Reinforcement learning for investment strategies with trading signals and transaction costs

Federico Giorgi Stefano Herzel
Paolo Pigato
Department of Economics and Finance
University of Rome "Tor Vergata"

Abstract. We apply Reinforcement Learning (RL) to determine a dynamic portfolio strategy in a financial asset subject to transaction costs and whose returns are predicable. To compare the performance of RL to a possible alternative, where an optimal strategy can be computed exactly, we consider a model where the forecasting factors are linear and the transaction costs are quadratic. Then we consider a more general and realistic setting where an optimal solution is not known and the use of RL is a viable alternative. We find that in a non-linear setting the RL agent outperforms the trader that approximates the market in a linear fashion and applies the theoretical optimum.

Keywords: Machine Learning; Dynamic Portfolios; Predictable Returns

1 Introduction

The most common method to approach a dynamic portfolio optimization problem is through the Bellman Equation. This is usually done by formulating an analytically tractable model for market dynamics, transaction costs, objective function. The classical result by Merton [?] finds the optimal strategy when the asset follows a geometric Brownian motion and there are no transaction costs. Gârleanu and Pedersen [?] extended it providing the solution for a dynamic policy when the asset price changes are linearly predicted by some mean reverting factors and the transactions costs are quadratic in the traded quantities.
For many practical situations, the model proposed by Gârleanu and Pedersen [?] presents some difficulties when the factors are not observable, or when their relation to the price changes is not linear or also when the transaction costs are not quadratic. In all such cases, it may be worthy to investigate whether RL is a viable alternative, as it should work under very general assumptions. In RL, an agent is interacting with the environment over time.

At each time, the agent observes the state of the environment and chooses an action; at the next time, the agent observes a new state of the environment and receives a reward for its action. Intuitively, RL algorithms provide a way to train the agent by letting it learn through positive reinforcement with the goal of maximizing its cumulative reward. Typically, the agent learns from the simulations of a large amount of "episodes", improving on the approximation of the value functions.
RL has already been shown to be applicable to construct dynamic strategies in finance: as for instance in [?], where an agent is trained to optimally trade a security following an Ornstein-Uhlenbeck process; or [?], where RL is applied to build a delta-hedging strategy for European options with geometric Brownian motion dynamics for the underlying, in the presence of transaction costs.
In this work, we propose an actionable RL algorithm that is able to train an agent to optimally trade a security with signal-predictable price changes. Our algorithm, inspired by the algorithm proposed in [?], combines the standard SARSA (state-action-reward-state-action) algorithm with a neural network to estimate the value function. It is able to consider a continuous set of states (current shares, market realization) and actions (shares to trade). The value function is estimated by generating consecutive "batches" of episodes: within each batch, the agent trained on previous batches is initially used, and then its choices are gradually improved via simulations of new rewards.
We consider two case studies. In the first one we use the same factor model as Gârleanu and Pedersen [?] and we observe that the RL algorithm produces a strategy whose performances are comparable to those of the theoretically optimal strategy. In the second case study we consider non-linear factor models and factors and we find that the RL agent outperforms an agent that linearizes the market dynamics in order to apply the GP strategy.
In Section **??** we briefly revisit the RL theory and describe the modified SARSA algorithm that we apply to our framework. In Section **??** we look at the applications to dynamic trading with transaction costs. In Section **??** we discuss some further developments that are currently being investigated.

2 Reinforcement Learning

In this section we review the main elements of the RL application.

2.1 The problem

In all RL problems, an agent is interacting with the environment over time. At each time, the agent is willing to take actions in the environment in order to maximize a notion of reward. At each time t, the agent observes a representation of the environment in the form of a state $s_t \in \mathcal{S}$, where $\mathcal{S}$ is a state space. Then, the agent selects an action $a_t \in \mathcal{A}(s_t)$. At time $t+1$, the agent receives a reward R_{t+1} from the environment, and it observes the

updated state of the world s_{t+1}. Both reward and new state are consequences of the action selected at the previous period.
In full generality, the actions are determined by means of a policy $a_t = \pi(s_t)$ and the reward obtained by selecting the action a_t after observing the state s_t is given by a reward function $R_{t+1} = R(s_t, a_t)$.
The dynamics of states, actions and rewards is given by a Markov decision process (MDP)

$$\mathbb{P}[s_{t+1} = s, R_{t+1} = r | s_0, a_0, s_1, a_1, r_1, \dots s_t, a_t, r_t] = \mathbb{P}[s_{t+1} = s, R_{t+1} = r | s_t, a_t] \tag{1}$$

The agent's goal is to maximize the expected cumulative reward

$$\pi^* = \operatorname*{argmax}_{\pi} \mathbb{E}_0[\sum_{t=1}^{T} \gamma^{t-1} R_t] \tag{2}$$

The sum in (**??**) can be finite ($T < +\infty$) or infinite ($T = +\infty$) and the constant $\gamma \in (0, 1]$ is a discount factor. In [**?**] it is discussed how expected utility maximization problems can be formulated as (**??**).
To each policy π we can associate a state-action value function, which expresses the value of starting in state s, selecting action a and following the policy π thereafter

$$q_\pi(s, a) = \mathbb{E}[\sum_{t=1}^{T} \gamma^{t-1} R_t | s_0 = s, a_0 = a] \tag{3}$$

where $\mathbb{E}$ denotes the expectation computed under the assumption that the policy π is followed, i.e. that $R_{t+1} = R(s_t, \pi(s_t))$. The optimal state-action value function is the value function corresponding to the solution π^* of the RL problem (**??**)

$$q_*(s, a) = \max_{\pi} q_\pi(s, a) = q_{\pi^*}(s, a) \tag{4}$$

and it is the best expected value in (**??**) that the agent can achieve across all policies, given the environment and the MDP. The optimal value function is solution to the Bellman equation

$$q_*(s, a) = \mathbb{E}[R + \gamma \max_{a' \in \mathcal{A}(s')} q_*(s', a') | s, a] \tag{5}$$

where the expectation is computed according to the transition law (**??**).
To each state-action value function q in (**??**) we can naturally associate its q-greedy policy, which consists in choosing the action a that, in a given state s, gives rise to the best value

$$\pi^q_{greedy}(s) = \operatorname*{argmax}_{a \in \mathcal{A}(s)} q(s, a) \tag{6}$$

The solution to the RL problem (**??**) does not need to be unique. The greedy policy (**??**) associated to the optimal value function (**??**) is *a* solution to the RL problem (**??**).

There are two main families of approaches to the solution of the RL problem (??). The first family are the policy search methods, where a parametrization $\pi_{\boldsymbol{\theta}}$ for the policy is assumed and the optimization is done directly in the policies space

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\, q_{\pi_{\boldsymbol{\theta}}}(s, \pi_{\boldsymbol{\theta}}(s))$$

We refer to [?] for a survey. Policy search methods have been used in [?] for trading and in [?] and [?] for hedging. These methods are robust on noisy datasets and typically fast converging. Furthermore, it is possible to build algorithms with safety guarantees that ensure an improvement of the policy at each update (see [?], [?], [?], [?]). However, policy search methods are often complex to implement and very objective-specific.
The second family, that is the one adopted in this paper, are value-based algorithms, which consist on approximating the optimal value function (??) and then taking the corresponding greedy policy (??)

$$\hat{q}(s,a) \approx q_*(s,a) \qquad \Rightarrow \qquad \pi^{\hat{q}}_{\text{greedy}}(s) = \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}}\, \hat{q}(s,a)$$

See [?] for an overview. The estimation of the optimal value function is typically done in a simulated environment, where the MDP assumption is used to generate a large number of paths, or "episodes", which are used to approximate the solution of the Bellman equation (??). Value-based algorithms have been used in [?] and [?] for hedging. Value-based algorithms are easy to implement for arbitrarily complex reward functions and MDPs. However, they are sensitive to hyperparameters and they need a large number of simulations to converge, making them hard to stress-test (see [?]).

2.2 The SARSA algorithm

The SARSA (state-action-reward-state-action) algorithm is an instance of value based methodology. It was introduced in [?], and refined in [?], to estimate the optimal value function q_* on discrete state-action spaces. In [?] a modified version of SARSA, based on the methodology discussed in [?], is used to train an agent to delta-hedge options.
In SARSA batch learning, the training is done on consecutive batches of experiments. Within each batch, a simulator is used to generate episodes $(s_0^{(j)}, a_0^{(j)}, s_1^{(j)}, a_1^{(j)}, r_1^{(j)}, \dots)$ of the MDP (??).
When selecting $a_t^{(j)}$ from $s_t^{(j)}$, the algorithm must balance exploration and exploitation. Exploitation consists in making use of the current best estimate of the optimal value function to allow the agent to select optimal actions during the simulation of the episodes. Since the estimate is not accurate, especially in the first phase of the training procedure, the agent is allowed to take random actions to explore other areas of the action space, in order not to overfit a poor estimate. This is called exploration, and it is done by considering the ϵ-greedy

policy

$$\pi^{\hat{q}}_{\epsilon\text{-greedy}}(s) = \begin{cases} \tilde{a}, & \text{if } u < \epsilon \\ \text{argmax}_a \, \hat{q}(s,a), & u \geq \epsilon \end{cases} \tag{7}$$

where $\epsilon \in (0,1)$, u is uniformly sampled on $(0,1)$ and $\tilde{a} \in \mathcal{A}(s)$ is a random action.
The episodes are used to estimate the optimal value function q^* by iteratively solving the Bellman equation (**??**) on a grid of points. Finally, the value function is approximated by means of a supervised regression model that is defined on continuous state-action variables. At the next batch, ϵ is reduced and the estimate $\hat{q}$ obtained at the previous batch is used to generate new episodes.
We summarize the main steps of the algorithm.

```
Initialize $\hat{q}^{(0)}$. For $n = 1, \ldots, N$ (number of batches)
    1 For $j = 1, \ldots, J$ (number of episodes)
        a Simulate state $s_0^{(j)}$, compute $a_0^{(j)} = \pi^{\hat{q}^{(n-1)}}_{\epsilon\text{-greedy}}(s_0^{(j)})$
        b For $t = 1, \ldots, T$
            i Receive reward $R_t^{(j)} = R(s_{t-1}^{(j)}, a_{t-1}^{(j)})$ and simulate new state $s_t^{(j)}$
            ii Compute new action $a_t^{(j)} = \pi^{\hat{q}^{(n-1)}}_{\epsilon\text{-greedy}}(s_t^{(j)})$
            iii Update value function
                $q_{t-1}^{(j)} = R_t^{(j)} + \gamma \hat{q}^{(n-1)}(s_t^{(j)}, a_t^{(j)})$
    2 Fit supervised regressor
        $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}} \sum_{t,j} (q_t^{(j)} - q_{\boldsymbol{\theta}}(s_t^{(j)}, a_t^{(j)}))^2$
    3 Define estimate for the $n$-th batch as
        $\hat{q}^{(n)} := q_{\boldsymbol{\theta}^*}$
    4 Decrease $\epsilon$
Output $\hat{q}^{(N)} \approx q_*$
```

3 Dynamic trading with predictable returns and transaction costs

The modern theory of dynamic asset allocation, or multiperiod portfolio choice, was initiated in [**?**] and [**?**] and is based on the assumption that a rational investor is willing to maximize the

present value of the utility stemming from the strategy final wealth w_T. The wealth evolution is stochastic; however, the investor is capable of influencing the wealth increments δw_t by means of its decisions, or policy. Such utility maximization problems are typically addressed via the theory of dynamic programming by solving the associated Hamilton-Jacobi-Bellman equations.
In [?] an analytically tractable model for dynamic asset allocation is proposed where trading is costly. A S-dimensional vector of securities price changes, in excess of the risk-free return $x_{t+1} = p_{t+1} - (1 + r^f)p_t$, is linearly predicted by means of a K-dimensional vector of factors

$$\begin{aligned} x_{t+1} &= Bf_t + u_{t+1} \\ \Delta f_{t+1} &= -\Phi f_t + \varepsilon_{t+1} \end{aligned} \tag{8}$$

where it is assumed that Φ is such that the factors are stationary and the noise is Gaussian and uncorrelated $u_t \sim N(0, \Sigma)$ i.i.d., $\varepsilon_t \sim N(0, \Omega)$ i.i.d., $u_t \perp \varepsilon_t$. The objective is to determine the strategy π, as defined by the shares $n_0, \ldots, n_{T-1}$, that maximizes the risk-cost penalized present value of the future expected gains (GP problem)

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}_0 \{ \sum_{t=1}^{T} \gamma^{t-1} \left(\gamma (n'_{t-1} x_t - \frac{\kappa}{2} n'_{t-1} \Sigma n_{t-1}) - c(\Delta n_{t-1}) \right) \} \tag{9}$$

In [?], Gârleanu and Pedersen find a solution to (??) for quadratic trading costs $c(a) = \frac{\lambda}{2} a' \Sigma a$, which can be expressed as a convex combination of a static strategy and an "aim" portfolio

$$n_t = GP_t := (1 - \eta)\, n_{t-1} + \eta \times aim_t \tag{10}$$

where the "trading rate" η depends on the model parameters. The solution in (??) has a nice interpretation[1]: when trading costs are extreme, or $\lambda \to +\infty$, the solution converges to the static strategy $n_t = n_{t-1}$; when there are no trading costs, or $\lambda \to 0$, the solution converges to the static Markowitz solution [?]

$$n_t = Markowitz_t := (\kappa \Sigma)^{-1} B f_t \tag{11}$$

The aim portfolio in the general solution in (??) is a weighted combination of the expected values of the future Markowitz portfolios (??). We refer to the original paper [?] for more details.

3.1 The reinforcement learning approach

The target definition in (??) naturally leads to a RL problem as in (??). The state space must contain all the relevant information the agent needs to observe at time t in order to

[1] Assuming no discount for simplicity, i.e. $\gamma = 1$

make a decision, or action, a_t; at the same time, the state space must be "minimal" in order not to overfit the problem. With this in mind, we define the state space as

$$\mathcal{S} = \{(f, n) | f \in \mathbb{R}^K, n \in \mathbb{Z}^S\} \tag{12}$$

where f is the current observed value of the signal and n is the current position on the securities[2]. For a given state $s_t = (f_t, n_{t-1}) \in \mathcal{S}$, the action a_t determines the number of shares to trade

$$\mathcal{A} = \mathbb{Z}^S \tag{13}$$

3.2 Case study 1

In our first experiment, we verify that the RL setting (Section **??**) can replicate the optimal solution (**??**). To do so, we need to define a reward function stemming from the trade in such a way that the expected cumulative reward maximization problem (**??**) coincides with the GP problem (**??**). The reward function we choose is given by

$$R_{t+1} = \gamma \left(n_t' B f_t - \frac{\kappa}{2} n_t' \Sigma n_t \right) - c(\Delta n_t) \tag{14}$$

We consider the West Texas Intermediate (WTI) spot price[3] and we assume that the investor is following a simple momentum strategy by considering a 5 periods moving average of the P&L

$$f_t = \frac{1}{5}(x_t + x_{t-1} + x_{t-2} + x_{t-3} + x_{t-4}) \tag{15}$$

The model (**??**) calibration yields the parameters

$B = -0.066$	$u_t \sim N(0, 1.755)$
$\Phi = 0.226$	$\varepsilon_t \sim N(0, 0.131)$

We consider a trading period of $T = 50$ days. The costs parameter is assumed to be $\lambda = 10^{-2}$, which yields a cost for unit trade of $c(1) \approx \$0.01$. We assume a risk aversion parameter of $\kappa = 10^{-3}$ and a 2% continuously compounded annualized rate, providing a daily discount factor equals to $\gamma = e^{-2\%/252}$.
We train the agent on $N = 6$ consecutive batches; in each batch, $J = 15,000$ episodes are generated by using the MDP (**??**). The epsilon-greedy policy starts with a parameter $\epsilon = 10\%$, which is then decreased as $\epsilon \leftarrow \epsilon/3$ at each batch iteration (see Table **??**).
The action space $\mathcal{A}$ in (**??**) is not numerically manageable in practice. In order to approximate it with a space that is large enough to contain the optimal actions, we simulate several

[2]Fractional shares can be considered upon choosing $n \in \mathbb{R}^S$

[3]Source: FRED (Federal Reserve Economic Data) - Economic Research Division - Federal Reserve Bank of St. Louis. Daily data from Jan-1986 to Jul-2019

Batch	ϵ	$\mathbb{E}_0[\sum_{t=1}^{T}\gamma^{t-1}R_t]$
1	10.000%	-1,297.497
2	3.333%	-14.624
3	1.111%	-0.090
4	0.470%	1.951
5	0.124%	1.760
6	0.041%	3.119

Table 1: We report the evolution of the training phase across batches.

episodes where the Markowitz strategy (??) is applied and we pick the realized maximum trade

$$\mathcal{A} \approx \{-M, \ldots, M\}, \quad M = \max_{t,j} |\pi^{Markowitz}(n_{t-1}^{(j)}, f_t^{(j)})|$$

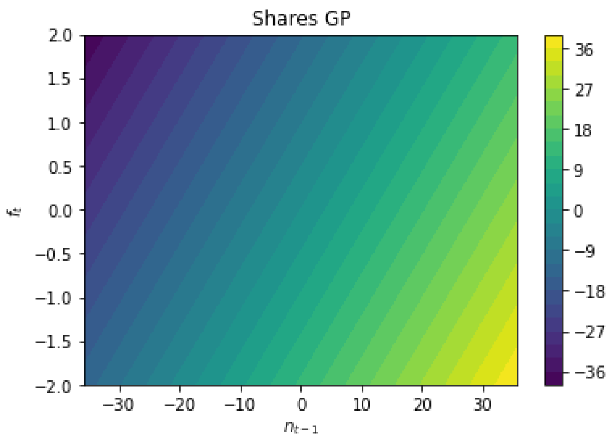
Indeed, we assume that the trades stemming from the Markowitz strategy, which does not take into account trading costs, are an upper bound for the optimal strategy (??).
During the creation of the episodes, the maximization defining the ϵ-greedy policy (??) is addressed by using three global optimizers (SHGO [?], dual annealing [?], differential evolution [?]) and picking the action that gives rise to the highest value. We tested a brute force method where the function's value is computed on a dense grid on point in order to find the optimum, but the computational time proved to be unfeasible[4].
In Figure ?? we show the shape of the function $s_t \to n_t(s_t)$ defining the shares owned by the investor while following the optimal strategy (??) and those owned by the agent trained with RL.
In order to backtest the strategies, we have considered a trading period of T days on the historical time series and we have compared the optimal, Markowitz and RL policies. For each of the strategy, we have computed the realized cumulative risk-cost penalized present value of the future expected gains, which we call wealth[5].

$$w_t = \sum_{s=1}^{t} \gamma^{s-1} \Big(\gamma (n'_{s-1} x_s - \frac{\kappa}{2} n'_{s-1} \Sigma n_{s-1}) - c(\Delta n_{s-1}) \Big) \tag{16}$$

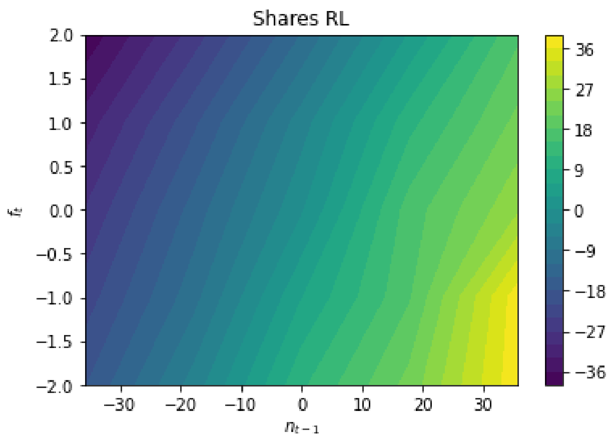
and costs. We can see in Figure ?? how the Markowitz strategy is more volatile than the optimal and RL strategies, as it does not consider transaction costs, resulting in a lower cumulative wealth. We see how the RL strategy closely resembles the optimal strategy.
We have simulated $J = 10,000$ paths and tested the hypothesis that the expected final wealth of the optimal and RL agents were equal with a two-sided Welch's t-test, which

[4]With the given setting, we must perform $N \times J \times T = 4,410,000$ global optimizations

[5]In reality, the "wealth" is a misnomer for the quantity (??), as it does not necessarily denote the true value of the investor's position. A more correct name would be "satisfaction". However, we stick to the nomenclature "wealth" as in related literature, see e.g. [?] or [?]



(a) GP strategy: shares



(b) RL strategy: shares

Figure 1: In Figure **??** we show the next-step shares provided by the optimal solution (**??**). In Figure **??** we show the shares owned by the RL agent given the state $s_t = (f_t, n_{t-1})$.

resulted in not rejecting the null hypothesis of equality

$$\begin{aligned}&H_0 : \mathbb{E}\{w_T^{RL}\} = \mathbb{E}\{w_T^{GP}\}\\&H_1 : \mathbb{E}\{w_T^{RL}\} \neq \mathbb{E}\{w_T^{GP}\}\\&t\text{-statistic} = -1.046, p = 0.296\end{aligned}$$

We then studied the linear relation $w_T^{RL} = \alpha + \beta w_T^{GP} + noise$, which yields

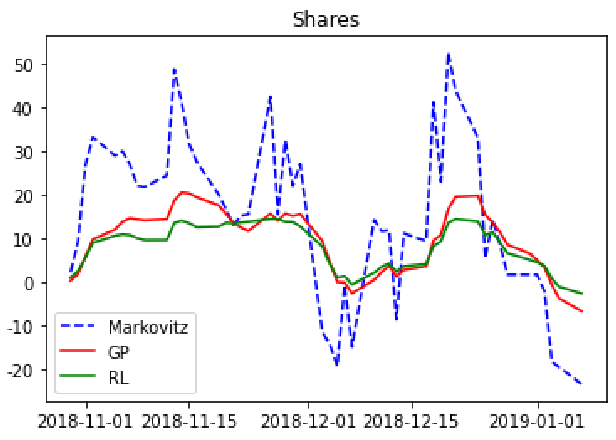
$$\begin{aligned}&\alpha = \underset{(-7.760,-4.184)}{-5.972} && H_0 : \beta = 1\\&\beta = \underset{(0.984,1.026)}{1.005} && H_1 : \beta \neq 1\\&R^2 = 0.466 && t\text{-statistic} = 0.487, p = 0.626\end{aligned}$$

We visualize these results in Figure **??**. We can see how the scatter plot is well arranged along the 45° line, showing that the random variables w_T^{RL}, w_T^{GP} are very similar, up to statistical error (as represented by the dispersion or the residuals *noise*) and up to the optimality of the GP solution (as represented by the negative intercept α).

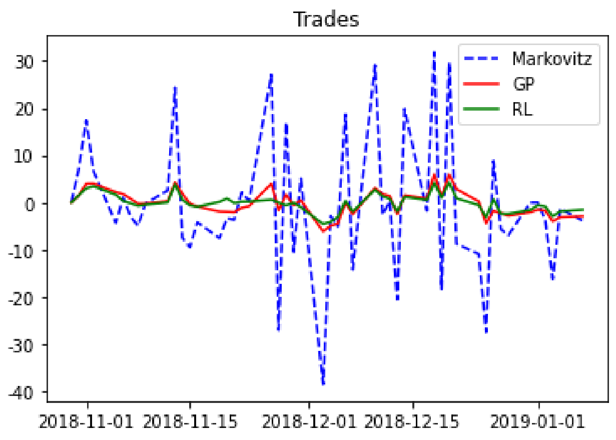
3.3 Case study 2

In our second experiment, we want to stress test the robustness of the RL approach to model risk if compared to the GP solution. We consider a market where the price changes in excess of the risk-free return is non-linear in the factors

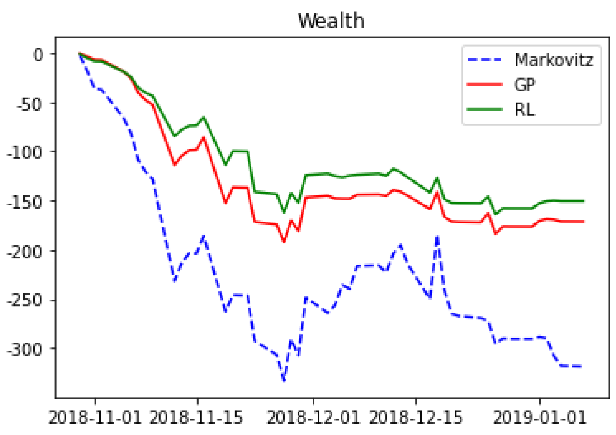
$$\begin{aligned}x_{t+1} &= g(f_t) + u_{t+1}\\\Delta f_{t+1} &= -\Phi f_t + \varepsilon_{t+1}\end{aligned} \tag{17}$$



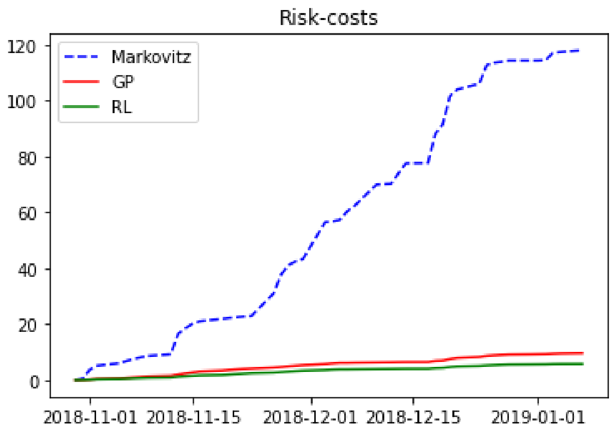
(a) Shares



(b) Trades



(c) Wealth

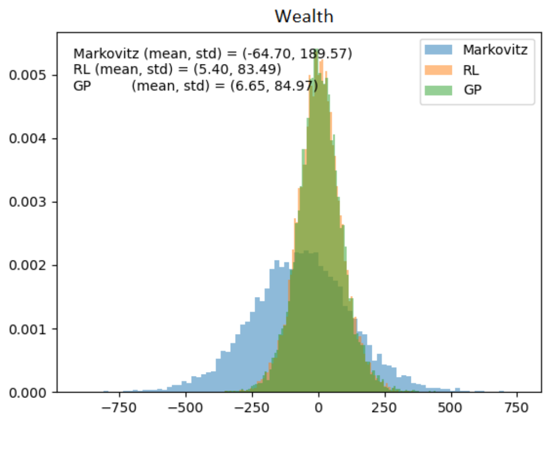


(d) Risk-costs

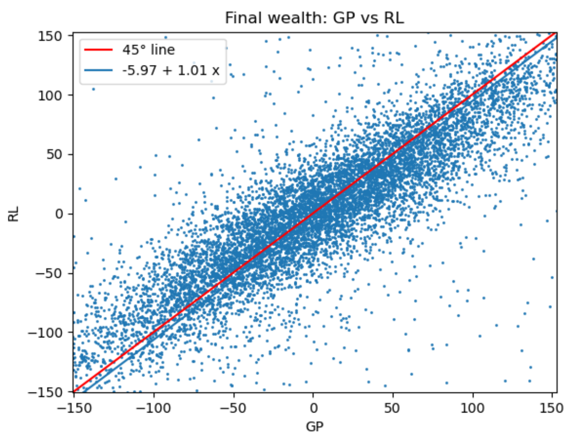
Figure 2: In Figure **??** we show the evolution of the portfolio position according to Markowitz, GP and RL. In Figure **??** we plot the trades implemented by the three policies. In Figure **??**-**??** we see the evolutions of wealth and risk-costs components.

where g is a non-linear function and the same assumptions as in (**??**) on the white noises and mean reversion hold. We visualize a scatter plot of factors and price changes in Figure **??**. We want to compare the strategies of two traders: the first trader (GP) will linearize the market structure (**??**) as $g(f_t) \approx Bf_t$ (the red line in Figure **??**) and it will apply the GP solution (**??**); the second trader (RL) will fit a non-linear function on the market structure (**??**) as $g(f_t) \approx g_\theta(f_t)$ (the black line in Figure **??**) and it will apply RL. For trader (RL), it is sufficient to change the reward definition (**??**) to

$$R_{t+1} = \gamma \left(n_t' g_\theta(f_t) - \frac{\kappa}{2} n_t' \Sigma n_t \right) - c(\Delta n_t) \tag{18}$$



(a) Final wealths



(b) Linear regression

Figure 3: In Figure ?? we show the distribution of the final wealth according to Markowitz, GP and RL. In Figure ?? we show the scatterplot of the final wealth according to GP and RL.

and run the same RL approach discussed in Section ??.

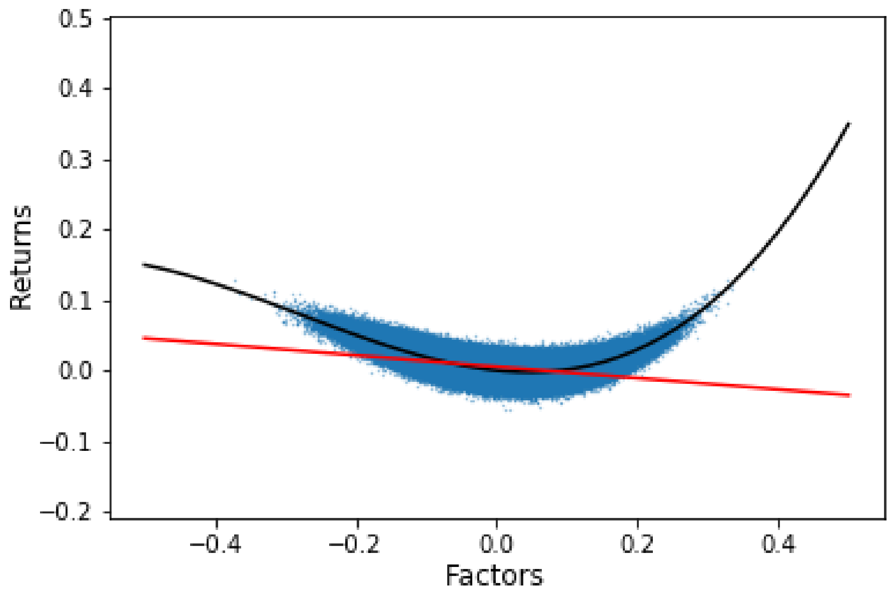
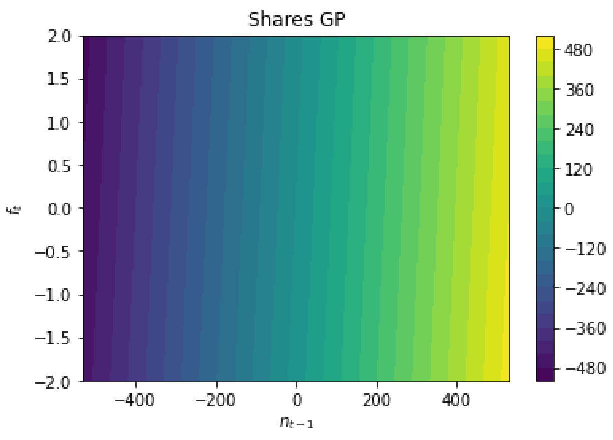
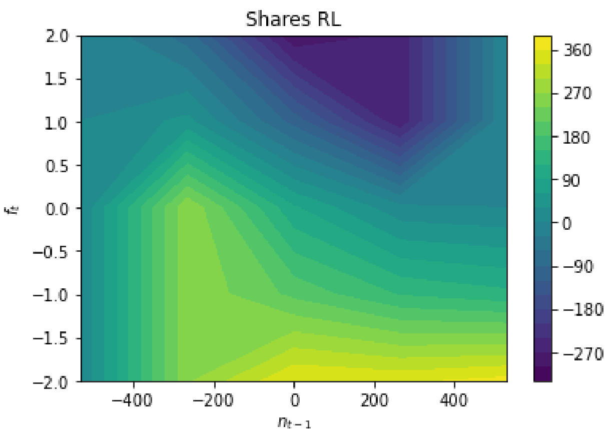


Figure 4: In Figure ?? we show the distribution of factors and P&Ls (f_t, x_{t+1}) (??). Trader (GP) fits a linear model (red line) and applies the GP strategy (??); trader (RL) fits a non-linear model (black line) and applies RL.

In Figure **??** we show the shape of the function $s_t \to n_t(s_t)$ defining the shares owned by the investor while following the GP strategy (**??**) (Figure **??**) and those owned by the agent trained with RL (Figure **??**). We notice how the RL agent is able to capture the non-linear structure of the market, not displaying a monotonic behavior of the strategy in the variables n_{t-1}, f_t.



(a) GP strategy: shares



(b) RL strategy: shares

Figure 5: In Figure **??** we show the next-step shares provided by the GP solution (**??**). In Figure **??** we show the shares owned by the RL agent given the state $s_t = (f_t, n_{t-1})$.

We have simulated $J = 10,000$ out of sample paths and tested the hypothesis of equal expected final wealth for the GP, RL and Markowitz strategies. As expected, the RL approach yields a greater expected cumulative wealth than the other strategies

$$\begin{array}{ll} H_0 : \mathbb{E}\{w_T^{RL}\} = \mathbb{E}\{w_T^{GP}\} & H_0 : \mathbb{E}\{w_T^{RL}\} = \mathbb{E}\{w_T^{Markowitz}\} \\ H_1 : \mathbb{E}\{w_T^{RL}\} > \mathbb{E}\{w_T^{GP}\} & H_1 : \mathbb{E}\{w_T^{RL}\} > \mathbb{E}\{w_T^{Markowitz}\} \\ t\text{-statistic} = 44.1837, p < 10^{-4} & t\text{-statistic} = 13.2352, p < 10^{-4} \end{array}$$

Interestingly, the non-linear structure of the market yields the GP strategy (**??**) performs worse than the Markowitz strategy

$$H_0 : \mathbb{E}\{w_T^{GP}\} = \mathbb{E}\{w_T^{Markowitz}\}$$
$$H_1 : \mathbb{E}\{w_T^{GP}\} < \mathbb{E}\{w_T^{Markowitz}\}$$
$$t\text{-statistic} = -29.0007, p < 10^{-4}$$

In reality, this result is not surprising if we look at the wealth (**??**) decomposition into the cumulative risk-cost of the strategy

$$rc_t = \sum_{s=1}^{t} \gamma^{s-1} \left(\frac{\gamma\kappa}{2} n'_{s-1} \Sigma n_{s-1} + c(\Delta n_{s-1}) \right) \tag{19}$$

and the risk-cost-net wealth $w_t^{rcn} = w_t + rc_t$.

Strategy	$\mathbb{E}\{w_T^{rcn}\}$	$\mathbb{S}d\{w_T^{rcn}\}$	$\mathbb{E}\{rc_T\}$	$\mathbb{S}d\{rc_T\}$
Markowitz	10.64	8.89	3.34	0.88
GP	5.31	4.49	0.86	0.23
RL	20.19	9.75	11.22	0.82

Due to the convex structure of the market, the most of the gain opportunities happen when the factor is far away from 0. Markowitz does not account for trading costs and it tends to spot gain opportunities, buying when $f_t < 0$ and selling when $f_t > 0$. Instead, GP (**??**) is prone to avoid trading due to transaction costs and it achieves much lower costs than Markowitz; by doing so, GP misses important gain opportunities. RL is able to automatically spot the convex structure of the market and it learns that it is more convenient to be aggressive despite the higher costs, as it is clear from the magnitude of the trades in Figure **??**.
We then studied the linear relation $w_T^{RL} = \alpha + \beta w_T^{GP} + noise$, which yield

$$\begin{array}{ll} \alpha = \underset{(2.189,2.547)}{2.368} & H_0 : \beta = 1 \\ \beta = \underset{(1.457,1.514)}{1.485} & H_1 : \beta \neq 1 \\ R^2 = 0.512 & t\text{-statistic} = 33.431, p < 10^{-3} \end{array}$$

confirming that RL achieves a higher wealth than GP. Indeed, a positive intercept α and a slope $\beta > 1$ denote that the random variable w_T^{RL} is likely to be greater than the random variable w_T^{GP}. We visualize these results in Figure **??**.

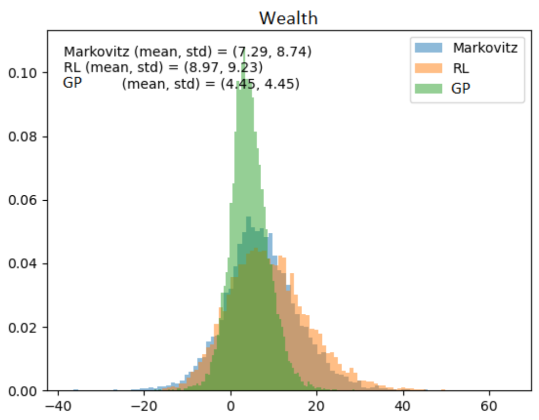
4 Work in progress

In this Section we describe further developments of the setting described so far that are currently being explored.

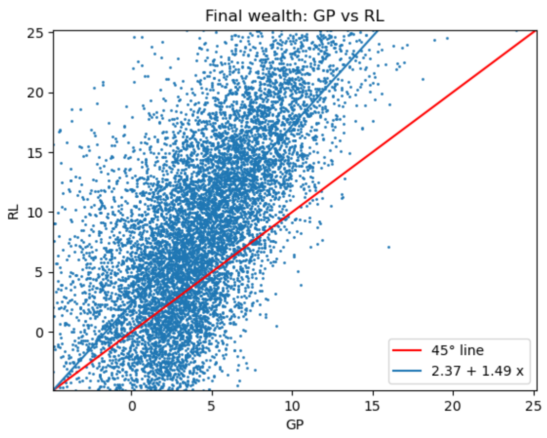
4.1 Non-linearly forecasting factors

Depending on the nature of the factors, the next step price changes can react differently to signals of different strengths. Panic factors are a proxy for the panic in the market: negative signals predict negative price changes, and with a higher variance, than positive signals do. One "almost linear" model that is a natural generalization of (**??**) in this direction is the threshold model defined as

$$x_{t+1} = \begin{cases} B^{(0)} f_t + u_{t+1}^{(0)} & \text{if } f_t < c \\ B^{(1)} f_t + u_{t+1}^{(1)} & \text{if } f_t \geq c \end{cases}, \quad u_{t+1}^{(i)} \sim N(\mu^{(i)}, \Sigma^{(i)}) \tag{20}$$



(a) Final wealths



(b) Linear regression

Figure 6: In Figure **??** we show the distribution of the final wealth according to Markowitz, GP and RL. In Figure **??** we show the scatterplot of the final wealth according to GP and RL.

where c is a threshold.
As an example, we consider the same security as in Case study 1 (Section **??**) and we fit the model (**??**) with $c = 0$, where the factors are the same as in (**??**), obtaining

$$\begin{array}{|rl|l|}\hline B^{(0)} & = 0.015 & u_t^{(0)} \sim N(0.021, 1.380) \\ B^{(1)} & = -0.277 & u_t^{(1)} \sim N(0.081, 1.324) \\ \hline \end{array}$$

We observe that this model can be easily incorporated in the RL framework upon properly adapting the reward definition (**??**).

4.2 Non-mean reverting factors

A mean-reverting model as in (**??**) can be applied only to a small class of factors, such as moving averages for commodity securities. In situations such as the stock market, where heteroscedasticity comes into the picture, more realistic models might include Self-Exciting Threshold AutoRegressive (SETAR) models (see [**?**]), GARCH models (see [**?**]), Threshold ARCH models (TARCH) (see [**?**]) or variations thereof.
Such refined models entail non-trivial computations in the dynamic optimization problem (**??**). Instead, they can be easily embedded in the RL framework, where only simulations of such dynamics are needed.

4.3 From P&Ls to general risk drivers

The optimal solution by Gârleanu and Pedersen [?] assumes the dynamics (**??**) for the price changes of the traded securities. In situations such as the stock market, where price changes are not stationary, alternative risk drivers such as returns or log-returns are more appropriate.
To the authors' knowledge, no analytical solution to the dynamic optimization problem (**??**) is available for more general risk drivers. On the other hand, the RL framework can be applied upon properly adapting the reward definition (**??**). As an example, consider a linear model on the returns

$$(p_{t+1} - p_t) Diag(p_t)^{-1} = Bf_t + u_{t+1}$$

where $Diag(\cdot)$ is the operator that transforms a vector into a diagonal matrix. Then, the proper definition of reward is

$$R_{t+1} = \gamma \left(n_t' Diag(p_t) B f_t - \frac{\kappa}{2} n_t' Diag(p_t) \Sigma Diag(p_t) n_t \right) - c(\Delta n_t) \tag{21}$$

4.4 Fitting stability and constrained strategies

In Step 2 of the algorithm described in Section **??**, it is recommended for the state-action variables to have a comparable magnitude in order to improve the fit of the supervised regressor. To do so, we can alternatively define the state (**??**) as

$$s_t = \left(f_t, \frac{n_{t-1}}{M} \right) \in \mathbb{R}^K \times \mathbb{R}^S \tag{22}$$

where n_{t-1} are the shares at the previous period (held by the trader at the beginning of period t) and M is a proper bound. Then, the action a_t is defined as

$$a_t = \frac{\Delta n_t}{M} \in \mathbb{R}^S \tag{23}$$

At the next period, the shares will be $n_t = \left(s_t^{(n)} + a_t \right) \times M$, where $s^{(n)}$ denotes all the components of the state that do not include the factor. In order to ensure that the next-step state $s_{t+1}^{(n)} \in (-1, 1)$, we search the action a_t in the span

$$a_t \in \left(-1 - s_t^{(n)}, 1 - s_t^{(n)} \right) \tag{24}$$

thereby considering a state-dependent action space (**??**).
We observe that the state-action formulation (**??**)-(**??**), together with the time-dependent action space (**??**), provides a way to include a budget constraint to the RL agent strategy, as defined by the bound M, thereby generalizing the unconstrained GP solution (**??**).

4.5 Dynamic updating of the value function

In Step 1-b-iii of the algorithm described in Section **??**, we update the approximation of the value function via the SARSA updating formula

$$q(s_{t-1}^{(j)}, a_{t-1}^{(j)}) \approx q_{t-1}^{(j)} = R_t^{(j)} + \gamma \hat{q}^{(n-1)}(s_t^{(j)}, a_t^{(j)}) \tag{25}$$

However, the values $q_{t-1}^{(j)}$ are not immediately used in the episodes simulation within the same batch, as they only intervene at Step 2 when fitting the supervised regressor.
It is possible to exploit these intermediate approximations in order to improve the current estimate of the value function by using a kernel function $K(x, y)$ such that $K(x, x) = 1$, such as a Gaussian kernel

$$\hat{q}^{(n-1)}(s, a) \leftarrow \hat{q}^{(n-1)}(s, a) + (q_{t-1}^{(j)} - \hat{q}^{(n-1)}(s, a)) \exp\left(-\frac{||(s, a) - (s_{t-1}^{(j)}, a_{t-1}^{(j)})||^2}{2h}\right) \tag{26}$$

where the constant h needs to be chosen small enough not to perturb the current estimate too much.
The kernel smoothing (**??**) can be also implemented out-of-sample: once the value function $\hat{q}$ is estimated in the training phase, the investor can use it to build its strategy and the real-world rewards R_t obtained during the trading period can be used to dynamically update the initial estimate. For long trading period, such approach is a viable alternative to a periodic re-calibration of the model, otherwise necessary to overcome the non-stationarity of price dynamics.

4.6 From observable to latent factors

In the training phase, it is always possible to use simulations to instruct the agent about the true future realizations of the price changes x_{t+1}. Therefore, rather than using the observed factor f_t to predict the future reward as in (**??**)-(**??**), we can use such simulations to provide the actual reward

$$R_{t+1} = \gamma \left(n_t' x_{t+1} - \frac{\kappa}{2} n_t' \Sigma n_t\right) - c(\Delta n_t) \tag{27}$$

as in the original problem (**??**).
In practice, it is not advisable to use the factor-agnostic reward definition (**??**) when factors are observable, since the realizations x_{t+1} incorporate the volatility of the residual u_{t+1} and therefore provide more disperse realizations of the reward, making the fit less stable.
However, in a latent factor model for the price changes, a reward function that does not include the observation of the factor is more suitable to training an agent which cannot observe the factor in the environment state. A latent factor framework of this sort is currently being investigated by the authors.

References

[1] Kenneth J Arrow. Essays in the theory of risk-bearing. Technical report, 1970.

[2] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[3] Jay Cao, Jacky Chen, John Hull, and Zissis Poulos. Deep hedging of derivatives using reinforcement learning. *The Journal of Financial Data Science*, 3(1):10–27, 2021.

[4] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and trends in Robotics*, 2(1-2):388–403, 2013.

[5] Stefan C Endres, Carl Sandrock, and Walter W Focke. A simplicial homology algorithm for lipschitz optimisation. *Journal of Global Optimization*, 72(2):181–217, 2018.

[6] R. F. Engle. Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50:987–1007, 1982.

[7] N. Gârleanu and L. H. Pedersen. Dynamic trading with predictable returns and transaction costs. *The Journal of Finance*, 68(6):2309–2340, 2013.

[8] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *In Proc. 19th International Conference on Machine Learning*. Citeseer, 2002.

[9] Petter N Kolm and Gordon Ritter. Dynamic replication and hedging: A reinforcement learning approach. *The Journal of Financial Data Science*, 1(1):159–171, 2019.

[10] Harry Markowitz. Portfolio selection in the journal of finance vol. 7. 1952.

[11] Robert C Merton. An intertemporal capital asset pricing model. *Econometrica: Journal of the Econometric Society*, pages 867–887, 1973.

[12] John Moody and Matthew Saffell. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, 12(4):875–889, 2001.

[13] Remi Munos, Leemon C Baird, and Andrew W Moore. Gradient descent approaches to neural-net-based solutions of the hamilton-jacobi-bellman equation. In *IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339)*, volume 3, pages 2152–2157. IEEE, 1999.

[14] Matteo Papini, Matteo Pirotta, and Marcello Restelli. Adaptive batch size for safe policy gradients. In *The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017.

[15] John W Pratt. Risk aversion in the small and in the large. In *Uncertainty in economics*, pages 59–79. Elsevier, 1978.

[16] Gordon Ritter. Machine learning for trading. *Available at SSRN 3015609*, 2017.

[17] Gavin A Rummery and Mahesan Niranjan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[18] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[19] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[20] Rainer Storn and Kenneth Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[21] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[22] Howell Tong. *Threshold models in non-linear time series analysis*, volume 21. Springer Science & Business Media, 2012.

[23] Hado Van Hasselt. Reinforcement learning in continuous state and action spaces. In *Reinforcement learning*, pages 207–251. Springer, 2012.

[24] Edoardo Vittori, Michele Trapletti, and Marcello Restelli. Option hedging with risk averse reinforcement learning. *arXiv preprint arXiv:2010.12245*, 2020.

[25] Y Xiang, DY Sun, W Fan, and XG Gong. Generalized simulated annealing algorithm and its application to the thomson model. *Physics Letters A*, 233(3):216–220, 1997.

[26] J. M. Zakoian. Threshold heteroskedastic models. *Unpublished paper, Institut National de la Statistique et des Etudes Economiques, Paris*, 1991.