\sqrt{\varepsilon}}\right)$ gradient evaluations.*

References

[1] Dmitrii Avdiukhin, Sergey Pupyrev, and Grigory Yaroslavtsev. Multi-dimensional balanced graph partitioning via projected gradient descent. *PVLDB*, 12(8):906–919, 2019. URL http://www.vldb.org/pvldb/vol12/p906-avdiukhin.pdf.

[2] D P Bertsekas. Nonlinear programming. *Journal of the Operational Research Society*, 48(3): 334–334, 1997. doi: 10.1057/palgrave.jors.2600425. URL https://doi.org/10.1057/palgrave.jors.2600425.

[3] Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points - online stochastic gradient for tensor decomposition. In *COLT*, 2015.

[4] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, and Michael I. Jordan. How to escape saddle points efficiently. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 1724–1732. JMLR.org, 2017. URL http://dl.acm.org/citation.cfm?id=3305381.3305559.

[5] Aryan Mokhtari, Asuman Ozdaglar, and Ali Jadbabaie. Escaping saddle points in constrained optimization. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3629–3639. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/7621-escaping-saddle-points-in-constrained-optimization.pdf.

[6] Katta G. Murty and Santosh N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39(2):117–129, Jun 1987. ISSN 1436-4646. doi: 10.1007/BF02592948. URL https://doi.org/10.1007/BF02592948.

[7] Yurii Nesterov and B.T. Polyak. Cubic regularization of newton method and its global performance. *Mathematical Programming*, 108(1):177–205, Aug 2006. ISSN 1436-4646. doi: 10.1007/s10107-006-0706-8. URL https://doi.org/10.1007/s10107-006-0706-8.

[8] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[9] Maher Nouiehed, Jason D Lee, and Meisam Razaviyayn. Convergence to second-order stationarity for constrained non-convex optimization. *arXiv preprint arXiv:1810.02024*, 2018.

[10] Yue Sun, Nicolas Flammarion, and Maryam Fazel. Escaping from saddle points on riemannian manifolds. *arXiv preprint arXiv:1906.07355*, 2019.

Appendix A. Missing Proofs From Section 4

One of the main tools in our analysis is the following lemma, which is the constrained-case analogue of the folklore descent lemma corollary.

Lemma 6 ([2], Section 2.3.2) *Let f have L-Lipschitz gradient and $0 < \eta < \frac{2}{L}$. Let $\mathbf{x}^{(t)} \in \mathcal{B}$, where $\mathcal{B}$ is a convex body, and $\mathbf{x}^{(t+1)} = \mathcal{P}_{\mathcal{B}}(\mathbf{x}^{(t)} - \eta \nabla f(\mathbf{x}^{(t)}))$, where $\mathcal{P}_{\mathcal{B}}(\mathbf{x})$ is a projection of $\mathbf{x}$ on $\mathcal{B}$. Then:*

$$f(\mathbf{x}^{(t)}) - f(\mathbf{x}^{(t+1)}) \geq \left(\frac{1}{\eta} - \frac{L}{2}\right) \|\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\|^2$$

Lemma 1 *For each iteration t of Algorithm 1, if $\|P_{\mathcal{A}(\mathbf{x}^{(t)})} \nabla f(\mathbf{x}^{(t)})\| \geq g_\theta$ and $\mathcal{A}(\mathbf{x}^{(t)}) = \mathcal{A}(\mathbf{x}^{(t+1)})$, then $\Delta_t = \tilde{\Omega}(\varepsilon^2)$.*

Proof Applying Lemma 6 to the body $\mathcal{M} \cap \mathcal{A}(\mathbf{x}^{(t)})$ and step size $\eta = \frac{c}{L} \leq \frac{1}{L}$, we have $\Delta_t \geq \frac{L}{2}\|\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\|^2$, and therefore:

$$\Delta_t \geq \frac{L}{2}\|\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\|^2 = \frac{L}{2}\|\eta P_{\mathcal{A}(\mathbf{x}^{(t)})} \nabla f(\mathbf{x}^{(t)})\|^2 \geq \frac{L}{2}\frac{c^2}{L^2} g_\theta^2 = \tilde{\Omega}(\varepsilon^2)$$

■

Lemma 2 *For any iteration t of Algorithm 1, if $\|P_{\mathcal{A}(\mathbf{x}^{(t)})} \nabla f(\mathbf{x}^{(t)})\| \geq g_\theta$ and $\mathcal{A}(\mathbf{x}^{(t)}) \neq \mathcal{A}(\mathbf{x}^{(t+1)})$, then $\Delta_t \geq 0$.*

Proof Applying Lemma 6 to the body $\mathcal{M} \cap \mathcal{A}(\mathbf{x}^{(t)})$, we have $\Delta_t \geq \frac{L}{2}\|\mathbf{x}^{(t)} - \mathbf{x}^{(t+1)}\|^2 \geq 0$. ■

Lemma 3 *For any iteration t of Algorithm 1, if $\|P_{\mathcal{A}(\mathbf{x}^{(t)})} \nabla f(\mathbf{x}^{(t)})\| < g_\theta$ and $\mathbf{x}^{(t)}$ is not an ε-SOSPAC, then w.h.p. $\mathbf{x}^{(t+1)} \neq \perp$. If $\mathbf{x}^{(t+1)} \neq \perp$ and $\mathcal{A}(\mathbf{x}^{(t)}) = \mathcal{A}(\mathbf{x}^{(t+1)})$, then $\Delta_t = \tilde{\Omega}(\varepsilon^{3/2})$.*

Proof Since $\|P_{\mathcal{A}(\mathbf{x}^{(t)})} \nabla f(\mathbf{x}^{(t)})\| < g_\theta$, Algorithm 2 is executed. If the set of active constraints changes, then $\mathbf{x}^{(t+1)} \neq \perp$.

Consider the case when the set of active constraints $\mathcal{A}(\mathbf{x})$ does not change. We will reduce the analysis to the unconstrained analysis in the projected space. Let $Z \in \mathbb{R}^{d \times \dim(\mathcal{A}(\mathbf{x}))}$ be an any orthonormal basis in the constrained space. We introduce the following parametrization of the constrained space $\mathcal{A}(\mathbf{x})$: $\mathbf{x} = \mu(\mathbf{y}) = \phi + Z\mathbf{y}$, where $\phi \perp \mathcal{A}(\mathbf{x})$ and $\mathbf{y} \in \mathbb{R}^{\dim(\mathcal{A}(\mathbf{x}))}$.

We can define function $g(\mathbf{y}) = f(\mu(\mathbf{y}))$. Then by the chain rule

$$\nabla g(\mathbf{y}) = Z^\top \nabla f(\mu(\mathbf{y})) \qquad\qquad \nabla^2 g(\mathbf{y}) = Z^\top \nabla^2 f(\mu(\mathbf{y})) Z.$$

By [8], Section 16.3, for orthonormal basis Z we have $P_{\mathcal{A}(\mathbf{x})} \nabla f(\mathbf{x}) = ZZ^\top \nabla f(\mathbf{x})$. We show that projected gradient descent step for f is equivalent to gradient descent step for g:

$$\mu(\mathbf{y} - \eta \nabla g(\mu(\mathbf{y}))) = \phi + Z(\mathbf{y} - \eta Z^\top \nabla f(\mu(\mathbf{y}))) = \mu(\mathbf{y}) - \eta ZZ^\top \nabla f(\mu(\mathbf{y})) = \mu(\mathbf{y}) - \eta P_{\mathcal{A}}(\mathbf{x})(\nabla f(\mathbf{x}))$$

Since Z is a matrix representing an orthonormal basis, the absolute value of the largest singular value of $Z^\top$ is at most 1, and therefore $\|\nabla g(\mathbf{y})\| \leq \|\nabla f(\mu(\mathbf{y}))\|$, which is less than ε for the initial point $\mathbf{x}$. Since $\mathbf{x}$ is not a ε-SOSPAC, $\lambda_{\min}(Z^\top \nabla^2 f(\mathbf{x}) Z) < -\sqrt{\rho\varepsilon}$ by definition. We reduced the problem to the unconstrained case and can directly apply result from [4], namely that after $O(\frac{1}{\sqrt{\varepsilon}})$ iterations the objective decreases by $\tilde{\Omega}(\varepsilon^{3/2})$ w.h.p. ■

Lemma 4 *For any iteration of Algorithm 1, if* $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}\nabla f(\mathbf{x}^{(t)})\| < g_\theta$, $\mathbf{x}^{(t+1)} \neq \perp$ *and* $\mathcal{A}(\mathbf{x}^{(t)}) \neq \mathcal{A}(\mathbf{x}^{(t+1)})$, *then* $|\Delta_t| = \tilde{O}(\varepsilon^2)$.

Proof We have to analyze $f(\mathbf{x}) - f(\tilde{\mathbf{x}}^{(t)})$ in Algorithm 2. By Lemma 6 applied to the body $\mathcal{M} \cap \mathcal{A}(\mathbf{x})$, for any iteration t we have $f(\tilde{\mathbf{x}}^{(t)}) - f(\tilde{\mathbf{x}}^{(t+1)}) \geq 0$. The remaining task is to estimate $f(\mathbf{x}) - f(\tilde{\mathbf{x}}^{(0)})$. By Taylor's theorem, there exists $\xi = \alpha\mathbf{x} + (1-\alpha)\tilde{\mathbf{x}}^{(0)}$ for $\alpha \in [0,1]$ such that:

$$\begin{aligned} |f(\tilde{\mathbf{x}}^{(0)}) - f(\mathbf{x})| &= |\nabla f^\top(\xi)(\tilde{\mathbf{x}}^{(0)} - \mathbf{x})| \\ &\leq (\|\nabla f(\mathbf{x})\| + \|\nabla f(\xi) - \nabla f(\mathbf{x})\|)r \\ &\leq (\varepsilon + Lr)r \\ &= \tilde{O}(\varepsilon^2), \end{aligned}$$

where we use that $r = \tilde{O}(\varepsilon)$ and $\|\nabla f(\mathbf{x})\| < \varepsilon$, ■

Theorem 5 *Algorithm 1 converges to an* ε*-SOSPAC w.h.p. and uses* $\tilde{O}\left(\frac{1}{\varepsilon^2} + \frac{k}{\sqrt{\varepsilon}}\right)$ *gradient evaluations.*

Proof To bound the number of graduate evaluations, it suffices to bound the number of iterations in Algorithm 1 and Algorithm 2. For t-th iteration of Algorithm 1 we have to consider the following cases:

Case 1. If $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}(\nabla f(\mathbf{x}^{(t)}))\| \geq g_\theta$ and $\mathcal{A}(\mathbf{x}^{(t)}) = \mathcal{A}(\mathbf{x}^{(t+1)})$, then by Lemma 1 we have $\Delta_t > \Omega(\varepsilon^2)$.

Case 2. If $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}(\nabla f(\mathbf{x}^{(t)}))\| \geq g_\theta$ and $\mathcal{A}(\mathbf{x}^{(t)}) \neq \mathcal{A}(\mathbf{x}^{(t+1)})$, then by Lemma 2 we have $\Delta_t \geq 0$. However, we don't give positive lower bound on the difference, which may results in an iterations without objective improvement.

Case 3. If $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}(\nabla f(\mathbf{x}^{(t)}))\| < g_\theta$ and $\mathbf{x}^{(t+1)} = \perp$, then by Lemma 3 $\mathbf{x}^{(t)}$ is an ε-SOSPAC w.h.p.

Case 4. If $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}(\nabla f(\mathbf{x}^{(t)}))\| < g_\theta$, $\mathbf{x}^{(t+1)} \neq \perp$ and $\mathcal{A}(\mathbf{x}^{(t)}) = \mathcal{A}(\mathbf{x}^{(t+1)})$, then we have $\Delta_t > f_\theta = \tilde{\Omega}(\varepsilon^{3/2})$. Since Algorithm 2 requires $O(t_\theta) = \tilde{O}(\frac{1}{\sqrt{\varepsilon}})$ iterations, the objective decreases by $\tilde{\Omega}(\varepsilon^2)$ on average per iteration.

Case 5. If $\|P_{\mathcal{A}(\mathbf{x}^{(t)})}(\nabla f(\mathbf{x}^{(t)}))\| < g_\theta$, $\mathbf{x}^{(t+1)} \neq \perp$ and $\mathcal{A}(\mathbf{x}^{(t)}) \neq \mathcal{A}(\mathbf{x}^{(t+1)})$, then by Lemma 4 after $O(t_\theta) = \tilde{O}(\frac{1}{\sqrt{\varepsilon}})$ iterations of Algorithm 2 the objective increases by at most $\tilde{O}(\varepsilon^2)$.

Note that Case 2 and Case 5 can only occur once per constraint, since each constraint can become active only once. Therefore, overall they can increase the objective by $\tilde{O}(k\varepsilon^2)$ and the number of iterations by $\tilde{O}(\frac{k}{\sqrt{\varepsilon}})$. Case 3 corresponds to termination of the algorithm. For Case 1 and Case 4 the objective decreases by $\tilde{\Omega}(\varepsilon^2)$ per iteration on average. Recall that $\Delta = f(\mathbf{x}^{(0)}) - f(\mathbf{x}^*)$. Then the total number of gradient evaluation is

$$\tilde{O}\left(\frac{\Delta + k\varepsilon^2}{\varepsilon^2} + \frac{k}{\sqrt{\varepsilon}}\right) = \tilde{O}\left(\frac{1}{\varepsilon^2} + \frac{k}{\sqrt{\varepsilon}}\right)$$

■