Mach Learn (2007) 69: 169–192
DOI 10.1007/s10994-007-5016-8

Logarithmic regret algorithms for online convex optimization

Elad Hazan · Amit Agarwal · Satyen Kale

Received: 3 October 2006 / Revised: 8 May 2007 /
Accepted: 25 May 2007 / Published online: 8 August 2007

Abstract In an online convex optimization problem a decision-maker makes a sequence of decisions, i.e., chooses a sequence of points in Euclidean space, from a fixed feasible set. After each point is chosen, it encounters a sequence of (possibly unrelated) convex cost functions. Zinkevich (ICML 2003) introduced this framework, which models many natural repeated decision-making problems and generalizes many existing problems such as Prediction from Expert Advice and Cover's Universal Portfolios. Zinkevich showed that a simple online gradient descent algorithm achieves additive *regret* $O(\sqrt{T})$, for an arbitrary sequence of T convex cost functions (of bounded gradients), with respect to the best single decision in hindsight.

In this paper, we give algorithms that achieve regret $O(\log(T))$ for an arbitrary sequence of strictly convex functions (with bounded first and second derivatives). This mirrors what has been done for the special cases of prediction from expert advice by Kivinen and Warmuth (EuroCOLT 1999), and Universal Portfolios by Cover (Math. Finance 1:1–19, 1991). We propose several algorithms achieving logarithmic regret, which besides being more general are also much more efficient to implement.

The main new ideas give rise to an efficient algorithm based on the Newton method for optimization, a new tool in the field. Our analysis shows a surprising connection between the natural follow-the-leader approach and the Newton method. We also analyze other algorithms, which tie together several different previous approaches including follow-the-leader, exponential weighting, Cover's algorithm and gradient descent.

Editors: Hans Ulrich Simon, Gabor Lugosi, Avrim Blum.

E. Hazan and S. Kale supported by Sanjeev Arora's NSF grants MSPA-MCS 0528414, CCF 0514993, ITR 0205594.

E. Hazan (✉)
IBM Almaden Research Center, 650 Harry Road, San Jose, 95120, USA
e-mail: ehazan@cs.princeton.edu

A. Agarwal · S. Kale
Department of Computer Science, Princeton University, Princeton, NJ, USA

Keywords Online learning · Online optimization · Regret minimization · Portfolio management

1 Introduction

In *online convex optimization*, an online player chooses a point in a convex set. After the point is chosen, a concave payoff function is revealed, and the online player receives payoff which is the concave function applied to the point she chose. This scenario is repeated for many iterations.

The online convex optimization framework generalizes many previous online optimization problems. For example, in the problem of online portfolio management an online investor wants to distribute her wealth on a set of n available financial instruments without knowing the market outcome in advance. The wealth distribution of the online investor can be thought of as a point in the set of all distributions over n items (the financial instruments), which is a convex set. The payoff to the online player is the change in wealth, which is a concave function of her distribution. Other examples which fit into this online framework include the problems of prediction from expert advice and online zero-sum game playing.

To measure the performance of the online player we consider two standard metrics. The first is called *regret*. Regret measures the difference in payoff between the online player and the best fixed point in hindsight. The second metric by which we measure performance is computational complexity, i.e. the amount of computer resources required to compute the online player's point for the upcoming iteration given the history of payoff functions encountered thus far.

Previous approaches for online convex optimization are based on first-order optimization, i.e. optimization using the first derivatives of the payoff functions. The regret achieved by these algorithms is proportional to a polynomial (square root) in the number of iterations. Besides the general framework, there are specialized algorithms, e.g. for portfolio management, which attain regret proportional to the logarithm of the number of iterations. However, these algorithms do not apply to the general online convex optimization framework and are less efficient in terms of computational complexity.

We introduce a new algorithm, ONLINE NEWTON STEP, which uses second-order information of the payoff functions and is based on the well known Newton–Raphson method for offline optimization. The ONLINE NEWTON STEP algorithm attains regret which is proportional to the logarithm of the number of iterations when the payoff functions are concave, and is computationally efficient.

In addition to the ONLINE NEWTON STEP algorithm, we also show two other approaches which can be used to achieve logarithmic regret in the case of some higher-order derivative assumptions on the functions.

1.1 Follow the leader

Perhaps the most intuitive algorithm for online convex optimization can be described as follows: at iteration t, choose the best point so far, i.e. the point in the underlying convex set that minimizes the sum of all cost functions encountered thus far.

Given the natural appeal of this algorithm, it was considered in the game theory literature for over 50 years. It is not difficult to show that for linear cost functions, the FOLLOW THE LEADER (FTL) algorithm does not attain any non-trivial regret guarantee (in the worst case it can be $\Omega(T)$ if the cost functions are chosen adversarially). However, Hannan (1957)

proposed a randomized variant of FTL, called perturbed-follow-the-leader, which attained $O(\sqrt{T})$ regret in the online game playing setting for linear functions over the simplex.[1]

As we show later, this regret bound is optimal. Merhav and Feder (1992) extend the FTL approach to strictly convex cost functions over the simplex, and prove that for such functions FTL attains regret which is logarithmic in the number of iterations. Similar results were obtained by Cesa-Bianchi and Lugosi (2006), and Gaivoronski and Stella (2000).

A natural question, asked explicitly by Cover and Ordentlich, Kalai and Vempala, and others, is whether FOLLOW THE LEADER provides any non-trivial guarantee for curved (but not necessarily strictly convex) cost functions. One application which is not covered by previous results is the problem of portfolio management.

In this paper (Sect. 3.3) we answer this question in the affirmative, and prove that in fact FOLLOW THE LEADER attains optimal regret for curved functions.

2 Preliminaries

2.1 Online convex optimization

In online convex optimization, an online player iteratively chooses a point from a set in Euclidean space denoted $\mathcal{P} \subseteq \mathbb{R}^n$. Following Zinkevich (2003), we assume that the set $\mathcal{P}$ is non-empty, bounded and closed. For reasons that will be apparent in Sects. 4 and 3.3.2, we also assume the set $\mathcal{P}$ to be convex.

We denote the number of iterations by T (which is unknown to the online player). At iteration t, the online player chooses $\mathbf{x}_t \in \mathcal{P}$. After committing to this choice, a convex cost function $f_t : \mathcal{P} \mapsto \mathbb{R}$ is revealed. The cost incurred to the online player is the value of the cost function at the point she committed to, i.e. $f_t(\mathbf{x}_t)$.

Consider an online player using a (possibly randomized) algorithm for online game playing $\mathcal{A}$. At iteration t, the algorithm $\mathcal{A}$ takes as input the history of cost functions $f_1, \ldots, f_{t-1}$ and produces a feasible point $\mathcal{A}(\{f_1, \ldots, f_{t-1}\})$ in the domain $\mathcal{P}$. When there is no ambiguity concerning the algorithm used, we simply denote $\mathbf{x}_t = \mathcal{A}(\{f_1, \ldots, f_{t-1}\})$. The regret of the online player using algorithm $\mathcal{A}$ at time T, is defined to be the total cost minus the cost of the best single decision, where the best is chosen with the benefit of hindsight. Formally

$$\text{Regret}(\mathcal{A}, \{f_1, \ldots, f_T\}) = \mathbf{E}\left[\sum_{t=1}^{T} f_t(\mathbf{x}_t)\right] - \min_{\mathbf{x}\in\mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x}).$$

Regret measures the difference in performance between the online player and a "static" player with the benefit of hindsight—i.e. a player that is constrained to choose a fixed point over all iterations. It is tempting to compare the online player to an adversary which has the benefit of hindsight but is otherwise unconstrained (i.e. can dynamically change her point every iteration). However, this allows the adversary to choose the optimum point $\mathbf{x}_t^* \triangleq \min_{x\in\mathcal{P}} f_t(\mathbf{x})$ each iteration, and the comparison becomes trivial in many interesting applications.

We are usually interested in an upper bound on the worst case guaranteed regret, denoted

$$\text{Regret}_T(\mathcal{A}) = \sup_{\{f_1,\ldots,f_t\}} \{\text{Regret}(\mathcal{A}, \{f_1, \ldots, f_t\})\}.$$

[1] This algorithm was rediscovered in (Kalai and Vempala 2005), who provide a much simpler analysis and many applications.

Intuitively, an algorithm attains non-trivial performance if its regret is sublinear as a function of T, i.e. $\text{Regret}_T(\mathcal{A}) = o(T)$, since this implies that "on the average" the algorithm performs as good as the best fixed strategy in hindsight.

Remark For some problems it is more natural to talk of "payoff" given to the online player rather than cost she incurs. In such cases, the online player receives payoff $f_t(\mathbf{x}_t)$, where f_t is a concave utility function. Regret is then defined to be

$$\text{Regret}(\mathcal{A}, \{f_1, \ldots, f_T\}) = \max_{\mathbf{x}\in\mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x}) - \mathbf{E}\left[\sum_{t=1}^{T} f_t(\mathbf{x}_t)\right].$$

The running time of an algorithm for online game playing is defined to be the worst-case expected time to produce $\mathbf{x}_t$, for an iteration $t \in [T]$[2] in a T iteration repeated game. Typically, the running time will depend on n, T and parameters of the cost functions and underlying convex set.

2.2 Notation and definitions

Recall that in online convex optimization, the online player iteratively chooses points from a closed, bounded and non-empty convex set $\mathcal{P} \subseteq \mathbb{R}^n$ and encounters convex cost functions $\{f_t : \mathcal{P} \mapsto \mathbb{R}\}$.

Denote by D the diameter of the underlying convex set $\mathcal{P}$, i.e.

$$D = \max_{\mathbf{x},\mathbf{y}\in\mathcal{P}} \|\mathbf{x} - \mathbf{y}\|_2.$$

Unless stated otherwise, we assume that the cost functions $\{f_t\}$ are twice differentiable and convex. These assumptions are satisfied by all applications described previously.

Recall that the gradient for a $f : \mathbb{R}^n \mapsto \mathbb{R}$ at point $\mathbf{x} \in \mathbb{R}^n$ is the vector $\nabla f(\mathbf{x})$ whose components are the partial derivatives of the function at $\mathbf{x}$. Its direction is the one in which the function has the largest rate of increase, and its magnitude is the actual rate of increase. We say that the cost functions have gradients upper bounded by a number G if the following holds:

$$\sup_{\mathbf{x}\in\mathcal{P}, t\in[T]} \|\nabla f_t(\mathbf{x})\|_2 \le G.$$

In some cases we are concerned with the ℓ_∞ norm of the gradient rather than the Euclidean norm, in which case we denote the upper bound by G_∞.

We also consider the analogue of second derivatives for multivariate functions. The Hessian of a function f at point $\mathbf{x}$ is a matrix $\nabla^2 f(\mathbf{x})$, such that $\nabla^2 f(\mathbf{x})[i, j] = \frac{\partial^2}{\partial \mathbf{x}_i, \mathbf{x}_j} f(\mathbf{x})$. Analogous to the one-dimensional case, a function f is convex at point $\mathbf{x}$ if and only if its Hessian is positive semidefinite, denoted by $\nabla^2 f(\mathbf{x}) \succeq \mathbf{0}$. We say that the Hessian of all cost functions is *lower bounded by a number* $H > 0$ if the following holds:

$$\forall \mathbf{x} \in \mathcal{P},\ t \in [T]: \quad \nabla^2 f_t(\mathbf{x}) \ \succeq\ H\mathbf{I}_n.$$

Here, $\mathbf{I}_n$ is the n-dimensional identity matrix and we denote $\mathbf{A} \succeq \mathbf{B}$ if the matrix $\mathbf{A} - \mathbf{B}$ is positive semidefinite, i.e. all its eigenvalues are nonnegative. Thus, H is a lower bound on the eigenvalues of all the Hessians of the constraints at all points in the domain. Such functions will be called *H-strong convex*.

[2] Here and henceforth we denote by $[n]$ the set of integers $\{1, \ldots, n\}$.

In the following chapters we will consider two different classes of cost functions. One class of cost functions are those which have bounded gradient and are H-strong convex for some $H > 0$. The second class of cost functions are those that satisfy the *α-exp-concavity* property: there is an $\alpha > 0$ such that $\exp(-\alpha f_t(\mathbf{x}))$ is a concave function of $\mathbf{x} \in \mathcal{P}$, for all t, i.e.

$$\forall \mathbf{x} \in \mathcal{P},\ t \in [T]: \quad \nabla^2[\exp(-\alpha f_t(\mathbf{x}))] \preceq \mathbf{0}.$$

The second class is more general than the first, since the α-exp-concavity condition is weaker than a bounded gradient and strict convexity. It is easy to show that functions that have gradients upper bounded by G and Hessian lower bounded by $H > 0$, are α-exp-concave for any $\alpha \leq H/G^2$. One can easily verify this for one-dimensional functions $f_t : \mathbb{R} \to \mathbb{R}$ by taking two derivatives,

$$h_t''(x) = ((\alpha f_t'(x))^2 - \alpha f_t''(x)) \exp(-\alpha f_t(x)) \leq 0 \quad \Longleftrightarrow \quad \alpha \leq \frac{f_t''(x)}{(f_t'(x))^2}.$$

We note that there are many interesting loss functions which are exp-concave but not strictly convex. A prominent example is the log-loss function, i.e. $f(\mathbf{x}) = -\log(\mathbf{x}^\top \mathbf{a})$ for a vector of constants $\mathbf{a}$. This loss function arises in the problem of universal portfolio management (Cover 1991). Henceforth we denote the natural logarithm by log.

When it is more natural to talk of maximization of payoff rather than minimization of cost (e.g. for portfolio management), we require the payoff functions to be concave instead of convex. The parameter H is then defined to be

$$\forall \mathbf{x} \in \mathcal{P},\ t \in [T] \quad \nabla^2 f_t(\mathbf{x}) \preceq -H\mathbf{I}_n,$$

and the payoff functions will be assumed to be $(-\alpha)$-exp-concave:

$$\forall \mathbf{x} \in \mathcal{P},\ t \in [T]: \quad \nabla^2[\exp(\alpha f_t(\mathbf{x}))] \preceq \mathbf{0}.$$

2.3 Summary of our results

A standard goal in machine learning and game theory is to achieve algorithms with guaranteed low regret. Zinkevich (2003) showed that one can guarantee $O(\sqrt{T})$ regret for an *arbitrary* sequence of differentiable convex functions of bounded gradient, which is tight up to constant factors. In fact, $\Omega(\sqrt{T})$ regret is unavoidable even when the functions come from a fixed distribution rather than being chosen adversarially.[3]

In this paper we describe three algorithms with regret which is bounded by a logarithm in the number of iterations T, as summarized in Table 1. In the running time column, T_{proj} stands for the time it takes to compute a projection onto the underlying convex set (see Sect. 4). Similarly, T^g_{proj} stands for the time to compute a generalized projection. All algorithms assume an oracle that given a point in the convex set returns the value of the cost function on that point and/or the gradient of the function. The O notation for the regret bounds hides constant factors. For the running time bounds the $\tilde{O}$ notation hides constant factors as well as polylogarithmic factors in n, T, G, H, D, α.

[3]This can be seen by a simple randomized example. Consider $K = [-1, 1]$ and linear functions $f_t(x) = r_t x$, where $r_t = \pm 1$ are chosen in advance, independently with equal probability. $\mathbf{E}_{r_t}[f_t(x_t)] = 0$ for any t and x_t chosen online, by independence of x_t and r_t. However, $\mathbf{E}_{r_1,\ldots,r_T}[\min_{x \in K} \sum_1^T f_t(x)] = \mathbf{E}[-|\sum_1^T r_t|] = -\Omega(\sqrt{T})$.

Table 1 Results from this paper. Zinkevich achieves $O(GD\sqrt{T})$ regret, even for $H = \alpha = 0$

Algorithm	Regret bound	Running time
OGD	$O(\frac{G^2}{H} \log T)$	$\tilde{O}(n) + T_{\text{proj}}$
ONS	$O((\frac{1}{\alpha} + GD)n \log T)$	$\tilde{O}(n^2) + T^g_{\text{proj}}$
FTAL	$O((\frac{1}{\alpha} + GD)n \log T)$	$\tilde{O}(n^2) + T^g_{\text{proj}}$
EWOO	$O(\frac{n}{\alpha} \log T)$	$poly(T, n)$

ONLINE GRADIENT DESCENT.

Inputs: convex set $\mathcal{P} \subset \mathbb{R}^n$, step sizes $\eta_1, \eta_2, \ldots \geq 0$, initial $\mathbf{x}_1 \in \mathcal{P}$.

- In iteration 1, use point $\mathbf{x}_1 \in \mathcal{P}$.
- In iteration $t > 1$: use point

$$\mathbf{x}_t = \Pi_{\mathcal{P}}(\mathbf{x}_{t-1} - \eta_t \nabla f_{t-1}(\mathbf{x}_{t-1}))$$

Here, $\Pi_{\mathcal{P}}$ denotes the *projection* onto nearest point in $\mathcal{P}$, $\Pi_{\mathcal{P}}(\mathbf{y}) = \arg\min_{\mathbf{x} \in \mathcal{P}} \|\mathbf{x} - \mathbf{y}\|_2$.

Fig. 1 The ONLINE GRADIENT DESCENT Algorithm (Zinkevich's online version of Stochastic Gradient Descent)

Since the exp-concavity assumption on the convex cost functions is a *weaker* assumption than the bounds on the gradients and Hessians (see previous subsection), we can compare the three regret bounds of Fig. 1. In these terms, ONLINE GRADIENT DESCENT (OGD) requires the strongest assumptions, whereas EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION (EWOO) requires only exp-concavity (and not even a bound on the gradient). Perhaps most interesting are ONLINE NEWTON STEP (ONS) and its close cousin, FOLLOW THE APPROXIMATE LEADER (FTAL) which require relatively weak assumptions and yet, as we shall see, are very efficient to implement (and whose analysis is the most technical).

Perhaps the most interesting applications of these logarithmic regret algorithms is the problem of portfolio management (Cover 1991). The portfolio management is a special case of online convex optimization, in which the payoff functions are logarithmic. These payoff functions are exp-concave (indeed, the exponent of the logarithm function is a linear function, which is of course concave), but not strongly concave!

Another application in which the payoff/loss functions are exp-concave but not strictly convex is linear regression (see Kivinen and Warmuth 1998 and referenced papers). In the basic setting of this application, the predictor is given a vector $a_t \in \mathbb{R}^n$ and needs to predict $x_t \in \mathbb{R}^n$, the loss is $f_t(x) = (a_t^\top x - b_t)^2$ for some $b_t \in \mathbb{R}$. The Hessian of such loss function has rank of one, and thus not strictly convex. Nevertheless, under some reasonable assumptions over the domains of x, a_t, b_t, the functions are exp-concave.

3 The algorithms

3.1 ONLINE GRADIENT DESCENT

The first algorithm that achieves regret logarithmic in the number of iterations is a twist on Zinkevich's ONLINE GRADIENT DESCENT algorithm, as defined in Fig. 1.

The ONLINE GRADIENT DESCENT algorithm is straightforward to implement, and the running time is $O(n)$ per iteration given the gradient. However, there is a projection step which may take longer. We discuss the computational complexity of computing projections in Sect. 4.

The following theorem establishes logarithmic bounds on the regret if the cost functions are strictly convex.

Theorem 1 ONLINE GRADIENT DESCENT *with step sizes* $\eta_t = \frac{1}{Ht}$ *achieves the following guarantee, for all* $T \geq 1$

$$\text{Regret}_T(OGD) \leq \frac{G^2}{2H}(1 + \log T).$$

Proof Let $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x})$. Recall the definition of regret (see Sect. 2)

$$\text{Regret}_T(OGD) = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^*).$$

Define $\nabla_t \triangleq \nabla f_t(\mathbf{x}_t)$. By using the Taylor series approximation, we have, for some point ζ_t on the line segment joining $\mathbf{x}_t$ to $\mathbf{x}^*$,

$$\begin{aligned} f_t(\mathbf{x}^*) &= f_t(\mathbf{x}_t) + \nabla_t^\top(\mathbf{x}^* - \mathbf{x}_t) + \frac{1}{2}(\mathbf{x}^* - \mathbf{x}_t)^\top \nabla^2 f_t(\zeta_t)(\mathbf{x}^* - \mathbf{x}_t) \\ &\geq f_t(\mathbf{x}_t) + \nabla_t^\top(\mathbf{x}^* - \mathbf{x}_t) + \frac{H}{2}\|\mathbf{x}^* - \mathbf{x}_t\|^2. \end{aligned}$$

The inequality follows from H-strong convexity. Thus, we have

$$2(f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*)) \leq 2\nabla_t^\top(\mathbf{x}_t - \mathbf{x}^*) - H\|\mathbf{x}^* - \mathbf{x}_t\|^2. \tag{1}$$

Following Zinkevich's analysis, we upper-bound $\nabla_t^\top(\mathbf{x}_t - \mathbf{x}^*)$. Using the update rule for $\mathbf{x}_{t+1}$ and the properties of projections (see Lemma 8), we get

$$\|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 = \|\Pi(\mathbf{x}_t - \eta_{t+1}\nabla_t) - \mathbf{x}^*\|^2 \leq \|\mathbf{x}_t - \eta_{t+1}\nabla_t - \mathbf{x}^*\|^2.$$

Hence,

$$\begin{aligned} \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2 &\leq \|\mathbf{x}_t - \mathbf{x}^*\|^2 + \eta_{t+1}^2\|\nabla_t\|^2 - 2\eta_{t+1}\nabla_t^\top(\mathbf{x}_t - \mathbf{x}^*), \\ 5\nabla_t^\top(\mathbf{x}_t - \mathbf{x}^*) &\leq \frac{\|\mathbf{x}_t - \mathbf{x}^*\|^2 - \|\mathbf{x}_{t+1} - \mathbf{x}^*\|^2}{\eta_{t+1}} + \eta_{t+1}G^2. \end{aligned} \tag{2}$$

Sum up (2) from $t = 1$ to T. Set $\eta_{t+1} = 1/(Ht)$, and using (1), we have:

$$\begin{aligned} 2\sum_{t=1}^{T} f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*) &\leq \sum_{t=1}^{T} \|\mathbf{x}_t - \mathbf{x}^*\|^2\left(\frac{1}{\eta_{t+1}} - \frac{1}{\eta_t} - H\right) + G^2\sum_{t=1}^{T}\eta_{t+1} \\ &= 0 + G^2\sum_{t=1}^{T}\frac{1}{Ht} \leq \frac{G^2}{H}(1 + \log T). \end{aligned}$$

□

3.2 Online Newton Step

If the Online Gradient Descent algorithm is the analogue of the Gradient Descent optimization method for the online setting, then Online Newton Step is the online analogue of the Newton–Raphson method. The Online Newton Step algorithm detailed in Fig. 2. The point chosen by the algorithm for a given iteration is a simple modification of the point chosen in the previous iteration: a vector is added to it. Whereas for the Online Gradient Descent algorithm this added vector is the gradient of the previous cost function, for Online Newton Step this vector is different: it is reminiscent to the direction in which the Newton–Raphson method would proceed if it were an offline optimization problem for the previous cost function. The Newton–Raphson algorithm would move in the direction of the vector which is the inverse Hessian multiplied by the gradient. In our case this direction is $\mathbf{A}_t^{-1}\nabla_t$, and the matrix $\mathbf{A}_t$ is related to the Hessian as will be shown in the analysis.

Since just adding a multiple of the Newton vector to the current point may result in a point outside the convex set, we project back into the set to obtain $\mathbf{x}_t$. This projection is somewhat different than the standard projection used by Online Gradient Descent in the previous section. It is the projection according to the norm defined by the matrix $\mathbf{A}_t$, rather than the Euclidean norm. The reason for using this projection is technical, and will be pointed out in the analysis.

The following theorem bounds the regret of Online Newton Step. The intuition which led to this theorem appears in the next section on follow-the-leader and its surprising connection to the Newton method.

Theorem 2 *Assume that for all t, the loss function $f_t : \mathcal{P} \to \mathbb{R}^n$ is α-exp-concave and has the property that $\forall \mathbf{x} \in \mathcal{P}$, $\|\nabla f(\mathbf{x})\| \leq G$. Then the algorithm* Online Newton Step *has the following regret bound*:

$$\mathrm{Regret}_T(\mathrm{ONS}) \leq 5\left(\frac{1}{\alpha} + GD\right) n \log T.$$

We begin with a lemma which shows how to approximate the cost functions up to the second order. Using the Taylor series we have $f(\mathbf{x}) = f(\mathbf{y}) + \nabla f(\mathbf{y})(\mathbf{x}-\mathbf{y}) + \frac{1}{2}(\mathbf{x}-\mathbf{y})^\top \nabla^2 f(\zeta)(\mathbf{x}-\mathbf{y})$ for some ζ on the line between $\mathbf{x}$ and $\mathbf{y}$. Instead of using this

ONS

- In iteration 1, use an arbitrary point $\mathbf{x}_1 \in \mathcal{P}$.
- Let $\beta = \frac{1}{2}\min\{\frac{1}{4GD}, \alpha\}$. In iteration $t > 1$, use point:vadjust

$$\mathbf{x}_t = \Pi_{\mathcal{P}}^{\mathbf{A}_{t-1}}\left(\mathbf{x}_{t-1} - \frac{1}{\beta}\mathbf{A}_{t-1}^{-1}\nabla_{t-1}\right)$$

  where $\nabla_\tau = \nabla f_\tau(\mathbf{x}_\tau)$, $\mathbf{A}_t = \sum_{i=1}^t \nabla_i \nabla_i^\top + \varepsilon \mathbf{I}_n$, $\varepsilon = \frac{1}{\beta^2 D^2}$, and $\Pi_{\mathcal{P}}^{\mathbf{A}_{t-1}}$ is the projection in the norm induced by $\mathbf{A}_{t-1}$, viz.,

$$\Pi_{\mathcal{P}}^{\mathbf{A}_{t-1}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{P}} (\mathbf{y}-\mathbf{x})^\top \mathbf{A}_{t-1}(\mathbf{y}-\mathbf{x})$$

Fig. 2 The Online Newton Step algorithm

approximation, we use a somewhat stronger approximation in which the Hessian of the cost function is not used, but rather only the gradient. Such an approximation is possible because we assume that the cost functions are α-exp-concave.

Lemma 3 *For a function $f : \mathcal{P} \to \mathbb{R}$, where $\mathcal{P}$ has diameter D, such that $\forall \mathbf{x} \in \mathcal{P}$, $\|\nabla f(\mathbf{x})\| \leq G$ and $\exp(-\alpha f(\mathbf{x}))$ is concave, the following holds for $\beta \leq \frac{1}{2}\min\{\frac{1}{4GD}, \alpha\}$:*

$$\forall x, y \in \mathcal{P}: \quad f(\mathbf{x}) \geq f(\mathbf{y}) + \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}) + \frac{\beta}{2}(\mathbf{x} - \mathbf{y})^\top \nabla f(\mathbf{y}) \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}).$$

Proof Since $\exp(-\alpha f(\mathbf{x}))$ is concave and $2\beta \leq \alpha$, the function $h(\mathbf{x}) \triangleq \exp(-2\beta f(\mathbf{x}))$ is also concave. Then by the concavity of $h(\mathbf{x})$,

$$h(\mathbf{x}) \leq h(\mathbf{y}) + \nabla h(\mathbf{y})^\top (\mathbf{x} - \mathbf{y}).$$

Plugging in $\nabla h(\mathbf{y}) = -2\beta \exp(-2\beta f(\mathbf{y})) \nabla f(\mathbf{y})$ gives,

$$\exp(-2\beta f(\mathbf{x})) \leq \exp(-2\beta f(\mathbf{y}))[1 - 2\beta \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})].$$

Simplifying

$$f(\mathbf{x}) \geq f(\mathbf{y}) - \frac{1}{2\beta}\log[1 - 2\beta \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})].$$

Next, note that $|2\beta \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})| \leq 2\beta GD \leq \frac{1}{4}$ and that for $|z| \leq \frac{1}{4}$, $-\log(1 - z) \geq z + \frac{1}{4}z^2$. Applying the inequality for $z = 2\beta \nabla f(\mathbf{y})^\top (\mathbf{x} - \mathbf{y})$ implies the lemma. □

We can now prove Theorem 2. The proof technique is reminiscent of Theorem 1, however the direction of curvature of the second derivative is taken into account. This gives rise to a potential function in terms of the metric A_t (which represents the sum of all Hessians up to iteration t), instead of the corresponding term in the analysis of Theorem 1 which included only the norms of the gradients.

Proof of Theorem 2 Let $\mathbf{x}^* \in \arg\min_{\mathbf{x} \in \mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x})$ be the best decision in hindsight. By Lemma 3, we have

$$f_t(\mathbf{x}_t) - f_t(\mathbf{x}^*) \leq R_t \triangleq \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^*) - \frac{\beta}{2}(\mathbf{x}^* - \mathbf{x}_t)^\top \nabla_t \nabla_t^\top (\mathbf{x}^* - \mathbf{x}_t) \tag{3}$$

for $\beta = \frac{1}{2}\min\{\frac{1}{4GD}, \alpha\}$. For convenience, define $\mathbf{y}_{t+1} = \mathbf{x}_t - \frac{1}{\beta}\mathbf{A}_t^{-1}\nabla_t$ so that according to the update rule of the algorithm $\mathbf{x}_{t+1} = \Pi_{S_n}^{\mathbf{A}_t}(\mathbf{y}_{t+1})$. Now, by the definition of $\mathbf{y}_{t+1}$:

$$\mathbf{y}_{t+1} - \mathbf{x}^* = \mathbf{x}_t - \mathbf{x}^* - \frac{1}{\beta}\mathbf{A}_t^{-1}\nabla_t, \tag{4}$$

$$\mathbf{A}_t(\mathbf{y}_{t+1} - \mathbf{x}^*) = \mathbf{A}_t(\mathbf{x}_t - \mathbf{x}^*) - \frac{1}{\beta}\nabla_t. \tag{5}$$

Multiplying the transpose of (4) by (5) we get

$$\begin{aligned}&(\mathbf{y}_{t+1} - \mathbf{x}^*)^\top \mathbf{A}_t(\mathbf{y}_{t+1} - \mathbf{x}^*) \\ &\quad = (\mathbf{x}_t - \mathbf{x}^*)^\top \mathbf{A}_t(\mathbf{x}_t - \mathbf{x}^*) - \frac{2}{\beta}\nabla_t^\top (\mathbf{x}_t - \mathbf{x}^*) + \frac{1}{\beta^2}\nabla_t^\top \mathbf{A}_t^{-1}\nabla_t.\end{aligned} \tag{6}$$

Since $\mathbf{x}_{t+1}$ is the projection of $\mathbf{y}_{t+1}$ in the norm induced by $\mathbf{A}_t$, it is a well known fact that (see Sect. 4 Lemma 8)

$$(\mathbf{y}_{t+1} - \mathbf{x}^*)^\top \mathbf{A}_t (\mathbf{y}_{t+1} - \mathbf{x}^*) \geq (\mathbf{x}_{t+1} - \mathbf{x}^*)^\top \mathbf{A}_t (\mathbf{x}_{t+1} - \mathbf{x}^*).$$

This inequality is the reason for using generalized projections as opposed to standard projections, which were used in the analysis of ONLINE GRADIENT DESCENT (see previous subsection). This fact together with (6) gives

$$\begin{aligned}\nabla_t^\top (\mathbf{x}_t - \mathbf{x}^*) \leq \frac{1}{2\beta} \nabla_t^\top \mathbf{A}_t^{-1} \nabla_t + \frac{\beta}{2} (\mathbf{x}_t - \mathbf{x}^*)^\top \mathbf{A}_t (\mathbf{x}_t - \mathbf{x}^*) \\ - \frac{\beta}{2} (\mathbf{x}_{t+1} - \mathbf{x}^*)^\top \mathbf{A}_t (\mathbf{x}_{t+1} - \mathbf{x}^*).\end{aligned}$$

Now, summing up over $t = 1$ to T we get that

$$\begin{aligned}\sum_{t=1}^T \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^*) &\leq \frac{1}{2\beta} \sum_{t=1}^T \nabla_t^\top \mathbf{A}_t^{-1} \nabla_t + \frac{\beta}{2} (\mathbf{x}_1 - \mathbf{x}^*)^\top \mathbf{A}_1 (\mathbf{x}_1 - \mathbf{x}^*) \\ &\quad + \frac{\beta}{2} \sum_{t=2}^T (\mathbf{x}_t - \mathbf{x}^*)^\top (\mathbf{A}_t - \mathbf{A}_{t-1}) (\mathbf{x}_t - \mathbf{x}^*) \\ &\quad - \frac{\beta}{2} (\mathbf{x}_{T+1} - \mathbf{x}^*)^\top \mathbf{A}_T (\mathbf{x}_{T+1} - \mathbf{x}^*) \\ &\leq \frac{1}{2\beta} \sum_{t=1}^T \nabla_t^\top \mathbf{A}_t^{-1} \nabla_t + \frac{\beta}{2} \sum_{t=1}^T (\mathbf{x}_t - \mathbf{x}^*)^\top \nabla_t \nabla_t^\top (\mathbf{x}_t - \mathbf{x}^*) \\ &\quad + \frac{\beta}{2} (\mathbf{x}_1 - \mathbf{x}^*)^\top (\mathbf{A}_1 - \nabla_1 \nabla_1^\top) (\mathbf{x}_1 - \mathbf{x}^*).\end{aligned}$$

In the last inequality we use the fact that $\mathbf{A}_t - \mathbf{A}_{t-1} = \nabla_t \nabla_t^\top$. By transferring the $\frac{\beta}{2} \sum_{t=1}^T (\mathbf{x}_t - \mathbf{x}^*)^\top (\mathbf{A}_t - \mathbf{A}_{t-1}) (\mathbf{x}_t - \mathbf{x}^*)$ term to the LHS, we get the expression for $\sum_{t=1}^T R_t$. Thus, we have

$$\sum_{t=1}^T R_t \leq \frac{1}{2\beta} \sum_{t=1}^T \nabla_t^\top \mathbf{A}_t^{-1} \nabla_t + \frac{\beta}{2} (\mathbf{x}_1 - \mathbf{x}^*)^\top (\mathbf{A}_1 - \nabla_1 \nabla_1^\top) (\mathbf{x}_1 - \mathbf{x}^*).$$

Using the facts that $\mathbf{A}_1 - \nabla_1 \nabla_1^\top = \varepsilon \mathbf{I}_n$ and $\|\mathbf{x}_1 - \mathbf{x}^*\|^2 \leq D^2$, and the choice of $\varepsilon = \frac{1}{\beta^2 D^2}$ we get

$$\begin{aligned}\text{Regret}_T(\text{ONS}) \leq \sum_{t=1}^T R_t \leq \frac{1}{2\beta} \sum_{t=1}^T \nabla_t^\top \mathbf{A}_t^{-1} \nabla_t + \frac{\varepsilon}{2} D^2 \beta \\ \leq \frac{1}{2\beta} \sum_{t=1}^T \nabla_t^\top \mathbf{A}_t^{-1} \nabla_t + \frac{1}{2\beta}.\end{aligned}$$

At this point we bound the first term above, which is our potential function. The linear algebraic facts required to bound this potential appear in Lemma 11 (see Appendix 2), which we apply with $\mathbf{V}_t = \mathbf{A}_t$, $\mathbf{u}_t = \nabla_t$, and $r = G$ to get the bound $\frac{n}{2\beta} \log(G^2 T/\varepsilon + 1)$. Now since

$\varepsilon = \frac{1}{\beta^2 D^2}$ and $\beta = \frac{1}{2}\min\{\frac{1}{4GD}, \alpha\}$, we get

$$\begin{aligned}\frac{1}{2\beta}\sum_{t=1}^{T} \nabla_t^\top \mathbf{A}_t^{-1} \nabla_t &\leq \frac{n}{2\beta}\log(G^2 T/\varepsilon + 1) \\ &\leq \frac{n}{2\beta}\log(TG^2\beta^2 D^2 + 1) \\ &\leq \frac{n}{2\beta}\log(T).\end{aligned}$$

Since $\beta = \frac{1}{2}\min\{\frac{1}{4GD}, \alpha\}$, we have $\frac{1}{\beta} \leq 8(GD + \frac{1}{\alpha})$. This gives the stated regret bound. □

3.2.1 Implementation and running time

The ONLINE NEWTON STEP algorithm requires $O(n^2)$ space to store the matrix $\mathbf{A}_t$. Every iteration requires the computation of the matrix $\mathbf{A}_t^{-1}$, the current gradient, a matrix-vector product and possibly a projection onto the underlying convex set $\mathcal{P}$.

A naive implementation would require computing the inverse of the matrix $\mathbf{A}_t$ every iteration. However, in case $\mathbf{A}_t$ is invertible, the matrix inversion lemma (Brookes 2005) states that for invertible matrix $\mathbf{A}$ and vector $\mathbf{x}$

$$(\mathbf{A} + \mathbf{x}\mathbf{x}^\top)^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\mathbf{x}\mathbf{x}^\top\mathbf{A}^{-1}}{1 + \mathbf{x}^\top\mathbf{A}^{-1}\mathbf{x}}.$$

Thus, given $\mathbf{A}_{t-1}^{-1}$ and ∇_t one can compute $\mathbf{A}_t^{-1}$ in time $O(n^2)$ using only matrix-vector and vector-vector products.

The ONLINE NEWTON STEP algorithm also needs to make projections onto $\mathcal{P}$, but of a slightly different nature than ONLINE GRADIENT DESCENT and other online convex optimization algorithms. The required projection, denoted by $\Pi_{\mathcal{P}}^{\mathbf{A}_t}$, is in the vector norm induced by the matrix $\mathbf{A}_t$, viz. $\|\mathbf{x}\|_{\mathbf{A}_t} = \sqrt{\mathbf{x}^\top\mathbf{A}_t\mathbf{x}}$. It is equivalent to finding the point $\mathbf{x} \in \mathcal{P}$ which minimizes $(\mathbf{x} - \mathbf{y})^\top\mathbf{A}_t(\mathbf{x} - \mathbf{y})$ where y is the point we are projecting. This is a convex program which can be solved up to any degree of accuracy in polynomial time, see Sect. 4.

Modulo the computation of generalized projections, the ONLINE NEWTON STEP algorithm can be implemented in time and space $O(n^2)$. In addition, the only information required is the gradient at each step (and the exp-concavity constant of the payoff functions).

3.3 FOLLOW THE APPROXIMATE LEADER

The intuition behind most of our algorithms stem from new observations regarding the well studied FOLLOW THE LEADER (FTL) method (see Hannan 1957; Kalai and Vempala 2005).

The basic FTL method, which by itself fails to provide sub-linear regret let alone logarithmic regret, simply chooses on period t the single fixed decision that would have been the best to use on the previous $t-1$ periods. This corresponds to choosing $\mathbf{x}_t = \arg\min_{\mathbf{x}\in\mathcal{P}} \sum_{\tau=1}^{t-1} f_\tau(\mathbf{x})$.

Below we prove that a simple modification of FTL, called FOLLOW THE APPROXIMATE LEADER (FTAL), guarantees logarithmic regret. The FOLLOW THE APPROXIMATE LEADER algorithm is given in two equivalent forms in Fig. 3. The first version is the FTL

FOLLOW THE APPROXIMATE LEADER (version 1)
Inputs: convex set $\mathcal{P} \subset \mathbb{R}^n$, and the parameter β.

- On period 1, play an arbitrary $\mathbf{x}_1 \in \mathcal{P}$.
- On period t, play the leader $\mathbf{x}_t$ defined as

$$\mathbf{x}_t \triangleq \arg\min_{\mathbf{x}\in\mathcal{P}} \sum_{\tau=1}^{t-1} \tilde{f}_\tau(\mathbf{x})$$

Where for $\tau = 1, \ldots, t-1$, define $\nabla_\tau = \nabla f_\tau(\mathbf{x}_\tau)$ and

$$\tilde{f}_\tau(\mathbf{x}) \triangleq f_\tau(\mathbf{x}_\tau) + \nabla_\tau^\top(\mathbf{x}-\mathbf{x}_\tau) + \frac{\beta}{2}(\mathbf{x}-\mathbf{x}_\tau)^\top \nabla_\tau \nabla_\tau^\top (\mathbf{x}-\mathbf{x}_\tau)$$

FOLLOW THE APPROXIMATE LEADER (version 2)
Inputs: convex set $\mathcal{P} \subset \mathbb{R}^n$, and the parameter β.

- On period 1, play an arbitrary $\mathbf{x}_1 \in \mathcal{P}$.
- On period $t > 1$: play the point $\mathbf{x}_t$ given by the following equations:

$$\begin{aligned}
\nabla_{t-1} &= \nabla f_{t-1}(\mathbf{x}_{t-1}) \\
\mathbf{A}_{t-1} &= \sum_{\tau=1}^{t-1} \nabla_\tau \nabla_\tau^\top \\
\mathbf{b}_{t-1} &= \sum_{\tau=1}^{t-1} \nabla_\tau \nabla_\tau^\top \mathbf{x}_\tau - \frac{1}{\beta}\nabla_\tau \\
\mathbf{x}_t &= \Pi_{\mathcal{P}}^{\mathbf{A}_{t-1}}\left(\mathbf{A}_{t-1}^{-1}\mathbf{b}_{t-1}\right)
\end{aligned}$$

Here, $\Pi_{\mathcal{P}}^{\mathbf{A}_{t-1}}$ is the projection in the norm induced by $\mathbf{A}_{t-1}$:

$$\Pi_{\mathcal{P}}^{\mathbf{A}_{t-1}}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{P}} (\mathbf{x}-\mathbf{y})^\top \mathbf{A}_{t-1}(\mathbf{x}-\mathbf{y})$$

$\mathbf{A}_{t-1}^{-1}$ denotes the Moore–Penrose pseudoinverse of $\mathbf{A}_{t-1}$.

Fig. 3 Two versions of the FOLLOW THE APPROXIMATE LEADER algorithm

variant, whereas the second version resembles the Newton method and the ONLINE NEWTON STEP algorithm. Lemma 4 below proves both versions to be equivalent, hence demonstrates the connection between the Newton method and FTL.

Lemma 4 *Both versions of the* FOLLOW THE APPROXIMATE LEADER *algorithm are equivalent.*

Proof In the first version of the FOLLOW THE APPROXIMATE LEADER algorithm, one needs to perform the following optimization at period t:

$$\mathbf{x}_t \triangleq \arg\min_{\mathbf{x}\in\mathcal{P}} \sum_{\tau=1}^{t-1} \tilde{f}_\tau(\mathbf{x}).$$

By expanding out the expressions for $\tilde{f}_\tau(\mathbf{x})$,

$$\begin{aligned}\sum_{\tau=1}^{t-1}\tilde{f}_\tau(\mathbf{x}) &= \sum_{\tau=1}^{t-1} f_\tau(\mathbf{x}_\tau)+\nabla_\tau^\top(\mathbf{x}-\mathbf{x}_\tau)+\frac{\beta}{2}(\mathbf{x}-\mathbf{x}_\tau)^\top\nabla_\tau\nabla_\tau^\top(\mathbf{x}-\mathbf{x}_\tau)\\ &= \sum_{\tau=1}^{t-1} f_\tau(\mathbf{x}_\tau)-(\beta\mathbf{x}_\tau^\top\nabla_\tau\nabla_\tau^\top-\nabla_\tau^\top)\mathbf{x}+\frac{\beta}{2}\mathbf{x}^\top\nabla_\tau\nabla_\tau^\top\mathbf{x}\\ &= \sum_{\tau=1}^{t-1} f_\tau(\mathbf{x}_\tau)-\beta\mathbf{b}_{t-1}^\top\mathbf{x}+\frac{\beta}{2}\mathbf{x}^\top\mathbf{A}_{t-1}\mathbf{x}.\end{aligned}$$

Therefore,

$$\begin{aligned}&\arg\min_{\mathbf{x}\in\mathcal{P}}\sum_{\tau=1}^{t-1}\tilde{f}_\tau(\mathbf{x})\\ &\quad=\arg\min_{\mathbf{x}\in\mathcal{P}}\left\{\frac{\beta}{2}\mathbf{x}^\top\mathbf{A}_{t-1}\mathbf{x}-\beta\mathbf{b}_{t-1}^\top\mathbf{x}\right\}\\ &\quad=\arg\min_{\mathbf{x}\in\mathcal{P}}\{\mathbf{x}^\top\mathbf{A}_{t-1}\mathbf{x}-2\mathbf{b}_{t-1}^\top\mathbf{x}\}\\ &\quad=\arg\min_{\mathbf{x}\in\mathcal{P}}\{(\mathbf{x}-\mathbf{A}_{t-1}^{-1}\mathbf{b}_{t-1})^\top\mathbf{A}_{t-1}(\mathbf{x}-\mathbf{A}_{t-1}^{-1}\mathbf{b}_{t-1})-\mathbf{b}_{t-1}^\top\mathbf{A}_{t-1}^{-1}\mathbf{b}_{t-1}\}.\end{aligned}$$

The solution of this minimization is exactly the projection $\Pi_{\mathcal{P}}^{\mathbf{A}_{t-1}}(\mathbf{A}_{t-1}^{-1}\mathbf{b}_{t-1})$ as specified by the second version. □

3.3.1 *Analysis of* FOLLOW THE APPROXIMATE LEADER

In this subsection we prove a performance guarantee for FOLLOW THE APPROXIMATE LEADER which is very similar to that for the ONLINE NEWTON STEP algorithm, albeit using a very different analysis. The analysis in this section is based on previous analyses of FOLLOW THE LEADER algorithms (Hannan 1957; Kalai and Vempala 2005). The standard approach to analyze such algorithms proceeds by inductively showing (see Lemma 10 in Appendix 1)

$$\text{Regret}_T(\text{FTL})=\sum_{t=1}^{T} f_t(\mathbf{x}_t)-\min_{\mathbf{x}\in\mathcal{P}}\sum_{t=1}^{T} f_t(\mathbf{x})\le\sum_{t=1}^{T}[f_t(\mathbf{x}_t)-f_t(\mathbf{x}_{t+1})]. \tag{7}$$

The standard analysis proceeds by showing that the leader doesn't change too much, i.e. $\mathbf{x}_t\approx\mathbf{x}_{t+1}$, which in turn implies low regret. Our analysis does not follow this paradigm directly, but rather shows *average stability* (i.e. that $\mathbf{x}_t\approx\mathbf{x}_{t+1}$ on the "average", rather than always).

Another building block, due to Zinkevich (2003), is that if we have another set of functions $\tilde{f}_t$ for which $\tilde{f}_t(\mathbf{x}_t)=f_t(\mathbf{x}_t)$ and $\tilde{f}_t$ is a lower-bound on f_t, so $\tilde{f}_t(\mathbf{x})\le f_t(\mathbf{x})$ for all $\mathbf{x}\in\mathcal{P}$, then it suffices to bound the regret with respect to $\tilde{f}_t$, because,

$$\text{Regret}_T=\sum_{t=1}^{T} f_t(\mathbf{x}_t)-\min_{\mathbf{x}\in\mathcal{P}}\sum_{t=1}^{T} f_t(\mathbf{x})\le\sum_{t=1}^{T}\tilde{f}_t(\mathbf{x}_t)-\min_{x\in\mathcal{P}}\sum_{t=1}^{T}\tilde{f}_t(\mathbf{x}). \tag{8}$$

We prove this fact in Lemma 9 in Appendix 1. Zinkevich uses this observation in conjunction with the fact that a convex function is lower-bounded by its tangent hyperplanes, to argue that it suffices to analyze online gradient descent for the case of linear functions.

We observe[4] that online gradient descent can be viewed as running FOLLOW THE LEADER on the sequence of functions $\tilde{f}_0(\mathbf{x}) = \|(\mathbf{x} - \mathbf{x}_1)\|^2/\eta$ and $\tilde{f}_t(\mathbf{x}) = f_t(\mathbf{x}_t) + \nabla f_t(\mathbf{x}_t)^\top(\mathbf{x} - \mathbf{x}_t)$. To do this, one need only calculate the minimum of $\sum_{\tau=0}^{t-1} \tilde{f}_\tau(\mathbf{x})$.

As explained before, any algorithm for the online convex optimization problem with linear functions has $\Omega(\sqrt{T})$ regret, and thus to achieve logarithmic regret one necessarily needs to use the curvature of functions. When we consider H-strong convex functions for some $H > 0$, we can lower bound the function f_t by a paraboloid,

$$f_t(\mathbf{x}) \geq f_t(\mathbf{x}_t) + \nabla f_t(\mathbf{x}_t)^\top(\mathbf{x} - \mathbf{x}_t) + \frac{H}{2}\|\mathbf{x} - \mathbf{x}_t\|^2,$$

rather than a linear function. The FOLLOW THE LEADER calculation, however, remains similar. The only difference is that the step-size $\eta_t = 1/(Ht)$ decreases linearly rather than as $O(1/\sqrt{t})$.

For α-exp-concave functions, Lemma 3 shows that they can be lower-bounded by a paraboloid $\tilde{f}_t(\mathbf{x}) = a + (\mathbf{v}^\top\mathbf{x} - b)^2$ where $\mathbf{v} \in \mathbb{R}^n$ is a multiple of $\nabla f_t(\mathbf{x}_t)$ and $a, b \in \mathbb{R}$.

The main technical step now is to show that FOLLOW THE LEADER, when run on a specific class of cost functions, which includes the paraboloid functions given above, has regret $O(\log T)$. This is the content of the following theorem, which is interesting in its own right because it also applies to the portfolio management problem, and shows that the simple FOLLOW THE LEADER strategy for the portfolio management problem achieves $O(\log T)$ regret.

Theorem 5 *Assume that for all t, the function $f_t : \mathcal{P} \to \mathbb{R}^n$ can be written as $f_t(\mathbf{x}) = g_t(\mathbf{v}_t^\top\mathbf{x})$ for a univariate convex function $g_t : \mathbb{R} \to \mathbb{R}$ and some vector $\mathbf{v}_t \in \mathbb{R}^n$. Assume that for some $R, a, b > 0$, we have $\|\mathbf{v}_t\|_2 \leq R$, and for all $\mathbf{x} \in \mathcal{P}$, we have $|g_t'(\mathbf{v}_t^\top\mathbf{x})| \leq b$ and $g_t''(\mathbf{v}_t^\top\mathbf{x}) \geq a$. Then the* FOLLOW THE LEADER *algorithm on the functions f_t satisfies the following regret bound*:

$$\text{Regret}_T(\text{FTL}) \leq \frac{2nb^2}{a}\left[\log\left(\frac{DRaT}{b}\right) + 1\right].$$

Before proving this Theorem, we show how it implies our main result concerning the FOLLOW THE APPROXIMATE LEADER algorithm.

Theorem 6 *Assume that for all t, the function $f_t : \mathcal{P} \to \mathbb{R}^n$ has the property that $\forall \mathbf{x} \in \mathcal{P}$, $\|\nabla f(\mathbf{x})\| \leq G$ and $\exp(-\alpha f(\mathbf{x}))$ is concave. Then the algorithm* FOLLOW THE APPROXIMATE LEADER *with $\beta = \frac{1}{2}\min\{\frac{1}{4GD}, \alpha\}$ has the following regret bound*:

$$\text{Regret}_T(\text{FTAL}) \leq 64\left(\frac{1}{\alpha} + GD\right)n(\log(T) + 1).$$

Proof We note that FOLLOW THE APPROXIMATE LEADER is just doing FOLLOW THE LEADER on the paraboloid functions $\tilde{f}_t$. By Lemma 3, we have $f_t(\mathbf{x}_t) = \tilde{f}_t(\mathbf{x}_t)$ and for all $\mathbf{x} \in \mathcal{P}$, $f_t(\mathbf{x}) \geq \tilde{f}_t(\mathbf{x})$. Thus, Lemma 9 (Appendix 1) implies that the regret assuming the

[4] Kakade has made a similar observation (Kakade 2005).

cost functions are $\tilde{f}_t$ rather than f_t is only greater, so it suffices to bound the regret with cost functions $\tilde{f}_t$. The function $\tilde{f}_t$ can be written as

$$\tilde{f}_t(\mathbf{x}) = f_t(\mathbf{x}_t) + \nabla_\tau^\top(\mathbf{x} - \mathbf{x}_\tau) + \frac{\beta}{2}[\nabla_\tau^\top(\mathbf{x} - \mathbf{x}_\tau)]^2$$

and thus satisfies the conditions of Theorem 5 with $g_t : \mathbb{R} \to \mathbb{R}$ defined as

$$g_t(y) \triangleq f_t(\mathbf{x}_t) + (y - \nabla_t^\top \mathbf{x}_t) + \frac{\beta}{2}(y - \nabla_t^\top \mathbf{x}_t)^2 \quad \text{and} \quad \mathbf{v}_t = \nabla_t.$$

We only need to evaluate the constants R, a, b. Note that $\|\mathbf{v}_t\| = \|\nabla_t\| \le G$, so we can take $R = G$. Next, $|g_t'(\mathbf{v}_t^\top \mathbf{x})| = |1 + \beta(\nabla_t^\top(\mathbf{x} - \mathbf{x}_t))| \le 1 + \beta GD \le 2$ since $\beta \le \frac{1}{8GD}$, so we can take $b = 2$. Finally, $g_t''(y) = \beta$, so we can take $a = \beta$. Plugging in the values, and using the fact that $\frac{DRa}{b} = \frac{\beta GD}{2} \le 1$, we get that

$$\text{Regret}_T(\text{FTAL}) \le \frac{8n}{\beta}(\log(T) + 1).$$

Finally, the stated regret bound follows because by definition $\frac{1}{\beta} \le \max\{8GD, \frac{2}{\alpha}\} \le 8(GD + \frac{1}{\alpha})$. □

Finally, we prove Theorem 5:

Proof of Theorem 5 Lemma 10 (Appendix 1) implies that the regret can be bounded as

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x}) \le \sum_{t=1}^{T} [f_t(\mathbf{x}_t) - f_t(\mathbf{x}_{t+1})].$$

We bound the RHS now.

For the sake of readability, we introduce some notation. Define the function $F_t \triangleq \sum_{\tau=1}^{t-1} f_\tau$. Let Δ be the forward difference operator, for example, $\Delta\mathbf{x}_t = (\mathbf{x}_{t+1} - \mathbf{x}_t)$ and $\Delta\nabla F_t(\mathbf{x}_t) = (\nabla F_{t+1}(\mathbf{x}_{t+1}) - \nabla F_t(\mathbf{x}_t))$.

Firstly, observe that for all τ, the gradient $\nabla f_\tau(\mathbf{x}) = g_\tau'(\mathbf{v}_\tau^\top \mathbf{x})\mathbf{v}_\tau$. Define $\nabla_t = \nabla f_t(\mathbf{x}_t) = g_t'(\mathbf{v}_t^\top \mathbf{x}_t)\mathbf{v}_t$. We use the gradient bound, which follows from the convexity of f_t:

$$f_t(\mathbf{x}_t) - f_t(\mathbf{x}_{t+1}) \le -\nabla f_t(\mathbf{x}_t)^\top(\mathbf{x}_{t+1} - \mathbf{x}_t) = -\nabla_t^\top \Delta\mathbf{x}_t. \tag{9}$$

Now, we have

$$\begin{aligned}
\nabla F_{t+1}(\mathbf{x}_{t+1}) - \nabla F_{t+1}(\mathbf{x}_t) &= \sum_{\tau=1}^{t} \nabla f_\tau(\mathbf{x}_{t+1}) - \nabla f_\tau(\mathbf{x}_t) && (10)\\
&= \sum_{\tau=1}^{t} [g_\tau'(\mathbf{v}_\tau^\top \mathbf{x}_{t+1}) - g_\tau'(\mathbf{v}_\tau^\top \mathbf{x}_t)]\mathbf{v}_\tau \\
&= \sum_{\tau=1}^{t} [\nabla g_\tau'(\mathbf{v}_\tau^\top \zeta_\tau^t)^\top(\mathbf{x}_{t+1} - \mathbf{x}_t)]\mathbf{v}_\tau && (11)\\
&= \sum_{\tau=1}^{t} g_\tau''(\mathbf{v}_\tau^\top \zeta_\tau^t)\mathbf{v}_\tau\mathbf{v}_\tau^\top(\mathbf{x}_{t+1} - \mathbf{x}_t). && (12)
\end{aligned}$$

Equation (11) follows by applying the Taylor expansion of the (multi-variate) function $g'_\tau(\mathbf{v}_\tau^\top \mathbf{x})$ at point $\mathbf{x}_t$, for some point ζ_τ^t on the line segment joining $\mathbf{x}_t$ and $\mathbf{x}_{t+1}$. Equation (12) follows from the observation that $\nabla g'_\tau(\mathbf{v}_\tau^\top \mathbf{x}) = g''_\tau(\mathbf{v}_\tau^\top \mathbf{x})\mathbf{v}_\tau$. Define $\mathbf{A}_t = \sum_{\tau=1}^{t} g''_\tau(\mathbf{v}_\tau^\top \zeta_\tau^t)\mathbf{v}_\tau \mathbf{v}_\tau^\top$. Since $g''_\tau(\mathbf{v}_\tau^\top \mathbf{x}_\tau) \geq a$ for all t, $\mathbf{A}_t$ is positive semidefinite. The RHS of (12) becomes $\mathbf{A}_t \Delta\mathbf{x}_t$. The LHS of (10) is

$$\nabla F_{t+1}(\mathbf{x}_{t+1}) - \nabla F_{t+1}(\mathbf{x}_t) = \Delta\nabla F_t(\mathbf{x}_t) - \nabla_t. \tag{13}$$

Putting (12) and (13) together, and by adding $\varepsilon\Delta\mathbf{x}_t$ we get

$$(\mathbf{A}_t + \varepsilon I_n)\Delta\mathbf{x}_t = \Delta\nabla F_t(\mathbf{x}_t) - \nabla_t + \varepsilon\Delta\mathbf{x}_t. \tag{14}$$

Pre-multiplying by $-\nabla_t^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}$, we get an expression for the gradient bound (9):

$$\begin{aligned} -\nabla_t^\top \Delta\mathbf{x}_t &= -\nabla_t^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}[\Delta\nabla F_t(\mathbf{x}_t) - \nabla_t + \varepsilon\Delta\mathbf{x}_t] \\ &= -\nabla_t^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}[\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t] + \nabla_t^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}\nabla_t. \end{aligned} \tag{15}$$

Thus, from (15) and (9) we have

$$\begin{aligned} &\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}_{t+1})] \\ &\quad\leq \sum_{t=1}^{T} -\nabla_t^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}[\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t] + \sum_{t=1}^{T}\nabla_t^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}\nabla_t. \end{aligned} \tag{16}$$

□

Claim 1 *The first term of* (16) *can be bounded as*

$$\sum_{t=1}^{T} -\nabla_t^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}[\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t] \leq \varepsilon D^2 T.$$

Proof We bound each term in the sum by εD^2. Since $\mathbf{x}_\tau$ minimizes F_τ over $\mathcal{P}$, we have (see (Boyd and Vandenberghe 2004))

$$\nabla F_\tau(\mathbf{x}_\tau)^\top(\mathbf{x} - \mathbf{x}_\tau) \;\geq\; 0 \tag{17}$$

for any point $\mathbf{x} \in \mathcal{P}$. Using (17) for $\tau = t$ and $\tau = t+1$, we get

$$0 \leq \nabla F_{t+1}(\mathbf{x}_{t+1})^\top(\mathbf{x}_t - \mathbf{x}_{t+1}) + \nabla F_t(\mathbf{x}_t)^\top(\mathbf{x}_{t+1} - \mathbf{x}_t) = -[\Delta\nabla F_t(\mathbf{x}_t)]^\top\Delta\mathbf{x}_t.$$

Reversing the inequality and adding $\varepsilon\|\Delta\mathbf{x}_t\|^2 = \varepsilon\Delta\mathbf{x}_t^\top\Delta\mathbf{x}_t$, we get

$$\begin{aligned} \varepsilon\|\Delta\mathbf{x}_t\|^2 &\geq [\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t]^\top\Delta\mathbf{x}_t \\ &= [\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t]^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}[\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t - \nabla_t] \\ &\qquad \text{(by solving for } \Delta\mathbf{x}_t \text{ in (14))} \\ &= [\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t]^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}(\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t) \\ &\quad - [\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t]^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}\nabla_t \end{aligned}$$

$$\geq -[\Delta\nabla F_t(\mathbf{x}_t) + \varepsilon\Delta\mathbf{x}_t]^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}\nabla_t$$
$$\text{(since } (\mathbf{A}_t + \varepsilon I_n)^{-1} \text{ is positive semidefinite).}$$

Finally, since the diameter of $\mathcal{P}$ is D, we have $\varepsilon\|\Delta\mathbf{x}_t\|^2 \leq \varepsilon D^2$. □

Claim 2 *The second term of* (16) *can be bounded as*

$$\sum_{t=1}^{T} \nabla_t^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}\nabla_t \leq \frac{nb^2}{a}\log(aR^2T/\varepsilon + 1).$$

Proof Now we bound the second term of (16). Define $\mathbf{B}_t = \sum_{\tau=1}^{t} a\mathbf{v}_t\mathbf{v}_t^\top$. Since for all τ, $g_\tau''(\mathbf{v}_\tau^\top\mathbf{x}_\tau) \geq a$, we have that $\mathbf{A}_t + \varepsilon I_n \succeq \mathbf{B}_t + \varepsilon I_n$, which implies that $(\mathbf{A}_t + \varepsilon I_n)^{-1} \preceq (\mathbf{B}_t + \varepsilon I_n)^{-1}$ and hence

$$\begin{aligned}\nabla_t^\top(\mathbf{A}_t + \varepsilon I_n)^{-1}\nabla_t &\leq \nabla_t^\top(\mathbf{B}_t + \varepsilon I_n)^{-1}\nabla_t \\ &= [g_t'(\mathbf{v}_t^\top\mathbf{x}_t)\mathbf{v}_t]^\top(\mathbf{B}_t + \varepsilon I_n)^{-1}[g_t'(\mathbf{v}_t^\top\mathbf{x}_t)\mathbf{v}_t] \\ &\leq \frac{b^2}{a}[\sqrt{a}\mathbf{v}_t]^\top(\mathbf{B}_t + \varepsilon I_n)^{-1}[\sqrt{a}\mathbf{v}_t].\end{aligned}$$

Sum up from $t = 1$ to T, and apply Lemma 11 (as used in the previous subsection, see Appendix 2 for statement and proof) with $\mathbf{V}_t = \mathbf{B}_t + \varepsilon I_n$, $u_t = \sqrt{a}\mathbf{v}_t$, and $r = \sqrt{a}R$, to get the stated bound.

Combining the two bounds from the claims above, and setting $\varepsilon = \frac{b^2}{aD^2T}$ we get

$$\begin{aligned}\sum_{t=1}^{T}[f_t(\mathbf{x}_t) - f_t(\mathbf{x}_{t+1})] &\leq \frac{nb^2}{a}\log(aR^2T/\varepsilon + 1) + \varepsilon D^2T \\ &\leq \frac{2nb^2}{a}\left[\log\left(\frac{DRaT}{b}\right) + 1\right]\end{aligned}$$

as required. □

3.3.2 *Implementation and running time*

The implementation of FOLLOW THE APPROXIMATE LEADER is straightforward: the point $\mathbf{x}_t$ chosen at iteration t is the optimum of the following mathematical program:

$$\mathbf{x}_t = \arg\min_{\mathbf{x}\in\mathcal{P}} \sum_{\tau=1}^{t-1} \tilde{f}_\tau(\mathbf{x}).$$

Since the approximate cost functions $\tilde{f}_t$ as well as the underlying set $\mathcal{P}$ are convex, this is a convex program which any general convex optimization algorithm applied to (here is another justification for our assumption that the set $\mathcal{P}$ is convex, see Sect. 2). An efficient implementation of the algorithm is FOLLOW THE APPROXIMATE LEADER (version 2). This uses the fact that all $\tilde{f}_t$ are quadratic polynomials, and maintains the sum of the coefficients of these polynomials. The algorithm requires $\tilde{O}(n^2)$ space to store the sum of all gradients and matrices of the form $\nabla_t\nabla_t^\top$. The time needed to compute the point $\mathbf{x}_t$ is $O(n^2)$ plus the time to perform a single generalized projection. This is because a generalized matrix inversion lemma (Riedel 1991) allows for iterative update of the pseudoinverse in $O(n^2)$ time.

EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION.
Inputs: convex set $\mathcal{P} \subset \mathbb{R}^n$, and the parameter α.

- Define weights $w_t(\mathbf{x}) = \exp(-\alpha \sum_{\tau=1}^{t-1} f_\tau(\mathbf{x}))$.
- On period t play $\mathbf{x}_t = \frac{\int_{\mathcal{P}} \mathbf{x} w_t(\mathbf{x}) d\mathbf{x}}{\int_{\mathcal{P}} w_t(\mathbf{x}) d\mathbf{x}}$.

  (*Remark:* choosing $\mathbf{x}_t$ at random with density proportional to $w_t(\mathbf{x})$ also gives our bounds.)

Fig. 4 The EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION algorithm

The time and space complexity is thus independent from the number of iterations, in contrast to other previous variants of FOLLOW THE LEADER.

We note that in practice, we have an excellent starting point to compute $\mathbf{x}_t$—the optimum of the convex program of the previous iteration $\mathbf{x}_{t-1}$. As shown in the analysis, on the average these two consecutive points are very close.

3.4 EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION

In this subsection we describe our EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION (EWOO) algorithm which gives logarithmic regret for a very general setting of online convex optimization. All that the algorithm requires is that the cost functions be α-exp-concave (ONLINE NEWTON STEP and FOLLOW THE APPROXIMATE LEADER need additionally a bound on the magnitude of the gradients). The algorithm does not seem to be directly related to FOLLOW THE LEADER. Rather, it is related to Cover's algorithm for universal portfolio management.

The downside of this algorithm is its running time. A trivial implementation of EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION would give exponential running time. Kalai and Vempala (2003) give a randomized polynomial time (polynomial both in n and in T) implementation of Cover's algorithm, based on random sampling techniques. The same techniques can be applied to the EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION algorithm as well. However, the polynomial in the running time is quite large and the overall implementation involved.

Remark In the implementation of EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION, choosing $\mathbf{x}_t$ at random with density proportional to $w_t(\mathbf{x})$, instead of computing the integral, also guarantees our regret bounds on the expectation. This is the basis for the (Kalai and Vempala 2003) polynomial time implementation.

Theorem 7 *Assume that for all t, the function $f_t : \mathcal{P} \to \mathbb{R}^n$ has the property that $\forall \mathbf{x} \in \mathcal{P}$, $\exp(-\alpha f(\mathbf{x}))$ is concave. Then the algorithm* EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION *has the following regret bound*:

$$\text{Regret}_T(\text{EWOO}) \leq \frac{1}{\alpha} n(1 + \log(T+1)).$$

Proof Let $\mathbf{x}^* \in \arg\min_{\mathbf{x} \in \mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x})$. Recall the definition of regret (see Sect. 2)

$$\text{Regret}_T(\text{EWOO}) = \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^*).$$

Let $h_t(\mathbf{x}) = e^{-\alpha f_t(\mathbf{x})}$. The algorithm can be viewed as taking a weighted average over points $\mathbf{x} \in \mathcal{P}$. Hence, by concavity of h_t,

$$h_t(\mathbf{x}_t) \geq \frac{\int_{\mathcal{P}} h_t(\mathbf{x}) \prod_{\tau=1}^{t-1} h_\tau(\mathbf{x})\, d\mathbf{x}}{\int_{\mathcal{P}} \prod_{\tau=1}^{t-1} h_\tau(\mathbf{x})\, d\mathbf{x}}.$$

Hence, we have by telescoping product,

$$\prod_{\tau=1}^{t} h_\tau(\mathbf{x}_\tau) \geq \frac{\int_{\mathcal{P}} \prod_{\tau=1}^{t} h_\tau(\mathbf{x})\, d\mathbf{x}}{\int_{\mathcal{P}} 1\, d\mathbf{x}} = \frac{\int_{\mathcal{P}} \prod_{\tau=1}^{t} h_\tau(\mathbf{x})\, d\mathbf{x}}{\mathrm{vol}(\mathcal{P})}. \tag{18}$$

By definition of $\mathbf{x}^*$ we have $\mathbf{x}^* \in \arg\max_{\mathbf{x}\in\mathcal{P}} \prod_{t=1}^{T} h_t(\mathbf{x})$. Following (Blum and Kalai 1997), define nearby points $S \subset \mathcal{P}$ by,

$$S = \left\{ \mathbf{x} \in S \;\middle|\; \mathbf{x} = \frac{T}{T+1}\mathbf{x}^* + \frac{1}{T+1}\mathbf{y},\ \mathbf{y} \in \mathcal{P} \right\}.$$

By concavity of h_t and the fact that h_t is non-negative, we have that,

$$\forall \mathbf{x} \in S \quad h_t(\mathbf{x}) \geq \frac{T}{T+1} h_t(\mathbf{x}^*).$$

Hence,

$$\forall \mathbf{x} \in S\text{:} \quad \prod_{\tau=1}^{T} h_\tau(\mathbf{x}) \geq \left(\frac{T}{T+1}\right)^T \prod_{\tau=1}^{T} h_\tau(\mathbf{x}^*) \geq \frac{1}{e} \prod_{\tau=1}^{T} h_\tau(\mathbf{x}^*).$$

Finally, since $S = \mathbf{x}^* + \frac{1}{T+1}\mathcal{P}$ is simply a rescaling of $\mathcal{P}$ by a factor of $1/(T+1)$ (followed by a translation), and we are in n dimensions, $\mathrm{vol}(S) = \mathrm{vol}(\mathcal{P})/(T+1)^n$. Putting this together with (18), we have

$$\prod_{\tau=1}^{T} h_\tau(\mathbf{x}_\tau) \geq \frac{\mathrm{vol}(S)}{\mathrm{vol}(\mathcal{P})} \frac{1}{e} \prod_{\tau=1}^{T} h_\tau(\mathbf{x}^*) \geq \frac{1}{e(T+1)^n} \prod_{\tau=1}^{T} h_\tau(\mathbf{x}^*).$$

The theorem is obtained by taking logarithms. □

3.4.1 Implementation and running time

The EXPONENTIALLY WEIGHTED ONLINE OPTIMIZATION algorithm can be approximated by sampling points according to the distribution with density proportional to w_t and then taking their mean. In fact, as far as an expected guarantee is concerned, our analysis actually shows that the algorithm which chooses a single random point x_t with density proportional to $w_t(x)$ achieves the stated regret bound, in expectation. Using recent random walk analyses of Lovász and Vempala (2003a, 2003b), m samples from such a distribution can be computed in time $\tilde{O}((n^4 + mn^3) \log \frac{R}{r})$. A similar application of random walks was used previously for an efficient implementation of Cover's Universal Portfolio algorithm (Kalai and Vempala 2003).

4 Computing projections

Some of the algorithms for online convex optimization described in this paper require computing projections onto convex sets. This corresponds to the following computational problem: given a convex set $\mathcal{P} \subseteq \mathbb{R}^n$, and a point $\mathbf{y} \in \mathbb{R}^n$, find the point in the convex set which is closest in Euclidean distance to the given vector, denoted

$$\Pi_{\mathcal{P}}[\mathbf{y}] \triangleq \min_{\mathbf{x}\in\mathcal{P}} \|\mathbf{x}-\mathbf{y}\|_2.$$

The ONLINE NEWTON STEP algorithm computes *generalized projections*, which are projections with respect to a norm other than the Euclidean norm, given by a positive semidefinite matrix. For a given positive semidefinite matrix $\mathbf{A}$, a generalized projection of $\mathbf{y} \in \mathbb{R}^n$ onto the convex set $\mathcal{P}$ is defined as

$$\Pi^{\mathbf{A}}_{\mathcal{P}}[\mathbf{y}] \triangleq \min_{\mathbf{x}\in\mathcal{P}} (\mathbf{x}-\mathbf{y})^\top \mathbf{A}(\mathbf{x}-\mathbf{y}).$$

Thus, the Euclidean projection can be seen to be a generalized projection with $\mathbf{A} = \mathbf{I}_n$. These projections satisfy the following well known fact:

Lemma 8 (folklore) *Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a convex set, $\mathbf{y} \in \mathbb{R}^n$ and $\mathbf{z} = \Pi^{\mathbf{A}}_{\mathcal{P}}[\mathbf{y}]$ be the generalized projection of $\mathbf{y}$ onto $\mathcal{P}$ according to positive semidefinite matrix $A \succeq 0$. Then for any point $\mathbf{a} \in \mathcal{P}$ it holds that*

$$(\mathbf{y}-\mathbf{a})^\top \mathbf{A}(\mathbf{y}-\mathbf{a}) \;\geq\; (\mathbf{z}-\mathbf{a})^\top \mathbf{A}(\mathbf{z}-\mathbf{a}).$$

If $\mathbf{A}$ is the identity matrix, this lemma is standard and follows from the fact that for any $\mathbf{a} \in \mathcal{P}$ the angle $\measuredangle(\mathbf{y}, \Pi_{\mathcal{P}}[\mathbf{y}], \mathbf{a})$ is obtuse. The latter is implied by the fact that for any point outside a convex body there exists a hyperplane which separates it from all points on the convex set.

For a general positive semidefinite matrix $\mathbf{A}$, the lemma can be proved by reduction to the simple case, as $\mathbf{A}$ generates a natural norm:[5] $\forall \mathbf{x} \in \mathbb{R}^n,\ \|\mathbf{x}\|_{\mathbf{A}} = \mathbf{x}^\top \mathbf{A}\mathbf{x}$. We include a proof for completeness.

Proof By the definition of generalized projections, the point $\mathbf{z}$ minimizes the function $f(\mathbf{x}) = (\mathbf{x}-\mathbf{y})^\top \mathbf{A}(\mathbf{x}-\mathbf{y})$ over the convex set. It is a well known fact in optimization (see Boyd and Vandenberghe 2004) that for the optimum $\mathbf{z}$ the following holds

$$\forall \mathbf{a} \in \mathcal{P}: \quad \nabla f(\mathbf{z})^\top (\mathbf{a}-\mathbf{z}) \geq 0$$

which implies

$$2(\mathbf{z}-\mathbf{y})^\top \mathbf{A}(\mathbf{a}-\mathbf{z}) \geq 0 \;\Rightarrow\; 2\mathbf{a}^\top \mathbf{A}(\mathbf{z}-\mathbf{y}) \geq 2\mathbf{z}^\top \mathbf{A}(\mathbf{z}-\mathbf{y}).$$

Now by simple calculation:

$$\begin{aligned}(\mathbf{y}-\mathbf{a})^\top \mathbf{A}(\mathbf{y}-\mathbf{a}) - (\mathbf{z}-\mathbf{a})^\top \mathbf{A}(\mathbf{z}-\mathbf{a}) &= \mathbf{y}^\top \mathbf{A}\mathbf{y} - \mathbf{z}^\top \mathbf{A}\mathbf{z} + 2\mathbf{a}^\top \mathbf{A}(\mathbf{z}-\mathbf{y}) \\ &\geq \mathbf{y}^\top \mathbf{A}\mathbf{y} - \mathbf{z}^\top \mathbf{A}\mathbf{z} + 2\mathbf{z}^\top \mathbf{A}(\mathbf{z}-\mathbf{y})\end{aligned}$$

[5] Note that because $\mathbf{A}$ can be singular, the norm may not be definite.

$$= \mathbf{y}^\top \mathbf{A}\mathbf{y} - 2\mathbf{z}^\top \mathbf{A}\mathbf{y} + \mathbf{z}^\top \mathbf{A}\mathbf{z}$$
$$= (\mathbf{y}-\mathbf{z})^\top \mathbf{A}(\mathbf{y}-\mathbf{z}) \geq 0.$$

The final inequality follows because $\mathbf{A} \succeq \mathbf{0}$. □

These projections are essentially convex programs. For convex polytopes, a projection reduces to a convex quadratic program with linear constraints. These type of convex programs can be solved more efficiently than general convex programs using interior point methods (Lobo et al. 1998). Another option is to efficiently approximate these convex programs using Lagrangian relaxation techniques (Hazan 2006).

Even more generally, $\mathcal{P}$ can be specified by a membership oracle $\chi_{\mathcal{P}}$, such that $\chi_{\mathcal{P}}(\mathbf{x}) = 1$ if $\mathbf{x} \in \mathcal{P}$ and 0 if $\mathbf{x} \notin \mathcal{P}$, along with a point $\mathbf{x}_0 \in \mathcal{P}$ as well as radii $R \geq r > 0$ such that the balls of radii R and r around $\mathbf{x}_0$ contain and are contained in $\mathcal{P}$, respectively. In this case $\Pi_{\mathcal{P}}^{\mathbf{A}}$ can be computed (to ε accuracy) in time $\tilde{O}(n^4 \log(\frac{R}{r}))$ using Vaidya's algorithm (Vaidya 1996).

However, for many simple convex bodies which arise in practical applications (e.g. portfolio management), projections can be computed much more efficiently. For the n-dimensional unit sphere, cube and the simplex these projections can be computed combinatorially in $\tilde{O}(n)$ time, rendering the online algorithms much more efficient when applied to these convex bodies (see Hazan 2006).

5 Conclusions

In this work, we presented efficient algorithms which guarantee logarithmic regret when the loss functions satisfy a mildly restrictive convexity condition. Perhaps the most interesting algorithm we describe is based on the Newton method from offline optimization. The intuition leading to this algorithm stems from new observations regarding the very natural follow-the-leader methodology, and answers open problems regarding this method.

Acknowledgements Many thanks to Adam Kalai for many critical contributions to this research. We would also like to thank Sanjeev Arora, Rob Schapire and an anonymous referee for helpful comments.

Appendix 1 Reductions for FOLLOW THE LEADER

Lemma 9 *Let f_t, for $t = 1, \ldots, T$, be a sequence of cost functions and let $\mathbf{x}_t \in \mathcal{P}$ be the point used in the t^{th} round. Let $\tilde{f}_t$ for $t = 1, \ldots, T$ be a sequence of cost functions such that $f_t(\mathbf{x}_t) = \tilde{f}_t(\mathbf{x}_t)$, and for all $\mathbf{x} \in \mathcal{P}$, $f_t(\mathbf{x}) \geq \tilde{f}_t(\mathbf{x})$. Then*

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x}) \leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{P}} \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}).$$

Proof Let $\mathbf{x}^* \in \arg\min_{\mathbf{x}\in\mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x})$. We have

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}^*) \leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}^*)$$
$$\leq \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}_t) - \min_{\mathbf{x}\in\mathcal{P}} \sum_{t=1}^{T} \tilde{f}_t(\mathbf{x}). \tag{19}$$

□

Lemma 10 *Let f_t, for $t = 1, \ldots, T$, be a sequence of cost functions and let* $\mathbf{x}_t = \arg\min_{x \in \mathcal{P}} \sum_{\tau=1}^{t} f_\tau(\mathbf{x})$. *Then*

$$\sum_{t=1}^{T} f_t(\mathbf{x}_t) - \min_{\mathbf{x} \in \mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x}) \leq \sum_{t=1}^{T} f_t(\mathbf{x}_t) - \sum_{t=1}^{T} f_t(\mathbf{x}_{t+1}).$$

Proof We prove inductively that

$$\sum_{t=1}^{T} f_t(\mathbf{x}_{t+1}) \leq \min_{\mathbf{x} \in \mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x}).$$

For $T = 1$ the two are equal by definition. Assume correctness for $T - 1$, and

$$\begin{aligned}
\sum_{t=1}^{T} f_t(\mathbf{x}_{t+1}) &\leq \min_{\mathbf{x} \in \mathcal{P}} \sum_{t=1}^{T-1} f_t(\mathbf{x}) + f_T(\mathbf{x}_{T+1}) \quad \text{by induction hypothesis} \\
&\leq \sum_{t=1}^{T-1} f_t(\mathbf{x}_{T+1}) + f_T(\mathbf{x}_{T+1}) \\
&= \min_{\mathbf{x} \in \mathcal{P}} \sum_{t=1}^{T} f_t(\mathbf{x}) \quad \text{by definition.}
\end{aligned}$$

Thus, the induction is complete. □

Appendix 2 Bounds on the potential function

The following two technical lemmas, which concern general facts from linear algebra, were used to bound the potential function used in the analysis of the ONLINE NEWTON STEP algorithm (as well as in the analysis of FOLLOW THE LEADER).

Lemma 11 *Let $\mathbf{u}_t \in \mathbb{R}^n$, for $t = 1, \ldots, T$, be a sequence of vectors such that for some $r > 0$, $\|\mathbf{u}_t\| \leq r$. Define $\mathbf{V}_t = \sum_{\tau=1}^{t} \mathbf{u}_t \mathbf{u}_t^\top + \varepsilon I_n$. Then*

$$\sum_{t=1}^{T} \mathbf{u}_t^\top \mathbf{V}_t^{-1} \mathbf{u}_t \leq n \log(r^2 T/\varepsilon + 1).$$

Proof For real numbers $a > b > 0$, the inequality $1 + x \leq e^x$ implies that $\frac{1}{a} \cdot (a - b) \leq \log \frac{a}{b}$ (taking $x = \frac{b}{a} - 1$). An analogous fact holds for positive definite matrices. Define, for matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times n}$ the product $\mathbf{A} \bullet \mathbf{B} = \sum_{i,j=1}^{n} \mathbf{A}_{ij} \mathbf{B}_{ij}$ (this is just the standard inner product when thinking of the matrices as vectors in $\mathbb{R}^{n^2}$). Then, for matrices $\mathbf{A} \succeq \mathbf{B} \succ \mathbf{0}$, $\mathbf{A}^{-1} \bullet (\mathbf{A} - \mathbf{B}) \leq \log \frac{|\mathbf{A}|}{|\mathbf{B}|}$, where $|\mathbf{A}|$ is the determinant of $\mathbf{A}$.[6] This is proved in Lemma 12.

[6] Recall our notation that $A \succeq B$ if the matrix $A - B \succeq 0$ is positive semi-definite.

Using this fact we have (for convenience, let $\mathbf{V}_0 = \varepsilon I_n$)

$$\begin{aligned}\sum_{t=1}^{T} \mathbf{u}_t^\top \mathbf{V}_t^{-1} \mathbf{u}_t &= \sum_{t=1}^{T} \mathbf{V}_t^{-1} \bullet \mathbf{u}_t \mathbf{u}_t^\top \\ &= \sum_{t=1}^{T} \mathbf{V}_t^{-1} \bullet (\mathbf{V}_t - \mathbf{V}_{t-1}) \\ &\leq \sum_{t=1}^{T} \log \frac{|\mathbf{V}_t|}{|\mathbf{V}_{t-1}|} = \log \frac{|\mathbf{V}_T|}{|\mathbf{V}_0|}.\end{aligned}$$

Since $\mathbf{V}_T = \sum_{t=1}^{T} \mathbf{u}_t \mathbf{u}_t^\top + \varepsilon I$ and $\|\mathbf{u}_t\| \leq r$, the largest eigenvalue of $\mathbf{V}_T$ is at most $r^2 T + \varepsilon$. Hence the determinant of $\mathbf{V}_T$ can be bounded by $|\mathbf{V}_T| \leq (r^2 T + \varepsilon)^n$. The stated bound in the lemma follows. □

Lemma 12 *Let $\mathbf{A} \succeq \mathbf{B} \succ 0$ be positive definite matrices. Then*

$$\mathbf{A}^{-1} \bullet (\mathbf{A} - \mathbf{B}) \leq \log \frac{|\mathbf{A}|}{|\mathbf{B}|}$$

where $|\mathbf{A}|$ denotes the determinant of matrix $\mathbf{A}$.

Proof For any positive definite matrix $\mathbf{C}$, denote by $\lambda_1(\mathbf{C}), \lambda_2(\mathbf{C}), \ldots, \lambda_n(\mathbf{C})$ its (positive) eigenvalues. Denote by $\mathbf{Tr}(C)$ the trace of the matrix, which is equal to the sum of the diagonal entries of C, and also to the sum of its eigenvalues.

Note that for the matrix product $A \bullet B = \sum_{i,j=1}^{n} \mathbf{A}_{ij} \mathbf{B}_{ij}$ defined earlier, we have $A \bullet B = \mathbf{Tr}(AB)$ (where AB is the standard matrix multiplication), since the trace is equal to the sum of the diagonal entries. Therefore,

$$\begin{aligned}\mathbf{A}^{-1} \bullet (\mathbf{A} - \mathbf{B}) &= \mathbf{Tr}(\mathbf{A}^{-1}(\mathbf{A} - \mathbf{B})) \\ &= \mathbf{Tr}(\mathbf{A}^{-1/2}(\mathbf{A} - \mathbf{B})\mathbf{A}^{-1/2}) \\ &= \mathbf{Tr}(I - \mathbf{A}^{-1/2}\mathbf{B}\mathbf{A}^{-1/2}) \\ &= \sum_{i=1}^{n} [1 - \lambda_i(\mathbf{A}^{-1/2}\mathbf{B}\mathbf{A}^{-1/2})] \quad \because \mathbf{Tr}(\mathbf{C}) = \sum_{i=1}^{n} \lambda_i(\mathbf{C}) \\ &\leq -\sum_{i=1}^{n} \log[\lambda_i(\mathbf{A}^{-1/2}\mathbf{B}\mathbf{A}^{-1/2})] \quad \because 1 - x \leq -\log(x) \\ &= -\log\left[\prod_{i=1}^{n} \lambda_i(\mathbf{A}^{-1/2}\mathbf{B}\mathbf{A}^{-1/2})\right] \\ &= -\log|\mathbf{A}^{-1/2}\mathbf{B}\mathbf{A}^{-1/2}| = \log\frac{|\mathbf{A}|}{|\mathbf{B}|}. \quad \because |\mathbf{C}| = \prod_{i=1}^{n} \lambda_i(\mathbf{C}).\end{aligned}$$

In the last equality we use the following facts about the determinant of matrices: $|\mathbf{AB}| = |\mathbf{A}||\mathbf{B}|$ and $|\mathbf{A}^{-1}| = \frac{1}{|\mathbf{A}|}$. □

References

Blum, A., & Kalai, A. (1997). Universal portfolios with and without transaction costs. In *COLT '97: proceedings of the tenth annual conference on computational learning theory* (pp. 309–313). New York: ACM.

Brookes, M. (2005). *The matrix reference manual*. http://www.ee.ic.ac.uk/hp/staff/dmb/matrix/intro.html.

Boyd, S., & Vandenberghe, L. (2004). *Convex optimization*. New York: Cambridge University Press.

Cesa-Bianchi, N., & Lugosi, G. (2006). *Prediction, learning, and games*. Cambridge: Cambridge University Press.

Cover, T. (1991). Universal portfolios. *Mathematical Finance*, *1*, 1–19.

Gaivoronski, A. A., & Stella, F. (2000). Stochastic nonstationary optimization for finding universal portfolios. *Annals of Operations Research*, *100*, 165–188.

Hannan, J. (1957). Approximation to bayes risk in repeated play. In M. Dresher, A.W. Tucker, & P. Wolfe (Eds.), *Contributions to the theory of games* (Vol. III, pp. 97–139).

Hazan, E. (2006). *Efficient algorithms for online convex optimization and their applications*. PhD thesis, Princeton University.

Kakade, S. (2005). Personal communication.

Kalai, A., & Vempala, S. (2003). Efficient algorithms for universal portfolios. *Journal of Machine Learning Research*, *3*, 423–440.

Kalai, A., & Vempala, S. (2005). Efficient algorithms for on-line optimization. *Journal of Computer and System Sciences*, *71*(3), 291–307.

Kivinen, J., & Warmuth, M. K. (1998). Relative loss bounds for multidimensional regression problems. In M. I. Jordan, M. J. Kearns, & S.A. Solla (Eds.), *Advances in neural information processing systems* (Vol. 10). Cambridge: MIT.

Kivinen, J., & Warmuth, M. K. (1999). Averaging expert predictions. In *Computational learning theory: 4th European conference (EuroCOLT '99)* (pp. 153–167). Berlin: Springer.

Lovász, L., & Vempala, S. (2003a). *The geometry of logconcave functions and an $o^*(n^3)$ sampling algorithm*. Technical Report MSR-TR-2003-04, Microsoft Research.

Lovász, L., & Vempala, S. (2003b). Simulated annealing in convex bodies and an $0^*(n^4)$ volume algorithm. In *Proceedings of the 44th symposium on foundations of computer science (FOCS)* (pp. 650–659).

Lobo, M. S., Vandenberghe, L., Boyd, S., & Lebret, H. (1998). Applications of second-order cone programming.

Merhav, N., & Feder, M. (1992). Universal sequential learning and decision from individual data sequences. In *COLT '92: Proceedings of the fifth annual workshop on computational learning theory* (pp. 413–427). New York: ACM.

Riedel, K. (1991). A Sherman–Morrison–Woodbury identity for rank augmenting matrices with application to centering. *SIAM Journal on Mathematical Analysis*, *12*(1), 80–95.

Vaidya, P. M. (1996). A new algorithm for minimizing convex functions over convex sets. *Mathematical Programming*, *73*(3), 291–341.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the twentieth international conference on machine learning (ICML)* (pp. 928–936).