HONES: A Fast and Tuning-free Homotopy Method For Online Newton Step

Yuting Ye
UC Berkeley
yeyt@berkeley.edu

Cheng Ju
UC Berkeley
cju@berkeley.edu

Lihua Lei
UC Berkeley
lihua.lei@berkeley.edu

Abstract

In this article, we develop and analyze a homotopy continuation method, referred to as HONES , for solving the sequential generalized projections in Online Newton Step (Hazan et al., 2006b), as well as the generalized problem known as *sequential standard quadratic programming*. HONES is fast, tuning-free, error-free (up to machine error) and adaptive to the solution sparsity. This is confirmed by both careful theoretical analysis and extensive experiments on both synthetic and real data.

1 Introduction

Online convex optimization (OCO) is an appealing framework that unifies online and sequential optimization problems in various areas. In OCO, a player sequentially makes decisions by choosing a point in a convex set and a concave payoff function is revealed after each decision. The player aims to "maximize" her cumulative payoff, or formally minimize the *regret*, which measures the gap between the average payoff of her decision strategy and that of the best fixed-action strategy from hindsight. One of the high-profile motivation is the universal portfolio management problem (Cover, 1991), where an investor seeks an online strategy to allocate her wealth on a set of financial instruments without making any assumption on the market behaviors. The payoff can be quantified by *logarithmic wealth growth ratio*, formulated as $\sum_{t=1}^{T} \log(x_t^T \gamma_t)$ where $x_t(j)$ $(j = 1, \ldots, n)$ is the share of the j-th stock in the portfolio and $\gamma_t(j)$ is the ratio of the closing price of stock j on time t to that on time $t-1$. In view of the prohibition of short sales in most markets, the decision space is thus the $(n-1)$-dimensional simplex $\Delta_{n-1} = \{x \in \mathbb{R}^n : \sum_{i=1}^n x_i = 1, x_i \geq 0\}$. The regret of a given strategy that outputs $\{x_t\}_{t=1}^T$ is then

$$\sup_{x \in \Delta_{n-1}} \sum_{t=1}^{T} \log(x^T \gamma_t) - \sum_{t=1}^{T} \log(x_t^T \gamma_t). \tag{1}$$

A rich class of algorithms has been developed since Cover (1991) which proposed an algorithm with regret $O(n \log T)$ but with exponential computation cost per period. Kalai & Vempala (2002) gave a polynomial-time implementation of this algorithm, though the order of polynomials is still high. Helmbold et al. (1998) developed an algorithm that reduces the computation cost to $O(n)$ but incurs a sub-optimal regret $O(\sqrt{T \log n})$ in terms of horizon dependence. In 2003, the pioneering work by Zinkevich (2003) proposed the influential *Online Gradient Method* which achieves $O(\sqrt{T})$ regret for general OCO problems. The next milestone, among others, is achieved by Hazan et al. (2006b), which proposed *Online Newton Step* (ONS) that achieves $O(n \log T)$ regret under mild conditions, satisfied in universal portfolio management problems, with a practical computation cost per period. Hazan et al. (2006b) also shows that Online Gradient Method is able to achieve $O(\log T)$ regret but requires the loss functions to be strongly convex and, more stringently, the player knowing the strong-convexity modulus apriori. We refer the readers to Shalev-Shwartz (2011) for the history and to Elad Hazan's thesis (Hazan et al., 2006a) for detailed description of Online Newton Step.

Despite the promising theoretical guarantee of Online Newton Step, the computation efficiency remains a considerable concern for practitioners as the algorithm involves solving a sequence of *generalized projections*. Specifically, at time t one needs to solve

$$\min \frac{1}{2} x^T A^{(t)} x - (r^{(t)})^T x, \quad s.t. \quad x \in \Delta_{n-1} \tag{2}$$

where $A^{(t)}$ (resp. $r^{(t)}$) is a sequence of matrices (resp. vectors) such that

$$A^{(t+1)} = A^{(t)} + g^{(t)} (g^{(t)})^T, \tag{3}$$

for some time-varying vectors $g^{(t)}$. In the special case $A = I$, (2) can be solved quite efficiently in $O(n)$ time (Duchi et al., 2008) due to the explicit form of the solution.

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

Unfortunately, in Online Newton Step, $A^{(t)}$ is never a scaled identity matrix and such benefits disappear for general matrices. Hazan et al. (2006b) suggests using iterative algorithms such as interior-point method (Wright, 1997). However, it is known that interior-point method has $O(n^3)$ computation cost per iteration, which could be prohibitive for large problems or high-frequency online problems. Luo et al. (2016) utilizes the sketching approximation to improve the computational efficiency of the second-order online learning. Nonetheless, it targets at the constraint of bounded $|x^T\gamma_t|$, which does not address the simplex constraint in (2). For these reasons, the sub-problem (2) remains a bottleneck of Online Newton Step, which motivates our work.

Without the rank-one update structure (3), we should not expect significant improvement over interior-point method as (2) leads to multiple unrelated quadratic programming problems. Nevertheless, (3) is a "huge bonus" that connects the consecutive problems: In fact $A^{(t)}$ is perturbed in only one direction at each step and hence the optimal solutions in consecutive steps should be close. A widely used strategy to exploit the minor change is warm-start, i.e. initializing the iterate as the optimal solution in the last step. Spectral projected gradient (SPG) method (Birgin et al., 2000) is a typical algorithm falling into this category, which combines projected gradient method with smart line search. However, a warm-start is not always allowed. For example, the interior-point method (Wright, 1997) requires the initializer to be an interior point of the constraint set, but as shown in various settings and applications, including our experiments in section 3, the solution in each step often lies on the boundary of the simplex. Another potential algorithm is Exponentiated Gradient Descent (Kivinen & Warmuth, 1997) or Mirror Descent (Beck & Teboulle, 2003). However, it also requires the initializer to be an interior point in that any zero entry will stay zero. Furthermore, it is lack of an efficient stopping rule, which might not be essential for solving a single problem but is quite important for solving thousands of problems.

On the other hand, it has been proved that the minimizer of a standard quadratic programming problem tends to be sparse under fairly general structural assumptions (Chen et al., 2013; Chen & Peng, 2015). We also observed the sparsity in both synthetic and real datasets; see section 3 for details. However, none of the existing algorithms take the solution sparsity into account. [1]

In summary, to the best of our knowledge, existing methods are neither tailored for the sequential problem with structure (3) nor designed to adapt to the solution sparsity. To exploit the structure (3), we resort to the homotopy method, which was proposed decades ago and widely used in optimizing highly non-convex problems such as polynomial systems (Chow et al., 1979; Li, 1983). The basic idea is to construct a bivariate function $H(x, w)$ on $\mathbb{R}^n \times [0, 1]$ with $H(x, 0) = g(x)$ and $H(x, 1) = f(x)$. In order to optimize $f(x)$ one can start from the optimizer of $g(x)$ and move towards $f(x)$ by gradually increasing w. Given sufficient smoothness of H along with the non-singularity of the Hessian matrix of H w.r.t x, one can obtain a smooth trajectory, or a solution path , penetrating the optimizers of $H(x, w)$ for all $w \in [0, 1]$ with the optimizer of $f(x)$ being the ending point. Homotopy methods for quadratic programming problems have been studied and applied for decades (Frank & Wolfe, 1956; Bank et al., 1982; Ritter, 1981; Murty & Yu, 1988; Best, 1996; Efron et al., 2004). However, all these methods are designed for a specific problem. Recently, the homotopy methods have been applied to sequential problems. For example, Garrigues & Ghaoui (2009) proposed a homotopy method to solve the online LASSO regression problem where the objectives are updated in a similar fashion as (3).

For our problem (2), we define the homotopy function in a zigzag fashion which moves $(A^{(t-1)}, r^{(t-1)})$ to $(A^{(t)}, r^{(t-1)})$ and to $(A^{(t)}, r^{(t)})$ then; See Section 2.2.1 for the explicit construction. We show that the solution path can be calculated efficiently and *exactly* using the Karush-Kuhn-Tucker (KKT) conditions. By carefully analyzing the evolution of solutions, we propose an algorithm, referred to as *Homotopy Online NEwton Step (HONES)*, which is *fast*, *tuning-free*, *error-free* (up to machine error) and *adaptive to solution sparsity*. In fact, *HONES* can be extended to a general box constraint, the discussion of which is deferred to the appendix. We focus on the simplex case for two reasons. First, it is one of the most common constraint in practice. Besides ONS, (2) is also the generic problem in other areas such as Markowitz's portfolio management (Markowitz, 1952) and resource allocation (Ibaraki & Katoh, 1988). This is referred to as *standard quadratic optimization* dating back to 1950s (Bomze, 1998). Secondly, the theory, derivation and implementation are more clear in this special case, in which the derivation is already non-trivial and sophisticated.

We compute the number of atomic operations exactly, up to an additive constant, in Theorem 3. As almost all other homotopy continuation methods, the theoretical complexity of our algorithm is in general incomparable to other iterative algorithms like SPG and interior-point method because the former is proportional to the number of turning points on the trajectory (see Section 2.4) while the latter is proportional to the number of iterations to achieve an accurate solution (see Section 2.3). For this reason, we compare the algorithms by the running time on both synthetic and real datasets. To conclude, HONES has a superior performance to SPG and interior-point method and the gain of computational efficiency of HONES is more significant when the solutions are sparser.

The rest of the paper is organized as follows: HONES algorithm is detailed in Section 2, followed by the theory

[1] Here "sparsity" is a "phenomenon" instead of an "assumption", observed in both theory and practice, that the solution of the optimization problem tends to be sparse.

and complexity analysis. The practical implementation is deferred to the Supplementary Material. In Section 3, we apply HONES to NYSE and NASDAQ data using ONS for universal portfolio management. We also conduct experiments on synthetic data and for Markowitz's portfolio management on real data. Section 4 concludes the article.

2 Proposed Algorithm

A generic framework to solve problem (2) with matrix flow (3) is summarized in Algorithm 1 where ALGO1 and ALGO2 could be arbitrary sub-routines producing the solution of (2) in step 0 and the following steps.

Algorithm 1 Framework to solve (2)

Inputs: Initial matrix $A^{(0)}$, vectors $\{g^{(t)}, r^{(t)}, t = 1, 2, \ldots\}$.

Procedure:

1: Initialize: $x^{(0)} \leftarrow \text{ALGO1}(A^{(0)}, r^{(0)})$;
2: **for** $t = 1, 2, \cdots$ **do**
3: $\quad x^{(t)} \leftarrow \text{ALGO2}(A^{(t-1)}, g^{(t)}, r^{(t)}; x^{(t-1)})$;
4: **end for**

Output: $\{x^{(t)} : t = 0, 1, \ldots\}$.

In this article, we will focus on the online part, namely ALGO2. The complexity of ALGO1 will be increasingly less important as t increases. ALGO1 can be simply chosen as any state-of-the-art algorithm such as the interior-point method. We follow the initialization approach used in Online Newton Step (Hazan et al., 2006b), that is, set $A^{(0)} = \epsilon I$, where ϵ is a small positive number, and hence $x^{(0)} = \frac{1}{n}\mathbf{1}$.

2.1 KKT Condition Within A Step

We first consider the problem for a given t. The aim is to minimize $\frac{1}{2}x^T Ax - r^T x$ over Δ_{n-1}, where A and r are abbreviation of $A^{(t)}$ and $r^{(t)}$. By strong duality, it is equivalent to minimizing the Lagrangian form

$$L(x; \mu_0, \mu) = \frac{1}{2}x^T Ax - r^T x + \mu_0(1 - \mathbf{1}^T x) - \mu^T x \quad (4)$$

where μ_0 and μ are Lagrangian multipliers with constraint $\mu_i \geq 0$ for $i = 1, \ldots, n$. Denote S_x by the support of vector x. To be concise, the subscript x is suppressed in the following context. KKT condition together with Slater's condition implies that (x, μ_0, μ) is the solution of (4) if and only if

$$Ax - \mu_0\mathbf{1} - \mu - r = 0; \quad (5)$$

$$\mathbf{1}^T x = 1; \quad (6)$$

$$\mu_i x_i = 0, \mu_i \geq 0, x_i \geq 0, \forall i = 1, \ldots, n. \quad (7)$$

Here (7) is dubbed *complementary slackness* condition. The definition of $S = \text{supp}(x)$ entails that x_{S^c}[2] $= 0$, and (7) further implies that $\mu_S = 0$. Then the condition (5) can be reformulated as

$$\begin{pmatrix} A_{SS} & A_{SS^c} \\ A_{S^cS} & A_{S^cS^c} \end{pmatrix} \begin{pmatrix} x_S \\ 0 \end{pmatrix} = \mu_0 \begin{pmatrix} \mathbf{1}_S \\ \mathbf{1}_{S^c} \end{pmatrix} + \begin{pmatrix} 0 \\ \mu_{S^c} \end{pmatrix} + \begin{pmatrix} r_S \\ r_{S^c} \end{pmatrix}$$

By separating S and S^c, we have the following equations for x_S and μ_{S^c}.

$$x_S = \mu_0 A_{SS}^{-1}\mathbf{1}_S + A_{SS}^{-1} r_S; \quad (8)$$

$$\begin{aligned} \mu_{S^c} &= A_{S^cS}x_S - \mu_0\mathbf{1}_{S^c} - r_{S^c} \\ &= -\mu_0(\mathbf{1}_{S^c} - A_{S^cS}A_{SS}^{-1}\mathbf{1}_S) - (r_{S^c} - A_{S^cS}A_{SS}^{-1}r_S). \end{aligned} \quad (9)$$

The other parameter μ_0 can be solved from (6) and (8). In fact,

$$1 = \mathbf{1}^T x = \mathbf{1}_S^T x_S = \mu_0 \mathbf{1}_S^T A_{SS}^{-1}\mathbf{1}_S + \mathbf{1}_S^T A_{SS}^{-1} r_S$$

which implies that

$$\mu_0 = \frac{1 - \mathbf{1}_S^T A_{SS}^{-1} r_S}{\mathbf{1}_S^T A_{SS}^{-1}\mathbf{1}_S}. \quad (10)$$

In summary, the quadruple $(S, x_S, \mu_{S^c}, \mu_0)$ which solves (8)-(10) produces the unique solution of (4). Moreover, given the correct support S, we can uniquely solve the other three parameters. Thus, determining S is the key part in this problem.

2.2 HONES Algorithm

2.2.1 Construction of Homotopy Continuation

Based on the above argument, the problem is reduced to updating support S with A replaced with $A + gg^T$, and r replaced by $r + \ell$, where g, ℓ are shorthand notations of $g^{(t)}$ and $r^{(t)} - r^{(t-1)}$. Heuristically, S will not be significantly disturbed when g, ℓ are small perturbations. However, in real problems, there is usually no such constraints on g. Instead, we can consider a homotopy from (A, r) to $(A+gg^T, r+\ell)$. The most natural one is $(A+\lambda gg^T, r+\underset{\sim}{\lambda}\ell)$ with $\lambda, \underset{\sim}{\lambda} \in [0, 1]$. In other words, if we denote $x(\lambda, \underset{\sim}{\lambda})$ be the solution of (4) with (A, r) replaced by $(A + \lambda gg^T, r + \underset{\sim}{\lambda}\ell)$, then $x(0, 0)$ is the solution in the last step and $x(1, 1)$ is the solution after the update. The idea of homotopy continuation method is to calculate $x(\lambda, \underset{\sim}{\lambda})$ over a path linking $(0, 0)$ to $(1, 1)$. Theoretically, any path suffices and the goal is to find a path which leads to a simple computation. In this article

[2]The notation $A_{S_1S_2}$ is referred to the submatrix that consists of the S_1-indexed rows and S_2-indexed columns in A, and x_S is referred to the subvector that consists of S-indexed entries in x.

we will consider the Manhattan path from $(0,0)$ to $(1,1)$, namely the union of two segments: $\{(z,0) : z \in [0,1]\}$ and $\{(1,z) : z \in [0,1]\}$. In other words we first minimize

$$H^{(1)}(\lambda) \triangleq \frac{1}{2}x^T(A+\lambda gg^T)x - r^T x$$

for each $\lambda \in [0,1]$ and then minimize

$$H^{(2)}(\underset{\sim}{\lambda}) \triangleq \frac{1}{2}x^T(A+gg^T)x - (r+\underset{\sim}{\lambda}\ell)^T x$$

for each $\underset{\sim}{\lambda} \in [0,1]$.

Although the problem is augmented, the update is efficient since the support S is shown to be a piecewise constant set on the path and explicit formulas, namely (8) - (10), can be used to compute (x_S, μ_{S^c}, μ_0) directly when S is fixed. Specifically, given the solution $x(\lambda, \underset{\sim}{\lambda})$ and its associated support $S = S_{x(\lambda,\underset{\sim}{\lambda})}$, the triple (x_S, μ_{S^c}, μ_0) is a simple function of $(\lambda, \underset{\sim}{\lambda})$ as shown in the following theorem. And S remains unchanged as λ in (11) or $\underset{\sim}{\lambda}$ in (12) increases, until one entry of (x_S, μ_{S^c}, μ_0) touches zero.

Theorem 1

1. *For a given $\underset{\sim}{\lambda}$, there exists vectors $u_1, u_2 \in \mathbb{R}^{n+1}$ and scalars $D_1, D_2 \in \mathbb{R}$, which only depend on S, such that*
$$\begin{pmatrix} x_S(\lambda) \\ -\mu_{S^c}(\lambda) \\ \mu_0(\lambda) \end{pmatrix} = \frac{u_1 - u_2\lambda}{D_1 - D_2\lambda}. \tag{11}$$

2. *For given λ, there exists vectors $\underset{\sim}{u}_1, \underset{\sim}{u}_2 \in \mathbb{R}^{n+1}$, which only depend on S, such that*
$$\begin{pmatrix} x_S(\underset{\sim}{\lambda}) \\ -\mu_{S^c}(\underset{\sim}{\lambda}) \\ \mu_0(\underset{\sim}{\lambda}) \end{pmatrix} = \underset{\sim}{u}_1 - \underset{\sim}{u}_2\underset{\sim}{\lambda}. \tag{12}$$

Proof

1. The proof is quite involved and we relegate it into Theorem B-4 in Appendix B. The theorem also gives the exact formula of u_1, u_2, D_1, D_2.

2. By (10), we have
$$\mu_0(\underset{\sim}{\lambda}) = \frac{1 - \mathbf{1}_S^T A_{SS}^{-1}(r_S + \underset{\sim}{\lambda}\ell_S)}{\mathbf{1}_S^T A_{SS}^{-1}\mathbf{1}_S} = \mu_0(0) - \frac{\mathbf{1}_S^T A_{SS}^{-1}\ell_S}{\mathbf{1}_S^T A_{SS}^{-1}\mathbf{1}_S}\underset{\sim}{\lambda}.$$
Then it follows from (8) that
$$\begin{aligned} x_S(\underset{\sim}{\lambda}) &= \mu_0(\underset{\sim}{\lambda})A_{SS}^{-1}\mathbf{1}_S + A_{SS}^{-1}(r_S + \underset{\sim}{\lambda}\ell_S) \\ &= x_S(0) - \left(\frac{\mathbf{1}_S^T A_{SS}^{-1}\ell_S}{\mathbf{1}_S^T A_{SS}^{-1}\mathbf{1}_S}A_{SS}^{-1}\mathbf{1}_S - A_{SS}^{-1}\ell_S\right)\underset{\sim}{\lambda}. \end{aligned}$$
Similarly, by (9), we obtain that
$$-\mu_{S^c}(\underset{\sim}{\lambda}) = -\mu_{S^c}(0) - \left(\frac{\mathbf{1}_S^T A_{SS}^{-1}\ell_S}{\mathbf{1}_S^T A_{SS}^{-1}\mathbf{1}_S}(\mathbf{1}_{S^c} - A_{S^cS}A_{SS}^{-1}\mathbf{1}_S) - (\ell_{S^c} - A_{S^cS}A_{SS}^{-1}\ell_S)\right)\underset{\sim}{\lambda}.$$

■

2.2.2 Update of Support

Once $S = S(\underset{\sim}{\lambda})$ is obtained for all $\underset{\sim}{\lambda}$, the solution path can be efficiently solved by Theorem 1. Heuristically, S is piecewise constant and the task is reduced to find the next $\underset{\sim}{\lambda}$ that $S(\underset{\sim}{\lambda})$ changes. We consider the update of S in optimizing $H^{(1)}(\lambda)$. The update of S in optimizing $H^{(2)}(\underset{\sim}{\lambda})$ can be obtained in the same way.

For a given $\lambda_0 \in [0,1]$, if $x_S(\lambda_0) > 0$ and $\mu_{S^c}(\lambda_0) > 0$, then (11) implies that there exists $\eta > 0$, such that for any $\lambda \in (\lambda_0 - \eta, \lambda_0 + \eta)$, both $x_S(\lambda)$ and $\mu_{S^c}(\lambda)$ remains positive by setting $S(\lambda) = S(\lambda_0)$. Since (8)-(10) are sufficient and necessary, we conclude that $S(\lambda) = S(\lambda_0)$. This argument remains valid until an entry of either x_S or μ_{S^c} hits zero. Denote j by the index of this entry. In the former case, j leaves S and S is updated to $S \setminus \{j\}$. In the latter case, j enters into S and S is updated to $S \cup \{j\}$. The other three parameters are then updated correspondingly by Theorem 1. Theorem 2 formalizes the above claim. The proof is omitted since it is a direct consequence of sufficiency and necessity of KKT conditions (8)-(10).

Theorem 2 *For any given $\lambda_0 \geq 0$, let λ^{new} be the next smallest λ such that one entry of either x_S or μ_{S^c} hits 0, i.e.*
$$\lambda^{\text{new}} = \min\nolimits_+ \left\{\frac{u_{1i}}{u_{2i}} : i = 1, 2, \ldots, n\right\},$$
where u_1 and u_2 are defined in (11) and $\min_+$ evaluates the minimum positive number in the set and defined to be ∞ if all elements are non-positive. Then $S(\lambda) \equiv S(\lambda_0)$ for $\lambda \in [\lambda_0, \lambda^{\text{new}})$. Further, let
$$I_1 = \{i \in S : u_{1i} = u_{2i}\lambda^{\text{new}}\},$$
$$I_2 = \{i \in S^c : u_{1i} = u_{2i}\lambda^{\text{new}}\},$$
then $S(\lambda^{\text{new}})$ is updated by
$$S(\lambda^{\text{new}}) = (S(\lambda_0) \setminus I_1) \cup I_2.$$

Remark 1 *According to our experience, $I_1 \cup I_2$ at most contains one element. In other words, S is updated by one element at each iteration.*

In summary, the algorithm starts from $\lambda = 0$ and searches for the next smallest λ such that one entry of x_S or

μ_{S^c} hits zero, then updates λ as well as the quadruple $(S, x_S, \mu_{S^c}, \mu_0)$. The procedure is repeated until λ crosses 1. In other words, there exist a sequence $0 = \lambda_0 < \lambda_1 < \ldots < \lambda_k = 1$, which we call *turning points*, such that $x(\lambda)$ has the same support between any two consecutive turning points and the value can be calculated by Theorem 1. A counterpart of Theorem 2 can be established for $\underset{\sim}{\lambda}$. The whole task reduces to finding all turning points and we call this procedure HONES algorithm. The complexity of HONES algorithm is determined by both the number of turning points and the complexity of the update between two consecutive turning points. For compact notation, we define v as a $n \times 1$ vector with

$$v_S = x_S, \quad v_{S^c} = -\mu_{S^c}.$$

To be more clear, we state the main steps in Algorithm 2 for optimizing $H^{(1)}(\lambda)$ holding $\underset{\sim}{\lambda} = 0$. As a convention, the minimum of an empty set is set to be infinity (line 3). The algorithm for optimizing $H^{(2)}(\underset{\sim}{\lambda})$ holding $\lambda = 1$ can be written in the same way as Algorithm 2 by changing λ into $\underset{\sim}{\lambda}$.

Algorithm 2 Main steps of HONES algorithm in optimizing $H^{(1)}(\lambda)$

Inputs: parameters A, y, r, g; initial optimum x (corresponding to A)

Procedure:

1: Initialize $\lambda \leftarrow 0, S \leftarrow \mathrm{supp}(x)$;
2: **while** $\lambda < 1$ **do**
3: $\quad \lambda = \min\{\lambda_1 > \lambda : v_i(\lambda_1) = 0 \text{ for some } i\}$;
4: $\quad I_1 \leftarrow \{i \in S : v_i(\lambda) = 0\}$;
5: $\quad I_2 \leftarrow \{i \in S^c : v_i(\lambda) = 0\}$;
6: $\quad$ **if** $\lambda \leq 1$ **then**
7: $\quad\quad S \leftarrow (S \setminus I_1) \cup I_2$;
8: $\quad$ **else**
9: $\quad\quad \lambda \leftarrow 1$;
10: $\quad$ **end if**
11: $\quad (x_S, \mu_{S^c}, \mu_0) \leftarrow (x_S(\lambda), \mu_{S^c}(\lambda), \mu_0(\lambda))$ via (8)-(10).
12: **end while**

Output: $(S, x_S, \mu_{S^c}, \mu_0)$.

2.3 Implementation and Complexity Analysis

Algorithm 2 presents the main idea without the implementation details. Although we can implement Algorithm 2 by directly computing quantities, e.g. u_1, u_2, in every step to find the next turning point as in line 3 and also directly computing the iterates via (8)-(10) as in line 11, it is fairly inefficient since many quantities appear in several computation steps and we can store them to save the computation. A careful derivation in Appendices B and C shows that the computation complexity is indeed low. For example, although u_1 and u_2 involves A_{SS}^{-1}, there is *no need* to calculate the matrix inverse directly. Theorem 3 summarizes the complexity for optimizing $H^{(1)}(\lambda), H^{(2)}(\underset{\sim}{\lambda})$ separately. As a convention, we assume the scalar-scalar multiplication takes a unit time and ignore the addition for simplicity when computing the complexity. Since the real implementation is involved, we state it as well as the proof of theorem 3 in Appendices B and C for two cases separately.

Theorem 3 *In step t, denote by k_A, k_r the number of turning points in optimizing $H^{(1)}(\lambda)$ and $H^{(2)}(\underset{\sim}{\lambda})$. Further let s be the maximum support size over the path of $(\lambda, \underset{\sim}{\lambda})$ and s_* by the size of union of all supports from step 1 to step t. Let C_{jt} be the computation cost of HONES algorithm in optimizing $H^{(j)}$, then*

1. $C_{1t} = ns_* + ns(3k_A + 1) + n(12k_A + 2) + O(k_A)$;
2. $C_{2t} = ns(2k_r + 1) + n(6k_r + 1) + O(k_r)$.

It is clear that the algorithm adapts to the sparsity when optimizing both $H^{(1)}(\lambda)$ and $H^{(2)}(\underset{\sim}{\lambda})$. For each step, the complexity is $\mathcal{O}\left(ns(k_A + k_r)\right)$, upper bounded by $\mathcal{O}(n^2(k_A + k_r))$ for the dense case, which is the same as other algorithms due to the inevitable multiplication of A by x. In some special regimes, the solution is guaranteed to be sparse with high probability, e.g., the data matrix is randomly generated form a certain distribution such as uniform and exponential distributions Chen et al. (2013); Chen & Peng (2015). We also observe this in our experiments; See Section 3 for details.

2.4 Number of Turning Points

Let S_t be the support of the optimum and k_t be the number of total turning points, then we can derive a generic bound that

$$k_t \geq |S_t \setminus S_{t-1}| + |S_{t-1} \setminus S_t| \tag{13}$$

provided that only one element is added to or removed from the support at each update; see Remark 1. This is because it requires at least $|S_{t-1} \setminus S_t|$ steps to pop out the elements in $S_{t-1} \setminus S_t$ and $|S_t \setminus S_{t-1}|$ steps to push in the elements in $S_t \setminus S_{t-1}$ to translate S_{t-1} into S_t.

On the other hand, suppose that no other coordinates than those in $S_t \cup S_{t-1}$ enter into the support in the path, then

$$k_t = |S_t \setminus S_{t-1}| + |S_{t-1} \setminus S_t|. \tag{14}$$

Heuristically, the equation (14) should hold since if a coordinate, not in $S_t \cup S_{t-1}$, entered into the support on the path, it must be popped out before the end, which, however, should be rare to happen. For both synthetic data and real data in section 3, we observe that there are at least 95% of steps with k_t satisfying (14) and over 99% of steps with $k_t \leq |S_t \setminus S_{t-1}| + |S_{t-1} \setminus S_t| + 6$, i.e. with at most 3

outside coordinates entered into the path. Thus, (14) is a highly reliable result for k_t.

As a direct consequence of (14), HONES algorithm is efficient when the support changes slowly in which case k_t is small. In addition, if the solution is sparse, then a rough bound suggests that $k_t \leq |S_t| + |S_{t-1}|$ is small. These phenomena are observed in various situations (see section 3) and (14) explains the good performance of HONES algorithm.

In general, the worst-case bound for the number of turning points can be exponential as Mairal & Yu (2012); Gärtner et al. (2009) pointed out for Lasso and SVM respectively. But the number of turning points is usually not large in practice. The same issue appears in Simplex method for linear programming. Although it is known that the worst case complexity is 2^n, it usually converges in $\mathcal{O}(n)$ operations;See Bertsimas & Tsitsiklis (1997).

3 Experiments

In this section, we compare the performance of HONES with SPG and the interior-point method on both real and synthetic data. We implement HONES in MATLAB [3] and implement SPG [4] and interior-point method[5] by using existing code. To make a fair comparison, SPG uses the solution to step t as the warm start for the solution to step $t+1$. To evaluate the performance, we display the cumulative running time as a measure of efficiency. All experiments are conducted on a machine with 3 GHz Intel Core i7 processor, OS X Yosemite system and Matlab 2015a.

3.1 Universal Portfolio Management

(Hazan & Arora, 2006) proposed a version of Online Newton Step (Figure 3.7) that is equivalent to (2) with

$$A^{(0)} = I, \quad g^{(t)} = \frac{\gamma_t}{x_t^T \gamma_t}, \quad r^t = \frac{1}{4} \sum_{\tau=1}^{t} \frac{\gamma_t}{x_t^T \gamma_t}.$$

We apply our algorithm on two datasets from NYSE and NASDAQ[6], with daily stock price data from Jan. 3, 2005 to May. 13, 2016. The NYSE dataset contains 1544 stocks and NASDAQ dataset contains 1101 stocks. This differs from classical studies where at most hundreds of stocks, such as S&P500, are incorporated. Still, we should emphasize that for some financial institutions like hedge funds, the number of base assets is huge and the computation efficiency becomes important when the trading frequency is high. Here we consider a large number of stocks to show the potential of HONES algorithm in optimizing a large basket of assets.

[3]Code available at https://github.com/Elric2718/HONES.
[4]https://www.cs.ubc.ca/ schmidtm/Software/minConf.html
[5]QUADPROG function in MATLAB
[6]Data available at https://github.com/Elric2718/HONES.

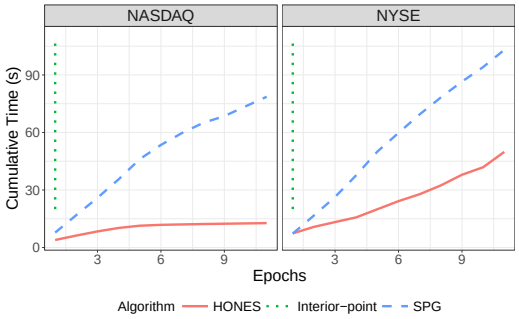

Figure 1: Cumulative running time of HONES , SPG and interior-point method on NYSE and NASDAQ dataset for universal portfolio management. Each epoch has 252 measurements.

The cumulative running time, measured in seconds, is reported in Figure 1. The interior-point method is quite inefficient as the running time for 10 steps (0.04 epochs) exceeds the total running time of HONES and SPG. Thus the proposal by Hazan et al. (2006b) is not desirable. In addition, HONES is much more efficient than both SPG, especially in the more volatile case (NASDAQ). In fact, HONES achieves a $2\times$ speedup on NYSE data and a $6\times$ speedup on NASDAQ data! By excluding the first 4 epochs, HONES even achieves a $12.5\times$ speedup on NASDAQ data.

To explain the different behavior on two datasets, we report the average solution sparsity in Table 1. Recall that HONES is accurate in every step and we confirm this by checking the KKT condition for each solution. Surprisingly, the solutions are generally sparse for both datasets, as suggested by existing theory (Chen et al., 2013; Chen & Peng, 2015). It is not surprising that the solutions are sparser on NASDAQ due to the high volatility. As indicated by our theory, the efficiency gain should be more significant in this case.

Table 1: Characteristics of HONES for universal portfolio management on NYSE and NASDAQ data. (Top) solution Sparsity on NYSE and NASDAQ datasets, including the average, standard error, maximum and minimum of the support size; (Bottom) distribution of e_t, the number of excess turning points, on NYSE and NASDAQ datasets.

Dataset	sparsity			
	mean	std.	max	min
NYSE	16.0	8.6	98	4
NASDAQ	6.46	3.7	64	2

Dataset	proportion of zeros	quantiles	
		99%	99.9%
NYSE	65.8%	8	22
NASDAQ	85.4%	4	11

Finally, we examine our conjecture in Section 2.4 on the number of turning points. As explained there, a benchmark for k_t is $|S_t \setminus S_{t-1}| + |S_{t-1} \setminus S_t|$. We refer to $e_t = (k_t - |S_t \setminus S_{t-1}| - |S_{t-1} \setminus S_t|)/2$ as the number of *excess turning points*; see Section 2.4 for details. For each synthetic dataset, we report the proportion of zero e_t in Table 1. It is clear that most steps (65.8% for NYSE data and 85.4% for NASDAQ data) are predicted by our conjecture and almost all steps (over 99% for both data) are not far away from our conjecture. This indicates that k_t is an accurate proxy for the number of turning points.

3.2 Synthetic Data

As discussed in Section 1, the problem (2) is indeed more general than universal portfolio management. To examine our algorithm comprehensively, we consider (2) under other setups. First we consider the problem with the following structure on synthetic data:

$$\min_{x \in \Delta_{n-1}} \frac{1}{2}(x-y)^T A^{(t)}(x-y). \tag{15}$$

Without the superscript t, this problem is called *standard quadratic programming problem* and has attracted the attention in various fields, e.g. Bomze (1998); Scozzari & Tardella (2008); Bomze et al. (2008). It is of particular interest to study the case where $A^{(t)}$ is a random matrix generated from some distribution. For instance, Chen et al. (2013) consider a Wigner matrix A with $\{A_{ij} : 1 \le i \le j \le n\}$ being i.i.d. random variables and $A_{ji} = A_{ij}$. In this article, we consider another important class of matrices in random matrix theory — covariance matrix of rectangular matrices with i.i.d. entries, i.e. $A^{(t)} = \sum_{s=1}^{t} g^{(s)}(g^{(s)})^T$, where $g^{(s)} \in \mathbb{R}^n$ has i.i.d. Gaussian entries; see Bai & Silverstein (2010) for more discussion. In this case the matrix flow $\{A^{(t)} : t = 1, 2, \ldots\}$ satisfies (3). To avoid singularity, we set $A^{(0)} = \epsilon I$ where $\epsilon = 10^{-4}$ is a small positive number. Then it is easy to see that $A^{(t)}$ is non-singular for all t with probability 1 so that the solution $x^{(t)}$ is unique. The vector y governs the sparsity of the solution. To see this, consider the isotropic case where $A = I$, the solution of (15) is the projection of y onto the simplex. If y is a zero vector, the optimum is a dense vector with all entries $\frac{1}{n}$. In contrast, if y has large entries, the simplex constraint will pull the optimum towards that direction and forces the other entries to be zero , in which case the solution is sparse. The same phenomenon is observed in anisotropic case as will be shown below.

Our goal is to explore the scalability, in terms of the dimension, and the adaptivity to solution sparsity of the algorithms. For the aspect of the dimension, we consider three dimensions: $\{100, 1000, 3000\}$; for the aspect of the sparsity, we set $y = cy_0$ with y_0 generated from $N(0, I_{n\times n})$ and $c \in \{0.01, 0.1\}$. For each case, we set the total number of steps as 5000 and treat every 250 steps as an epoch (20 epochs in total). Similar to the previous case, we report the cumulative running time in Figure 2 and other information in Table 2 and Table 3. Here we exclude the interior-point method since it is too slow.

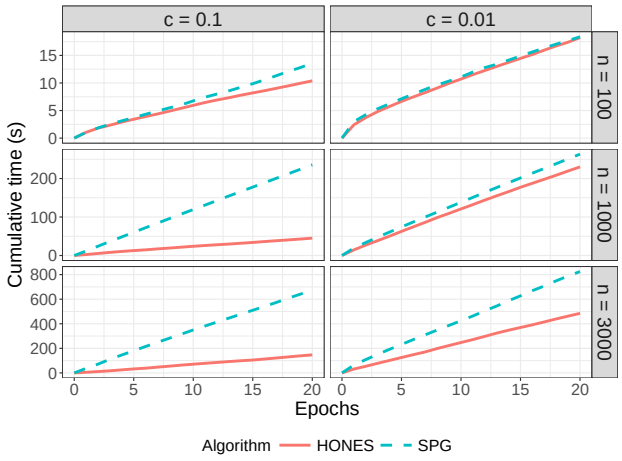

Figure 2: Cumulative running time of HONES and SPG on synthetic datasets. Each row corresponds to a dimension n and each column corresponds to a factor c.

Table 2: Solution Sparsity and overall computation gain of HONES over SPG on synthetic datasets. The first two columns correspond to the dimension and the factor c; the third column gives the mean of support size with its standard deviation (in the parentheses); the fourth column gives the maximum support size along the path; the last two columns show the ratio of overall running time between SPG and HONES .

scenarios		sparsity		speed ratio
n	c	mean (s.e.)	max	
100	0.01	81.3 (3.5)	88	0.97
1000	0.01	158.3 (30.6)	190	1.13
3000	0.01	146.4 (34.9)	225	1.68
100	0.1	19.5 (2.3)	26	1.34
1000	0.1	22.0 (5.3)	32	5.26
3000	0.1	18.3 (4.2)	27	4.51

First we notice that HONES is more scalable in high dimension. Moreover, as expected, a larger c gives sparser solutions along the path and HONES significantly outperforms SPG when the solution is sparse ($c = 0.1$) especially for large-scale problems ($n = 1000, 3000$), in which HONES is over 4.5 times faster than SPG. When the solution is not sparse ($c = 0.01$), HONES is similar to SPG in small-scale problem ($n = 100$) and increasingly more efficient when the size of the problem grows. Finally the results in Table 3 show that our conjecture in Section 2.4 is extremely accurate in this setting.

Table 3: Distribution of e_t, the number of excess turning points, on synthetic datasets. The first two columns correspond to the dimension and the factor c; the third column gives the proportion of zero e_t; the last two columns give the 99% and 99.9% quantiles of e_t.

scenarios		proportion of zeros	quantiles	
n	c		99%	99.9%
100	0.01	99.8%	0	1
1000	0.01	98.9%	1	4
3000	0.01	98.7%	1	6
100	0.1	99.9%	0	0
1000	0.1	99.8%	0	1
3000	0.1	99.8%	0	1

3.3 Markowitz Portfolio Selection

In this Subsection, we consider the application of HONES algorithm on sequential Markowitz portfolio selection problem. In this problem, the vector of stock prices is assumed to be a random vector with mean μ and covariance matrix Σ and the investor observes a realization from this distribution. The goal is to minimize the risk, measured by the variance $x^T\Sigma x$ of a given portfolio while maintaining a reasonably high average return $x^T\mu$. The problem is usually formulated as follows:

$$\min_{x_t\in\mathbb{R}^n} \frac{1}{2}x_t^T\Sigma x_t - \lambda x_t^T\mu.$$

For simplicity we assume $\lambda = 0$. In practice, μ and Σ are unknown and one has to replace them by estimators. The most natural estimators $\hat{\Sigma}^{(t)}$ and $\hat{\mu}^{(t)}$ are the sample covariance matrix and the sample mean, i.e.

$$\hat{\mu}^{(t)} = \frac{1}{t}\sum_{s=1}^{t} w^{(s)}, \hat{\Sigma}^{(t)} = \frac{1}{t}\sum_{s=1}^{t}(w^{(s)}-\hat{\mu}^{(s)})(w^{(s)}-\hat{\mu}^{(s)})^T,$$

where $w^{(t)}$ is the vector of daily gains, measured by the entrywise log return $\log(\gamma_t)$, of all assets of interest. Via some algebra, it can be shown that the problem is equivalent to (2) with

$$A^{(t)} = t\hat{\Sigma}^{(t)}, r^{(t)} = t\hat{\mu}^{(t)}, g^{(t)} = \sqrt{\frac{t-1}{t}}\left(w^{(t)} - \hat{\mu}^{(t-1)}\right).$$

We should emphasize that the solution in this way is optimal from hindsight, which is different from the notions in online learning regret minimization. Nonetheless, it is an interesting and important problem in the context of back testing and risk management since the result can reveal the hidden structure of the assets; see Brodie et al. (2009); Fan et al. (2008, 2012) for more details.

Similar to the Section 3.1, we report the cumulative running time of HONES and SPG in Figure 3 and report other information in Table 4. Again, HONES is more efficient than SPG. We also try the interior-point method on each dataset. It is 67 times slower than HONES on NYSE dataset and 31 times slower than HONES on NASDAQ dataset. Finally, the conjecture on the number of turning points is also validated by the results in the bottom panel of Table 4.

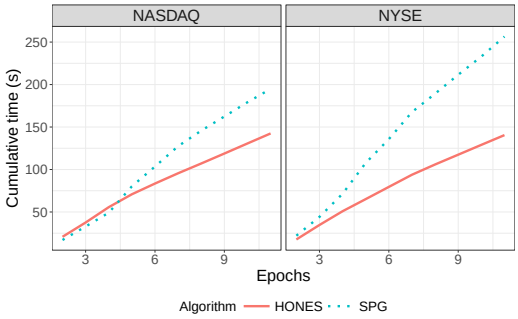

Figure 3: Cumulative running time of HONES and SPG on NYSE and NASDAQ dataset for Markowitz portfolio management. Each epoch has 252 measurements.

Table 4: (Top) solution Sparsity on NYSE and NASDAQ datasets for Markowitz portfolio management, including the average, standard error, maximum and minimum of the support size; (Bottom) distribution of e_t, the number of excess turning points, on NYSE and NASDAQ datasets.

Dataset	sparsity			
	mean	s.e.	max	min
NYSE	49.8	22.7	108	30
NASDAQ	148.9	17.8	192	127

Dataset	proportion of zeros	quantiles	
		99%	99.9%
NYSE	97.9%	1	10
NASDAQ	96.2%	3	22

4 Conclusion

In this article, we propose an efficient algorithm HONES to solve the sequential generalized projection problem (2) with rank-one update (3), appeared as the building block and the bottleneck of Online Newton Step. HONES is a homotopy continuation method that interpolates the consecutive objectives. By a careful derivation, we calculate the exact number of atomic operations, up to an additive constant (Theorem 3) and show that HONES has a good performance when the support of the solution changes slowly with time or is sparse as in many applications. We also provide a heuristic conjecture on the number of turning points which plays an important role in the computation complexity. The heuristic conjecture and the efficiency of HONES algorithm are supported and confirmed by extensive experiments on both synthetic and real data.

References

Bai, Zhidong and Silverstein, Jack W. *Spectral analysis of large dimensional random matrices*, volume 20. Springer, 2010.

Bank, B, Guddat, J, Klatte, D, Kummer, B, and Tammer, K. Non-linear parametric optimization. *Akademie-Verlag, Berlin*, 1982.

Beck, Amir and Teboulle, Marc. Mirror descent and nonlinear projected subgradient methods for convex optimization. *Operations Research Letters*, 31(3):167–175, 2003.

Bertsimas, Dimitris and Tsitsiklis, John N. *Introduction to linear optimization*, volume 6. Athena Scientific Belmont, MA, 1997.

Best, Michael J. An algorithm for the solution of the parametric quadratic programming problem. In *Applied mathematics and parallel computing*, pp. 57–76. Springer, 1996.

Birgin, Ernesto G, Martínez, José Mario, and Raydan, Marcos. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4): 1196–1211, 2000.

Bomze, Immanuel M. On standard quadratic optimization problems. *Journal of Global Optimization*, 13(4):369–387, 1998.

Bomze, Immanuel M, Locatelli, Marco, and Tardella, Fabio. New and old bounds for standard quadratic optimization: dominance, equivalence and incomparability. *Mathematical Programming*, 115(1):31–64, 2008.

Brodie, Joshua, Daubechies, Ingrid, De Mol, Christine, Giannone, Domenico, and Loris, Ignace. Sparse and stable markowitz portfolios. *Proceedings of the National Academy of Sciences*, 106(30):12267–12272, 2009.

Chen, Xin and Peng, Jiming. New analysis on sparse solutions to random standard quadratic optimization problems and extensions. *Mathematics of Operations Research*, 40 (3):725–738, 2015.

Chen, Xin, Peng, Jiming, and Zhang, Shuzhong. Sparse solutions to random standard quadratic optimization problems. *Mathematical Programming*, 141(1-2):273–293, 2013.

Chow, Shui-Nee, Mallet-Paret, John, and Yorke, James A. A homotopy method for locating all zeros of a system of polynomials. In *Functional differential equations and approximation of fixed points*, pp. 77–88. Springer, 1979.

Cover, Thomas M. Universal portfolios. *Mathematical finance*, 1(1):1–29, 1991.

Duchi, John, Shalev-Shwartz, Shai, Singer, Yoram, and Chandra, Tushar. Efficient projections onto the l_1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pp. 272–279. ACM, 2008.

Efron, Bradley, Hastie, Trevor, Johnstone, Iain, Tibshirani, Robert, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Fan, Jianqing, Zhang, Jingjin, and Yu, Ke. Asset allocation and risk assessment with gross exposure constraints for vast portfolios. *Available at SSRN 1307423*, 2008.

Fan, Jianqing, Zhang, Jingjin, and Yu, Ke. Vast portfolio selection with gross-exposure constraints. *Journal of the American Statistical Association*, 107(498):592–606, 2012.

Frank, Marguerite and Wolfe, Philip. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

Garrigues, Pierre and Ghaoui, Laurent E. An homotopy algorithm for the lasso with online observations. In *Advances in neural information processing systems*, pp. 489–496, 2009.

Gärtner, Bernd, Jaggi, Martin, and Maria, Clément. An exponential lower bound on the complexity of regularization paths. *arXiv preprint arXiv:0903.4817*, 2009.

Hazan, Elad and Arora, Sanjeev. *Efficient algorithms for online convex optimization and their applications*. Princeton University, 2006.

Hazan, Elad, Kalai, Adam, Kale, Satyen, and Agarwal, Amit. Logarithmic regret algorithms for online convex optimization. In *International Conference on Computational Learning Theory*, pp. 499–513. Springer, 2006a.

Hazan, Elad, Kalai, Adam, Kale, Satyen, and Agarwal, Amit. Logarithmic regret algorithms for online convex optimization. In *International Conference on Computational Learning Theory*, pp. 499–513. Springer, 2006b.

Helmbold, David P, Schapire, Robert E, Singer, Yoram, and Warmuth, Manfred K. On-line portfolio selection using multiplicative updates. *Mathematical Finance*, 8 (4):325–347, 1998.

Ibaraki, Toshihide and Katoh, Naoki. *Resource allocation problems: algorithmic approaches*. MIT press, 1988.

Kalai, Adam and Vempala, Santosh. Efficient algorithms for universal portfolios. *Journal of Machine Learning Research*, 3(Nov):423–440, 2002.

Kivinen, Jyrki and Warmuth, Manfred K. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63, 1997.

Li, Tien-Yien. On chow, mallet-paret and yorke homotopy for solving systems of polynomials. *Bull. Inst. Math. Acad. Sinica*, 11(3):433–437, 1983.

Luo, Haipeng, Agarwal, Alekh, Cesa-Bianchi, Nicolo, and Langford, John. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems*, pp. 902–910, 2016.

Mairal, Julien and Yu, Bin. Complexity analysis of the lasso regularization path. *arXiv preprint arXiv:1205.0079*, 2012.

Markowitz, Harry. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952.

Murty, Katta G and Yu, Feng-Tien. *Linear complementarity, linear and nonlinear programming*. Citeseer, 1988.

Ritter, Klaus. On parametric linear and quadratic programming problems. Technical report, DTIC Document, 1981.

Scozzari, Andrea and Tardella, Fabio. A clique algorithm for standard quadratic programming. *Discrete Applied Mathematics*, 156(13):2439–2448, 2008.

Shalev-Shwartz, Shai. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.

Wright, Stephen J. *Primal-Dual Interior-Point Methods*, volume 54. SIAM, 1997.

Zinkevich, Martin. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 928–936, 2003.