Hedge Algorithm and Subgradient Methods

Michel Baes, Michael Bürgisser
Institute for Operations Research
ETH Zurich
Raemistrasse 101, 8092 Zurich, Switzerland
{michel.baes,michael.buergisser}@ifor.math.ethz.ch

October 5, 2010

Abstract

We show that the Hedge Algorithm, a method that is widely used in Machine Learning, can be interpreted as a particular subgradient algorithm for minimizing a well-chosen convex function, namely as a Mirror Descent Scheme. Using this reformulation, we establish three modificitations and extensions of the Hedge Algorithm that are better or at least as good as the standard method with respect to worst-case guarantees. Numerical experiments show that the modified and extended methods that we suggest in this paper perform consistently better than the standard Hedge Algorithm.

Keywords: Hedge Algorithm, Subgradient Methods, Online Learning, Convex Optimization, Black-Box Model

1 Introduction

The Weighted Majority Algorithm, discovered by Littlestone and Warmuth [10] in the mid-nineties, has acquired a prominent status in Online Machine Learning. This algorithm aims at learning how to place smarter and smarter bets on the outcome of a repeated game. Freund and Schapire [5] have introduced its natural generalization, namely the Hedge Algorithm. Using a particularly simple and elegant reasoning, they have obtained convergence guarantees for this algorithm, and they showed how it can be related to their now widely used AdaBoost Algorithm.

In this paper, we propose an alternative viewpoint on the Hedge Algorithm, using methods and techniques from Convex Analysis and Convex Optimization. We show how the Hedge Algorithm can be expressed as a particular subgradient algorithm for minimizing a well-chosen convex function, namely as a Mirror Descent Scheme [11]. An immediate adaptation of

standard complexity results on subgradient algorithms allows us to derive optimal parameters for the Hedge Algorithm and to improve the convergence guarantee of the Hedge Algorithm obtained in [5] slightly. Freund and Schapire's analysis of the Hedge Algorithm is based on an assumption that we do not need when we interprete their algorithm as a subgradient method. Eliminating this unnecessary restriction enables us to formulate an alternative version of the Hedge algorithm with a worst case guarantee that is further improved.

Some extensions of these Mirror Descent Algorithms with provable convergence guarantees have recently been introduced by Nesterov in [14]. Adapting these extensions to the Hedge Algorithm, we obtain a new method that requires less *a priori* information on the repeated game, but performs in the worst case asymptotically at least as well as previous schemes. Comparing the different methods with respect to the computational time they need, all modifications and extensions of the Hedge Algorithm that we derive in this paper are roughly as efficient as the Hedge Algorithm. Finally, numerical results show that these modified or extended methods perform consistently better than the standard Hedge Algorithm.

We review the necessary concepts and algorithms of Convex Optimization in Section 2. We start from the most simplistic Gradient and Subgradient Algorithms. We show how these elementary Subgradient Algorithms have extended naturally to Mirror Descent Methods [11], which were further extended to Primal-Dual Subgradient Algorithms [14]. These two last classes of Subgradient Methods are meant to minimize a convex function over a convex set using from the objective function exclusively first-order information. Moreover, Nemirovski and Yudin [11] have proved that these methods have an *optimal* worst-case complexity, in the sense that whatever first-order method one might use, there always exists a convex function whose minimization needs a computational effort that is at least proportional to this worst-case complexity.

Mirror Descent Methods and Primal-Dual Subgradient Algorithms are flexible and robust enough to allow the use of *approximate* first-order information on the objective function. Of particular interest is the situation where the objective function is the expectation of some parametrized integrable function. If we do not have access to the exact value of the objective function and to its subgradient, we can try to approximate these objects by sampling the underlying random variable. Nesterov showed that the expected output of Primal-Dual Subgradient Algorithms is the solution to the original problem. We complete his result in Theorem 2.5 by providing a probabilistic guarantee on this output.

We briefly recall how the Hedge Algorithm works in Section 3, The algorithm attributes scores to some possible strategies. Those scores are altered individually at every iteration in view of the loss that the corresponding strategy has faced. The way these scores are modified depends on how these losses are perceived by the algorithm. We describe simple strategies that ensure the convergence of the corresponding Hedge Algorithm. Interestingly, one of them can be interpreted as an approximation of the utility function over gains and losses used in Prospect Theory [8, 9].

In Section 4, we show how the Hedge Algorithm can be expressed as a Mirror Descent Algorithm that uses (approximate) first-order information. This observation allows us to address several issues concerning the standard Hedge Algorithm from a new viewpoint: what is an optimal update strategy? How do we optimally perceive losses? What do we achieve by extending the Hedge Algorithm the very same fashion that Nesterov extended Mirror Descent Schemes into Primal-Dual Subgradient Algorithms? Based on these questions, we derive new methods, the Standard Hedge Algorithm with Optimal Update Parameter, the Optimal Hedge Algorithm, and the Optimal Exponential Weights Update Method.

We test these three schemes versus the Hedge Algorithm in Section 5. The proof of a technical result is left to the Appendix of the paper.

2 Subgradient methods in convex optimization

We give in this section a brief account on subgradient methods for convex optimization problems. As this topic has been studied extensively and developed dramatically during the last forty years (see e.g. [11, 15, 16] and the references therein), we focus on a few results that are relevant for our analysis of the Hedge Algorithm.

The most elementary gradient method solves unconstrained convex optimization problems, where the objective function $f : \mathbb{R}^n \to \mathbb{R}$ is convex and differentiable. Starting from a given point x_0, the algorithm generates a sequence $x_{k+1} := x_k - \lambda_k f'(x_k)$, where $\lambda_k > 0$ is an appropriate step-size and the gradient $f'(x_k)$ is the vector for which

$$f'(x_k)^T h = \lim_{\lambda \downarrow 0} \frac{f(x_k + \lambda h) - f(x_k)}{t} \qquad \forall\ h \in \mathbb{R}^n.$$

(Note the choice of the standard dot product in the left-hand side.) Many strategies exist for fixing the step-size, from a simple approximate line-search method to extremely elaborated techniques (see e.g. [17]).

If the convex function f is not differentiable, but its epigraph is still closed, i.e., f is closed, we can use a *subgradient* g_k as a substitute for the gradient $f'(x_k)$, that is an element of the set:

$$\partial f(x_k) := \{g \in \mathbb{R}^n : f(x) \geq f(x_k) + g^T(x - x_k) \text{ for all } x \in \mathbb{R}^n\}. \tag{1}$$

The above set comprises all the vectors that describe affine spaces passing through $(x_k, f(x_k))$ and defining half-spaces that contain the entire epigraph of f. Closedness and convexity of f ensure that this set is not empty on points x_k in the relative interior of the domain of f. If the convex function f is differentiable in x_k, the set $\partial f(x_k)$ contains only one element, namely $f'(x_k)$. We will use later the following equivalent way of defining the elementary subgradient

algorithm $x_{k+1} := x_k - \lambda_k g_k$, where $\lambda_k > 0$ is an appropriate step-size and $g_k \in \partial f(x_k)$:

$$x_{k+1} := \arg\min_{x \in \mathbb{R}^n} \left\{ \left(\sum_{i=0}^{k} \lambda_i g_i \right)^T (x - x_0) + \frac{1}{2} ||x - x_0||_2^2 \right\}. \tag{2}$$

For constrained problems, the simplest strategy consists in using for x_{k+1} the Euclidean projection of the point $x_k - \lambda_k g_k$ on the feasible set.

This elementary subgradient algorithm suffers from several inconsistencies. From a purely mathematical viewpoint, the most visible one is due to the definitions of gradient and subgradient. These objects depend on a scalar product — namely the dot product — that we have chosen arbitrarily. However, there is no reason to believe that our optimization problem is particularly well adapted to the geometry that this scalar product induces. In the next subsections, we briefly describe how this issue and some others have been addressed, leading to the design of new subgradient-type algorithms.

2.1 Mirror Descent Schemes

The family of Mirror Descent Schemes, designed by Nemirovski and Yudin (see Chapter 3 in [11]), constitutes an answer to the mathematical inconsistency we have pointed out above. As many variants of these algorithms exist, we have preferred to use the version that can be best compared with Nesterov's Primal-Dual Algorithm introduced in the next subsection.

Let us formalize our setting. We write E for a Euclidean space of dimension n and $\langle \cdot, \cdot \rangle$ for its scalar product. We designate by E^* its dual, i.e., the set of all linear applications from E to $\mathbb{R}$. We denote by $|| \cdot ||$ a norm on E — not necessarily the norm induced by the scalar product — and by $|| \cdot ||_*$ its corresponding dual, defined as:

$$||h||_* := \max_{||x||=1} \langle h, x \rangle, \qquad h \in E^*.$$

The definition (1) of subgradient can be easily extend to our more general setting (i.e., when the scalar product is not necessarily the standard dot product):

$$\partial f(x_k) := \{ g \in \mathbb{R}^n : f(x) \geq f(x_k) + \langle g, x - x_k \rangle \text{ for all } x \in \mathbb{R}^n \}.$$

We will use this definition of subgradient from now on.

Mirror Descent Schemes solve convex optimization problems that are formulated as:

$$f^* := \min_{x \in Q} f(x),$$

where $Q \subseteq E$ is a closed convex set. The function $f : Q \to \mathbb{R}$ is closed, convex, and equipped with an *oracle*, that is a device (e.g. a piece of compiled code) that takes as input a point

$x \in Q$ and returns the value $f(x)$ and a subgradient $g \in \partial f(x)$. In addition, we equip Q with a *mirror operator*: a function V_Q that maps a point in E^* to a point in Q. The precise form of the mirror operator V_Q will be defined later.

A Mirror Descent Scheme accumulates subgradients in a *dual* variable, that is, a variable in E^*, and projects this variable back (or *mirrors* it) in Q using V_Q.

Algorithm 2.1 *Set $s_0 := 0$, select a set of step-sizes $\{\lambda_k\}_{k\geq 0}$ and a starting point $x_0 \in Q$.*
for *$k \geq 0$,*
Determine a $g_k \in \partial f(x_k)$.
Set $s_{k+1} := s_k - \lambda_k g_k$.
Compute $x_{k+1} := V_Q(s_{k+1})$.
end ■

Remark 2.1 *The above algorithm corresponds to the scheme described in Section 4.1 in [4], where, according to their notation, we have taken $X \equiv Y$. In particular, our simplified version does not require the use of a separator from Q.* ■

Let us now discuss the construction of the mapping V_Q. Mirror Descent Schemes require a *prox-function* $d : Q \to \mathbb{R}$, that is, a *strongly convex* continuously differentiable function: there exists $\sigma > 0$ such that for every $x, y \in Q$,

$$d(y) \geq d(x) + \langle d'(x), y - x\rangle + \frac{\sigma}{2}||y - x||^2.$$

Also, we assume that d has a minimizer x_0 on Q. This minimizer is then unique by strong convexity, and we use it as starting point in the above algorithm.

Let us now define:

$$V_Q(s) := \arg\max_{x\in Q}\{\langle s, x - x_0\rangle - d(x)\}.$$

This maximizer is well-defined and unique since d is strongly convex. We assume that the structure of Q and of d are simple enough for $V_Q(s)$ to be easily computable. Note that the standard subgradient algorithm for unconstrained problems is exactly a Mirror Descent Scheme with $d(x) := ||x - x_0||_2^2/2$ (for $s := -\sum_{i=0}^k \lambda_i g_i$, compare the definition of V_Q with (2)). Also, the mapping V_Q corresponds to the standard Euclidean projection when the prox-function $d(x)$ is $||x||_2^2/2$, for which $x_0 = 0$.

Nemirovski and Yudin have determined the complexity of the above algorithm. We display here the version appearing in the paper [14], which corresponds better to our setting.

Theorem 2.1 (Consequence of Theorem 1 in [14]) *Assume that there exists a constant D such that $D \geq d(x^*)$, where $x^* \in Q$ and $f(x^*) = f^*$. With $f_k := \min\{f(x_i) : 0 \leq i \leq k\}$,*

we have:

$$f_k - f^* \leq \frac{1}{\sum_{i=0}^{k} \lambda_i} \left(D + \frac{1}{2\sigma} \sum_{i=0}^{k} \lambda_i^2 ||g_i||_*^2 \right).$$

■

If the subgradients are uniformly bounded, in particular if there is a constant Γ for which $||g_i||_* \leq \Gamma$ for all i, Nesterov has observed in [14] that the above algorithm is guaranteed to converge as long as $\sum_{i=0}^{k} \lambda_i$ diverges and $\sum_{i=0}^{k} \lambda_i^2$ converges as k goes to infinity. The latter condition implies that the weights λ_k converge to 0. However, the common sense dictates that new subgradients should be treated with more consideration than old ones as they are likely to contain more relevant information. Nesterov's Primal-Dual Subgradient Algorithm resolves this apparent contradiction.

2.2 Nesterov's Primal-Dual Subgradient Algorithm

Nesterov's Primal-Dual Subgradient Algorithm [14] can be understood as a generalization of the Mirror Descent Algorithm presented in the previous section. This method also solves convex problems of the form:

$$\min_{x \in Q} f(x),$$

where the only information on f is available through an oracle that returns the value of f and one of its subgradient at an input point. As for standard Mirror Descent Schemes, we choose a prox-function $d : Q \to \mathbb{R}$ that is σ-strongly convex with respect to the norm $||\cdot||$ and minimized on Q in x_0, where $d(x_0) = 0$. We use this prox-function to construct a *parametric* mirror operator. Given a parameter value $\beta > 0$, we set:

$$V_{Q,\beta}(s) := \arg\max_{x \in Q} \langle s, x - x_0 \rangle - \beta d(x).$$

Following Nemirovski and Yudin's strategy, Nesterov's Algorithm constructs a sequence of points in the dual space and project it on the feasible set Q.

Algorithm 2.2 *Set $s_0 := 0$, select a set of step-sizes $\{\lambda_k\}_{k \geq 0}$ and a non-decreasing sequence $\{\beta_k\}_{k \geq 0}$ of projection parameters. Set $x_0 := \arg\min\{d(x) : x \in Q\}$.*
for *$k \geq 0$,*
Determine a $g_k \in \partial f(x_k)$.
Set $s_{k+1} := s_k - \lambda_k g_k$.
Compute $x_{k+1} := V_{Q,\beta_{k+1}}(s_{k+1})$.
end ■

For $\beta_k = 1$ for any k, we recover Nemirovski and Yudin's Mirror Descent Scheme 2.1. In his paper, Nesterov [14] has determined the complexity of the generalized algorithm. In order to bound the duality gap, Nesterov employed a quantity, which we define in this paper as:

$$R_k := \max \left\{ \sum_{i=0}^{k} \lambda_i \langle g_i, x_i - x \rangle : \ x \in Q, \ d(x) \leq D \right\}.$$

Observe that the above quantity can be easily related to the notion of *regret* used in the Machine Learning community. For unitary weights $\lambda_i = 1$, the above quantity coincides with the standard definition of regret from Machine Learning. We will discuss this quantity in more details in the subsequent sections.

Theorem 2.2 (Theorem 1 in [14])
Let D be an upper bound on $d(x^*)$, where $x^* \in Q$ with $f(x^*) = f^*$. With $f_k := \min_{0 \leq i \leq k} f(x_i)$, we have:

$$f_k - f^* \leq \frac{R_k}{\sum_{i=0}^{k} \lambda_i} \leq \frac{1}{\sum_{i=0}^{k} \lambda_i} \left(\beta_{k+1} D + \frac{1}{2\sigma} \sum_{i=0}^{k} \frac{\lambda_i^2}{\beta_i} ||g_i||_*^2 \right). \tag{3}$$

■

This theorem suggests several possible strategies for selecting the weights β_k and the step-sizes λ_k. In particular, we are not forced to take decreasing step-sizes as in Nemirovski's Mirror Descent Algorithm. For instance, we can set $\lambda_k := 1$, $\beta_0 = 1$, and $\beta_{k+1} := \sum_{i=0}^{k} 1/\beta_i$ for all $k \geq 0$.

A more refined analysis shows that with $\lambda_k := 1$, one can take:

$$\beta_{k+1} := \nu \hat{\beta}_{k+1}, \quad \text{where} \quad \hat{\beta}_0 := 1, \ \hat{\beta}_{k+1} := \sum_{i=0}^{k} \frac{1}{\hat{\beta}_i}, \ \text{ and } \ \nu := \frac{\Gamma}{\sqrt{2\sigma D}}. \tag{4}$$

As in the previous subsection, Γ is an upper bound on the dual norm of the subgradient. Alternatively, in order to give more weight to last subgradients — they are arguably more relevant than the first one because they are closer from the minimizer — we can consider for instance $\lambda_k := (k+1)^2$ and $\beta_k := \nu k^{2.5}$, with which the right-hand side of (3) is still in $\mathcal{O}(k^{-0.5})$, provided that the subgradients output by the oracle are uniformly bounded. The coefficient ν is, as above, equal to $\Gamma/\sqrt{2\sigma D}$. More comments on the choice of ν are given in the next section.

2.3 Convex stochastic optimization

In order to model optimization problems where the objective to optimize is subject to some uncertainties, for instance the outcome of a random process, a possible choice consists in

replacing this uncertain objective function by its expectation. Given a Borel probability space $(\Omega, \mathcal{B}, P)$ and an objective function $\phi : Q \times \Omega \to \mathbb{R}$ that is P-integrable for each fixed x and where $Q \subseteq E$ is the feasible set, we aim at solving:

$$f^* := \min_{x \in Q} E_P[\phi(x, \omega)].$$

Here, we have $f(x) := E_P[\phi(x, \omega)] := \int_\Omega \phi(x, \omega) dP(\omega)$. If the function $\phi(\cdot, \omega)$ is convex and closed for every $\omega \in \Omega$ and if Q is convex and closed, the resulting optimization problem is convex and can be solved by any of the subgradient schemes from the last subsections, provided that the subgradient of $E_P[\phi(x, \omega)]$ is easily computable. So, we assume from now on that $\phi(\cdot, \omega)$ is convex and closed for every $\omega \in \Omega$ and that Q is convex and closed.

However, in most cases, it is impossible to compute exactly the value of the function f at a given point, let alone its subgradient. Instead of using $g_k \in \partial f(x_k)$, we have to content ourselves with an approximation of it, usually generated with (quite) a few samples $\{\omega_{k,\alpha}\}_{1 \leq \alpha \leq L_k} \subseteq \Omega$. The estimation of $g_k \in \partial f(x_k)$, or *stochastic subgradient of f at x_k*, is simply the average $\tilde{g}_k := \sum_{\alpha=1}^{L_k} \nabla_x \phi(x_k, \omega_{k,\alpha})/L_k$, where $\nabla_x \phi(x_k, \omega_{k,\alpha}) \in \partial_x \phi(x_k, \omega_{k,\alpha})$. Note the slight abuse of notations by which we write $\tilde{g}_k$ instead of $\tilde{g}(\omega_k)$ and by which we denote by $\nabla_x \phi(x_k, \omega_{k,\alpha})$ a subgradient of $\phi(\cdot, \omega_{k,\alpha})$ at x_k.

What is the effect of using inaccurate gradients on the algorithms? For his Primal-Dual Subgradient Algorithm, Nesterov answers this question at least partially. In the statement below, we use the notation:

$$E_{P^m}[F(x, \omega_1, \ldots, \omega_m)] := \int_{\Omega^m} F(x, \omega_1, \ldots, \omega_m) dP(\omega_1) \cdots dP(\omega_m).$$

Theorem 2.3 (Theorem 7 in [14]) *Assume that $d(x^*) \leq D$, where $x^* \in Q$ such that $f(x^*) = f^*$, and that the stochastic subgradients are uniformly bounded in the norm $||\cdot||_*$ by Γ. We construct the sequence $x_0, x_1, \ldots$ as in Algorithm 2.2 where g_k is replaced by $\tilde{g}_k$. With $M_k := \sum_{i=0}^k L_i$ and*

$$\tilde{f}_k := E_{P^{M_k}} \left[\min_{0 \leq i \leq k} \sum_{\alpha=1}^{L_i} \phi(x_i, \omega_{i,\alpha})/L_i \right],$$

we have:

$$\tilde{f}_k - f^* \leq \frac{1}{\sum_{i=0}^k \lambda_i} \left(\beta_{k+1} D + \frac{1}{2\sigma} \sum_{i=0}^k \frac{\lambda_i^2}{\beta_i} ||\tilde{g}_i||_*^2 \right).$$

■

The above result also holds when $\beta_k := 1$ for every $k \geq 0$, that is, for the Mirror Descent Scheme introduced earlier in this section, where the oracle produces stochastic subgradients instead of subgradients. Instrumental in the proof of the above Theorem, Nesterov derives a bound on the *weighted stochastic regret*:

$$\tilde{R}_k := \max\left\{\sum_{i=0}^{k} \lambda_i \langle \tilde{g}_i, x_i - x\rangle : \ x \in Q, \ d(x) \leq D\right\} \leq \beta_{k+1} D + \frac{1}{2\sigma}\sum_{i=0}^{k} \frac{\lambda_i^2}{\beta_i} ||\tilde{g}_i||_*^2. \qquad (5)$$

This notion of weighted stochastic regret will be used later in the text.

Using Hoeffding's inequality, we can obtain a probabilistic guarantee of the output of Nesterov's Algorithm with stochastic subgradients.

Theorem 2.4 (Hoeffding's Inequality [7]) *Let $Z_1, \ldots, Z_T$ be independent random variables on the probability space $(\Omega, \mathcal{B}, P)$. Assume that there exists $a \neq b \in \mathbb{R}$ for which*

$$P[Z_i \in [a, b]] = 1$$

for each $1 \leq i \leq T$, and that all these random variables have the same expectation μ. We have for every $\epsilon > 0$:

$$P\left[\sum_{i=1}^{T} Z_i/T \geq \mu + \epsilon\right] \leq \exp(-2\epsilon^2 T/(b-a)^2).$$

■

The proof of the following theorem combines ideas from Nesterov (especially from Theorem 1 in [14]) with Hoeffding's Inequality.

Theorem 2.5 *Assume that the conditions of Theorem 2.3 hold and let*

$$V := \max\left\{\phi(x, \omega) - \phi(x, \omega') : \ \omega, \omega' \in \Omega, \ x \in Q\right\} < \infty.$$

For every $\epsilon > 0$, the inequality

$$\min_{0 \leq i \leq k} f(x_i) - f^* \leq \frac{1}{\sum_{i=0}^{k} \lambda_i}\left(\beta_{k+1} D + \frac{1}{2\sigma}\sum_{i=0}^{k} \frac{\lambda_i^2}{\beta_i}\Gamma^2\right) + 2\epsilon$$

holds with a probability of at least

$$1 - 2\exp\left(-\frac{2\epsilon^2\left(\sum_{j=0}^{k} \lambda_j\right)^2}{M_k V^2} \min_{0 \leq i \leq k} \frac{L_i^2}{\lambda_i^2}\right).$$

Proof
As in Theorem 1 in [14], we define:

$$\tilde{R}_k := \max\left\{\sum_{i=0}^{k} \lambda_i \langle \tilde{g}_i, x_i - x\rangle : \ x \in Q, \ d(x) \leq D\right\}.$$

In the mentioned theorem, Nesterov proves that

$$\tilde{R}_k \leq \beta_{k+1} D + \frac{1}{2\sigma} \sum_{i=0}^{k} \frac{\lambda_i^2}{\beta_i} ||\tilde{g}_i||_*^2.$$

We proceed to determine a highly probable lower bound on $\tilde{R}_k$. By convexity of $x \mapsto \phi(x, \omega)$ for each $\omega \in \Omega$, we have:

$$\tilde{R}_k \geq \max\left\{\sum_{i=0}^{k} \lambda_i \left(\sum_{\alpha=1}^{L_i} \frac{\phi(x_i, \omega_{i,\alpha})}{L_i} - \frac{\phi(x, \omega_{i,\alpha})}{L_i}\right) : \ x \in Q, \ d(x) \leq D\right\}. \tag{6}$$

For any $0 \leq i \leq k$ and any $1 \leq \alpha \leq L_i$, let:

$$Z_{i,\alpha} := \frac{M_k}{L_i} \frac{\lambda_i \left(f(x_i) - \phi\left(x_i, \omega_{i,\alpha}\right)\right)}{\sum_{j=0}^{k} \lambda_j}.$$

Observe that $\{Z_{i,\alpha}\}_{i,\alpha}$ constitutes a collection of independent random variables with null expectation. They are bounded because ϕ is, and their spread is not greater than

$$\frac{M_k V}{\sum_{j=0}^{k} \lambda_j} \max_{0 \leq i \leq k} \frac{\lambda_i}{L_i}.$$

The first term of the right-hand side in (6) can be successively estimated as:

$$\begin{aligned}
\sum_{i=0}^{k} \sum_{\alpha=1}^{L_i} \frac{\lambda_i \phi(x_i, \omega_{i,\alpha})}{L_i} &= \left(\sum_{i=0}^{k} \sum_{\alpha=1}^{L_i} \frac{\lambda_i f(x_i)}{L_i} - \frac{\sum_{j=0}^{k} \lambda_j Z_{i,\alpha}}{M_k}\right) \\
&= \left(\sum_{j=0}^{k} \lambda_j\right) \left(\frac{\sum_{i=0}^{k} \lambda_i f(x_i)}{\sum_{j=0}^{k} \lambda_j} - \sum_{i=0}^{k} \sum_{\alpha=1}^{L_i} \frac{Z_{i,\alpha}}{M_k}\right) \\
&\geq \sum_{i=0}^{k} \lambda_i f(x_i) - \left(\sum_{j=0}^{k} \lambda_j\right) \epsilon,
\end{aligned}$$

where, according to Hoeffding's Inequality, the last relation is valid with a probability of at least:

$$1 - \exp\left(-\frac{2\epsilon^2 \left(\sum_{j=0}^{k} \lambda_j\right)^2}{M_k V^2} \min_{0 \le i \le k} \frac{L_i^2}{\lambda_i^2} \right).$$

Similarly, we can guarantee that:

$$\sum_{i=0}^{k} \sum_{\alpha=1}^{L_i} \frac{\lambda_i \phi(x, \omega_{i,\alpha})}{L_i} \le \left(\sum_{j=0}^{k} \lambda_j \right) (f(x) + \epsilon)$$

with the same probability as above. Thus:

$$\frac{\tilde{R}_k}{\sum_{j=0}^{k} \lambda_j} \ge \max_{x \in Q} \frac{\sum_{i=0}^{k} \lambda_i (f(x_i) - f(x))}{\sum_{j=0}^{k} \lambda_j} - 2\epsilon \ge \min_{0 \le i \le k} f(x_i) - f^* - 2\epsilon$$

with probability at least

$$1 - 2\exp\left(-\frac{2\epsilon^2 \left(\sum_{j=0}^{k} \lambda_j\right)^2}{M_k V^2} \min_{0 \le i \le k} \frac{L_i^2}{\lambda_i^2} \right).$$

■

Of course, the results also holds for the stochastic version of the Mirror Descent Scheme. Note that the above theorem echoes Lemma 6 in [6], where the authors give probabilistic guarantees on the outcome of a multiplicative weights algorithm.

2.4 The Black-Box Model: how fast can subgradient methods be?

The Black-Box Model for optimization has been created by Nemirovski and Yudin [11] to formalize the complexity study of some families of methods on some classes of problems.

In this framework, the optimizer knows in advance several characteristics of the problem she wants to solve. For instance, she might know that her problem is convex, with a differentiable objective, and she might also know the Lipschitz constant of her objective and so on. Given this information, the optimizer chooses an optimization algorithm to solve the particular instance of her problem. According to the Black-Box Model, the only information the algorithm can get from the particular instance it is dealing with is given by an *oracle*. An oracle is an entity that takes as input a test point and returns some relevant information on the instance, such as the value of the function and/or the value of one of its subgradients at that point. Most importantly, the information that the oracle give should be *local*, that is, if the instance is

modified outside of a certain neighborhood of the test point, the answer of the oracle should stay identical. In fact, this condition is used to create hard instances by designing them as nastily as allowed, given the current information the optimizer has already gathered. In convex optimization, subgradient methods use a *first-order oracle*, which returns the value of the objective function and of one of its subgradients at the input point.

Many negative complexity results emerged from this framework. A classical account on this topic can be found in [11]. Of particular interest to us is the following theorem, which concerns subgradient methods and shows the optimality of the presented algorithms.

Theorem 2.6 (Theorem 3.2.1 in [12])
Consider the unconstrained minimization problem $\min f(x)$*, where* $f : \mathbb{R}^n \to \mathbb{R}$ *is a convex function, and let* $x_0 \in \mathbb{R}^n$*. Assume that there exists a minimizer* x^* *of* f *and that*

$$x^* \in B := \{x \in \mathbb{R}^n : ||x - x_0||_2 \leq R\}.$$

Assume also that f *is Lipschitz continuous on* B *with Lipschitz constant* M*.*

We have at our disposal a first-order oracle $x \mapsto (f(x), g) \in \mathbb{R} \times \partial f(x)$*. If our optimization method constructs a sequence of points* $x_0, x_1, \ldots$ *such that*

$$x_k \in x_0 + span\{g_0, \ldots, g_{k-1}\},$$

then, for every $0 \leq k < n$*, we have:*

$$f(x_k) - f(x^*) \geq \frac{MR}{2(1 + \sqrt{k+1})}.$$

■

3 The Hedge Algorithm

3.1 The classical Hedge Algorithm

The Hedge Algorithm [5] is a generic method that encompasses many well-known schemes in Machine Learning. The problem this algorithm aims at solving can be described as follows. We have a pool of experts who place each one bet on the outcome of an event, e.g. the return of a basket of financial products. Every day, these experts face a loss (positive or negative) due to their initial bet. We would like to make use of the opinions of these experts to constitute our own bet. Unlike these experts, we have the possibility of placing a different bet every day.

The Hedge Algorithm gives a strategy whose average loss per day converges to the average daily loss of the best expert in the pool. The idea consists in assigning at every time step

a score to each expert, based on her previous losses. Then, the bet we place is a linear combination of all the experts' bets, with weights that are proportional to their respective scores.

Let us describe this algorithm more formally. Let $(\Omega, \mathcal{B}, P)$ be a Borel probability space. The *loss* is a P-measurable function $\ell : \Omega \to [-\mu, \rho]^n$, where n is the number of experts in the pool, $\mu \geq 0$ is the maximal gain they can get, and $\rho \geq 0$ the maximal loss they can face. We assume that these quantities are known and finite. The component j of ℓ refers to the loss of expert j. Observe that this setting does not rule out the possibility of having a deterministic loss function.

We evaluate, or perceive, these losses through a decreasing function $U_\gamma : [-\mu, \rho] \to (0, 1]$. In a first step, we assume that U_γ satisfies:

$$\gamma^{\frac{\mu+t}{\mu+\rho}} \leq U_\gamma(t) \leq 1 - (1-\gamma)\frac{\mu+t}{\mu+\rho}, \tag{7}$$

where $\gamma \in (0, 1)$ and $t \in [-\mu, \rho]$. The above condition is used by Freund and Schapire [5] to establish the convergence guarantees of the resulting algorithm. We will discuss several standard choices of loss evaluation functions together with their corresponding parameter γ.

The algorithm looks as follows. The jth component of the vector $w_k \in \mathbb{R}^n$, denoted here as $w_{k,j}$, represents the score of expert j at iteration k.

Algorithm 3.1 *Let* $\gamma \in]0, 1[$ *and* $w_0 := (1/n, \ldots, 1/n)^T$.
for $0 \leq k \leq T$,
Set $p_k := w_k / \sum_{j=1}^n w_{k,j}$.
Bet $p_{k,j}$ *of our unitary budget like expert* j.
Draw/endure/experience $\omega_k \in \Omega$.
Set $w_{k+1,j} := w_{k,j} U_\gamma(\ell_j(\omega_k))$.
end. ■

At iteration k, our strategy faces a loss of $\mathcal{L}_k := \sum_{j=1}^n p_{k,j} \ell_j(\omega_k)$.

Instead of drawing only one sample per time step, it can be relevant to draw several of them, say $(\omega_{k,\alpha})_{1 \leq \alpha \leq L_k}$ at iteration k. The score we give to expert j is in that case an average of the score of each individual draws. Namely, we set:

$$w_{k+1,j} := w_{k,j} \left(\prod_{\alpha=1}^{L_k} U_\gamma(\ell_j(\omega_{k,\alpha})) \right)^{1/L_k}.$$

Freund and Schapire have determined a bound on the convergence speed of the above scheme. In their paper, they have considered the situation where $\mu := 0$ and $\rho := 1$. The immediate extension of their reasoning to our more general setting yields the following result.

Theorem 3.1 (See Theorem 2 in [5]) *With the above notation and for $T > 0$:*

$$\sum_{k=0}^{T}(\mu + \mathcal{L}_k) \leq \frac{\mu+\rho}{1-\gamma}\ln(n) - \frac{\ln(\gamma)}{1-\gamma} \cdot \min_{1\leq j\leq n}\left(\sum_{k=0}^{T}(\mu + \ell_j(\omega_k))\right).$$

■

3.2 The function U_γ

The function $\ln(U_\gamma(\cdot)) : [-\mu, \rho] \to \mathbb{R}$ represents the *utility* of the losses incurred. According to the bounds (7), its minimum equals γ and is attained at ρ.

The simplest choice for this utility is an affine decreasing function:

$$\ln(U_\gamma(t)) := a\frac{\mu+t}{\mu+\rho},$$

where $a < 0$. This function complies with the requirement (7) for $\gamma := \exp(a)$.

For $\gamma := \exp(a)$, we can suggest a value for a from Theorem 3.1. If we have an upper bound $B > 0$ to the *loss* $\sum_{k=0}^{T}(\mu + \ell_j(\omega_k))$, we can take:

$$a := \ln\left(\frac{\sqrt{B}}{\sqrt{2(\mu+\rho)\ln(n)} + \sqrt{B}}\right) < 0 \tag{8}$$

(see Section 2.2 in [5]). Theorem 3.1 can be rewritten as follows, using the inequality

$$\ln(1/\gamma) \leq \frac{1-\gamma^2}{2\gamma}.$$

Theorem 3.2 *With the above notation and for $T > 0$:*

$$\sum_{k=0}^{T}\mathcal{L}_k - \min_{1\leq j\leq n}\sum_{k=0}^{T}\ell_j(\omega_k) \leq (\mu+\rho)\ln(n) + \sqrt{2(\mu+\rho)\ln(n)B}.$$

■

If the number T of iterations in known in advance, we can always set $B := (T+1)(\mu+\rho)$, implying in view of (8):

$$\gamma := \frac{1}{\sqrt{2\ln(n)/(T+1)} + 1}, \tag{9}$$

and, by Theorem 3.2, we obtain:

$$\sum_{k=0}^{T} \mathcal{L}_k - \min_{1\leq j\leq n} \sum_{k=0}^{T} \ell_j(\omega_k) \leq (\mu+\rho)\left(\ln(n) + \sqrt{2\ln(n)(T+1)}\right).$$

In some situations, it is meaningful to evaluate losses through a utility function that differs from the utility function for gains. For instance, we can use a function U_γ for which $\ln(U_\gamma(\cdot)) : [-\mu, \rho] \to \mathbb{R}$ becomes piecewise linear:

$$\begin{cases} \ln(U_\gamma(t)) := \frac{\mu+t}{\mu+\rho} a_+ & \text{for } t \in [-\mu, 0] \\ \ln(U_\gamma(t)) := \frac{a_- t + a_+ \mu}{\mu+\rho} & \text{for } t \in [0, \rho]. \end{cases}$$

For negative losses, this continuous function has a slope of $a_+/(\mu+\rho)$, while for positive ones, the slope equals $a_-/(\mu+\rho)$. Also, the appropriate value for γ is here:

$$\gamma := \exp\left(\frac{a_-\rho + a_+\mu}{\mu+\rho}\right).$$

In order to ensure condition (7), the utility for positive losses must drop at least as steep as for gains, that is $a_- \leq a_+ < 0$, and we must have

$$(1-a_-)\exp(a_+) \leq 1. \tag{10}$$

We refer the reader to the Appendix for a demonstration. Note that if the scaling of the utility is arbitrary, the condition (10) can always be enforced by multiplying a_- and a_+ by a sufficiently large constant κ, for instance $\kappa \geq 2|a_-|/a_+^2$.

Interestingly, the now well-established *Prospect Theory* in Econometrics [3, 8, 9], proposes an analytic form for the utility $\ln(U_\gamma(\cdot))$. This theory aims at taking into account subjective choices of investors, with the ultimate goal of explaining the striking variability of stocks returns over time. According to this theory, investors have a natural aversion for uncertain outcomes and for losses. Based on experimental data, Kahneman and Tversky propose the following subjective utility function for investors [9] (up to a convenient multiple). Observe that the perceived utility of losses depreciates more than twice as fast as the utility of gains appreciates.

$$\begin{cases} \ln(U_\gamma(t)) := -2\left(\frac{\mu+t}{\mu+\rho}\right)^{0.88} & \text{for } t \in [-\mu, 0] \\ \ln(U_\gamma(t)) := 2.5\left(\frac{\mu}{\mu+\rho}\right)^{0.88} - 4.5\left(\frac{\mu+t}{\mu+\rho}\right)^{0.88} & \text{for } t \in [0, \rho]. \end{cases}$$

Unfortunately, because of the 0.88-power, there is no $\gamma < 1$ for which the bounds (7) are satisfied. Nevertheless, we can approximate this power by 1 as it is sometimes done in the Prospect Theory literature, and get back to the piecewise-linear model presented above.

Following the idea of a different evaluation of gains and losses, Arora et al [2] suggest a score function of the following form:

$$U_\gamma : [-\mu, \rho] \to \mathbb{R} : \ t \mapsto \begin{cases} U_\gamma(t) := (1+\gamma)^{-t/\max\{\mu,\rho\}} & \text{for } t \in [-\mu, 0] \\ U_\gamma(t) := (1-\gamma)^{t/\max\{\mu,\rho\}} & \text{for } t \in [0, \rho]. \end{cases} \tag{11}$$

Note that this score function slightly extends the one initially introduced in [2], as — in contrast to our setting — we have $\rho \geq \mu$ as further requirement in [2]. We observe that the above score function U_γ does not comply with the assumptions that a score function needs to satisfy in the setting of the standard Hedge Algorithm.

With the above score function, we obtain the following algorithm, which is known as Multiplicative Weights Update Method.

Algorithm 3.2 *Let* $\gamma \in (0,1)$ *and* $w_0 := (1/n, \ldots, 1/n)^T$.
for $0 \leq k \leq T$,
Set $p_k := w_k / \sum_{j=1}^n w_{k,j}$.
Bet $p_{k,j}$ *of our unitary budget like expert* j.
Draw/endure/experience $\omega_k \in \Omega$.
Set $w_{k+1,j} := \begin{cases} w_{k,j}(1+\gamma)^{-l_j(\omega_k)/\max\{\mu,\rho\}} & \text{if } l_j(\omega_k) \in [-\mu, 0] \\ w_{k,j}(1-\gamma)^{l_j(\omega_k)/\max\{\mu,\rho\}} & \text{if } l_j(\omega_k) \in [0, \rho]. \end{cases}$
end ■

4 The Hedge Algorithm is a Mirror Descent Algorithm

In this section, we reformulate the problem solved by the Hedge Algorithm into an optimization problem, and we show that a particular Stochastic Mirror Descent Scheme for minimizing its objective function generates *exactly* the same sequence $\{p_k\}_{k\geq 0}$ as the Hedge Algorithm.

4.1 Random losses

Let us fix $\gamma \in (0,1)$ and a score function U_γ, that is, a decreasing function from $[-\mu, \rho]$ to $\mathbb{R}_+$. Importantly, the score function is not required to satisfy bounds (7). We assume that at each turn the loss incurred by every expert is stochastic, i.e., given a Borel probability space $(\Omega, \mathcal{B}, P)$, the loss

$$\ell : \Omega \to [-\mu, \rho]^n$$

is a P-measurable function. An option to deal with this uncertainty is to minimize the utility expectation:

$$\min_{x\in\Delta_n}\left\{f_{U_\gamma}(x) := E_P\left[-\sum_{j=1}^{n} x_j \ln(U_\gamma(\ell_j(\omega)))\right]\right\} = \min_{x\in\Delta_n} -\sum_{j=1}^{n} x_j E_P\left[\ln(U_\gamma(\ell_j(\omega)))\right],$$

where the set Δ_n is the standard simplex of $\mathbb{R}^n$:

$$\Delta_n := \{x \in \mathbb{R}^n_+ : \sum_{j=1}^{n} x_j = 1\}.$$

The minimizer of the function f_{U_γ} indicates the best expert. Also, this function is linear, thus convex, and can therefore be minimized by any of the stochastic subgradient algorithms described in Section 2. In our context of the Hedge Algorithm, the most convenient first-order oracle returns one stochastic subgradient at every iteration:

$$-\ln(U_\gamma(\ell(\omega))) \in [-\ln(U_\gamma(-\mu)), -\ln(U_\gamma(\rho))]^n.$$

Strong evidence (see [13], Section 4.1) suggests that the most appropriate prox-function for the n-dimensional simplex is the *entropy function*:

$$d : \Delta_n \to \mathbb{R}, \quad x \mapsto d(x) := \sum_{j=1}^{n} x_j \ln(x_j) + \ln(n),$$

with $x_0 := (1/n, \ldots, 1/n)^T$. The entropy function is 1-strongly convex with respect to the 1-norm, and gives for $D := \max\{d(x) : x \in \Delta_n\}$ a value of $\ln(n)$. The corresponding mirror operator takes the following form:

$$V_Q(s) = \arg\max\{\langle s, x - x_0\rangle - d(x) : x \in \Delta_n\} = \left[\frac{\exp(s_j)}{\sum_{i=1}^{n}\exp(s_i)}\right]_{1\le j\le n}.$$

The Stochastic Mirror Descent Scheme can be written as follows.

Algorithm 4.1 *Set $s_0 := 0$, $x_0 := (1/n, \ldots, 1/n)^T$, and select a set of step-sizes $\{\lambda_k\}_{k\ge 0}$.*
for *$k \ge 0$,*
Call the oracle to obtain a stochastic subgradient $\tilde{g}_k = -\ln(U_\gamma(\ell(\omega_k)))$.
Set $s_{k+1} := s_k - \lambda_k \tilde{g}_k$.
Compute

$$x_{k+1} := V_Q(s_{k+1}) = \left[\frac{\prod_{m=0}^{k} U_\gamma(\ell_j(\omega_m))^{\lambda_m}}{\sum_{i=1}^{n}\prod_{m=0}^{k} U_\gamma(\ell_i(\omega_m))^{\lambda_m}}\right]_{1\le j\le n}.$$

end ■

For a score function U_γ that complies with bounds (7) and with $\lambda_k := 1$ for all $k \geq 0$, the sequence $\{x_k\}_{k\geq 0}$ corresponds exactly to the sequence $\{p_k\}_{k\geq 0}$ constructed by Algorithm 3.1. Additionally, with a score function of the form (11) and with unitary λ_k's, we recover Algorithm 3.2. With unitary step-sizes, the vector s_k represents our accumulated knowledge, as $(-s_k)$ is the sum of the utility vectors $\{\ln(U_\gamma(\ell(\omega_m)))\}_{0\leq m\leq k}$. Finally,

$$x_{k+1} := \arg\max_{x\in\Delta_n} \left\{ \sum_{m=0}^{k} \langle \ln(U_\gamma(\ell(\omega_m))), x\rangle - \sum_{j=1}^{n} x_j \ln(x_j) \right\}$$

is the maximizer of the sum of the accumulated knowledge and the entropy, which reflects the information contained in the current distribution over the experts.

4.2 Adversarial losses

As an alternative setting, we may think of losses that are chosen in an adversarial way. In case of adversarial losses, we are confronted with the optimization problem:

$$\min_{x\in\Delta_n} \left\{ f_{U_\gamma}(x) := \sup_{\ell\in\bar{Q}} \left\{ -\sum_{j=1}^{n} x_j \ln(U_\gamma(\ell_j)) \right\} \right\},$$

where $\bar{Q} \subset [-\mu, \rho]^n$ is a closed set (and thus compact). We assume that the score function U_γ is not only decreasing, but continuous, which means that the supremum in the above problem can be replaced by a maximum. Observe that the function f_{U_γ} is closed and convex, as it is defined as the maximum of convex and closed functions. Furthermore, the above problem becomes trivial when $\bar{Q} = [-\mu, \rho]^n$.

For any $x \in \Delta_n$, we have:

$$S(x) := \operatorname{conv} \left\{ -\ln(U_\gamma(\ell)) : \ \ell \in \bar{Q} \text{ such that } f_{U_\gamma}(x) = -\sum_{j=1}^{n} x_j \ln(U_\gamma(\ell_j)) \right\} \subset \partial f_{U_\gamma}(x).$$

We make the following assumption: when we call the first-order oracle with input $x \in \Delta_n$, the oracle returns an element g that belongs to $S(x)$ and that is of the form $g = -\ln(U_\gamma(\ell))$ for an $\ell \in \bar{Q}$. It is easy to verify that applying Algorithm 2.1 with $\lambda_k = 1$ for all k and with the prox-function $d(x) := \sum_{j=1}^{n} x_j \ln(x_j) + \ln(n)$, $x \in \Delta_n$, generates the same sequence $\{p_k\}_{k\geq 0}$ of probability vectors as in Algorithm 3.1. Since the utility function in Algorithm 3.2 can be positive, this scheme does not fall strictly in the framework of Algorithm 3.2. However, by running Algorithm 2.1 with the same step-sizes and prox-function as before and with a score function of the form 11, we exactly generate the sequence $\{p_k\}_{k\geq 0}$ from Algorithm 3.2.

4.3 Optimal update parameter for the standard Hedge Algorithm

All the following results can be derived for both random and adversarial losses. However, for notational simplicity, we only treat the case of random losses and suppose that all the assumptions made in Subsection 4.1 hold.

If we choose the same affine utility as Freund and Schapire [5], namely

$$\ln(U_\gamma(t)) = \ln(\gamma)\frac{\mu + t}{\mu + \rho},\ \gamma \in (0,1),$$

we can recover a result similar to Theorem 3.2 from Nesterov's bound on the stochastic regret (5) by taking $\lambda_k = \beta_k = 1$ for all k. With the above affine utility, we have:

$$\|\ln(U_\gamma(t))\|_\infty \leq -\ln(\gamma) \qquad \forall\ t \in [-\mu, \rho].$$

Provided that the number of steps T is known in advance, the bound (5) yields:

$$\begin{aligned}
\ln(n) + \frac{\ln^2(\gamma)}{2}(T+1) &\geq \max\left\{\sum_{m=0}^{T}\langle -\ln(U_\gamma(\ell(\omega_m))), x_m - x\rangle :\ x \in \Delta_n\right\} \\
&= \max\left\{\sum_{m=0}^{T}\langle -\ln(\gamma)\frac{\mu\bar{1} + \ell(\omega_m)}{\mu+\rho}, x_m - x\rangle :\ x \in \Delta_n\right\} \\
&= \frac{-\ln(\gamma)}{\mu+\rho}\max\left\{\sum_{m=0}^{T}\langle \ell(\omega_m), x_m - x\rangle :\ x \in \Delta_n\right\} \\
&= \frac{-\ln(\gamma)}{\mu+\rho}\left(\sum_{m=0}^{T}\mathcal{L}_m - \min_{1\leq j\leq n}\sum_{m=0}^{T}\ell_j(\omega_m)\right),
\end{aligned}$$

where $\bar{1}$ denotes the n-dimensional all one vector. Optimizing over the parameter γ, we find:

$$\gamma^* = \exp\left(-\sqrt{\frac{2\ln(n)}{T+1}}\right), \tag{12}$$

and thus:

$$(\mu+\rho)\sqrt{2\ln(n)(T+1)} \geq \sum_{m=0}^{T}\mathcal{L}_m - \min_{1\leq j\leq n}\sum_{m=0}^{T}\ell_j(\omega_m). \tag{13}$$

The right-hand side is precisely the *regret* standardly used in Machine Learning. Observe that the stochastic regret with unitary weights coincides with the usual notion of regret in Machine Learning, multiplied by $\ln(1/\gamma)/(\mu+\rho)$. The above result improves the bound in Theorem 3.2 by an additive quantity of $(\mu+\rho)\ln(n)$.

4.4 Optimal affine utility functions

Let $\gamma \in (0,1)$, $a > 0$, and $b \in \mathbb{R}$. Consider the following affine utility function:

$$\ln(U_\gamma(t)) := \ln(\gamma)(at + b), \qquad t \in [-\mu, \rho].$$

With the above utility function, we have:

$$\|\ln(U_\gamma(l))\|_\infty \leq -\ln(\gamma) a \Gamma(a,b) \qquad \forall\ l \in [-\mu, \rho]^n,$$

where $\Gamma(a,b) := \max\{|-\mu + b/a|, |\rho + b/a|\}$. Assume that we perform $T+1$ iterations of Algorithm 4.1, where we choose constant step-sizes $\lambda_k = 1$. Applying Nesterov's bound (5) with $\beta_k = 1$ for all k, we obtain:

$$\ln(n) + \frac{(T+1)\ln^2(\gamma) a^2 \Gamma^2(a,b)}{2} \geq -\ln(\gamma) a \left(\sum_{m=0}^{T} \mathcal{L}_m - \min_{1 \leq j \leq n} \sum_{m=0}^{T} \ell_j(\omega_m) \right).$$

This bound is minimized by parameters γ^*, a^*, and b^* that satisfy:

$$\gamma^* = \exp\left(-\frac{2}{a^*(\mu+\rho)} \sqrt{\frac{2\ln(n)}{T+1}} \right) \qquad \text{and} \qquad b^* := \frac{\mu - \rho}{2} a^*,$$

where $a^* > 0$. For any optimal parameter choice, we obtain:

$$\sum_{k=0}^{T} \mathcal{L}_k - \min_{1 \leq j \leq n} \sum_{k=0}^{T} \ell_j(\omega_k) \leq (\mu + \rho) \sqrt{\frac{\ln(n)(T+1)}{2}}.$$

This result improves bound (13) by a factor of $1/2$.

4.5 An optimal algorithm setup that is independent from the number of iterations

In view of our discussion in Section 2, we can use Nesterov's Primal-Dual Subgradient Algorithm for minimizing the convex function f_{U_γ}, whatever its form is. The only adaptation of Algorithm 4.1 resides in the coefficients $\{\beta_k\}_{k \geq 0}$, which appear as powers in the score update.

Algorithm 4.2 (Exponential Weights Update Method) *Set $s_0 := 0$, select a sequence of step-sizes $\{\lambda_k\}_{k \geq 0}$ and a non-decreasing sequence $\{\beta_k\}_{k \geq 0}$. Set $x_0 := (1/n, \ldots, 1/n)^T$.*
for *$k \geq 0$,*
Call the oracle and obtain $\tilde{g}_k$.

Set $s_{k+1} := s_k - \lambda_k \tilde{g}_k$.
Compute

$$x_{k+1} := V_{Q,\beta_{k+1}}(s_{k+1}) = \left[\frac{\exp\left(s_{k+1,j}/\beta_{k+1}\right)}{\sum_{i=1}^{n}\exp\left(s_{k+1,i}/\beta_{k+1}\right)}\right]_{1\leq j\leq n}.$$

end ■

Alternatively, we may write the update rule for x_{k+1} as:

$$x_{k+1} := V_{Q,\beta_{k+1}}(s_{k+1}) = \left[\frac{\prod_{m=0}^{k} U_\gamma(\ell_j(\omega_m))^{\lambda_m/\beta_{k+1}}}{\sum_{i=1}^{n}\prod_{m=0}^{k} U_\gamma(\ell_i(\omega_m))^{\lambda_m/\beta_{k+1}}}\right]_{1\leq j\leq n}.$$

That is, we simply replace U_γ by $U_\gamma^{1/\beta_{k+1}}$ at iteration k.

Interestingly, the very introduction of the sequence $\{\beta_k\}_{k\geq 0}$ allows us to define strategies for the sequences $\{\beta_k\}_{k\geq 0}$, $\{\lambda_k\}_{k\geq 0}$ for which the best parameter γ is *independent* of the desired number of iterations T. Indeed, with a strategy of the form (4):

$$\lambda_k := 1, \quad \beta_{k+1} := \nu\hat{\beta}_{k+1}, \qquad \text{where} \qquad \hat{\beta}_0 := 1,\ \hat{\beta}_{k+1} := \sum_{i=0}^{k} 1/\hat{\beta}_i, \quad \text{for a } \ \nu > 0, \tag{14}$$

the convergence bound on the stochastic regret given in (5) becomes:

$$\max\left\{\sum_{m=0}^{T}\langle\tilde{g}_m, x_m - x\rangle :\ x\in\Delta_n\right\} \leq \hat{\beta}_{T+1}\left(\nu\ln(n) + \frac{\Gamma^2}{2\nu}\right),$$

where Γ bounds the dual norm of f's stochastic subgradients. Now, only the factor $\hat{\beta}_{T+1}$ depends on the iteration number. Thus, one does not need to know the iteration number in advance to optimize the various coefficients involved here.

Assume now that we choose an affine utility function of the form:

$$\ln(U_\gamma(t)) := \ln(\gamma)(at+b), \qquad t\in[-\mu,\rho],$$

where $\gamma\in(0,1)$, $a>0$, and $b\in\mathbb{R}$. With the above bound on the stochastic regret, we immediately obtain:

$$\left(\sum_{m=0}^{T}\mathcal{L}_m - \min_{1\leq j\leq n}\sum_{m=0}^{T}\ell_j(\omega_m)\right) \leq \frac{1}{a\ln(1/\gamma)}\hat{\beta}_{T+1}\left(\nu\ln(n) + \frac{\Gamma^2}{2\nu}\right). \tag{15}$$

Recall:

$$||\ln(U_\gamma(\ell))||_\infty \leq \max_{1\leq j\leq n}|\ln(\gamma)(a\ell_j+b)| \leq -\ln(\gamma)a\max\{|-\mu+b/a|, |\rho+b/a|\} =: \Gamma.$$

Therefore, the upper bound (15) can be expressed as a function of the ratio $\ln(1/\gamma)/\nu$, a and b. Optimizing over these parameters, we observe that bound (15) becomes minimal for γ^*, ν^*, a^*, and b^* that satisfy:

$$\frac{\ln(1/\gamma^*)}{\nu^*} = \frac{2\sqrt{2\ln(n)}}{a^*(\mu+\rho)} \qquad \text{and} \qquad b^* = \frac{\mu-\rho}{2}a^*,$$

where $a^* > 0$ and $\gamma^* \in (0,1)$. For these optimal parameters, we obtain;

$$\left(\sum_{m=0}^{T}\mathcal{L}_m - \min_{1\le j\le n}\sum_{m=0}^{T}\ell_j(\omega_m)\right) \le (\mu+\rho)\hat{\beta}_{T+1}\sqrt{\frac{\ln(n)}{2}}.$$

It can be proved (see Lemma 3 in [14]) that

$$\sqrt{2T+1} \le \hat{\beta}_{T+1} \le \sqrt{2T+1} + \frac{1}{1+\sqrt{3}}.$$

This bound allows us to recover approximately the complexity bound obtained for the Mirror Descent Scheme.

5 Numerical results

We test the methods presented for different sequences of random losses $\{\ell(\omega_k)\}_{k=0}^{T}$, each generated by realizations of a multivariate normally distributed random vector with mean $\bar{\mu}$ and covariance matrix Σ. The data $(\bar{\mu}, \Sigma)$ is taken from [1]. We consider three different vector sizes for $\bar{\mu}$, that is, three different pools of experts of sizes 30, 167, and 808 experts, respectively. We perform $T := 7800$ iterations, which corresponds to the number of transactions at New York Stock Exchange (NYSE) during one month (20 trading days of 6h30), provided that there is one transaction per minute. All experiments are run 10 times, and the obtained losses are averaged afterwards.

We want to find a sequence $\{x_k\}_{0\le k\le T}$ of probability distributions yielding averaged expected losses:

$$\frac{\mathcal{L}_k}{k+1} = \frac{1}{k+1}\sum_{m=0}^{k}\langle x_m, \ell(\omega_m)\rangle, \qquad \text{where } 0 \le k < T,$$

that converge to the best expert's averaged expected loss:

$$\min\left\{\frac{1}{k+1}\sum_{m=0}^{k}\langle x, \ell(\omega_m)\rangle : \ x \in \Delta_n\right\}.$$

Here, the scalar product $\langle\cdot,\cdot\rangle$ is the standard dot product.

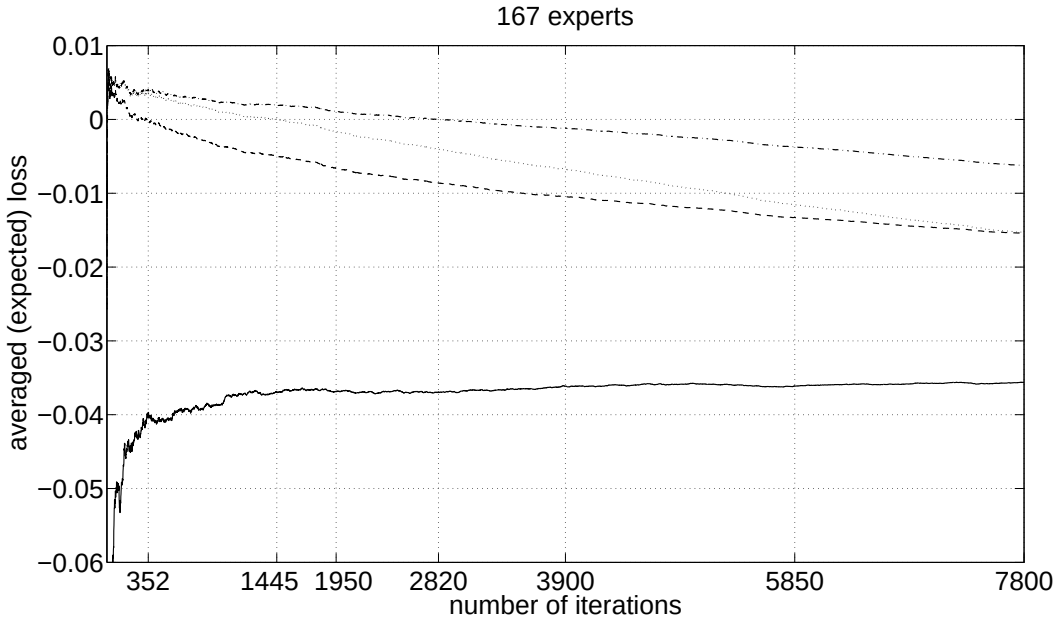

Figure 1: Averaged expected losses $\mathcal{L}_k/(k+1)$ with a pool of 167 experts. The continuous line represents the averaged expected losses of the best expert. The dashed, dotted, and dashed-dotted lines correspond respectively to the averaged expected losses of the Optimal Exponential Weights Update Algorithm, of the Optimal Hedge Algorithm, and of the Standard Hedge Algorithm. The numbers 352, 1145, and 2820 represent the last iteration, where the Optimal Exponential Weights Update Algorithm, the Optimal Hedge Algorithm, and the Standard Hedge Algorithm, respectively, suffer an averaged expected loss.

According to our discussion in the last section, we choose different utility functions. In a first setting, we select the natural extension of Freund and Schapire's utility function [5], which is:

$$\ln(U_\gamma(t)) := \ln(\gamma)\frac{\mu+t}{\mu+\rho}, \qquad t \in [-\mu, \rho].$$

We refer to Algorithm 3.1, where we choose U_γ as above and γ as suggested by Freund and Schapire (see (9)), as the **Standard Hedge Algorithm**. By the **Standard Hedge Algorithm with Optimal Update Parameter**, we denote Algorithm 3.1 with U_γ as above and optimal update parameter γ as defined in (12).

We call Algorithm 4.1 the **Optimal Hedge Algorithm**, provided that we have unitary

30 **experts** ($\mu = 0.4748$, $\rho = 0.4577$)**:**

Number of iterations	1950	3900	5850	7800
Best expert	−0.0043	−0.0039	−0.0042	−0.0041
Standard Hedge	0.0049	0.0041	0.0032	0.0028
Standard Hedge with Opt. Update Parameter	0.0049	0.0041	0.0032	0.0027
Optimal Hedge	0.0042	0.0028	0.0017	0.0011
Optimal Exponential Weights Update	0.0030	0.0020	0.0013	0.0009

167 **experts** ($\mu = 1.3252$, $\rho = 1.3324$)**:**

Number of iterations	1950	3900	5850	7800
Best expert	−0.0368	−0.0368	−0.0411	−0.0356
Standard Hedge	0.0012	−0.0012	−0.0039	−0.0062
Standard Hedge with Opt. Update Parameter	0.0011	−0.0013	−0.0039	−0.0064
Optimal Hedge	−0.0016	−0.0067	−0.0118	−0.0153
Optimal Exponential Weights Update	−0.0066	−0.0104	−0.0133	−0.0154

808 **experts** ($\mu = 5.5516$, $\rho = 5.6021$)**:**

Number of iterations	1950	3900	5850	7800
Best expert	−0.0420	−0.0423	−0.0411	−0.0402
Standard Hedge	0.0213	0.0196	0.0186	0.0175
Standard Hedge with Opt. Update Parameter	0.0212	0.0195	0.0185	0.0174
Optimal Hedge	0.0198	0.0171	0.0152	0.0131
Optimal Exponential Weights Update	0.0175	0.0154	0.0144	0.0134

Table 1: Averaged expected losses $\mathcal{L}_k/(k+1)$ after 1950, 3900, 5850, and 7800 iterations, i.e., after 1, 2, 3, and 4 weeks of trading.

step-sizes $\lambda_k = 1$ for all k and that we have the following optimal affine utility function:

$$\ln(U_\gamma(t)) := \ln(\gamma)\left(t + \frac{\mu - \rho}{2}\right), \qquad t \in [-\mu, \rho],$$

where:

$$\gamma := \exp\left(-\frac{2}{\mu + \rho}\sqrt{\frac{2\ln(n)}{T+1}}\right).$$

We refer to Exponential Weights Update Method 4.2 as **Optimal Exponential Weights Update Algorithm**, if we are in the following situation: we choose a non-decreasing sequence $\{\beta_k\}_{k\geq 0}$ defined as in (14), where $\nu := 1$. Additionally, we have unitary step-sizes $\lambda_k = 1$, and we set:

$$\ln(U_\gamma(t)) := \ln(\gamma)\left(t + \frac{\mu - \rho}{2}\right), \qquad t \in [-\mu, \rho],$$

where:

$$\gamma := \exp\left(-\frac{2\sqrt{2\ln(n)}}{\mu + \rho}\right).$$

In all the experiments, we observe that all the extensions of the Hedge Algorithm yield averaged loss sequences $\{\mathcal{L}_k/(k+1)\}_{0\leq k\leq T}$ that are consistently better than the averaged losses suffered by the Standard Hedge Algorithm; see Figure 1 and Table 1. Whereas the Standard Hedge Algorithm with Optimal Update Parameter shows a performance that is very similar to the efficiency of the Standard Hedge Algorithm, the Optimal Hedge Algorithm and the Optimal Exponential Weights Update Algorithm clearly outperform the Standard Hedge Algorithm. According to this observation, using the optimal affine utility function is much more important than implementing the optimal update parameter γ.

In the experiment with a pool of 167 experts, the best expert ends up with an accumulated profit of 277.77; see Table 1. The best method, the Optimal Exponential Weights Update Algorithm, yields an accumulated profit of 120.20, which corresponds to 43.27% of the best expert's profit. This is more than twice as good as the performance of the Standard Hedge Algorithm, which generates an accumulated profit of 48.69 (17.53% of the best expert's profit).

All the methods are implemented in Matlab. For the computations, we use a computer with two AMD Athlon processors with a CPU of 2.9 GHz, and with 3.7 GB of RAM. The CPU times required by the three methods are of the same order; see Table 2.

Number of experts	30	167	808
Standard Hedge Alg.	0.3172"	0.5175"	1.4592"
Standard Hedge Alg. with Opt. Update Parameter	0.3106"	0.5193"	1.4473"
Optimal Hedge Alg.	0.3109"	0.5160"	1.4459"
Optimal Exponential Weights Update Alg.	0.3416"	0.5738"	1.5995"

Table 2: CPU times required to perform 7800 iterations for different numbers of experts.

6 Appendix

6.1 Hedge Algorithm with piecewise linear utility

We provide here a proof for the assertions on the Hedge Algorithm with the piecewise linear utility as described in Subsection 3.2.

Theorem 6.1 *Let $\mu, \rho \geq 0$, let $a_-, a_+ < 0$ and let $h : [0,1] \to [-\mu, \rho]$ be the function*

$$\begin{cases} h(t) := \frac{\mu+t}{\mu+\rho} a_+ & \text{for } t \in [-\mu, 0] \\ h(t) := \frac{a_- t + a_+ \mu}{\mu+\rho} & \text{for } t \in [0, \rho]. \end{cases}$$

If $a_- \leq a_+$ and $(1 - a_-)\exp(a_+) \leq 1$,

$$\ln(\gamma)\frac{\mu+t}{\mu+\rho} \leq h(t) \leq \ln\left(1 - (1-\gamma)\frac{\mu+t}{\mu+\rho}\right) \tag{16}$$

with

$$\gamma := \exp\left(\frac{a_-\rho + a_+\mu}{\mu+\rho}\right).$$

Proof
As $a_+ \geq a_-$, the function h is concave. The lower bound and the upper bound in (16) match when $t = \rho$, in which case $h(\rho) = \ln(\gamma)$, given our expression for γ. The lower bound is an affine function which equals $h(-\mu)$ at $-\mu$, establishing the lower bound. As the case $a_+ = a_-$ is trivial, we disregard it in the rest of this proof. In order to comply with the upper bound, which is a concave function in t, the function h should merely be smaller than $\ln(1 - (1-\gamma)\mu/(\mu+\rho))$ in its non-differentiable point 0, i.e., with $\lambda := \mu/(\mu+\rho)$:

$$\exp(\lambda a_+) + \lambda(1 - \exp(\lambda a_+ + (1-\lambda)a_-)) \leq 1.$$

Obviously, there is nothing to prove when λ equals 0 or 1. Therefore, we assume that $0 < \lambda < 1$.

Denoting $v := \exp(a_-) \in (0,1)$ and $M := \exp(a_+)/\exp(a_-) \geq 1$, we can rewrite this condition after some manipulations as:

$$g(\lambda) := \left(\frac{v^\lambda - \lambda v}{1-\lambda}\right)^{1/\lambda} \leq \frac{1}{M}.$$

Observe that the numerator $v^\lambda - \lambda v$ is positive because $v^\lambda > v > \lambda v$ when $0 < \lambda < 1$ and $v < 1$. Obviously, this condition can be checked for a particular choice of a_- and a_+ given μ and ρ, but it gives little insight on how to modify this choice to fit the desired bounds (16). In order to simplify this condition, we determine some conditions on a_- and a_+ that ensure that $Mg(\lambda) \leq 1$ for every $\lambda \in (0,1)$.

1. We have $g(\lambda) \leq 1$ for $0 < \lambda < 1$. Indeed, by concavity of the logarithm function, we have:
$$\ln(v^\lambda) = \lambda \ln(v) + (1-\lambda)\ln(1) \leq \ln(\lambda v + 1 - \lambda),$$
and therefore $v^\lambda - \lambda v \leq 1 - \lambda$. Thus $g(\lambda)^\lambda \leq 1$, and $g(\lambda) \leq 1$.

2. The function g is increasing on $]0,1[$. Since g is positive, we prove the equivalent property that $\ln(g(\cdot))$ is increasing. The derivative of this function is:
$$[\ln(g(\lambda))]' = \frac{1}{\lambda^2}\left(\lambda\left(\frac{v^\lambda \ln(v) - v}{v^\lambda - \lambda v} + \frac{1}{1-\lambda}\right) - \ln\left(\frac{v^\lambda - \lambda v}{1-\lambda}\right)\right).$$
Let $\alpha := \lambda v^{1-\lambda}$. Note that $0 < \alpha < \lambda$. Checking that the above derivative is nonnegative is equivalent to checking that for all $0 < \alpha < \lambda < 1$:
$$\begin{aligned}
0 \leq & \frac{v^\lambda \ln(v^\lambda) - \lambda v}{v^\lambda - \lambda v} + \frac{\lambda}{1-\lambda} + \ln\left(\frac{1-\lambda}{v^\lambda - \lambda v}\right) \\
= & \frac{\ln(v^\lambda) - \lambda v^{1-\lambda}}{1 - \lambda v^{1-\lambda}} + \frac{\lambda}{1-\lambda} + \ln\left(\frac{1-\lambda}{1-\lambda v^{1-\lambda}}\right) - \ln\left(v^\lambda\right) \\
= & \frac{\ln\left(v^\lambda\right) - \alpha}{1-\alpha} + \frac{\lambda}{1-\lambda} + \ln\left(\frac{1-\lambda}{1-\alpha}\right) - \ln\left(v^\lambda\right) \\
= & \ln\left(\frac{\alpha}{\lambda}\right)\frac{\lambda}{1-\lambda}\frac{\alpha}{1-\alpha} + \frac{\lambda}{1-\lambda} - \frac{\alpha}{1-\alpha} + \ln\left(\frac{1-\lambda}{1-\alpha}\right),
\end{aligned}$$
or:
$$0 \leq (1-\alpha)(1-\lambda)\ln\left(\frac{1-\lambda}{1-\alpha}\right) - \alpha\lambda\ln\left(\frac{\lambda}{\alpha}\right) + \lambda - \alpha. \tag{17}$$
This inequality can probably be checked in many ways. Admittedly, the least elegant way of proving it is by viewing the right-hand side as a function of λ, and checking that

this function is itself an increasing function on $]\alpha, 1[$. The derivative of this function is, after a few simplifications:

$$-(1-\alpha)\ln\left(\frac{1-\lambda}{1-\alpha}\right) - \alpha\ln\left(\frac{\lambda}{\alpha}\right) \geq -\ln(1) = 0,$$

where the inequality comes from the concavity of the logarithmic function. Therefore, the inequality (17) has only to be checked when $\lambda \downarrow \alpha$. The limit happens to exist and to equal 0.

3. Putting everything together, we have:

$$\sup_{0<\lambda<1} g(\lambda) = \lim_{\lambda\uparrow 1} g(\lambda) \leq \lim_{\lambda\uparrow 1} g(\lambda)^{\lambda} = \lim_{\lambda\uparrow 1} v - v^{\lambda}\ln(v) = v(1-\ln(v)).$$

The first equality holds as g is an increasing function, the inequality because $g(\lambda) \leq 1$ for $0 < \lambda < 1$, the third relation follows from de l'Hospital's rule.[1] If $(1-a_-)\exp(a_+) \leq 1$, we conclude that $\sup\{g(\lambda) : 0 < \lambda < 1\} \leq 1/M$. ■

References

[1] Portfolio Selection Instances from Yahoo Finance. Available from http://tabu.diegm.uniud.it/portfolio/.

[2] S. Arora, E. Hazan, and S. Kale. Multiplicative weights method: a meta-algorithm and applications. Technical report, Computer Science Department, Princeton, 2005.

[3] N. Barberis, M. Huang, and T. Santos. Prospect Theory and Asset Prices. *The Quarterly Journal Of Economics*, 116(1):1–53, 2001.

[4] A. Ben-Tal, T. Margalit, and A. Nemirovski. The ordered subsets mirror descent optimization method with applications to tomography. *SIAM Journal on Optimization*, 12(1):79–108, 2001.

[5] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

[6] Y. Freund and R. Schapire. Adaptative game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1999.

[7] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58:13–30, 1963.

[1] In fact, the inequality above is an equality, and our bound $v(1-\ln(v))$ is tight.

[8] D. Kahneman and A. Tversky. Prospect Theory: An Analysis of Decision under Risk. *Econometrica*, 47(2):263–292, 1979.

[9] D. Kahneman and A. Tversky. Advances in Prospect Theory: Cumulative Representation of Uncertainty. *Journal of Risk and Uncertainty*, 5:297–323, 1992.

[10] N. Littlestone and M. Warmuth. The Weighted Majority Algorithm. *Information and Computation*, 108(2):212–261, 1994.

[11] A. Nemirovski and D. Yudin. *Problem Complexity and Method Efficiency in Optimization*. John Wiley, 1983.

[12] Y. Nesterov. *Introductory lectures on convex optimization: a basic course*, volume 87 of *Applied Optimization*. Kluwer Academic Publishers, 2003.

[13] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103(1):127–152, 2005.

[14] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical Programming*, 120(1):221–259, 2009.

[15] B. Polyak. *Introduction to Optimization*. Optimization Software Inc. Publications Division, 1987.

[16] N. Shor. *Minimization Methods for Non-Differentiable Functions*, volume 3 of *Springer Series in Computational Mathematics*. Springer, 1985.

[17] Y. Yuan. A new stepsize for the steepest descent method. *Journal of Computational Mathematics*, 24:149156, 2006.