Pursuit-Evasion Without Regret, with an Application to Trading

Lili Dworkin LDWORKIN@SEAS.UPENN.EDU
Michael Kearns MKEARNS@CIS.UPENN.EDU
Yuriy Nevmyvaka YURIY.NEVMYVAKA@GMAIL.COM
Computer and Information Science, University of Pennsylvania

Abstract

We propose a state-based variant of the classical online learning problem of tracking the best expert. In our setting, the actions of the algorithm and experts correspond to local moves through a continuous and bounded state space. At each step, Nature chooses payoffs as a function of each player's current position and action. Our model therefore integrates the problem of prediction with expert advice with the stateful formalisms of reinforcement learning. Traditional no-regret learning approaches no longer apply, but we propose a simple algorithm that provably achieves no-regret when the state space is any convex Euclidean region. Our algorithm combines techniques from online learning with results from the literature on *pursuit-evasion games*. We describe a natural quantitative trading application in which the convex region captures inventory risk constraints, and local moves limit market impact. Using historical market data, we show experimentally that our algorithm has a strong advantage over classic no-regret approaches.

1. Introduction

A well-studied setting in online learning is that of prediction with expert advice. In the classic model, at each time step, an algorithm receives advice from a set of experts, and must then choose an action. Both experts and algorithm receive some payoff, chosen exogenously by Nature, as a function of their action. The goal of the algorithm is to compete favorably with the best expert in hindsight. This setting is *stateless*, in the sense that previous action choices have no effect on the current actions available to a player, nor on the payoff it receives. Here we consider a state-based variant on the model in which these properties no longer hold. The experts and algorithm now operate in a continuous and bounded state space, and their actions correspond to local moves within this state space. Players are forbidden to choose actions that would take them outside the state space. Furthermore, payoff is determined as a function of both state and action.

There are many natural motivations for this new model. In particular, we shall describe in detail an application in quantitative trading, but we also imagine that our setting would be appropriate for a variety of control and optimization problems, such as those studied in robotics.

Our model's dependence on state is reminiscent of reinforcement learning, with the key difference being that we allow arbitrary payoffs in each round. Thus, whereas the goal in reinforcement learning is to compete with the best fixed policy, here we can only hope to compete with the best expert in hindsight. Our goal is to recover standard no-regret guarantees, but due to the state constraints, classic methods no longer apply. We provide an algorithm that can achieve no-regret in any state space in which the algorithm can "capture" the best expert. In particular, we consider convex Euclidean regions in $\mathbb{R}^n$, though capture is possible in other domains as well (Alexander et al., 2006). Our approach is a hybrid that takes advantage of algorithms for two distinctly different problems: "lazy" online prediction, in which an algorithm should not change the expert it is following too often, and pursuit-evasion games.

We present a detailed application of our framework and algorithm to a natural and common quantitative trading problem. States represent *inventory positions* — that is, the number of shares (long or short) held in each of a number of stocks or other financial assets. The allowed states obey a convex constraint that captures risk limits, which are measured by the volatility or variance of the inventory position. The shape of this region is elliptical, and determined by the correlations between the assets. A player's movement through the state space corresponds to trades (long or short in each stock), and each move is constrained to be small (local) in order to limit market impact. At each step,

Proceedings of the 31^{st} *International Conference on Machine Learning*, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the authors.

the payoff of a player is the sum of the cost of its current trade (which could be positive or negative, depending on whether shares are bought or sold in each stock) and the change in the liquidation value of its inventory position.

We experimentally demonstrate that our algorithm achieves higher payoff than traditional no-regret alternatives on a large dataset of prices of two highly correlated exchange-traded funds (ETFs), SPY and IWM. We consider two types of experts, *directional* and *relative*, each of which receives (possibly noisy) signals about the future returns of each fund. These experts model profitable trading strategies that flourish under differing conditions: whereas the directional expert thrives when the underlying stocks exhibit strong momentum (positive or negative) of returns with low volatility, the relative expert deliberately exploits the correlation of stocks to maintain more hedged long/short or short/long positions. Using these two experts as the substrategies, our algorithm significantly outperforms (state-constrained variants of) traditional no-regret approaches on average, and under certain conditions, our algorithm has a sizeable advantage.

2. Related Work

There is a large literature on the problem of prediction with expert advice, but little work has been done to extend the model to state-based settings. A notable attempt was made by de Farias and Megiddo (2006), who also consider an experts problem in which the algorithm's past choices affect the current state of the environment. A similar reactive setting was analyzed by Poland (2008). Both of these models assume that all actions are available to the algorithm in all states, which is the main difficulty overcome by our approach.

A restriction on the set of allowed actions is considered in *online convex optimization*. Here the distribution over actions must obey a convex constraint in the simplex. In this case, a variant of the multiplicative weights (MW) algorithm can be used to achieve no-regret to the best fixed action in hindsight (Arora et al., 2012). There has been more recent work along these lines, such as an algorithm for online learning with general polytope constraints (Banerjee & Wang, 2012). These approaches are quite distinct from ours, where there is the further restriction to only local moves, and in which we achieve no-regret to the best *expert*, who is allowed to choose a new action in each round.

3. Preliminaries

Our setting is as follows. We have an algorithm a and a set of k experts, $e_1, \ldots, e_k$. All players are constrained to the convex region $S \subset \mathbb{R}^n$. We denote the current state of an expert i and algorithm a at time t by the n-dimensional vectors $s(i)^t, s(a)^t \in S$, respectively. First each expert i chooses an action $x(i)^t$ and transitions to the new state $s(i)^{t+1} = s(i)^t + x(i)^t$. We restrict the magnitude of each action to $\|x(i)^t\| \leq \epsilon$, for some fixed ϵ. Additionally, we require $s(i)^{t+1} \in S$, else $x(i)^t$ is a forbidden action. After witnessing these moves, the algorithm chooses an action $x(a)^t$, where $\|x(a)^t\| \leq \epsilon$, and makes the corresponding transition. Again, we require $s(a)^{t+1} \in S$. Finally, for each state-action pair (s, x), Nature reveals a payoff $p^t(s, x)$. As is standard, we require that these payoffs belong to a bounded interval $[-b, b]$.

Additionally, our results will require that Nature chooses the payoff function for each time step *before* the game begins. Thus, a player's payoff is only a function of its current state and action, and does not depend on its past behavior. This is known as the "oblivious" or non-adaptive model, and prevents Nature from adversarially adapting to the choices of the algorithm. Other than this non-adaptivity constraint, the sequence of payoffs chosen by Nature is arbitrary, and may not exhibit any regularity or obey any statistical model.

The cumulative payoff of algorithm a is denoted $P_a = \sum_{t=1}^{T} p^t(s(a)^t, x(a)^t)$. The cumulative payoff of an expert i is denoted P_i and defined similarly. The *regret* R of a is the difference between the total payoff of the algorithm and the total payoff of the best expert in hindsight, i.e. $R = \max_i P_i - P_a$. The average per-step regret is therefore R/T. As is standard, we say a is a *no-regret* algorithm if the per-step regret goes to zero in the limit of T, or equivalently, if R is sublinear in T.

Standard no-regret approaches have no concept of state, and therefore cannot hope to perform well in our setting. In fact, it may be impossible even to run a classic no-regret algorithm in our model, because the algorithm may try to take an action that is not available in its current state. In the next section, we present an approach that extends a stateless no-regret algorithm by introducing phases in which adjustments are made to account for state constraints.

4. Main Algorithm

A traditional approach to prediction with expert advice is to "follow the leader." At each time step, the algorithm determines which expert currently has maximum cumulative payoff, and then copies the action of this "leader." To ensure that this algorithm achieves no-regret, it is first necessary to perturb the cumulative payoffs with random noise. This algorithm is known as Follow the Perturbed Leader (FPL) and has a regret guarantee of

$$P_{\text{FPL}} \geq (1 - \eta) P_e - \frac{b \log(k)}{\eta},$$

where η is a parameter specifying the algorithm's learning rate and P_e is the payoff of the best expert (Kalai & Vempala, 2005).

There are two main problems with running FPL in our modified setting. First, consider a step at which the leading expert changes. If FPL is not in the same state as this expert, then the algorithm may not be able to copy the expert's action. In particular, if FPL is near the boundary of the convex region, then any move that would carry the algorithm out of the region is forbidden. Thus, FPL can become stuck, and must wait until the leading expert chooses an action that can be safely copied. In the intervening steps, there is no guarantee on the algorithm's regret.

Second, recall that we have defined payoff as a function of both state and action. So even if the expert does choose an action available to FPL, the expert may be in a "better" state and therefore receive higher payoff. For this reason, it is easy to see that FPL, or any other standard no-regret algorithm, cannot achieve no-regret in a state-based model.

The problems described above arise only when the leading expert changes, but in general there is no limit on the number of such changes for FPL. In our state-based setting it will turn out to be important to minimize the frequency of this event. The Follow the Lazy Leader (FLL) algorithm is a modified version of FPL with the same regret guarantee, and has the additional property that the leading expert only switches ηbT times (Kalai & Vempala, 2005). Recall that η is the learning rate of the algorithm, and b is the bound on per-step payoff magnitude. Thus, when $\eta \in O(1/\sqrt{T})$ and $b \in o(\sqrt{T})$, the number of switches is $O(\sqrt{T})$. FLL achieves this behavior[1] by correlating the perturbations that are added to the cumulative payoffs of the experts before determining the leader. Initially, FLL picks a random set G of n-dimensional points spaced exactly $1/\eta$ apart. Then at each step t, if P^t is a vector of length k denoting the cumulative payoffs of each expert, the vector of perturbations z^t is chosen so that $P^t + z^t$ rarely differs from $P^{t-1} + z^{t-1}$. More precisely, the vector z^t is chosen so that $P^t + z^t$ is equal to the unique point in the intersection of G with $P^t + [0, 1/\eta)^k$. It can be shown that the probability that this point changes from one round to the next is ηb. Thus, we expect a change to occur ηbT times.

Merely bounding the number of switches is still insufficient to achieve no-regret in our setting. Each time a switch occurs, we risk the problems of being unable to follow the new leader, and of following the leader from a different state, and therefore receiving different payoff. Our solution is to "capture" the new expert; i.e., chase it until we are both in the same state, and then resume copying its actions. There is a large body of literature on such *pursuit-evasion games*. We consider the version of this game in which one "pursuer" in a fixed domain is attempting to capture one "evader". Our notion of capture requires the pursuer to move to the exact location of the evader.[2]

This game can be played on both continuous and discrete domains. We restrict our analysis to an instance of continuous case, but our algorithm can in fact be applied to *any* domain on which efficient pursuit is possible. As enumerated in a survey by Chung et. al. (2011), this class includes a wide variety of graphs (Nowakowski and Winkler provide an algorithmic characterization (1983)) as well as all metric spaces satisfying the CAT(0) condition (Alexander et al., 2006).

In the continuous case, in each round, the evader first moves by a distance bounded by ϵ to a new position in the domain. The pursuer witnesses this choice and then moves by a distance also bounded by ϵ to its position of choice. We consider convex regions in $\mathbb{R}^n$, for which it is a known result (Kopparty & Ravishankar, 2005; Alexander et al., 2006) that the pursuer can capture the evader in a finite number of moves. By combining FLL with any pursuit algorithm that achieves this result, we obtain Algorithm 1, Pursuit-Evasion Without Regret (PEWR). At the top level, our algorithm runs a copy of FLL. When the leading expert changes, the algorithm uses a subroutine PURSUIT to reach the same state as the new expert. Once capture is achieved, the algorithm resumes running FLL.

There are at least a couple of alternative choices for the pursuit subroutine used. Perhaps the one with the cleanest and most complete theoretical analysis for our purposes is an algorithm known as SPHERES (Kopparty & Ravishankar, 2005), which we will assume to obtain our formal regret bound for PEWR in Theorem 1 below. For SPHERES, the number of steps required to catch an evader in a convex body of diameter[3] d is $O((d/\epsilon)^2)$. The quadratic dependence on d is in fact tight for certain initial configurations of the pursuer and evader (Sgall, 2001). Note that this bound has no explicit dependence on the dimensionality n, since the pursuer always travels along the direct line to the evader's current position.

For our experimental examination of PEWR (discussed in Section 6), rather than using SPHERES for the pursuit subroutine, we instead implement a simpler method of direct

[1] It is worth noting that there are other algorithms that also achieve limited switching behavior while preserving the original no-regret guarantee. The "shrinking dartboard" algorithm of Geulen et. al. was the first to achieve such behavior (2010). Another solution was proposed more recently by Devroye et. al. (2013). We restrict our attention to FLL, but the alternatives would work equally well.

[2] This notion can be relaxed as long as we additionally require a continuity or Lipschitz condition on payoff functions.

[3] Defined as the longest Euclidean distance between two points in the body.

pursuit as described by Alexander et. al. (2006). Applied to our setting, the pursuer chooses $s(a)^{t+1}$ to be a point at most the allowed distance ϵ along a line from $s(a)^t$ (the algorithm's current position) to $s(i)^{t+1}$ (the pursued expert i's new position). In practice we expect this approach to be simpler and to perform equally well as SPHERES.

Algorithm 1 Pursuit-Evasion Without Regret (PEWR)

```
Input: convex region $S \subset \mathbb{R}^n$, set of $k$ experts, sequence length $T$, learning rate $\eta$, maximum step size $\epsilon$
$s(a)^0 = s(i)^0 = s^0, \forall i \leq k$
$P(a) = P(i) = 0, \forall i \leq k$
Choose $p \in [0, \frac{1}{\eta}]^k$ uniformly {Initialization for FLL}
$G = \{p + \frac{1}{\eta} z \mid z \in \mathbb{Z}^k\}$
for $t = 1$ to $T$ do {Main loop}
  $s(i)^t \leftarrow s(i)^{t-1} + x(i)^t, \forall i \leq k$ {Experts move}
  $P(i) \leftarrow P(i) + p^t(s(i)^t, x(i)^t), \forall i \leq k$
  $g(i) \leftarrow G \cap (P(i) + [0, \frac{1}{\eta})), \forall i \leq k$
  $l \leftarrow \arg\max_i g(i)$ {Determine leader}
  if $s(a)^{t-1} == s(l)^{t-1}$ then {Use FLL}
    $x(a)^t \leftarrow x(l)^t$
  else {Use pursuit strategy}
    $x(a)^t \leftarrow$ PURSUIT$(S, \epsilon, s(l)^{t-1}, s(l)^t, s(a)^{t-1})$
  end if
  $s(a)^t \leftarrow s(a)^{t-1} + x(a)^t$ {Algorithm moves}
  $P(a) \leftarrow P(a) + p^t(s(a)^t, x(a)^t)$
end for
```

Theorem 1. *For any convex set $S \subset \mathbb{R}^n$ of diameter d, set of k experts, payoff bound b, maximum step size ϵ, and sequence length T, if the learning rate η is set to $1/\sqrt{T}$ and the pursuit subroutine used is SPHERES, then PEWR achieves a regret guarantee of*

$$P_{\text{PEWR}} \geq P_e - (2b\log(k) + b^2 c(d/\epsilon)^2)\sqrt{T},$$

where P_e is the payoff of the best expert.

Proof. Recall that the original FLL regret bound is

$$P_{\text{FLL}} \geq (1-\eta)P_e - \frac{b\log(k)}{\eta}.$$

As guaranteed by FLL, the expected number of times we switch experts is ηbT. As guaranteed by SPHERES, each time we switch, there are $c((d/\epsilon)^2)$ steps on which we can incur maximum regret of b, for some constant c. On every other step, PEWR receives the same payoff as FLL. It follows that the regret guarantee is

$$P_{\text{PEWR}} \geq (1-\eta)P_e - (b\log(k)/\eta) - (\eta bT)c(d/\epsilon)^2 b.$$

Choosing $\eta = 1/\sqrt{T}$ and using $P_e \leq bT$ yields

$$\begin{aligned} P_{\text{PEWR}} &\geq P_e - b\sqrt{T} - b\log(k)\sqrt{T} - b^2 c(d/\epsilon)^2\sqrt{T} \\ &\geq P_e - (2b\log(k) + b^2 c(d/\epsilon)^2)\sqrt{T}. \end{aligned}$$

□

The dependence on b and $\log(k)$ are standard; the term involving $(d/\epsilon)^2$ is the additional regret we may suffer due to switches of the leading expert and the ensuing costs of pursuit. If we view d as constant with respect to T, the above expression for cumulative payoff yields per-step regret to the best expert of $O(1/\sqrt{T})$, as desired.[4]

In Theorem 1, as the diameter d of the convex region decreases, the pursuit time decreases, and the algorithm's regret approaches that of standard FLL. We later present experiments that illustrate this trend, and show a significant boost in our algorithm's performance when run on tighter constraints. The experiments also demonstrate the advantage of our algorithm over a "constrained" version of FLL.

5. Application: Trading with Inventory Risk

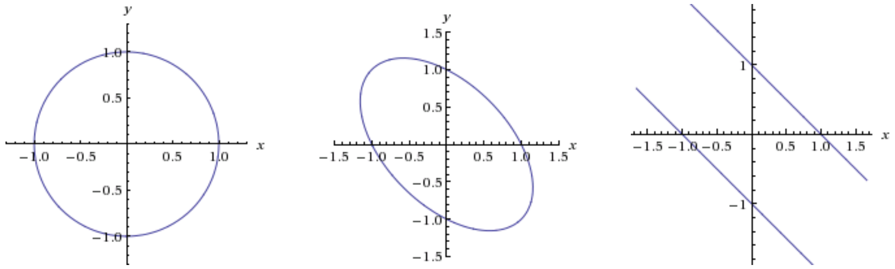
The theory developed above can be directly applied to a natural and common quantitative trading problem. In each time step, a player will choose to buy or sell[5] shares in each of a set of stocks. The player's payoff is determined by the net cost or revenue of this trade, as well as the change in liquidation value (at current prices) of its inventory. As is standard in practice, trade sizes at each step are bounded in order to limit market impact, and inventory amounts are bounded to limit holding risk to acceptable levels. Note that even if the inventory restrictions are tight, a player may be able to accrue payoff by frequently "buying low and selling high" — profitable strategies involve repeatedly trading in and out of the same stocks. On the other hand, if inventory restrictions are loose, a player may profit from accumulating large inventory positions in anticipation of future upward movement. Ideally, a player would perform best by choosing between both strategies, according to current market circumstances.

To design an algorithm that competes favorably with a set of trading experts, we can directly map this setting to our state-based model. Consider a set of n stocks. The players move through a convex region $S \subset \mathbb{R}^n$, defined in "inventory space". That is, a player's position $s^t \in S$ is an n-dimensional vector representing its current holdings of each stock. The bounds of the region S represent the inventory limits. At time t, the price of stock i is $p_i^t \leq m$, for some upper bound m. Each player can choose to buy or sell x_i^t fractional shares of stock i. We restrict trade size to $\|x^t\| \leq \epsilon$, for some fixed ϵ, to limit market impact.[6] A

[4] Note that no-regret is still obtained (albeit at a slower rate) even if we allow the constraints to relax with T at rates obeying $d = o(\sqrt{T})$, which might be important in some applications.

[5] We permit the common practice of short selling, which can be viewed as borrowing shares to sell at current prices, with the requirement that the loan be repaid in shares at a later time.

[6] By market impact, we do not mean trading commissions and fees, but the well-known empirical phenomenon of adversarial



(a) $\rho = 0$ (b) $\rho = 0.5$ (c) $\rho = 1$

Figure 1. Effect of correlation coefficient ρ on inventory limits. (a) Uncorrelated stocks ($\rho = 0$): risk limit dictates that any combination of long/short inventory in the two stocks less than some total absolute position is permitted. (b) Moderate correlation ($\rho = 0.5$): larger absolute inventory is allowed for offsetting (hedged) long/short or short/long positions, while long/long and short/short positions are riskier and thus more restricted. (b) Perfect correlation ($\rho = 1$): long/long and short/short positions remain sharply curtailed, but sufficiently offsetting long/short and short/long positions of arbitrary volume still obey the risk limit.

positive value of x_i^t represents a purchase, and a negative value represents a sale.

The per-step payoff of a player is the sum of its *trading* payoff and *inventory* payoff. We define these quantities so that the player's cumulative payoff will be equal to its net expenditure plus the liquidation value of its inventory at the end of the series. The trading payoff received in a given step is the amount of cash spent or received in exchange for shares bought or sold, and is formally defined $p_{\mathrm{T}} = -p^t x^t$. That is, the player pays $p_i^t x_i^t$ when making a purchase of x_i^t shares of stock i, and earns $p_i^t x_i^t$ when making the analogous sale. The inventory payoff is the amount by which the player's inventory has changed in value over the last step, and is defined $p_{\mathrm{I}} = p^t s^t - p^{t-1} s^{t-1}$. The player's cumulative payoff is therefore $P = \sum_{t=1}^{T} -p^t x^t + p^t s^t - p^{t-1} s^{t-1} = p^T s^T + \sum_{t=1}^{T} -p^t x^t$, as desired.

We can now directly apply the result of Theorem 1 to obtain a bound on the regret of PEWR when run in this setting. Because $p_i^t \leq m$ and $\|x^t\| \leq \epsilon$, payoffs fall in the interval $[-m\epsilon, m\epsilon]$. Therefore, our dependence on b translates to a dependence on $m\epsilon$.

5.1. Inventory Risk as a Convex Region

Although the theory holds for any convex region S, our goal in this application is to define the state space such that the player is restricted to inventories of limited risk. We use the most common quantitative measure of the risk of an inventory position, which is the variance of the corresponding dollar investment. The variance of an investment in a set of stocks depends on the correlations between the prices of these stocks. For instance, if two stocks are highly correlated, then it is risky to hold long or short positions in both. However, a long position in one can be "cancelled out" by a short position in the other.

To encode this formally in the case of the two stocks, we proceed as follows. We first convert the price time series of each stock, denoted p_1 and p_2, to a series of returns, denoted r_1 and r_2, where $r_i^t = p_i^{t+1}/p_i^t - 1$. Let μ_i and σ_i denote the mean and variance of r_i, and let $\Sigma = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}$ denote the covariance matrix, where ρ is the correlation coefficient of the two stocks. Assuming joint normality of the returns series,[7] μ and Σ parameterize a 2-dimensional normal distribution over the data. Let (a, b) denote a random variable from this distribution, and let x and y denote the dollar value of an investment in each stock, respectively. The dollar value of an investment (x, y) is therefore $u = xa + yb$. Because u is a linear combination of the components of a normal distribution, u is also normally distributed, with mean $x\mu_1 + y\mu_2$ and variance $(x\ y)\Sigma(x\ y)' = \sigma_1^2 x^2 + \sigma_2^2 y^2 + 2\rho\sigma_1\sigma_2 xy$.

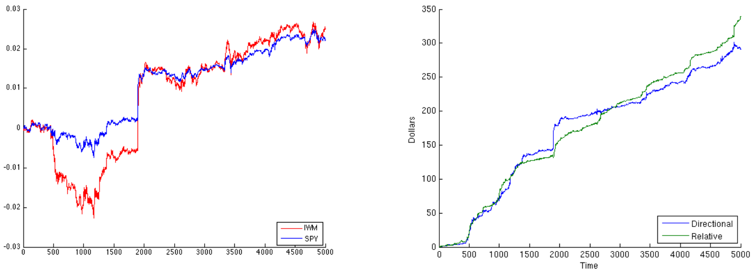
Thus, to upper bound the risk of an investment (x, y) by R, we use the constraint $\sigma_1^2 x^2 + \sigma_2^2 y^2 + 2\rho\sigma_1\sigma_2 xy \leq R$. This equation represents an ellipse centered at the origin. The effect of the correlation coefficient ρ on the shape of the ellipse is illustrated in Figure 1. The ellipses shown are for $\sigma_1 = \sigma_2 = 1$ and $R = 1$. As the correlation increases, inventories containing both long/long or short/short holdings become riskier, and so the constraint puts a tighter bound on such positions.

One remaining discrepancy is that this constraint is defined in dollar space, whereas the algorithm's inventory has thus far been defined in share space. To address this, we use the initial dollar prices p_1^1 and p_2^1 as a benchmark[8] to per-

price movements caused by trading large volumes in short periods.

[7] Normality is approximately obeyed by our experimental data. We note that our methods could be applied to other distributional models over returns.

[8] This is standard practice, since typically risk limits are determined in advance of the trading period of interest.



(a) Cumulative Returns

(b) Cumulative Payoffs

Figure 2. Performance of the directional and relative expert on a particular subsequence of the price data. The returns of the stocks on this period are shown in (a), and the payoffs of each expert in (b). At around time 2000, both stocks have steep positive returns. The directional expert can go long in both stocks, and is therefore better able to take advantage of the joint upward movement. In the remainder of the sequence, the directional expert is restricted by inventory constraints. The relative expert continues to make trades that exploit the small differences in returns, and becomes the new leader.

form a conversion. Then given a share inventory of (v, w), the corresponding dollar investment is $(p_1^1 v, p_2^1 w)$. We use this vector to obtain a constraint $\sigma_1^2 (p_1^1 v)^2 + \sigma_2^2 (p_2^1 w)^2 + 2\rho\sigma_1\sigma_2 p_1^1 p_2^1 vw \leq R$.

6. Experiments

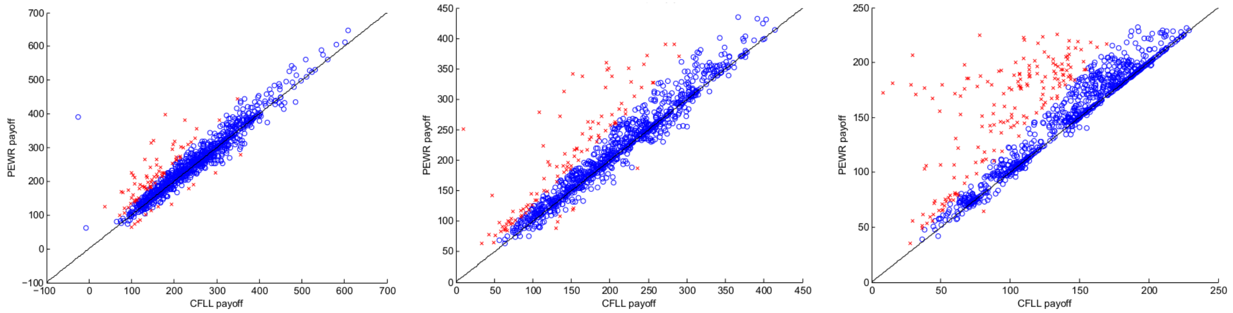
We analyze the performance of our algorithm on the application discussed above using a real dataset containing prices of two exchange-traded funds, the S&P 500 (SPY) and the Russell 2000 (IWM). These were chosen deliberately due to their high historical and structural correlation; the Russell 2000 index contains the S&P 500. The dataset contains 135,548 observations drawn from the months of October and November of 2013. Prices were sampled daily at 30 second intervals during regular market hours. The cumulative returns of each fund are shown in Figure 2. The correlation of the two raw price timeseries is 0.9642 and the correlation of returns is 0.7097.

Our experiments compare the performance of PEWR to a variant of the FLL algorithm, called Constrained FLL (CFLL in the sequel), that behaves as follows. When the leading expert changes and CFLL is not able to follow because of the region boundary, the algorithm remains stationary and waits until the leader makes a move that it can follow. During this waiting period, CFLL will still receive inventory payoff, but can make no trading payoff. We expect that the algorithm's regret will suffer during this period. However, because PEWR incurs regret while pursuing the new expert, it is unclear which approach is better. It may be the case that CFLL profits from valuable inventory, while PEWR wastes time during an unnecessary chase. As we see in our experiments, PEWR's approach is indeed preferable to that of CFLL, particularly when inventory constraints are tight.

6.1. Experts and Parameters

The experts used in our experiments reflect two different trading strategies. We imagine that each expert first receives some possible noisy signal about the future performance of the two stocks. More formally, for each stock $i \in \{1, 2\}$, the expert receives the return value $r_i^t = p_i^{t+1}/p_i^t - 1 + \gamma r$, where r is a random number chosen from $[-1, 1]$, and γ is a parameter used to scale the random noise. The *directional expert* buys stock i if $r_i^t > 0$ and sells stock i if $r_i^t < 0$. The magnitude of each transaction is chosen in proportion to r_i^t. The *relative expert* trades in proportion to the difference $d^t = r_1^t - r_2^t$ between the two returns. If $d^t > 0$, the expert buys stock 1 and sells stock 2, and if $d^t < 0$, the expert does the opposite. The value of the maximum allowed step size ϵ was chosen so that the typical trade size of the experts and algorithms was approximately two shares in magnitude. Additionally, the learning rate η of both algorithms was set to be 0.05.

Given that they receive signals on future returns, we expect both experts to be profitable, but in different ways. The directional expert has greater flexibility in the types of trades it is allowed to make, and has the advantage of being able to follow large upward or downward moves in prices. However, it will likely accumulate large inventory, which makes it vulnerable to constraint boundaries. The relative expert is forced to maintain a more conservative hedged position on the 45° line. Because SPY and IWM are strongly correlated, the differences in returns will be small and balanced, so the relative expert will remain close to the origin and maintain smaller inventory. As a result, we expect its payoffs to benefit from greater stability. An example illustrating the different behavior of each expert is shown in Figure 2. Because each expert has an advantage in different market circumstances, it is most interesting to consider an algorithm that will switch between them.



(a) $R = 1$. PEWR outperforms CFLL by 4.4% on average.

(b) $R = 0.5$. PEWR outperforms CFLL by 6.5% on average.

(c) $R = 0.1$. PEWR outperforms CFLL by 12.2% on average.

Figure 3. Effect of ellipse size on the comparative performance of PEWR and CFLL. Each scatterplot illustrates the same set of 1000 trials, each on a random price sequence of length 5000. The x axis measures the cumulative payoff of CFLL on each trial, while the y axis measures cumulative payoff of PEWR. Points above the diagonal are wins for PEWR. The red crosses highlight points where the winning algorithm outperforms the losing algorithm by more than 25%. The directional expert was parameterized with $\gamma = 0.002$ and the relative expert with $\gamma = 0$. (a) Under weak risk limits, PEWR only slightly outperforms CFLL, as long pursuit times handicap PEWR and CFLL may encounter constraints only infrequently. (b) Under moderate risk limits, pursuit times for PEWR are faster, and CFLL may frequently hit risk constraints that prevent it from following experts. (c) Under tight risk limits, PEWR significantly outperforms CFLL, which is frequently unable to follow the current leading expert, and is often in unfavorable inventory positions relative to PEWR. PEWR outperforms CFLL by 25% or more on 17.4% of trials, while the reverse occurs in only a single trial.

6.2. Results

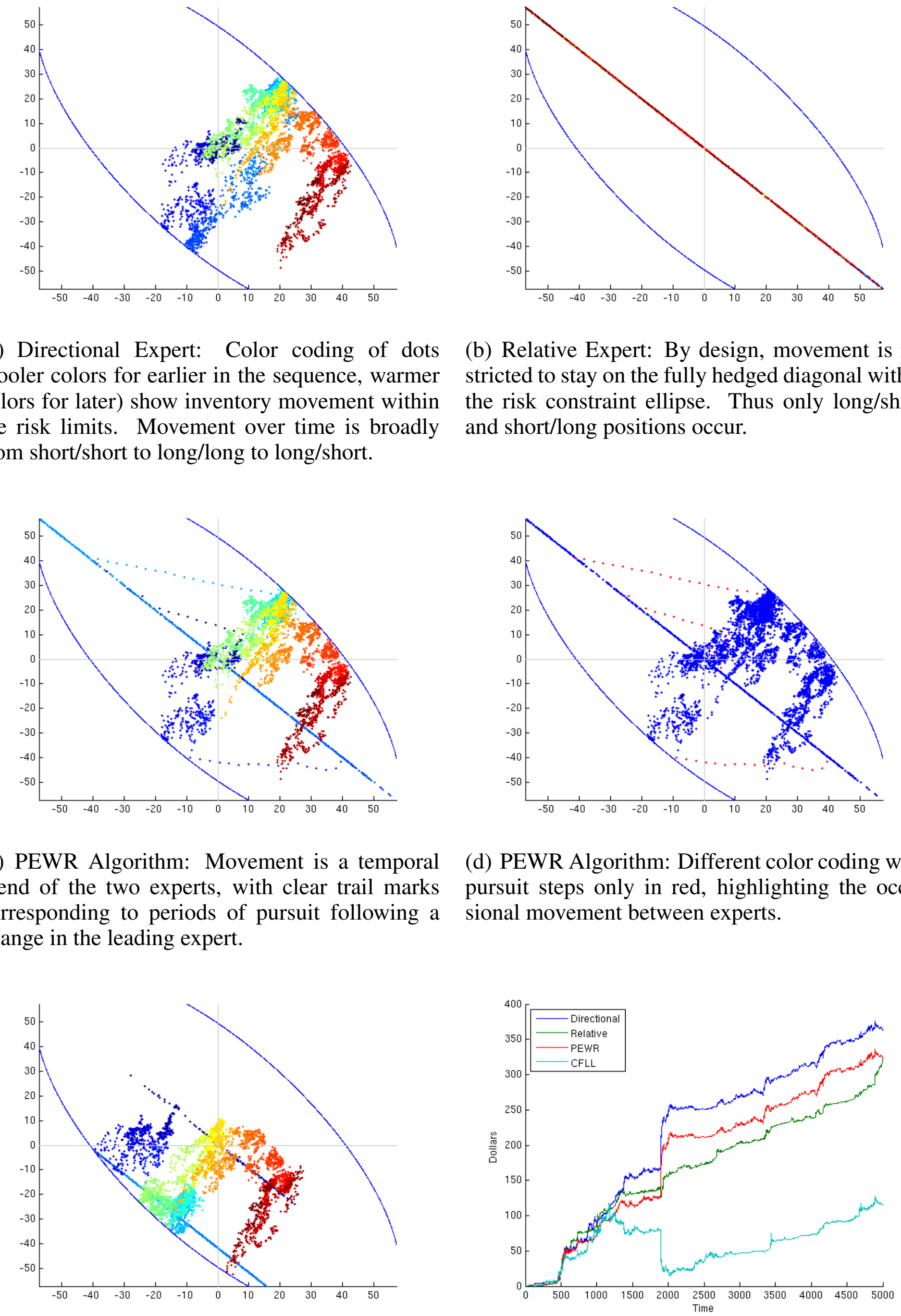
Our experiments demonstrate the advantages of PEWR over CFLL in a state-based setting. We compare the performance of each algorithm for varying ellipse sizes in Figure 3. These plots depict the results of 1000 trials, each on a random subsequence of length 5000, in which ellipse sizes were varied between $R = 1$, $R = 0.5$, and $R = 0.1$. In each case, there are many trials in which PEWR's payoff far surpasses that of CFLL. When CFLL does receive higher payoff, the difference is quite small. These trends become more pronounced as ellipse size decreases. When the state space is large, CFLL rarely runs into boundaries, but PEWR may incur unnecessary regret during its pursuit phases. As a result, PEWR's advantage at ellipse size $R = 1$ is modest. On the other hand, when inventory limits are tight, CFLL is often handicapped by the constraints, whereas PEWR can achieve capture quickly. Thus, at ellipse size $R = 0.1$, we see that PEWR outperforms CFLL by significant amounts in a large number of trials. See Figure 3 caption for details.

Figure 4 (on the following page) illustrates a typical trial, and draws attention to the differences in behavior of the two algorithms. We see that CFLL suffers from running into boundary constraints, as well as from following leading experts from inferior inventory positions. Because PEWR pursues each leading expert, both problems are avoided, and PEWR receives 2.8 times as much payoff as CFLL. Interestingly, the size of the ellipse used in this example is $R = 3$, a relatively loose risk constraint on which PEWR has only a small advantage on average. However, our experiments identified many subsequences of the dataset on which PEWR significantly outperforms CFLL, even on large ellipses. The main conclusion is that, while PEWR never loses to CFLL by much, there are certain conditions (such as particular price sequences or small ellipses) under which PEWR has a sizeable advantage.

Finally, we note a connection between PEWR's theoretical regret bound and our experimental results. Recall that the bound of Theorem 1 has a regret term arising from PEWR's worst case pursuit time. There is also a strong experimental correlation between PEWR's regret and the fraction of time that the algorithm spends in the pursuit subroutine. On average over 1000 trials, for ellipse size $R = 0.5$, the correlation coefficient is 0.5314.

7. Future Work

We believe the state-based model proposed here opens a wide variety of possible research avenues. One question is whether there exists an algorithm that can achieve no-regret in continuous state spaces in the non-oblivious adversary model. We can also generalize our setting to discrete state spaces in which players move through a finite state machine and actions are determined by a transition function. In this model, even more remains to be explored. As noted previously, our algorithm can easily be applied to state machines in which pursuit (or a slightly weaker notion) is possible. However, we do not yet have a complete characterization of the machines on which it is possible to achieve no-regret.



(a) Directional Expert: Color coding of dots (cooler colors for earlier in the sequence, warmer colors for later) show inventory movement within the risk limits. Movement over time is broadly from short/short to long/long to long/short.

(b) Relative Expert: By design, movement is restricted to stay on the fully hedged diagonal within the risk constraint ellipse. Thus only long/short and short/long positions occur.

(c) PEWR Algorithm: Movement is a temporal blend of the two experts, with clear trail marks corresponding to periods of pursuit following a change in the leading expert.

(d) PEWR Algorithm: Different color coding with pursuit steps only in red, highlighting the occasional movement between experts.

(e) CFLL Algorithm: Movement also switches between experts, but without pursuit as in PEWR. Thus we see multiple separate periods of following the relative expert's diagonal movement, but from the "wrong" inventory position, as well as periods where this causes CFLL to be against the risk constraint. Similarly, CFLL usually follows the directional expert from a different state as well.

(f) Cumulative Payoffs: As a result, CFLL performance badly lags PEWR and both experts.

Figure 4. Illustrative and typical example in which PEWR significantly outperforms CFLL. The ellipse size is $R = 3$, the directional expert is parameterized with $\gamma = 0.0001$, and the relative expert with $\gamma = 0$.

References

Alexander, S., Bishop, R., and Ghrist, R. Pursuit and evasion in non-convex domains of arbitrary dimension. In *Robotics: Science and Systems*. The MIT Press, 2006.

Arora, S., Hazan, E., and Kale, S. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

Banerjee, Arindam and Wang, Huahua. Online alternating direction method. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Chung, Timothy H., Hollinger, Geoffrey A., and Isler, Volkan. Search and pursuit-evasion in mobile robotics. *Autonomous Robots*, 31(4):299–316, November 2011. ISSN 0929-5593.

Devroye, L., Lugosi, G., and Neu, G. Prediction by random-walk perturbation. In *Proceedings of the Twenty-Sixth Conference on Computational Learning Theory*, pp. 460–473, 2013.

Farias, Daniela Pucci De and Megiddo, Nimrod. Combining expert advice in reactive environments. *Journal of the ACM*, 53(5):762–799, September 2006.

Geulen, S., Voecking, B., and Winkler, M. Regret minimization for online buffering problems using the weighted majority algorithm. In *Proceedings of the Twenty-Third Conference on Computational Learning Theory*, pp. 132–143, 2010.

Kalai, A. and Vempala, S. Efficient algorithms for online decision problems. *Journal of Compututer and System Sciences*, 71(3):291–307, October 2005.

Kopparty, S. and Ravishankar, C. V. A framework for pursuit evasion games in $\mathbb{R}^n$. *Information Processing Letters*, 96(3):114–122, 2005.

Nowakowski, Richard and Winkler, Peter. Vertex-to-vertex pursuit in a graph. *Discrete Mathematics*, 43(2-3): 235239, 1983.

Poland, Jan. Nonstochastic bandits: Countable decision set, unbounded costs and reactive environments. *Theoretical Computer Science*, 397(1-3):77–93, May 2008. ISSN 0304-3975.

Sgall, J. Solution of David Gale's lion and man problem. *Theoretical Computer Science*, 259(1-2):663–670, 2001.