

ABSTRACT

Title of dissertation: REINFORCEMENT LEARNING METHODS
FOR CONIC FINANCE

Sahil Chopra
Doctor of Philosophy, 2020

Dissertation directed by: Professor Dilip Madan
AMSC

Conic Finance is a world of two-prices, a more grounded reality than the theory of one-price. The world, however, is constructed by considering nonadditive expectations of risks or value functions. This makes some of the optimization algorithms incompatible with this universe, if not infeasible. It is more evident in the application of Reinforcement Learning algorithms where the underlying principle of TD learning and Bellman equations are based on the additivity of value functions. Hence, the task undertaken here is to mold the recent advances in the field of Distributional Reinforcement Learning to be conducive to learning in the setting of nonadditive dynamics. Algorithms for discrete and continuous actions are described and illustrated on sample problems in finance.

REINFORCEMENT LEARNING METHODS FOR CONIC FINANCE

by

Sahil Chopra

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2020

Advisory Committee:
Professor Dilip Madan, Chair/Advisor
Professor Radu Balan
Professor Wojciech Czaja
Professor Mark Loewenstein
Dr. Ilya Ryzhov

Acknowledgements

First and foremost, I am indebted to Prof. Dilip Madan for taking me under his wing and educating me on the vast world of Mathematical Finance; the puzzle that was revealed bit by bit, chaos eventually making sense. Going to the RITs and learning mathematical finance from him was akin to learning prose by hearing the “Bard of Avon” speak. I am thankful not only for his continued support of my mostly stupid and occasional wild ideas, but also for his consistent adherence to sticking to the bigger picture of it all. I also find it very difficult imagining undertaking the shift in my research life without the academic support and guidance that came from Prof. Madan.

I would also like thank the dissertation committee, Prof. Radu Balan, Prof. Czaja, Prof. Loewenstein, and Dr. Ryzhov, for making invaluable revision and embellishment suggestions. Some of the ideas would be worthy additions to the commandments of research I must carry forward.

I have had the pleasure of interacting with some brilliant colleagues, and too many of them have left an indelible mark to mention by name here. The whole experience has been a sizable chunk of my adult life and it’s difficult to recall the exact way each and every interaction guided me in the quest; the petal in the river gets direction from many pebbles. I am sincerely in appreciative of all the advice, suggestions, ideas, and criticism.

Finally, I would like to thank my family; my awesome parents for providing un-moving love and support through all my missteps and miscalculations, my sister and brother-in-law for consistent encouragements and guidance, and my favorite human, my niece Navya, for being the foremost source of joy in my life.

Table of Contents

Acknowledgements	ii
Table of Contents	iii
1 Conic Finance	1
1.1 Introduction	1
1.2 Acceptable Risks and Coherent Risk Measures	2
1.3 Pricing under two price theory	3
1.4 Prices as Distorted Expectations	5
1.5 Distorted expectations as sum of weighted quantiles	7
1.6 Discrete Approximation	8
1.7 Performance Indicators	9
2 Deep Reinforcement Learning: Background and Recent Advances	12
2.1 Introduction	12
2.2 Background	13
2.3 Distributional	15
2.4 QR implementation	19
2.5 Policy Gradient Methods	23
3 Conic Q-learning	28
3.1 Motivation	28
3.2 Constructing the conic value functions	28
3.3 Bid-optimal policy	29
3.4 Remarks on consistency	30
3.5 Implementation	31
3.6 Application: Pairs Trading	32
4 Deterministic policy gradient and distorted least squares	45
4.1 Motivation	45
4.2 Constructing the new loss function	46
4.3 Implementation using Distributional Reinforcement Learning	47
4.4 Application of DLS: tracking an ETF	50
5 Conclusion and further work	70

List of Tables

3.1	2D Convolutional Network for Distributional Learning using QR	37
3.2	Convergence of 10 step moving average of mean reward for Q nets. Training done on 8 GB NVIDIA GeForce GTX 1650.	38
3.3	Sharpe Ratio on training data.	39
3.4	Sharpe Ratio on test data.	39
3.5	Gain Loss ratio on training data.	39
3.6	Gain Loss ratio on test data.	39
3.7	Maximum Drawdown on training data.	40
3.8	Maximum Drawdown on test data.	40
3.9	Acceptability Index on training data.	40
3.10	Acceptability Index on test data.	40
4.1	Breakdown of top 10 (by cap weight in percentage) of Basket Holdings for popular ETFs - September 2019	50
4.2	Dilated Convolutional Network - Actor	55
4.3	Dilated Convolutional Network with Quantile Regression - Critic	56
4.4	Convergence of 10 step moving average of DMSE for actor nets. Training done on 4 GB NVIDIA GTX 1050 Ti.	57
4.5	Sharpe Ratio on training data.	58
4.6	Sharpe Ratio on testing data.	58
4.7	Gain Loss ratios on training data.	58
4.8	Gain Loss ratios on testing data.	59
4.9	Maximum Drawdown on training data.	59
4.10	Maximum Drawdown on testing data.	59
4.11	Acceptability Index on training data.	59
4.12	Acceptability Index on testing data.	59

List of Figures

3.1	Evaluation results for neural net training	38
3.2	Acceptability indices comparison	41
3.3	Sharpe Ratio comparison	42
3.4	Gain-Loss ratio comparison	43
3.5	Maximum Drawdown comparison	44
4.1	Convergence results for neural net training - Actor	57
4.2	Acceptability indices comparison	61
4.3	Sharpe Ratio comparison	62
4.4	Gain-Loss ratio comparison	63
4.5	Maximum Drawdown comparison	64
4.6	Weight distribution comparison on training and testing dataset for $\lambda = 0.0$	65
4.7	Weight distribution comparison on training and testing dataset for $\lambda = 0.5$	66
4.8	Weight distribution comparison on training and testing dataset for $\lambda = 0.75$	67
4.9	Weight distribution comparison on training and testing dataset for $\lambda = 1.0$	68
4.10	Weight distribution comparison on training and testing dataset for $\lambda = 1.5$	69

Chapter 1

Conic Finance

1.1 Introduction

Much of the theoretical background with regards conic finance can be traced from “Applied Conic Finance” by Madan and Schoutens[1]. The focus of study here are random variables X denoting the payoff at some future date, say T . We consider these random variables X promised at T and defined on a probability space (Ω, \mathcal{F}, P) while assuming $E[|X|] < \infty$. Such a random variable X is called a *risk* and we denote the set of these risks as \mathcal{A} . Furthermore, we associate two properties of such sets if they satisfy the corresponding conditions:

Definition 1. *A set of risks \mathcal{A} is called*

- **convex** if $X, Y \in \mathcal{A} \implies \alpha X + (1 - \alpha)Y \in \mathcal{A} \quad \forall \alpha \in [0, 1]$
- **cone** if $X \in \mathcal{A} \implies cX \in \mathcal{A} \quad \forall c \in \mathbb{R}^+$

We further define some concepts and properties related to the measures of risk on this space.

Definition 2. A *monetary risk measure*, denoted by ρ , is a functional on \mathcal{A} that assigns a non-negative real number to a risk. That is,

$$\rho : \mathcal{A} \rightarrow \mathbb{R}^+ \cup \{0\}$$

The higher this value $\rho(X)$ is, the riskier the underlying asset is. In addition, this functional must be

1. (Monotonic) $X \leq Y \implies \rho(X) \leq \rho(Y)$
2. (Cash invariant) $\rho(X + c) = \rho(X) + c$

1.2 Acceptable Risks and Coherent Risk Measures

We consider here risk measures introduced by Artzner et. al[2] that satisfy the corresponding properties.

Definition 3 (Artzner et. al). A monetary risk measure $\rho : \mathcal{A} \rightarrow \mathbb{R}^+ \cup \{0\}$ is called a *coherent risk measure* if it satisfies the following properties for any non-negative random variables $X, Y \in \mathcal{A}$

1. (Sub-additivity) $\rho(X + Y) \leq \rho(X) + \rho(Y)$
2. (Positive homogeneity) $\rho(cX) = c\rho(X) \quad \forall c \in \mathbb{R}^+$
3. (Monotonicity) $P(X \leq Y) = 1 \implies \rho(X) \leq \rho(Y)$

Theorem 1 (Artzner et. al). For a given set of probability measures \mathcal{M} , any risk measure ρ defined as

$$\rho(X) = \sup_{Q \in \mathcal{M}} E_Q[X] \tag{1.1}$$

satisfies the aforementioned properties and, hence, is a **coherent risk measure**. The converse is also true, that is, any coherent risk measure must be of this form.

For a convex set of probability measures \mathcal{M} and interest rate r , we define a **cone of acceptable risks** \mathcal{A} as following

$$\mathcal{A} = \{Z | \exp(-rT)E_Q[Z] \geq 0 \quad \forall Q \in \mathcal{M}\} \quad (1.2)$$

Furthermore, the two primary valuation operators in the two-price economy, the bid price and the ask price are characterized as following by Madan and Schoutens[1]

$$bid(X) := \exp(-rT) \inf_{Q \in \mathcal{M}} E_Q[X] \quad (1.3)$$

$$ask(X) := \exp(-rT) \sup_{Q \in \mathcal{M}} E_Q[X] \quad (1.4)$$

By the very properties of inf and sup, the bid and the ask price functionals are concave and convex respectively. This above characterization of the markets as convex cones of acceptable cash flows lends the adjective “conic” to finance.

1.3 Pricing under two price theory

We start by constructing a market model by specifying a (convex) set \mathcal{M} of supporting measures. The bid and the ask prices can then be calculated by taking infimum and supremum respectively of the expectations over these support measures. We’ll shift to using distortions instead of using infimums and supremums, by using a result of Kusuoka’s [4]. To get there, though, we need a few more concepts and assumptions:

Definition 4. *Two risks X, Y are said to be **comonotone** if they are completely driven*

by one single factor. That is, if F_X and F_Y are the corresponding cumulative distribution functions, then there exists an r.v. U on $[0, 1]$ such that $X = F_X^{-1}(U)$ and $Y = F_Y^{-1}(U)$.

Proposition 1. *We further have that the bid/ask valuation operators are additive for comonotone risks. That is, for comonotone risks X, Y*

$$bid(X + Y) = bid(X) + bid(Y) \quad (1.5)$$

$$ask(X + Y) = ask(X) + ask(Y) \quad (1.6)$$

We then say that the bid and ask functionals are comonotone additive.

Definition 5. *A concave function $\Psi : [0, 1] \rightarrow [0, 1]$ on a distribution is called a concave distortion function. Following are the most commonly used distortion functions as introduced in [1]*

1. (MINVAR) $\Psi_\lambda^{MINVAR}(u) = 1 - (1 - u)^{1+\lambda} \quad \lambda \geq 0$
2. (MAXVAR) $\Psi_\lambda^{MAXVAR}(u) = u^{\frac{1}{1+\lambda}} \quad \lambda \geq 0$
3. (MAXMINVAR) $\Psi_\lambda^{MAXMINVAR}(u) = (1 - (1 - u)^{1+\lambda})^{\frac{1}{1+\lambda}} \quad \lambda \geq 0$
4. (MINMAXVAR) $\Psi_\lambda^{MINMAXVAR}(u) = 1 - \left(1 - u^{\frac{1}{1+\lambda}}\right)^{1+\lambda} \quad \lambda \geq 0$

The results in [4] lead to the following characterization of bid and ask prices in [1]

Proposition 2. *Under the assumptions of a.) comonotone additivity of the bid/ask operators and b) law invariance, the bid/ask prices must be nonlinear expectations under concave distortions. That is, there must exist a concave distortion Ψ for any risk X with distribution function F_X such that its bid price is given by*

$$bid(X) = \exp(-rT) \int_{-\infty}^{\infty} x d\Psi(F_X(x)) \quad (1.7)$$

Then, using the properties of infimum and supremum, the corresponding ask price can be calculated using the equation $bid(X) = -ask(-X)$. In terms of integrals, that can be written as

$$ask(X) = -\exp(-rT) \int_{-\infty}^{\infty} x d\Psi(F_{-X}(x)) \quad (1.8)$$

Furthermore, if we have access to the density of X , the corresponding bid price formula is

$$bid(X) = \exp(-rT) \int_{-\infty}^{\infty} x \Psi'(F_X(x)) f_X(x) dx \quad (1.9)$$

Owing to the fact that distorted expectations are also expectations under a measure change $\Psi'(F_X(x))$, for x nearing negative infinity or $F_X(x)$ nearing zero, we should ideally have loss aversion by having $\Psi'(u)$ approach positive infinity as u approaches zero. Conversely, as x tends to positive infinity (and $F_X(x)$ approaches one), we require that $\Psi'(u)$ approaches zero as u approaches one to ensure against being enticed by large gains. Some of the concave distortions that have these properties are MINMAXVAR, MAXMINVAR as introduced earlier. Another distortion function with these properties is the WANG-TRANSFORM, which we skip for the time being. In the following sections, unless specified otherwise, the choice of distortion function is fixed to be MINMAXVAR. The underlying theory remains the same.

1.4 Prices as Distorted Expectations

One may write the standard linear expectation of a risk X as

$$E[X] = \int_{-\infty}^{\infty} x dF_x(x) = - \int_{-\infty}^0 F_x(x) dx + \int_0^{\infty} (1 - F_x(x)) dx$$

A conservative alternative to expectation maximization may be the maximization of a non-linear expectation weighted more heavily on the left hand side. Such nonlinear expectations introduced earlier can be denoted in a general form using the notation for distorted expectations

$$D^\Psi[X] = \int_{-\infty}^{\infty} x d\Psi(F_x(x)) = - \int_{-\infty}^0 \Psi(F_x(x)) dx + \int_0^{\infty} (1 - \Psi(F_x(x))) dx \quad (1.10)$$

If X represents a cash flow paid out at time T , then the **bid price** or the price paid from the market in return for the cash flow is defined as

$$bid(X) = \exp^{-rT} D^\Psi[X] \quad (1.11)$$

The price one pays to the market, or the **ask price**, is computed using the result that $bid(X) = -ask(-X)$. If we define $\hat{\Psi}(u) = 1 - \Psi(1 - u)$, then

$$bid(X) = \exp\{-rT\} \left(- \int_0^{\infty} \Psi(1 - F_{-x}(x)) dx + \int_0^{\infty} \hat{\Psi}(1 - F_x(x)) dx \right) \quad (1.12)$$

$$ask(X) = \exp\{-rT\} \left(- \int_0^{\infty} \Psi(1 - F_x(x)) dx + \int_0^{\infty} \hat{\Psi}(1 - F_{-x}(x)) dx \right) \quad (1.13)$$

1.5 Distorted expectations as sum of weighted quantiles

Suppose that $F : (a, b) \rightarrow \mathbb{R}$ is an increasing function which is not necessarily strictly increasing. Let

$$c := \lim_{x \downarrow a} F(x)$$

$$d := \lim_{x \uparrow b} F(x)$$

Definition 6. Given a distribution function F_X , we define the inverse function F_X^{-1} and F_X^{-1+} of F_X as

$$F_X^{-1}(p) := \inf\{x \in \mathbb{R} | F_X(x) \geq p\} = \sup\{x \in \mathbb{R} | F_X(x) < p\} \quad (1.14)$$

and,

$$F_X^{-1+}(p) := \inf\{x \in \mathbb{R} | F_X(x) > p\} = \sup\{x \in \mathbb{R} | F_X(x) \leq p\} \quad (1.15)$$

for $p \in [0, 1]$, where by convention, $\inf \emptyset = +\infty$ and $\sup \emptyset = -\infty$.

Furthermore, given some probability level p , $F_X^{-1}(p)$ is the p^{th} quantile of X which will now be denoted as $q(p)$. That is,

$$q_X(p) := \inf\{x \in \mathbb{R} | F_X(x) \geq p\} = \sup\{x \in \mathbb{R} | F_X(x) < p\} \quad (1.16)$$

This q_X is now onward referred to as the **quantile function** of X .

Theorem 2 (Denuit et al. [5]). Given Ψ is a continuous distortion function, we can

re-write the distorted expectation $\mathcal{D}^\Psi[X]$ as the weighted sum of quantiles. That is,

$$\mathcal{D}^\Psi[X] = \int_{[0,1]} F_X^{-1}(q) d\Psi(q) \quad (1.17)$$

Furthermore, if Ψ is absolutely continuous, then

$$\mathcal{D}^\Psi[X] = \mathbb{E}[F_X^{-1}(U)\Psi'(U)] \quad (1.18)$$

where $U \sim \text{Uniform}[0, 1]$.

1.6 Discrete Approximation

When working in contexts where we may not have the distribution function coming from a model, but from data, one may only be able to compute the discrete approximation of the expectation. Say X is a random variable with the following parametric distribution

$$X = x_i \quad \text{w.p. } p_i, \quad i = 1, \dots, n, \quad \sum_{i=1}^n p_i = 1.$$

Then the expected value of X is $E[X] = \sum x_i p_i$. To compute the corresponding distorted expectation, we use the following algorithm[1].

1. **Sort x_i 's:** $x_{(1)} \leq x_{(2)} \leq \dots x_{(n)}$ with the corresponding probabilities $p_{(1)}, p_{(2)}, \dots p_{(n)}$.
2. Compute the **cumulative sum function:**

$$F_{(i)} = \sum_{j=1}^i p_{(j)} = P(X \leq x_{(i)}) \quad i = 1, \dots, n$$

3. Compute the **distorted probabilities** for a given distortion function Ψ

$$\hat{p}_{(i)} = \Psi(F_{(i)}) - \Psi(F_{(i-1)}) \quad i = 1, \dots, n \text{ with } F(0) = 0$$

4. Compute the **discrete sum** as

$$D^\Psi[X] = \sum_{i=1}^n x_{(i)} \cdot \hat{p}_{(i)} \tag{1.19}$$

In uses where we don't have the continuous distribution functions, but only the parametric estimates, one may use the above algorithm to approximate the bid and the ask prices of risks.

1.7 Performance Indicators

For each of the applications we later consider, PNLs are constructed from bootstrapped trajectories of returns. For pairs trading, each trajectory is constructed in 5 minute intervals, iterated over 79 timesteps to make a trading unit of one day. Furthermore, since our purpose is to compare the shift in these metrics, we assume for the time being the risk free return to be zero. For each of these PNL trajectories over the defined duration, we denote by r_i the return for day $i = 1, \dots, N$, N being the size of the sample. Then calculate the following:

1.7.1 Sharpe Ratio

Assuming risk free rate to be zero, the Sharpe ratio is computed as the ratio of mean return per day (μ) and its corresponding standard deviation (σ). Furthermore, this ratio is annualized by multiplying the square root of 252, the approximate number of trading days in the year.

$$SR = \sqrt{252} \frac{\mu}{\sigma} \quad (1.20)$$

Trading strategies Sharpe ratio around 2 are usually considered above par.

1.7.2 Gain loss ratio

To compute the gain to loss ratio, we compute the average positive return and average negative return, and compute their ratio. That is,

$$\begin{aligned} \text{average gain} &= \frac{\sum_{i=1}^N \mathbb{1}_{r_i > 0} r_i}{\sum_{i=1}^N \mathbb{1}_{r_i > 0}} \\ \text{average loss} &= \frac{\sum_{i=1}^N \mathbb{1}_{r_i < 0} r_i}{\sum_{i=1}^N \mathbb{1}_{r_i < 0}} \\ GL &= \frac{\text{average gain}}{\text{average loss}} \end{aligned}$$

Trading strategies that may net the Gain loss ratio around 1 nets us more than a fair coin toss, and may be considered desirable.

1.7.3 Maximum Drawdown

For each sequence of daily, we have running previous maximums (peaks) and minimums (trough). Let P denote the peak value before the largest drop in cash flows, and L denote

the lowest value before new maximum is reached. Then,

$$MD = \frac{P - L}{L} \tag{1.21}$$

Conservative traders may want to only trade strategies within 5% range of maximum drawdown. Others, slightly more adventurous may trade 5% – 10%.

1.7.4 Acceptability index

Considering the cash flows $\{x_1, x_2, \dots, x_N\}$, we use the definition of Acceptability index[1] as the highest stress level γ under which the cash flows remain acceptable. For a given concave distortion function Ψ , and using the previously described algorithm to compute the approximate distorted expectation of cash flows,

$$AI_\Psi = \arg \max_{\gamma > 0} \{\mathbb{D}^{\Psi_\lambda}[X] \geq 0\} \tag{1.22}$$

Since Ψ is assumed to be concave, a cash flow with negative expectation can never be acceptable. By convention, we assign Acceptability Index value of -1 to that case. One may consider values corresponding to 0.12 to 0.18 above par.

Chapter 2

Deep Reinforcement Learning: Background and Recent Advances

2.1 Introduction

The fundamental principles behind reinforcement learning is setting up the problem in a manner where there is an agent (an actor that makes the decisions) and a surrounding environment, and the agent is allowed to improve by repeatedly interacting with the environment based on the reward signals it receives. Usually in these settings, the aim of the agent is to maximize the expected utility from any starting point. These interactions can be modeled as a Markov Decision Process (MD) $(\mathcal{X}, \mathcal{A}, \mathcal{R}, P, \gamma)$ as introduced in Sutton and Barto[6], Puterman[8], Fu et al.[9]. The state space, denoted by \mathcal{X} , is the set of information that the agent sees, and acts upon. The action space, denoted by \mathcal{A} is the collection of all the actions that the agent is allowed to take for any given state. \mathcal{R} is the reward function. Provided by the environment, the reward function allocates a signal to each state-action pair. P provides the state transition kernel. We denote by $P(x'|x, a)$ the

probability of transitioning from $x \in \mathcal{X}$ to $x' \in \mathcal{X}$ when action $a \in \mathcal{A}$ is taken. Finally, γ denotes the discount factor associated with the preference for immediate rewards vs. later rewards. In this setting, each step of the training process consists of the agent selecting an action based on its current state and the environment responding with a reward and the next state. The agent then learns from this interaction, and repeats the process while updating. This iterative process continues till the agent reaches “optimality”.

2.2 Background

We now consider the agent having a certain fixed method of acting, and call it the policy function. This policy function $\pi : \mathcal{X} \rightarrow \mathcal{A}$ maps each state $x \in \mathcal{X}$ to an action in \mathcal{A} . For this fixed policy π , we can define the corresponding state value function as

$$V^\pi(x) := \mathbb{E}_{P,\pi} \left[\sum_{t=0}^{\infty} \gamma^t R_t(x_t, a_t) \mid x_0 = x \right] \quad (2.1)$$

Similarly, we define the quality of a state action pair as the Q function as the expected return obtained from taking action a in state x , and then following π as

$$Q^\pi(x, a) := \mathbb{E}_{P,\pi} \left[\sum_{t=0}^{\infty} \gamma^t R_t(x_t, a_t) \mid x_0 = x, a_0 = a \right] \quad (2.2)$$

These equations defining the value functions can further be “unrolled” and re-written in a manner that leads to the Bellman’s equation for the value function

$$Q^\pi(x, a) = \mathbb{E}[R(x, a)] + \gamma \mathbb{E}_{P,\pi}[Q^\pi(x', a')] \quad (2.3)$$

Since the objective here is to learn the optimal policy and not evaluate particular policies, we define the optimal policy π^* as the policy of taking actions in each state that maximize the expected future returns. Such a policy corresponds to the following value function

$$Q^*(x, a) = \max_{\pi \in \Pi} Q^\pi(x, a) \quad \forall (x, a) \in \mathcal{X} \times \mathcal{A} \quad (2.4)$$

Defined this way, the above optimal Q function satisfies the Bellman optimality equation and, hence, we have that

$$Q^*(x, a) = \mathbb{E}[R(x, a)] + \gamma \mathbb{E}_P \max_{a' \in \mathcal{A}} [Q^*(x', a')] \quad (2.5)$$

This equation has a unique fixed point in Q^* , and our algorithm for learning this optimal value function is hence called Q-learning. The optimal policy π^* can then be defined as the following

$$\pi^*(\cdot|x) = \arg \max_{a \in \mathcal{A}} Q^*(x, a) \quad (2.6)$$

Q-learning, is then, the fixed point iteration algorithm using the aforementioned Bellman optimality condition. Going forward, for ease of notation, we define the Bellman operator \mathcal{T}^π and the Bellman optimality operator \mathcal{T} as following:

$$\mathcal{T}^\pi Q(x, a) := \mathbb{E}[R(x, a)] + \gamma \mathbb{E}_{P, \pi} [Q^\pi(x', a')] \quad (2.7)$$

$$\mathcal{T}Q(x, a) := \mathbb{E}[R(x, a)] + \gamma \mathbb{E}_P \max_{a' \in \mathcal{A}} [Q(x', a')] \quad (2.8)$$

2.3 Distributional

An alternate method of formulating a policy is to consider the underlying distributions rather than the expectations. This leads us to consider the corresponding underlying random variables of the value functions. With some redundancy involved, let's consider (for a fixed policy) the random variable that is the sum of all future rewards starting from state x and action a , and denote it by $Z^\pi(x, a) = \sum_{t=0}^{\infty} \gamma^t R_t(x_t, a_t)$ where $x_0 = x$, and $a_0 = a$. Then the Q-function defined earlier is nothing but the expectation of such a random variable. However, the hurdle here is that we can no longer rely on the Bellman equations to obtain these distributions and, hence, must formulate alternate ways of learning. This was tackled by the team of Bellmare, Dabney, and Munos [11], who introduced a distributional analog of the Bellman equations, defining the corresponding distributional Bellman operator for a fixed policy π to be

$$\mathcal{T}^\pi Z(x, a) \stackrel{\text{D}}{=} R(x, a) + \gamma Z_{P, \pi}(x', a') \quad (2.9)$$

2.3.1 Theory

Definition 7. *Before delving into the convergence and the corresponding optimality operator, we need to define a metric over distributions that is going to be useful. The **p-Wasserstein metric** W_p for $p \in [1, \infty]$ between two distributions U and Y is given by*

$$W_p(U, Y) = \left(\int_0^1 |F_Y^{-1}(\omega) - F_U^{-1}(\omega)|^p d\omega \right)^{1/p} \quad p \in [1, \infty) \quad (2.10)$$

and

$$W_\infty(U, Y) = \sup_{\omega \in [0, 1]} |F_Y^{-1}(\omega) - F_U^{-1}(\omega)| \quad (2.11)$$

where $F_Y^{-1}(\omega) := \inf \{y \in \mathbb{R} : \omega \leq F_Y(y)\}$.

In the distributional context, let \mathcal{Z} denote the space of action value distributions with finite moments.

$$\mathcal{Z} = \{Z : \mathcal{X} \times \mathcal{A} \rightarrow \mathcal{P}(\mathbb{R}) : \mathbb{E}[|Z(x, a)|^p] < \infty \quad \forall (x, a) \text{ and } p \geq 1\}$$

Then for two action-value distributions $Z_1, Z_2 \in \mathcal{Z}$, we use the maximal form of the Wasserstein metric introduced in [11].

$$\tilde{d}_p(Z_1, Z_2) := \sup_{x, a} W_p(Z_1(x, a), Z_2(x, a)) \quad (2.12)$$

Lemma 1. \tilde{d}_p is a metric over the set of value distributions \mathcal{Z} .

Lemma 2. The distributional Bellman operator defined above, $\mathcal{T}^\pi : \mathcal{Z} \rightarrow \mathcal{Z}$ is a γ -contraction in \tilde{d}_p . Furthermore, we can conclude using Banach's fixed point theorem that \mathcal{T}^π has a unique fixed point. By definition, Z^π must be the random variable that corresponds to Q^π

$$\{Z_k\} \xrightarrow{\tilde{d}_p} Z^\pi \quad 1 \leq p \leq \infty \quad (2.13)$$

Note: \mathcal{T}^π is not a contraction in total variation, KL divergence, or Kolmogorov distance.

The above results shows the convergence for a fixed policy π . However, we are interested in the control setting and obtaining the optimal policy. Let Π denote the set of all policies. As a placeholder, we can denote by

$$\Pi^{**} = \{\pi \in \Pi | Q^\pi(x, a) \geq Q^{\pi'} \forall (x, a) \in \mathcal{X} \times \mathcal{A}, \text{ and for } \pi' \in \Pi\} \quad (2.14)$$

The theory developed here relies on a subset of this aforementioned set of “optimal” policies, where we get the optimal distributions through fixed point iteration and $\Pi^* \subset \Pi^{**}$ are the policies corresponding to those. As a consequence, not all value distributions with expectation Q^* are optimal; they must match the full distribution of the return.

Definition 8. Let Π^* denote the set of optimal policies. An **optimal value distribution**, then, is the value distribution of an optimal policy. The set of optimal value distributions is

$$\mathcal{Z}^* = \{Z^{\pi^*} : \pi^* \in \Pi^*\}$$

Definition 9. A greedy policy π for $Z \in \mathcal{Z}$ maximizes the expectation of Z . The set of greedy policies for Z is

$$\mathcal{G}_Z := \left\{ \pi : \sum_a \pi(a|x) \mathbb{E}[Z(x, a)] = \max_{a' \in \mathcal{A}} \mathbb{E}[Z(x, a')] \right\}$$

The Bellman optimality operator for Q is

$$\mathcal{T}Q(x, a) = \mathbb{E}[R(x, a)] + \gamma \mathbb{E}_P[\max_{a' \in \mathcal{A}} Q(x', a')]$$

The corresponding distributional optimality operator for Z is any operator that implements a greedy selection rule $\mathcal{T}Z = \mathcal{T}^\pi Z$ for some $\pi \in \mathcal{G}_Z$ where we want the sequence of iterates $Z_{k+1} := \mathcal{T}Z_k$ with $Z_0 \in \mathcal{Z}$.

Lemma 3. Let $Z_1, Z_2 \in \mathcal{Z}$. Then

$$\|\mathbb{E}[\mathcal{T}Z_1] - \mathbb{E}[\mathcal{T}Z_2]\|_\infty \leq \gamma \|\mathbb{E}[Z_1] - \mathbb{E}[Z_2]\|_\infty$$

and in particular, $\mathbb{E}[Z_k] \rightarrow Q^*$ exponentially quickly.

Definition 10. A non-stationary optimal value distribution (NOVD) Z^{**} is the value distribution corresponding to a sequence of optimal policies. The set of NOVD is \mathcal{Z}^{**}

Theorem 3. Let \mathcal{X} be measurable and suppose that \mathcal{A} is finite. Then

$$\lim_{k \rightarrow \infty} \inf_{Z^{**} \in \mathcal{Z}^{**}} \tilde{d}_p(Z_k(x, a), Z^{**}(x, a)) = 0 \quad \forall x, a$$

If \mathcal{X} is finite, then Z_k converges to Z^{**} uniformly. Furthermore, if there is a total ordering \prec on Π^* such that for any $Z^* \in \mathcal{Z}^*$, $\mathcal{T}Z^* = \mathcal{T}^\pi Z^*$ with $\pi \in \mathcal{G}_{Z^*}$, $\pi \prec \pi' \quad \forall \pi' \in \mathcal{G}_{Z^*} \setminus \{\pi\}$, then \mathcal{T} has a unique fixed point $Z^* \in \mathcal{Z}^*$

This above result shows that we have a weak convergence under the distributional analog of Bellman optimality equation. This can be used to then find the underlying optimal distribution and choose the optimal policy accordingly.

Algorithm 1: Distributional Algorithm for Q-learning:

input: A transition $x, a, r, x', \gamma \in [0, 1]$

Compute $Q(x', a') = \mathbb{E}[Z(x', a')] \quad \forall a'$

$a^* \leftarrow \arg \max_{a'} Q(x', a')$

Implement the distributional Bellman update

$$\tilde{Z}(x, a) \leftarrow r(x, a) + \gamma Z(x', a^*)$$

output: $W_p(Z(x, a), \tilde{Z}(x, a))$ #the Wasserstein distance

2.4 QR implementation

2.4.1 Quantile Regression

Introduction

Since minimizing the Wasserstein based loss function is difficult to implement with stochastic gradient descent, instead the algorithm introduced in the original paper by Bellamare et al. [11] on distributional learning was developed for minimizing the KL divergence on a sequence of projected distributions. The projected distributions were called C-51 (51 for the choice of atoms the distribution was projected upon). More suited to our needs, and also shown to outperform the C-51 method is the one based on quantile regression introduced in a follow up paper by Bellamare et al.[12]. The approach introduced here is

built upon estimating the approximate quantile distribution instead.

Background

Let \mathcal{Z} be the space of value distributions with finite moments. That is, $\mathcal{Z} = \{Z : \mathcal{X} \times \mathcal{A} \rightarrow \mathcal{P}(\mathbb{R}) \mid E[|Z(x, a)|^p] < \infty \forall (x, a), p \geq 1\}$. We now let \mathcal{Z}_Q the space of quantile distributions for a fixed $N \in \mathbb{N}$ where N is the number of quantiles, and denote the corresponding cumulative probabilities as $\tau_i = \frac{i}{N}$ for $i = 1, \dots, N$, and $\tau_0 = 0$. Then, a distribution $Z_\theta \in \mathcal{Z}_Q$ is a mapping of each state-action pair (x, a) to a uniform probability distribution supported on $\{\theta_i(x, a)\}$. That is,

$$Z_\theta(x, a) := \frac{1}{N} \sum \delta_{\theta_i(x, a)} \quad (2.15)$$

The advantages of this variation of projecting include not being restricted to pre-specified bounds in reward space, in addition to the ease of computation of distorted expectation that we'll later use.

We further denote by Π_{W_1} , for an arbitrary distribution $Z \in \mathcal{Z}$, as the minimizer of the 1-Wasserstein distance in the projected space.

$$\Pi_{W_1} := \arg \min_{Z_\theta \in \mathcal{Z}_Q} W_1(Z, Z_\theta) \quad (2.16)$$

Let Y be a distribution with a bounded first moment and U a uniform distribution over N diracs as defined earlier with support $\{\theta_1, \dots, \theta_N\}$. Then,

$$W_1(Y, U) = \sum_{i=1}^N \int_{\tau_{i-1}}^{\tau_i} |F_Y^{-1}(\omega) - \theta_i| d\omega \quad (2.17)$$

Lemma 4. For any $\tau, \tau' \in [0, 1]$ with $\tau < \tau'$ and cumulative distribution function F with inverse F^{-1} , the set of $\theta \in \mathbb{R}$ minimizing

$$\int_{\tau}^{\tau'} |F_Y^{-1}(\omega) - \theta_i| d\omega \quad (2.18)$$

is given by $\left\{ \theta \in \mathbb{R} \mid F(\theta) = \left(\frac{\tau + \tau'}{2} \right) \right\}$. Therefore, the values of $\{\theta_i, \dots, \theta_N\}$ that minimize $W_1(Y, U)$ are given by $\theta_i = F_Y^{-1}(\hat{\tau}_i)$ where $\hat{\tau}_i = \frac{\tau_{i-1} + \tau_i}{2}$

Lemma 5. The value of the quantile function $F_Z^{-1}(\tau)$ for a given a distribution Z and a quantile τ is also the minimizer of the quantile regression loss.

$$\mathcal{L}_{QR}^{\tau}(\theta) := \mathbb{E}_{\hat{Z} \sim Z} [\rho_{\tau}(\hat{Z} - \theta)]$$

$$\text{where } \rho_{\tau}(u) = u(\tau - \delta_{\{u < 0\}}), \quad \forall u \in \mathbb{R}$$

Correspondingly, the values $\{\theta_i, \dots, \theta_N\}$ minimizing $W_1(Z, Z_{\theta})$ are also the minimizers of the following loss function

$$\sum_{i=1}^N \mathbb{E}_{\hat{Z} \sim Z_{\theta}} [\rho_{\hat{\tau}_i}(\hat{Z} - \theta_i)] \quad (2.19)$$

.

However, it is to be noted that the quantile regression loss is not smooth at zero.

So instead, to alleviate this issue, the modified Huber loss function is considered.

$$\mathcal{L}_\kappa(u) = \begin{cases} \frac{1}{2}u^2 & \text{if } |u| \leq \kappa \\ \kappa(|u| - \frac{1}{2}\kappa) & \text{otherwise} \end{cases} \quad (2.20)$$

$$\rho_\tau^\kappa(u) = |\tau - \delta_{\{u < 0\}}| \mathcal{L}_\kappa(u) \quad (2.21)$$

2.4.2 Implementation

Proposition 3 (Bellamare et.al[12]). *Let Π_{W_1} be the quantile projection defined as above, and when applied to value distributions gives the projection for each state-value distribution. For any two value distributions $Z_1, Z_2 \in \mathcal{Z}$ for an MDP with countable state and action spaces*

$$\tilde{d}_\infty(\Pi_{W_1} \mathcal{T}^\pi Z_1, \Pi_{W_1} \mathcal{T}^\pi Z_2) \leq \gamma \tilde{d}_\infty(Z_1, Z_2) \quad (2.22)$$

We therefore conclude that the combined operator $\Pi_{W_1} \mathcal{T}^\pi$ has a fixed point \hat{Z}^π and the repeated application of this operator (or its stochastic approximation) converges to \hat{Z}^π .

Additionally, since $\tilde{d}_p \leq \tilde{d}_\infty$, we conclude that the convergence occurs for all $p \in [1, \infty]$

Algorithm 2: Quantile Regression Algorithm: Q-learning:

Require: N - number of quantiles, κ - Huber loss hyperparameter, $\theta_i(x, a) \approx F_{Z(x,a)}^{-1}(\hat{\tau}_i)$

input: A transition $x, a, r, x', \gamma \in [0, 1)$

Compute the projection of $\mathcal{T}z_j$

$$Q(x', a') := \sum_j q_j \theta_j(x', a')$$

$$a^* \leftarrow \arg \max_{a'} Q(x', a')$$

$$\mathcal{T}\theta_j \leftarrow r + \gamma \theta_j(x', a^*) \quad \forall j$$

Compute the quantile regression loss from (2.21).

$$\rho_\tau^\kappa(u) = |\tau - \delta_{\{u < 0\}}| \mathcal{L}_\kappa(u)$$

output: $\sum_{i=1}^N \mathbb{E}_j [\rho_{\hat{\tau}_i}^\kappa(\mathcal{T}\theta_j - \theta_i(x, a))]$

2.5 Policy Gradient Methods

2.5.1 Introduction

In the context of learning the optimal behavior in a stochastic environment modeled as an Markov Decision Process (MDP), we explored the methods to do so indirectly through the optimal state action value function. This may be a feasible approach for applications with

discrete action space. However, in applications with continuous action space, the problem may become infeasible owing to high optimization cost of searching over the whole action space. This challenge can then be tackled by learning the policy function directly using the policy gradient methods. A parameterized policy π_θ can be used to select an action for each state $x \in \mathcal{X}$, which can be improved in the course of training to pick the optimal actions.

2.5.2 Policy Gradient Methods

We start by denoting the transition density of moving from state x to x' in time t by $p(x \rightarrow x', t, \pi)$. And observe that the integral $\int_{\mathcal{S}} \sum_{t=1}^{\infty} \gamma^{t-1} p_1(x) p(x, x', t, \pi) dx$ can be written as $\rho^\pi(x')$. Since the main goal of the agent is to maximize the cumulative discounted expected reward from a starting state $x \in \mathcal{X}$, we can write the objective function with respect to the policy parameter θ as the following

$$J(\theta) = \int_{\mathcal{X}} \rho^\pi(x) \left(\int_{\mathcal{A}} \pi_\theta(x, a) r(x, a) da \right) dx = \mathbb{E}_{x \sim \rho^\pi, a \sim \pi_\theta} [Q(x, a)] \quad (2.23)$$

For the objective function $J(\theta)$ where the policy function is stochastic, we can use the following theorem from Sutton[6] in our training algorithm to learn the policy.

Theorem 4 (Stochastic Policy Gradient). *For a policy π parameterized by θ , the gradient can be written as the following:*

$$\begin{aligned} \nabla_\theta J(\pi_\theta) &= \int_{\mathcal{S}} \rho^\pi(s) \left(\int_{\mathcal{A}} \nabla \pi_\theta(a|s) Q^\pi(s, a) da \right) ds \\ &= \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta} [\nabla_\theta \log(\pi_\theta(a|s)) Q^\pi(s, a)] \end{aligned} \quad (2.24)$$

Since the policy gradient depends on knowing the value function, we may further

parameterize the value function as $Q_\omega^\pi(a|x)$ and learn it simultaneously. This approach, of learning the policy and value function simultaneously, is called the **actor-critic** method. The above theorem remains unchanged for the value approximator, and training step expands into two main steps: first, the critic updates the value function parameters ω based on the training sample. Then the updated critic is used to compute the policy gradient and update the parameters accordingly. This two-pronged approach of training the interacting approximators is a quick learning tool, which can, upon culmination of the training algorithm yield both the value function and the corresponding policy function. Under suitable conditions, this approximator can be used to formulate the gradient as follows:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s \sim \rho^\pi, a \sim \pi_\theta} [\nabla_\theta \log(\pi_\theta(a|s)) Q^w(s, a)] \quad (2.25)$$

Things were trickier for policy gradient under the deterministic policy. It wasn't clear if the gradient even existed for such a policy until a proof of existence and consistency was given by Silver [15]. Under the assumption of a deterministic policy function parameterized by θ as $\mu_\theta : \mathcal{X} \rightarrow \mathcal{A}$, the following theorem gives us the framework for steps forward.

Theorem 5 (Deterministic Policy Gradient). *Under suitable conditions, the gradients $\nabla_\theta \mu_\theta(s)$ and $\nabla_a Q^\mu(s, a)$ exist and that the deterministic policy gradient exists and is given as*

$$\begin{aligned} \nabla_\theta J(\mu_\theta) &= \int_{\mathcal{S}} \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s, a)|_{a=\mu_\theta(s)} ds \\ &= \mathbb{E}_{s \sim \rho^\mu} [\nabla_\theta \log(\mu_\theta(s)) \nabla_a Q^\mu(s, a)|_{a=\mu_\theta(s)}] \end{aligned} \quad (2.26)$$

Just like before, the actor-critic method can be molded accordingly with a determin-

istic policy using the aforementioned theorem to compute the actor critic. This method was illustrated in the paper by Lillicrap et al. [16] using neural net approximators for the policy and value functions. For the corresponding critic update, the DDPG paper uses the objective as minimization of the TD error given by

$$L(w) = \mathbb{E}_\rho \left[(Q_w(x, a) - \mathcal{T}_{\pi_{\theta'}} Q_{w'}(x, a))^2 \right] \quad (2.27)$$

Following the development in the usage of deterministic policy, and improvement in the results showcased in DDPG and Distrbutional DQN, the authors of D4PG[17] combined the two concepts and introduced distributional gradient in the critic in addition to parallelism. Since $Q_\pi(x, a) = \mathbb{E}[Z_\pi(x, a)]$, we can rewrite this using distributional Bellman and the corresponding loss function to be minimized for the critic becomes

$$L(w) = \mathbb{E}_\rho \left[d(Z_w(x, a), \mathcal{T}_{\pi_{\theta'}} Z_{w'}(x, a)) \right] \quad (2.28)$$

where d distributional TD error introduced by Bellamare et al.[12]. Stripping away the parallelism component, the algorithm is as follows

Algorithm 3: D4PG:

1. **input:** batch size M , trajectory length N , exploration constant ϵ , initial learning rates α_0 and β_0 , distributional learning parameter N_{QUANTS} . Then define $\tau_j = j/N_{QUANTS}$ for $j = 0, \dots, N_{QUANTS}$.
 2. Initialize network weights (θ, ω) at random
 3. Initialize target weights $(\theta', \omega') \leftarrow (\theta, \omega)$
 4. **for** $t = 1, \dots, T$ **do**
 5. Sample M transitions $(\mathbf{x}_{i:i+N}, \mathbf{a}_{i:i+N-1}, r_{i:i+N-1})$ of length N from replay with the priority p_i
 6. Construct the target distributions $Y_i = \left(\sum_{n=0}^{N-1} \gamma^n r_{i+n} \right) + \gamma^N Z_{\omega'}(\mathbf{x}_{i+N}, \pi_{\theta'}(\mathbf{x}_{i+N}))$
 7. $\mathbb{E}[\nabla_{\mathbf{a}} Z_{\omega}(\mathbf{x}_i, \mathbf{a})]_{\mathbf{a}=\pi_{\theta}(\mathbf{x}_i)} = \sum_{i=1}^{N_{QUANTS}} [\tau_i - \tau_{i-1}] \nabla_{\mathbf{a}} Z_{\omega}(\mathbf{x}_i, \mathbf{a})_{\mathbf{a}=\pi_{\theta}(\mathbf{x}_i)}$
 8. Compute the actor and critic updates

$$\delta_{\omega} = \frac{1}{M} \sum_j \nabla_{\omega} d(Y_j, Z_{\omega}(\mathbf{x}_j, \mathbf{a}_j))$$

$$\delta_{\theta} = \nabla_{\theta} \frac{1}{M} \sum_{j=1}^M \sum_{i=1}^{N_{QUANTS}} [\tau_i - \tau_{i-1}] Z_{\omega}(\mathbf{x}_{i,j}, \mathbf{a})_{\mathbf{a}=\pi_{\theta}(\mathbf{x}_{i,j})}$$

$$= \frac{1}{M} \sum_{j=1}^M \sum_{i=1}^{N_{QUANTS}} \nabla_{\theta} \pi_{\theta}(\mathbf{x}_{i,j}) \mathbb{E}[\nabla_{\mathbf{a}} Z_{\omega}(\mathbf{x}_{i,j}, \mathbf{a})]_{\mathbf{a}=\pi_{\theta}(\mathbf{x}_{i,j})}$$
 10. Update the network parameters $\theta \leftarrow \theta + \alpha_t \delta_{\theta}$, $\omega \leftarrow \omega + \beta_t \delta_{\omega}$
 11. If $t = 0 \bmod t_{target}$, update the target networks $(\theta', \omega') \leftarrow (\theta, \omega)$
 12. If $t = 0 \bmod t_{actor}$, replicate the network weights to the actor
 13. **end for**
 14. **return** policy parameters θ
-

Actor

1. **repeat**
 2. Sample action $\mathbf{a} = \pi_{\theta}(\mathbf{x}) + \epsilon \mathcal{N}(0, 1)$
 3. Execute action \mathbf{a} , observe reward r and state \mathbf{x}'
 4. Store $(\mathbf{x}, \mathbf{a}, r, \mathbf{x}')$ in replay
 5. **until** learner finishes.
-

Chapter 3

Conic Q-learning

3.1 Motivation

We go back to the de-construction and reconstruction of value functions using value distributions; for a fixed policy π , the cumulative discounted return as a random variable $Z^\pi = \sum_{t=0}^{\infty} \gamma^t R_t$. Using this definition, we defined our standard value functions $V(x)$ and $Q(x, a)$ as the respective conditional linear expectations.

In the context of conic finance, where $Z(x, a)$ may represent a cash flow starting from state-action pair (x, a) , we are more interested in maximizing the bid price of this cash flow. Doing so requires us to formulate the corresponding value functions a bit differently than the standard literature.

3.2 Constructing the conic value functions

We rely on constructing the corresponding nonlinear expectations computed using conic distortion function MINMAXVAR, and analogously define the new **conic value func-**

tions: $W_\lambda(x)$ **the conic value function** of starting in state x ,

$$W_{\Psi_\lambda}^\pi(x) := \mathbb{D}_\lambda^\Psi [Z^\pi(x)] = \mathbb{D}_\lambda^\Psi \left[\sum_{t=0}^{\infty} \gamma^t R_t(x_t, a_t) \middle| x_0 = x \right] \quad (3.1)$$

and $K_\lambda(x, a)$ **the conic Q function** of starting with state-action pair (x, a) .

$$K_{\Psi_\lambda}^\pi(x, a) := \mathbb{D}_\lambda^\Psi [Z^\pi(x, a)] = \mathbb{D}_\lambda^\Psi \left[\sum_{t=0}^{\infty} \gamma^t R_t(x_t, a_t) \middle| x_0 = x, a_0 = a \right] \quad (3.2)$$

We denote the distortion parameter by λ to avoid confusing it with the discount rate. Since $\lambda = 0$ corresponds to linear expectations, notice that $V^\pi(x) = W_{\Psi_0}^\pi(x)$ and $Q^\pi(x, a) = K_{\Psi_0}^\pi(x, a)$.

3.3 Bid-optimal policy

The big hurdle remains that in order to maximize such a non-linear expectation, the standard approach of TD learning[7] is rendered ineffective due to the lack of additivity of nonlinear expectations. In this case, having access to the underlying value distribution is a possible work around to that problem.

Again, using the conic value function defined above, we can define the larger placeholder set of “optimal” policies as

$$\Pi_{\Psi_\lambda}^{**} = \{\pi \in \Pi \mid K_{\Psi_\lambda}^\pi(x, a) \geq K_{\Psi_\lambda}^{\pi'} \forall (x, a) \in \mathcal{X} \times \mathcal{A}, \text{ and for } \pi' \in \Pi\} \quad (3.3)$$

The set we want to get to is $\Pi^* \subset \Pi^{**}$, the policies corresponding optimal value distributions with expectation $K_{\Psi_\lambda}^*$ and matching the full distribution of the return.

Definition 11. *Given the state space \mathcal{X} , and a finite action space \mathcal{A} , the **bid-greedy***

policy function in conic finance may be defined as picking action that maximizes the bid price of cash flows received from that state onwards. That is,

$$\pi^*(\cdot|x) = \arg \max_a K_{\Psi_\lambda}^*(x, a) \quad (3.4)$$

where K_{Ψ}^* corresponds to the nonlinear expectation of an optimal value distribution Z^* .

Although the underlying random variable may not be unique, we can still perceive K_{Ψ}^* as optimal nonlinear value function.

$$K_{\Psi_\lambda}^*(x, a) = \mathbb{D}_\lambda^\Psi[Z^*(x, a)] \quad (3.5)$$

3.4 Remarks on consistency

The underlying approach here is developed here for a time-homogeneous Markov Decision Process. There is no pre-commitment to actions at a later point, but only an adherence to the notion of optimality defined for each state. For an application like pairs-trading, one step transitions when considered as the full episode lead to the objective of conservatively maximizing the reward in that step and moving on to the next state, wherein the agent has a different concept of optimality. However, if one were to use time dependence, the construction of our objective function would fail to be dynamically consistent as defined by Artzner et al.[25]. If we had a random sequence of payoffs $\{Z_t\}_{t=0}^T$ dependent on the policy over a set of consecutive periods $\mathbb{T} = \{0, 1, \dots, T\}$, one would like to find a policy π so as to fulfill the following objective

$$\max_{\pi \in \Pi} \mathbb{D}[Z_0^\pi + \gamma \mathbb{D}[Z_1^\pi + \dots]] \quad (3.6)$$

Future works optimizing for this objective could use the distributional analog of Bellman update with further motivation from works by Boda et al.[26] and Shapiro[27]. One may also explore other weaker forms of consistency discussed by Roorda and Schumacher [28].

3.5 Implementation

The risk averse Q-learning algorithm can be implemented by undertaking the distributional Bellman update, and then following the bid-greedy policy.

Algorithm 4: Distributional Algorithm for Conic Q-learning:

input: A transition $x, a, r, x', \gamma \in [0, 1]$

Compute $K_\lambda(x', a') = \mathbb{D}_\lambda[Z(x', a')] \quad \forall a'$

$a^* \leftarrow \arg \max_{a'} K_\lambda(x', a')$

Implement the distributional Bellman update

$$\tilde{Z}(x, a) \leftarrow r(x, a) + \gamma Z(x', a^*)$$

output: $W_p(Z(x, a), \tilde{Z}(x, a))$ #the Wasserstein distance

However, the implementation challenges require us to switch up and use the quantile regression algorithm as defined earlier. It is computationally convenient that the distorted expectations can be re-written in terms of the quantiles, as characterized by Dhaene et al. [5].

Algorithm 5: Distributional Algorithm for Conic Q-learning using quantile regression:

Require: N - number of quantiles, κ - Huber loss hyperparameter, $\theta_i(x, a) \approx F_{Z(x,a)}^{-1}(\hat{\tau}_i)$

input: A transition $x, a, r, x', \gamma \in [0, 1]$

Compute the projection of $\mathcal{T}z_j$

$K_\lambda(x', a') := \sum_j \hat{p}_j \theta_j(x', a')$ where $\hat{p}_j = \Psi(q_j) - \Psi(q_{j-1})$

$a^* \leftarrow \arg \max_a K(x', a)$

$\mathcal{T}\theta_j \leftarrow r + \gamma \theta_j(x', a^*) \quad \forall j$

Compute the quantile regression loss.

$\rho_\tau^\kappa(u) = |\tau - \delta_{\{u < 0\}}| \mathcal{L}_\kappa(u)$

output: $\sum_{i=1}^N \mathbb{E}_j [\rho_{\tau_i}^\kappa(\mathcal{T}\theta_j - \theta_i(x, a))]$

3.6 Application: Pairs Trading

3.6.1 Introduction

Pairs trading is a well known trading strategy in hedge funds and investment banks. Based on a simple concept, its considered a special case of “statistical arbitrage” wherein one takes opposite positions in two stocks simultaneously based on some predetermined

principle, and then unwinds the position later. The idea is that the two stocks are chosen based on historical data displaying little deviation in their price difference (or spread). If one believes that the deviation, over long term, is mean reverting and consistent, any divergence in the spread can be treated as a trading signal. By buying the decreasing stock (long position), and simultaneously selling (short position) the increasing stock, one can hope to generate a profit when the spread reverts to the mean. Developed in the 1980's as a "market neutral" trading strategy, pairs trading has been a cornerstone of quant based investment strategies, and further history can be found in Gatev et al.[23].

The implementation of this simple process can vary a lot. The two main steps i) finding two stocks that move together, and ii) determining the entry and exit criteria can both vary according to the trader preferences. One may devise simple tests or rules on the historical data, or model the spread as a mean reverting stochastic process and implement those rules on the process parameters. One such approach can be found in Elliot et al.[24] wherein they propose a mean reverting Gaussian Markov chain model for the spread. However, the thresholds for entry and exit strategies in addition to the problem of picking the two stocks to trade remains an unsettled question.

As an application of RL in the conic finance universe, we consider the problem of pairs trading. Furthermore, we choose here to be model agnostic and refrain from modeling the spread as a stochastic process. Instead we look at solving the dilemma of picking the stocks to trade in as well as determining the entry signal. This is done by letting our Reinforcement Learning agent make both those decisions.

3.6.2 Setting up the problem

Objective

We consider the events defined on a probability space (Ω, \mathcal{F}, P) and consider an infinite sequence of payoffs $\{Z_t\}_{t \geq 0}$ over a set of consecutive periods $\mathbb{T} = \{0, 1, \dots\}$. The relevant cash flow at each time step t is the reward earned on the gross underlying notional amount. Under this setting, our objective is to find the policy that, in each timestep τ , maximizes the bid price of sum of discounted cash flows from that step onwards. If the cash flows are determined by the choice of actions $a \in \mathcal{A}$, the pairs and positions chosen to trade in, our objective can, hence, be written as the following:

$$\max_{a_t \in \mathcal{A}} \mathbb{D}_\Psi \left[\sum_{t \geq \tau} \gamma^{t-\tau} Z_t \right], \forall \tau \geq 0 \quad (3.7)$$

Modeling this as a time-homogeneous MDP, the issue of time-consistency doesn't arise; in a world where the uncertainty resolution may change the nature of the underlying policy, and the agent may be retrained, the objective fulfills some sense of risk averse optimality. The objective, then, is to maximize the bid price of the cash flows received from trading equities over fixed horizons in each time step. Since the action space is finite, this is an apt problem to showcase Q-learning on. In order to train this agent, we model the interaction as an MDP $(\mathcal{X}, \mathcal{A}, R, P)$.

State space - \mathcal{X}

The training domain of the problem is restricted to minute interval price data over eleven stocks and ETFs; AAPL, JNJ, INTC, IBM, GE, MSFT, ORCL, XLF, XLE, XLV, XLY. We then train various trading agents for a choice of the stress parameter $\lambda = 0$ values in

our objective function. The state space is encoded with the price history, represented in three features (Close, High, Low), of these 11 holdings over the last 5 time steps. This gives a training sample of dimension $3 \times 11 \times 5$. Furthermore, the price history is normalized over the feature space, across the stocks. For a given feature k , the feature value f_{ij} for equity i and step j is encoded as

$$\hat{f}_{ijk} = \frac{f_{ijk}}{\sum_i^N f_{ijk}}.$$

We consider the period 08/08/2019 - 08/16/2019 as the training period, and 08/19/2019 - 08/26/2019 as the testing period.

Action space - \mathcal{A}

We let our agent pick the stocks and direction to trade in. For the general case of N stocks, each action $a \in \mathcal{A} \subset \mathbb{R}^N$

$$\mathbf{a}_{ij} = (a_1, a_2, \dots, a_N); \quad a_k = \begin{cases} 1, & k = i \\ -1, & k = j \\ 0, & \text{else} \end{cases}$$

Since $N = 11$, \mathcal{A} is a discrete set of 110 elements. That's is the depth of action space in our application. We skip the exit strategy aspect and automatically unwind after 5 timesteps to maintain a pliable action space.

Reward function - R

For each action $a \in \mathcal{A}$, we have two simultaneous equity trades in opposite directions. We consider those actions 1 and -1 as dollars invested, so this is a zero cost trading

strategy. Letting r_P and r_N denote the net absolute returns on the underlying 1 dollar in each direction, observe that the net return on the gross notional is $(r_P + r_N)/2$. This is the per step reward of the state-agent interaction. Now given that r_1, r_2, \dots, r_M are the step returns for M steps in a day, the daily return is arrived at by compounding these M returns. That is,

$$\tilde{r} = \prod_{i=1}^M (1 + r_i) - 1 \tag{3.8}$$

3.6.3 Neural Net architecture and implementation remarks

The original DQN paper by Mnih et al.[20] had great success with Atari games using Convolutional Neural Networks. The further improvements made in the distributional reinforcement learning literature built on that architecture, and modified only the final layer to be atoms of projected distribution. Therefore, we stick with Conv2D neural nets for this simple application and explore the performance. For exploration purposes, the agent is designed to be ϵ -greedy. We start with $\epsilon = 1$ and gradually take it to 0.01 over half a million steps. This is helpful in solving the exploration vs. exploitation dilemma [6].

```

DCNN(

(dconv): Sequential(

(0): Conv2d(3, 16, kernel size=(1, 2), stride=(1, 1))

(1): ReLU()

(2): Conv2d(16, 16, kernel size=(1, 2), stride=(1, 1))

(3): ReLU()

(4): Conv2d(16, 16, kernel size=(1, 2), stride=(1, 1))

(5): ReLU()

(6): Conv2d(16, 16, kernel size=(1, 2), stride=(1, 1))

(7): ReLU())

(fc): Sequential(

(0): Linear(in features=1056, out features=256, bias=True)

(1): ReLU()

(2): Linear(in features=256, out features=22000, bias=True)))

```

Table 3.1: 2D Convolutional Network for Distributional Learning using QR

3.6.4 Results

We compare the optimization performance of agents trained using conic Q-learning with different distortion parameter values. In addition, an agent is also trained using the standard Q-learning algorithm with all else constant.

λ	Distortion parameter for Q-nets				
	td-Q	0.0	0.25	0.5	1.0
mean-reward	13.330	27.649	19.049	21.772	16.845
minutes	180.256	1505.788	1887.677	1325.482	1765.126
training steps	2275.00k	2540.44k	2577.46k	2325.35k	2469.20k

Table 3.2: Convergence of 10 step moving average of mean reward for Q nets. Training done on 8 GB NVIDIA GeForce GTX 1650.

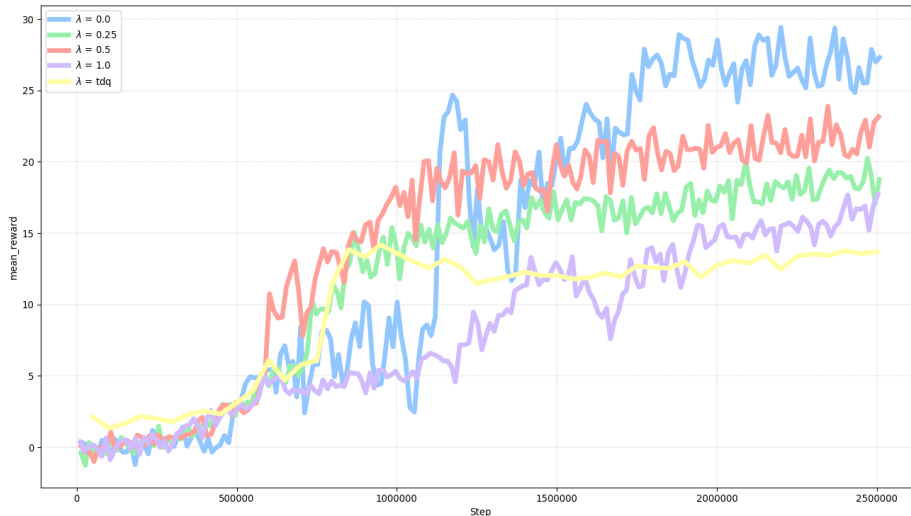


Figure 3.1: Evaluation results for neural net training

However, it is not enough to look at the average rewards in the financial context. The following tables and images showcase the difference in metrics of actual financial performance. The trained neural net is then tested on the training and test datasets by computing the trading performance on randomly sampling states, and their corresponding actions and rewards. We consider 2000 such samples of size 50 each. Starting with dollar capital at time $t = 0$, the random returns are accumulated, thereby constructing a trajectory of daily profits and losses for 50 timesteps. These 2000 random trajectories are then evaluated in terms of various financial metrics; we report Sharpe Ratio, Gain-Loss ratio, Maximum Drawdown, and Acceptability indices. First, the tables provide

pre-specified quantile levels of each of these metrics on both the training and testing data.

λ	Quantiles				
	1%	25%	50%	75%	99%
TD-Q	-2.292	1.503	2.964	4.149	7.044
0.0	-2.024	2.084	3.281	4.359	6.821
0.25	-2.159	2.035	3.159	4.157	7.052
0.5	-0.730	2.936	3.893	4.755	7.374
1.0	-1.525	2.231	3.368	4.305	7.087

Table 3.3: Sharpe Ratio on training data.

λ	Quantiles				
	1%	25%	50%	75%	99%
TD-Q	-2.040	1.493	3.018	4.199	7.203
0.0	-2.241	2.034	3.319	4.375	6.927
0.25	-2.442	1.916	3.214	4.214	7.075
0.5	-0.558	2.959	3.923	4.877	7.754
1.0	-1.671	2.395	3.402	4.368	6.981

Table 3.4: Sharpe Ratio on test data.

λ	Quantiles				
	1%	25%	50%	75%	99%
TD-Q	0.515	1.014	1.395	1.896	3.892
0.0	0.598	1.249	1.680	2.278	4.622
0.25	0.609	1.198	1.630	2.299	4.863
0.5	0.701	1.457	2.022	2.920	6.266
1.0	0.710	1.323	1.843	2.591	5.630

Table 3.5: Gain Loss ratio on training data.

λ	Quantiles				
	1%	25%	50%	75%	99%
TD-Q	0.510	1.020	1.413	1.935	4.450
0.0	0.628	1.196	1.647	2.233	4.727
0.25	0.589	1.167	1.602	2.286	4.865
0.5	0.741	1.490	2.057	2.881	6.185
1.0	0.697	1.333	1.877	2.620	5.599

Table 3.6: Gain Loss ratio on test data.

It can be observed from the quantiles that there is a clear shift to the right in the Maximum Drawdown metric, implying the fulfillment of our risk-averse objective. These results also carry over to the test data.

λ	Quantiles				
	1%	25%	50%	75%	99%
TD-Q	0.545	1.493	2.222	3.358	7.512
0.0	0.563	1.374	1.969	2.767	5.974
0.25	0.402	0.855	1.249	1.827	4.297
0.5	0.316	0.608	0.851	1.240	2.655
1.0	0.346	0.685	0.998	1.455	3.141

Table 3.7: Maximum Drawdown on training data.

λ	Quantiles				
	1%	25%	50%	75%	99%
TD-Q	0.648	1.501	2.250	3.423	7.437
0.0	0.545	1.420	1.994	2.820	5.897
0.25	0.384	0.848	1.267	1.855	4.044
0.5	0.305	0.603	0.850	1.250	2.561
1.0	0.328	0.702	0.987	1.435	3.127

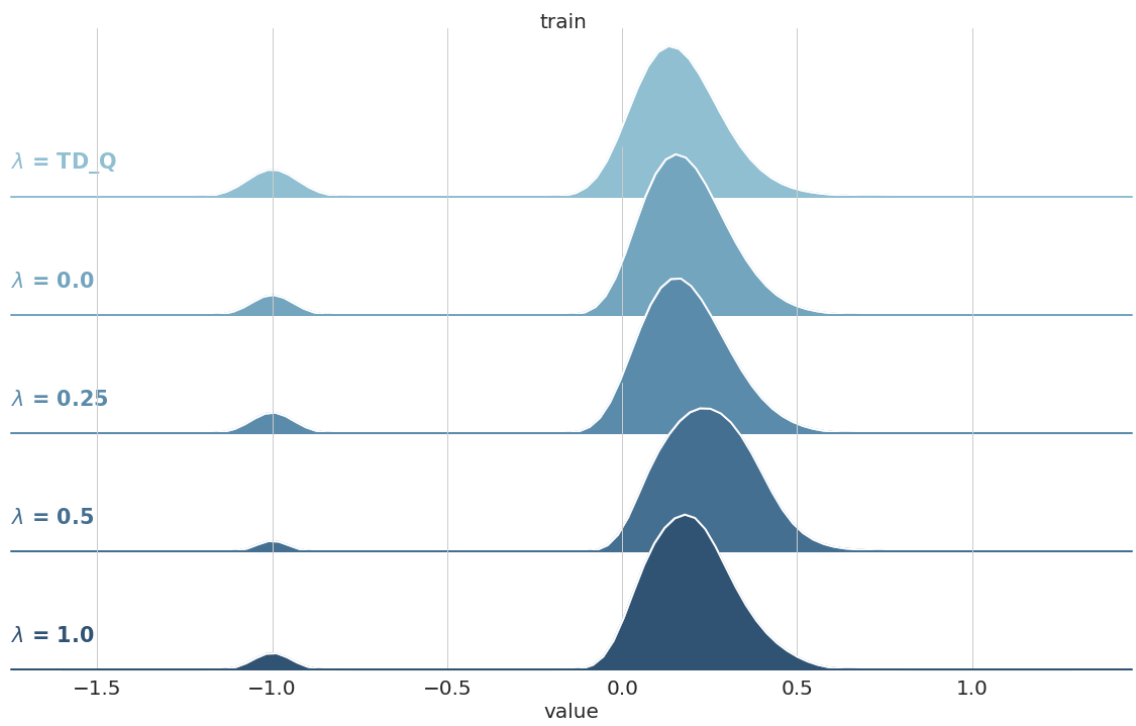
Table 3.8: Maximum Drawdown on test data.

λ	Quantiles				
	1%	25%	50%	75%	99%
TD-Q	-1.000	0.065	0.138	0.219	0.475
0.0	-1.000	0.092	0.164	0.246	0.468
0.25	-1.000	0.089	0.162	0.246	0.477
0.5	-1.000	0.148	0.237	0.328	0.557
1.0	-1.000	0.101	0.183	0.264	0.497

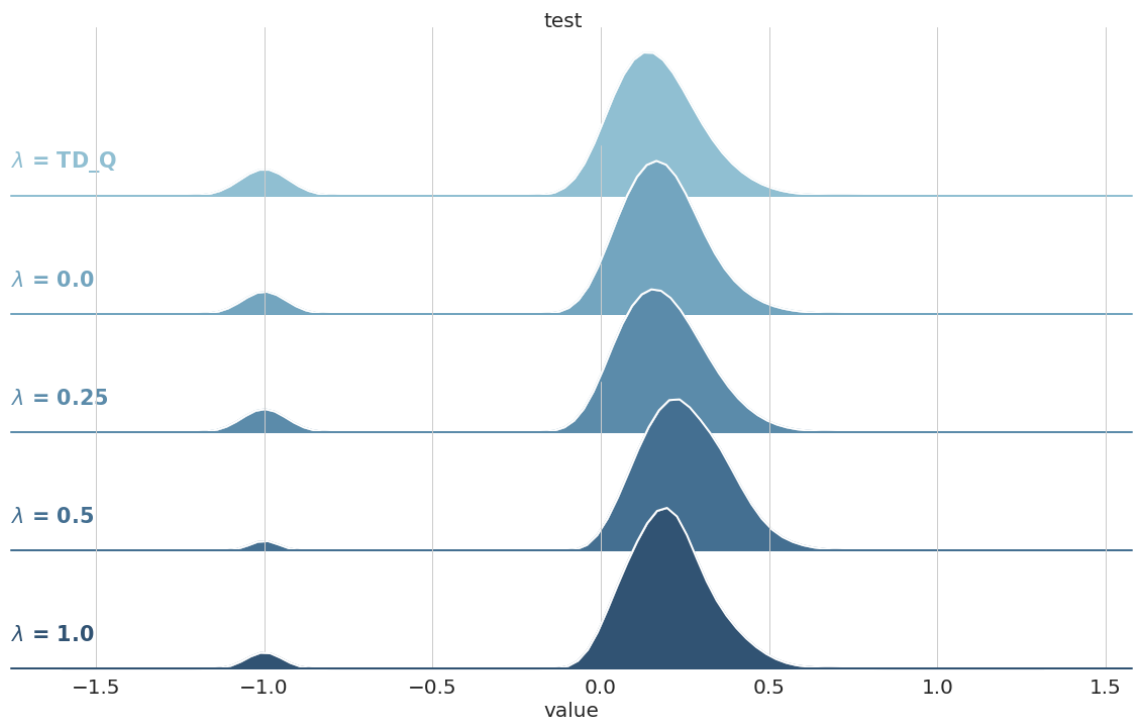
Table 3.9: Acceptability Index on training data.

λ	Quantiles				
	1%	25%	50%	75%	99%
TD-Q	-1.000	0.062	0.141	0.220	0.471
0.0	-1.000	0.089	0.165	0.242	0.475
0.25	-1.000	0.080	0.164	0.249	0.474
0.5	-1.000	0.154	0.237	0.327	0.554
1.0	-1.000	0.113	0.190	0.263	0.501

Table 3.10: Acceptability Index on test data.

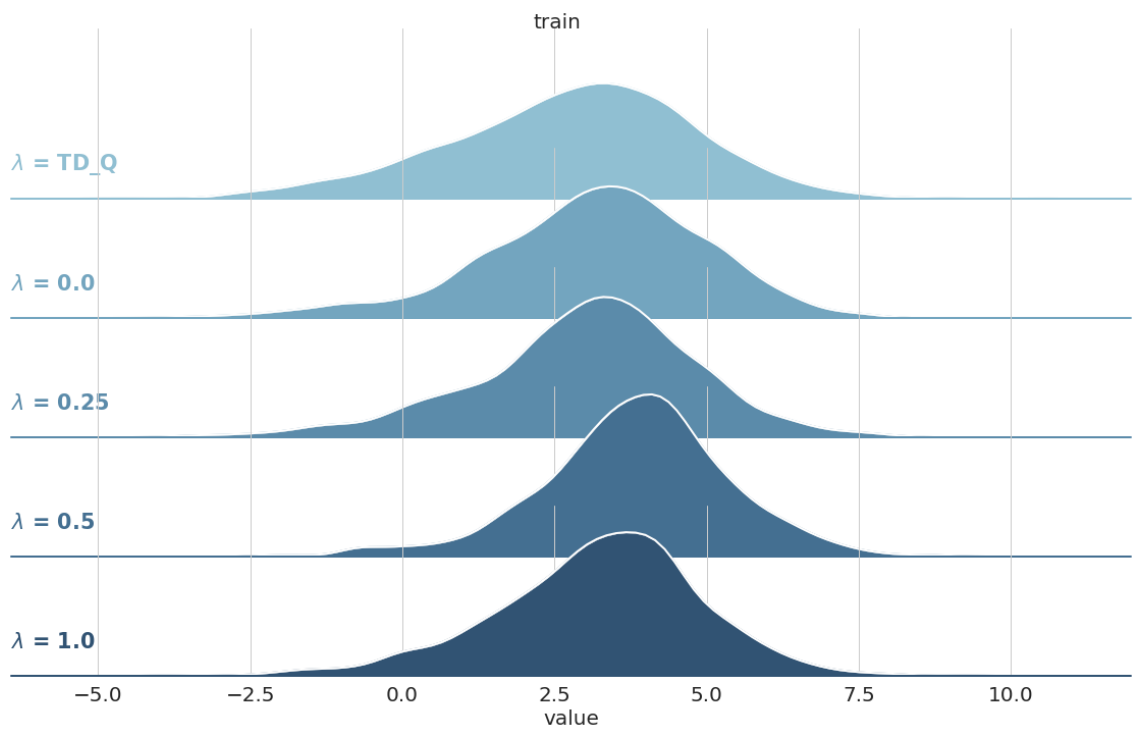


(a) Training data

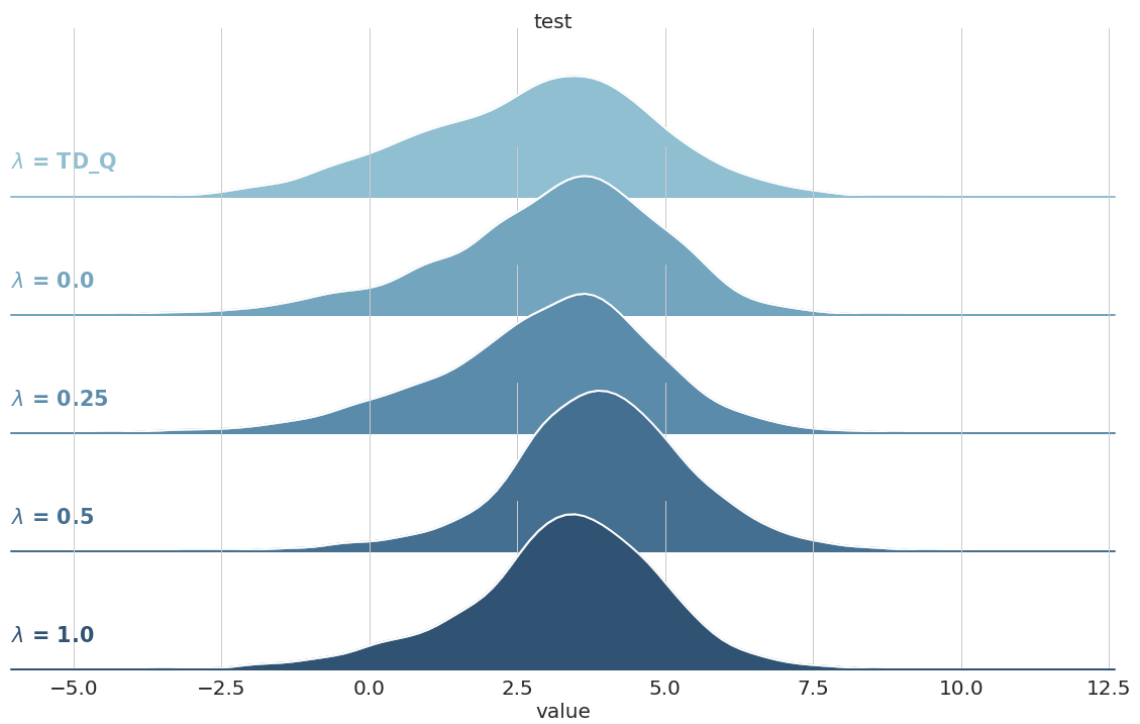


(b) Testing data

Figure 3.2: Acceptability indices comparison

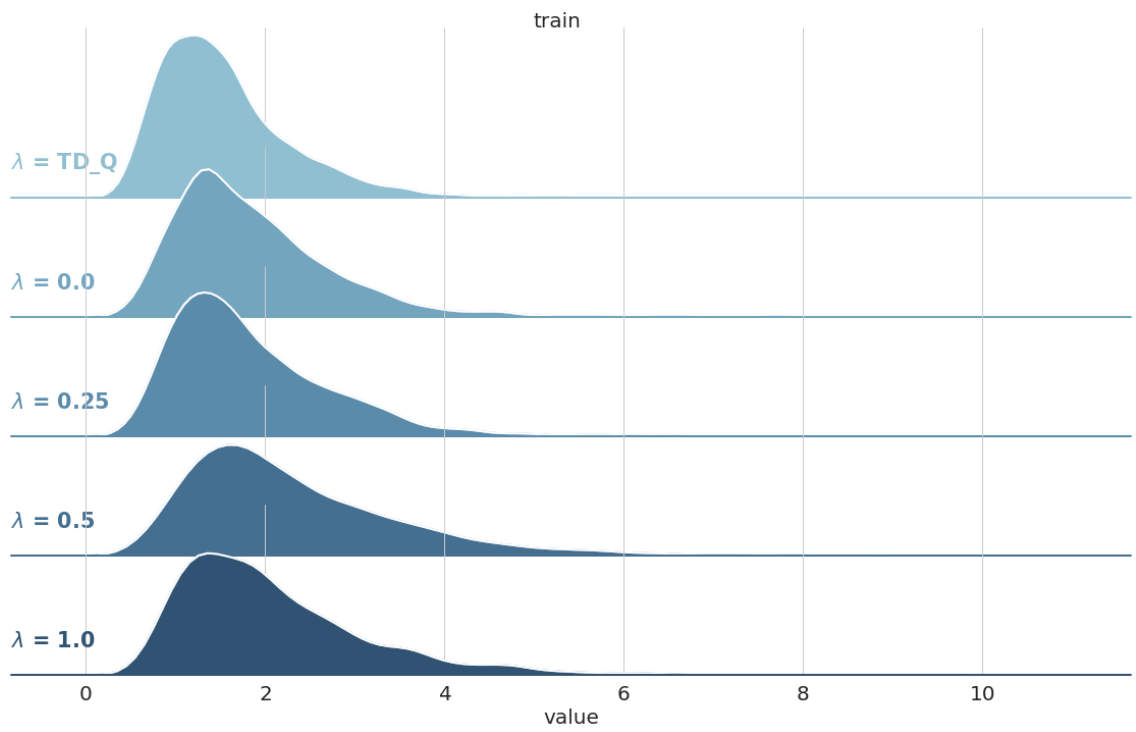


(a) Training data

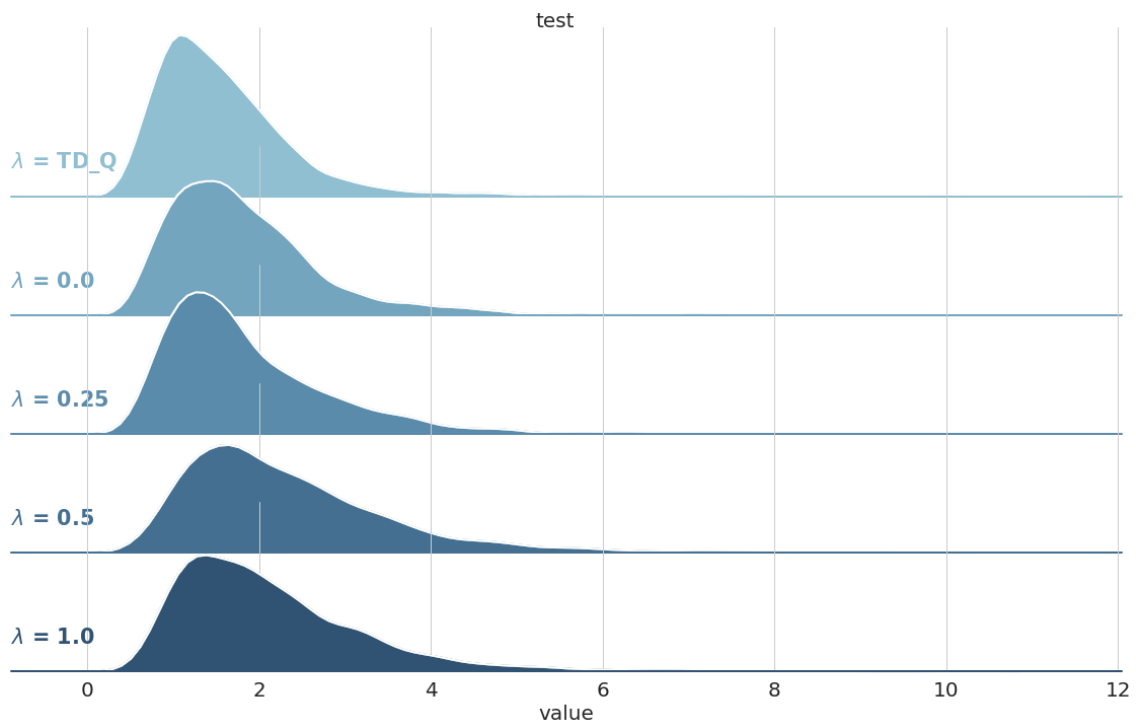


(b) Testing data

Figure 3.3: Sharpe Ratio comparison

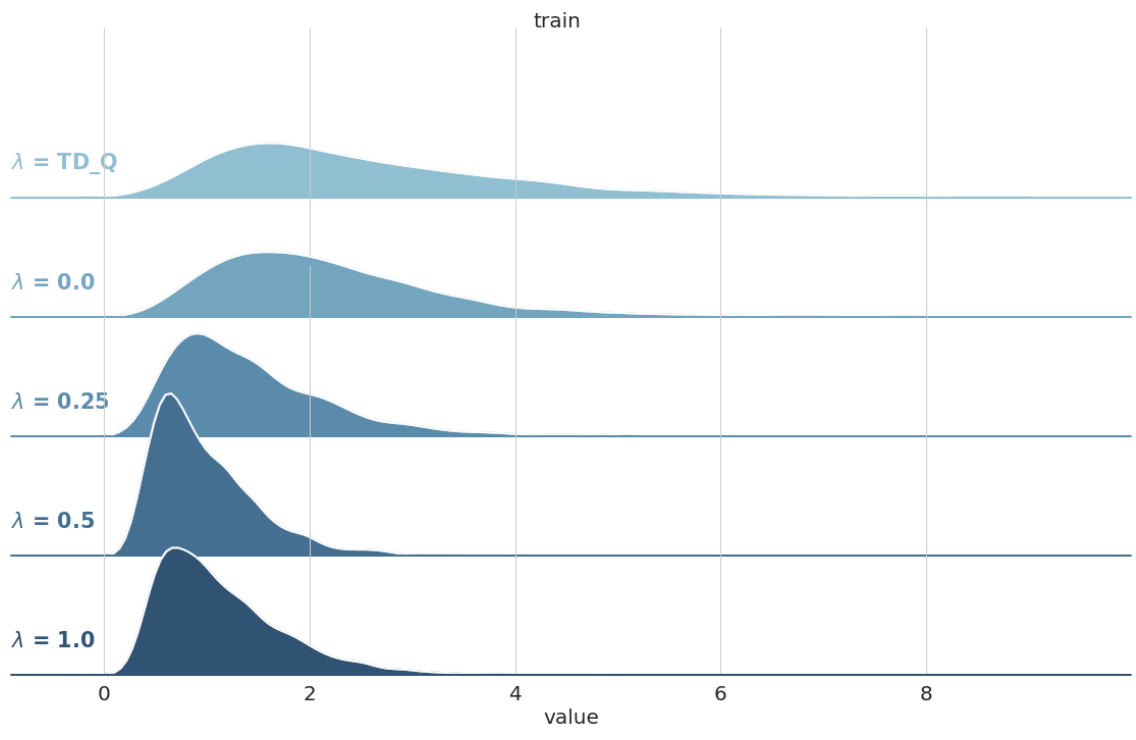


(a) Training data

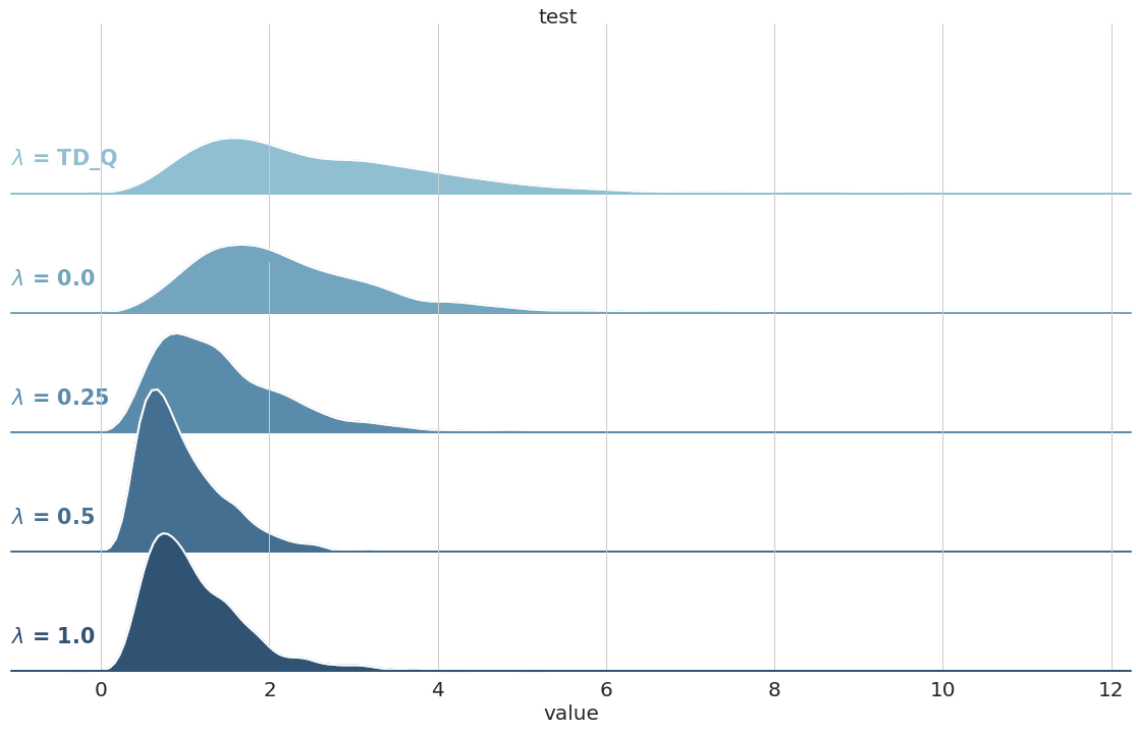


(b) Testing data

Figure 3.4: Gain-Loss ratio comparison



(a) Training data



(b) Testing data

Figure 3.5: Maximum Drawdown comparison

Chapter 4

Deterministic policy gradient and distorted least squares

4.1 Motivation

Mean squared error (MSE), as a loss function, is one of the standard choices one may use for training prediction and forecasting algorithms. However, there may be applications where one may not want to be direction agnostic to the error and, instead, prefer to directionally pursue the target. This direction specific forecasting or prediction, hence, must modify the respective loss function. Owing to the literature by Madan and Schoutens[3], we have what is called the *distorted mean squared error*. Herein, the statistical probabilities on one side are inflated while deflating the probabilities on the other side. Done so using distortion functions on the cumulative distribution functions, the solution to such a distorted least squares optimization, as a consequence, is biased. We try and use actor-critic method with distributional networks to minimize such a loss function. However, such an optimization needs novel methods for the primal problem owing to the lack of gradient on the sorting

operation we end up using in our distortion application. In the following sections, we explore the application of Policy Gradient methods using Distributional Reinforcement Learning for those. We summarize the performance by comparing the algorithm on a financial problem pertaining ETFs.

4.2 Constructing the new loss function

We discuss here the approach of distorted least squares as introduced Madan and Schoutens[3].

The starting point of this discussion is to consider the span of parameterized function $f_\theta(x)$ such that a sequence of target outcomes y_i can be predicted based on observed features x_i as $y = f_\theta(x) + \epsilon$. The standard approach of solving this prediction problem is to optimize for θ so that some scalar multiple of the mean squared distance,

$$\frac{1}{N} \sum_i^N (y_i - f_\theta(x_i))^2 \tag{4.1}$$

is minimized, say for some optimal θ^* . This was shown to be the minimizer of the euclidean distance to the conditional expectation $g(x) := \frac{1}{N} \sum_i^N y_i$ on the manifold defined by $f_\theta(x)$. In order to obtain such a minimizer on a conditional expectation obtained using distorted expectations, we refer to the following construction

- For a random variable Y , obtain the respective target quantiles as $q_i = \frac{1}{N} \sum_j^N \mathbf{1}_{y_j \leq y_i}$
- Define the distorted weights from quantiles as $w_i := \Psi(q_i) - \Psi(q_i - \frac{1}{N})$, for some concave(convex) distortion function Ψ .

- Define the conditional distorted expectation of Y

$$\mathbb{G}_\Psi(Y) := \sum_i^N y_i w_i \tag{4.2}$$

- For the prediction problem, response variable y being estimated by a parametrized function $f_\theta(x)$, the i^{th} residual is $r_{i,\theta} = y_i - f_\theta(x_i)$.

The corresponding *distorted least squares* problem then is of minimizing, w.r.t. θ ,

$$DMSE(\theta) = \sum_i^N w_i (y_i - f_\theta(x_i))^2 = \sum_i^N w_i r_{i,\theta}^2 = \mathbb{D}_\Psi(R^2) \tag{4.3}$$

The choice of Ψ , obviously, dictates the kind of distorted expectation we are aiming for. The intuition is that the minimizer here, $f_{\theta^*}(x)$, is the point on the manifold of $f_\theta(x)$ that is closest to the conditional distorted expectation, as opposed to the conditional linear expectation. The choice of Ψ , obviously, dictates the kind of distorted expectation we are aiming for.

4.3 Implementation using Distributional Reinforcement Learning

In this section, we demonstrate a method for approximately minimizing distorted least squares. Upon building the general notation, and making some design choices, we explore the results of such an approach to efficiently track an ETF.

As reviewed earlier, distributional deterministic policy gradient based actor-critic methods give us access to the underlying distribution, which can be used to optimize for distorted expectations. We learn the approximate value function $Q(s, a)$ and the ideal policy func-

tion $\pi(\cdot|x)$ simultaneously, both represented by neural networks. We approximate the value function as before and compute the gradients to update the policy function μ_θ by gradient ascent, and then the value function. This interweaving method eventually leads to convergence to policy μ that maximizes the objective function J , or minimizes the distorted mean squared error. Crucial to this work was the **DPG theorem** since the characterization of the gradient doesn't need integration over action spaces and may, as a consequence, require fewer samples to train.

Algorithm 6: Distributional Distorted Policy Gradient:

1. **input:** batch size M , trajectory length N , exploration constant ϵ , initial learning rates α_0 and β_0 , distributional learning parameter N_{ileS} , and the distortion function Ψ . Then define $\tau_j = j/N_{QUANTS}$ for $j = 0, \dots, N_{QUANTS}$.
 2. Initialize network weights (θ, ω) at random
 3. Initialize target weights $(\theta', \omega') \leftarrow (\theta, \omega)$
 4. **for** $t = 1, \dots, T$ **do**
 5. Sample M transitions $(\mathbf{x}_{i:i+N}, \mathbf{a}_{i:i+N-1}, r_{i:i+N-1})$ of length N from replay with the priority p_i
 6. Construct the target distributions $Y_i = \left(\sum_{n=0}^{N-1} \gamma^n r_{i+n} \right) + \gamma^N Z_{\omega'}(\mathbf{x}_{i+N}, \pi_{\theta'}(\mathbf{x}_{i+N}))$
 7. $\mathbb{E}[\nabla_{\mathbf{a}} Z_{\omega}^2(\mathbf{x}_i, \mathbf{a})] \Big|_{\mathbf{a}=\pi_{\theta}(\mathbf{x}_i)} = \sum_{i=1}^{N_{QUANTS}} [\Psi(\tau_i) - \Psi(\tau_{i-1})] \nabla_{\mathbf{a}} (Z_{\omega}^{\Psi(\tau_i)}(\mathbf{x}_i, \mathbf{a}) \Big|_{\mathbf{a}=\pi_{\theta}(\mathbf{x}_i)})^2$
 8. Compute the actor and critic updates

$$\delta_{\omega} = \frac{1}{M} \sum_j \nabla_{\omega} d(Y_j, Z_{\omega}(\mathbf{x}_j, \mathbf{a}_j))$$

$$\delta_{\theta} = -\nabla_{\theta} \frac{1}{M} \sum_{j=1}^M \sum_{i=1}^{N_{QUANTS}} [\Psi(\tau_i) - \Psi(\tau_{i-1})] (Z_{\omega}^{\Psi(\tau_i)}(\mathbf{x}_{i,j}, \mathbf{a}) \Big|_{\mathbf{a}=\pi_{\theta}(\mathbf{x}_{i,j})})^2$$

$$= -\frac{1}{M} \sum_{j=1}^M \sum_{i=1}^{N_{QUANTS}} \nabla_{\theta} \pi_{\theta}(\mathbf{x}_{i,j}) \mathbb{E}[\nabla_{\mathbf{a}} Z_{\omega}^2(\mathbf{x}_{i,j}, \mathbf{a})] \Big|_{\mathbf{a}=\pi_{\theta}(\mathbf{x}_{i,j})}$$
 9. Update the network parameters $\theta \leftarrow \theta + \alpha_t \delta_{\theta}$, $\omega \leftarrow \omega + \beta_t \delta_{\omega}$
 10. If $t = 0 \bmod t_{target}$, update the target networks $(\theta', \omega') \leftarrow (\theta, \omega)$
 11. If $t = 0 \bmod t_{actor}$, replicate the network weights to the actor
 12. **end for**
 13. **return** policy parameters θ
-

Actor

1. **repeat**
 2. Sample action $\mathbf{a} = \pi_{\theta}(\mathbf{x}) + \epsilon \mathcal{N}(0, 1)$
 3. Execute action \mathbf{a} , observe reward r and state \mathbf{x}'
 4. Store $(\mathbf{x}, \mathbf{a}, r, \mathbf{x}')$ in replay
 5. **until** learner finishes.
-

4.3.1 Further Remarks

1. Although τ are uniform and represent the probabilities of the corresponding quantile values, we apply Ψ to it based on the λ value. This reweighting of the quantiles allows us to use the result by Dhaene and compute the approximate distorted expectation.
2. The neural nets are trained using the **ADAM** algorithm.
3. The agent **AgentD4PG** is defined to be a noisy agent to accompany the deterministic actor. Doing so ensures sufficient exploration and convergence of the corresponding approximation functions.

4.4 Application of DLS: tracking an ETF

ETF trading has grown tremendously over the last decade, offering consumers an easy way of replicating and implementing quant strategies. Some of the most frequently traded ETFs and their top 10 composite equities are given in the table below.

SPY	MSFT	AAPL	AMZN	FB	BRK/B	GOOG	JPM	GOOGL	JNJ	V
21.55	4.28	3.74	3.05	1.81	1.59	1.49	1.47	1.45	1.37	1.3
XLE	XOM	CVX	COP	SLB	EOG	PSX	OXY	KMI	MPC	VLO
77.42	23.27	21.5	5.78	4.52	4.23	4.02	3.88	3.79	3.32	3.11
XLB	LIN	ECL	DD	APD	SHW	NEM	PPG	BLL	DOW	LYB
65.491	15.48	7.89	7.8	7.48	6.54	4.89	4.21	4.04	3.85	3.31
XLY	AMZN	HD	MCD	SBUX	NKE	LOW	BKNG	TJX	TGT	GM
65.75	21.79	11.17	7.38	5.1	4.9	3.95	3.75	2.99	2.47	2.25
XLV	JNJ	MRK	UNH	PFE	ABT	MDT	AMGN	TMO	ABBV	LLY
50.62	9.99	6.54	6.39	5.95	4.46	4.28	3.72	3.47	2.93	2.89
XLF	BRK/B	JPM	BAC	WFC	C	AXP	CME	USB	CB	GS
54.78	12.52	11.52	7.74	6.02	4.81	2.59	2.52	2.52	2.32	2.22

Table 4.1: Breakdown of top 10 (by cap weight in percentage) of Basket Holdings for popular ETFs - September 2019

For illustration purposes, we look at the actively tracking the daily returns of the financial sector ETF **XLF** using its top 10 composite stocks. There could possibly be a few

motivating factors here: one may not want to include illiquid assets in the ETF, or there may be deviations from market cap based considerations, or diversification risk. Additionally, one may be look to discover alpha generating strategies that directly attempt to outperform a benchmark, an ETF in this case. For a passively managed tracker, one could replicate the whole portfolio of stocks in the Index/Sector and weigh them according to their market cap. Passively managing the tracker can be costly if we include all the composite stocks, an on the other hand may have a has high error if we reduce the number of basket holdings. Since the purpose of this discussion is to track in a direction specific manner, and not dimension reduction, we pick the top 10 composite stocks as our basket holdings and illustrate the difference in tracking when minimizing the distorted mean squared error with different distortion parameter of our distortion function. Even then, we must preserve some characteristics of our original ETF. Once a basket of stocks/equities have been picked to replicate, the corresponding ETF must demonstrate a *similar* risk-return profile. Although, the exact definition of a similar risk-return profile remains a party specific choice.

This tracking problem is very much like a portfolio managing problem with one key difference. Whereas in a portfolio managing problem, the objective may be to maximize a certain risk-return profile, or to maximize a performance metric, tracking a benchmark needs one to follow a moving target. Some funds offer embedded leverage with higher returns on the upside, but lower return on the downside. The principal of Distorted Least Squares allow us to further track in that specific manner, so as to match the risk-return profile and yet (stochastically) dominate the return. Another way of looking at this may be in the context of statistical arbitrage where one may be looking to construct a basket of long-short positions. If one can track an ETF in a direction specific manner, one may

generate alpha return by holding the tracker and the benchmark in opposite directions.

4.4.1 Setting up the problem

Objective

As previously discussed in the pairs-trading section, the events are defined on a probability space (Ω, \mathcal{F}, P) and we consider a sequence of residuals $\{Z_t\}_{t \geq 0}$ over a set of consecutive periods $\mathbb{T} = \{0, 1, \dots\}$. The residuals at each time step are computed by taking the difference between weighted returns of N basket holdings X_{it} $i = 1, \dots, N$ and the underlying trackee return Y_t at time t . The weights $a_{i(t-1)}$ for each of these holdings are chosen one time step prior. The residual at time t can be written as the following

$$Z_t = Y_t - \sum_{i=1}^N a_{i(t-1)} X_{it} \quad (4.4)$$

Under this setting, our objective is to find the policy that, in each timestep t , minimizes the distorted expectation of the residual squared in the next time step. If the residuals are determined by the choice of actions $a \in \mathcal{A}$, the weights of our tracker portfolio of basket holdings, our objective can, hence, be written as the following:

$$\min_{a_t \in \mathcal{A}} \mathbb{D}_\Psi [Z_{t+1}^2], \quad t = 0, \dots, T \quad (4.5)$$

To this end, we consider setting this problem in the context of a simple one step Markov Decision Process. By taking continuous actions in each step, we use the policy gradient method in the distributional context and try to find the policy that yields the optimal policy which minimizes the distorted mean square error in each step. Additionally, this

problem doesn't suffer from time-consistency issues because of the static decision making only for one state at a time, and not worrying about the future states.

State space - \mathcal{X}

- We encode the state space with the price history, represented in three features (Close, High, Low), of these 11 holdings (basket plus the ETF) over the last 10 time steps. This gives a training sample of dimension $3 \times 11 \times 10$.
- Furthermore, the price history is normalized over the feature space, by taking log relative prices with respect to the pen price. For a given feature k , the feature value f_{ij} for equity i and step j is encoded as

$$\hat{f}_{ijk} = \log \left(\frac{f_{ijk}}{open_{ij}} \right)$$

- We consider the period 01/02/2008 - 12/29/2017 as the training period, and 01/02/2018 - 08/30/2019 as the testing period.

Tracking the ETF is a passive enough problem that we consider daily returns and rebalance our tracker at the end of each day. For each day, we are interested in choosing the weights so that the cumulative return of our basket most closely mimics the ETF.

Action space - \mathcal{A} and interaction

Let $a_{i,t}$ denote the weight chosen at time t for holding i . For the general case of N stocks, each action $\mathbf{a}_t \in \mathcal{A} \subset \mathbb{R}^N$

$$\mathbf{a}_t = (a_{1,t}, a_{2,t}, \dots, a_{N,t}); \quad \sum a_{k,t} = 1$$

The action space, thus, is an N dimensional simplex.

Reward function - R

In timestep $t + 1$, let y_{t+1} denote the percentage return from the ETF. By letting $r_{i,t+1}$ denote the percentage return in holding i in timestep $t + 1$, we see that the cumulative return on the tracker is $\hat{y}_{t+1} = \sum_i a_{i,t} r_{i,t+1}$. For the state action (x_t, a_t) we define the residual random variable as $R_{t+1} := \hat{y}_{t+1} - y_{t+1}$. The main objective then is to find a policy minimizing $\mathbb{D}^\Psi[R_{t+1}^2]$ for each t .

4.4.2 Neural Net architecture and implementation remarks

Although LSTMs have shown to be remarkable in forecasting time-series data, they are expensive to train. One alternate, as discussed by Borovykh et al.[18], is the usage of CNNs and Dilated CNNs. In the standard literature on distributional Reinforcement Learning, the authors at DeepMind have used CNNs (albeit, not for time series data). The network proposed uses stacked dilated convolutions that allow it to access data in a time-sensitive manner, a feature of Recurrent Networks that we discard. An added benefit of CNNs is that it is well equipped to process correlated data.

```

DCNN2Actor(

(dconv): Sequential(

(0): Conv2d(3, 32, kernel size=(1, 2), stride=(1, 1))

(1): ReLU()

(2): Conv2d(32, 32, kernel size=(1, 2), stride=(1, 1))

(3): ReLU()

(4): Conv2d(32, 32, kernel size=(1, 2), stride=(1, 1))

(5): ReLU()

(6): Conv2d(32, 32, kernel size=(1, 2), stride=(1, 1))

(7): ReLU()

(fc): Sequential(

(0): Linear(in features=2112, out features=512, bias=True)

(1): ReLU()

(2): Linear(in features=512, out features=128, bias=True)

(3): ReLU()

(4): Linear(in features=128, out features=10, bias=True)

(5): Softmax(dim=None))

```

Table 4.2: Dilated Convolutional Network - Actor

```

DCNN2QRCritic(

(dconv): Sequential(

(0): Conv2d(3, 32, kernel size=(1, 2), stride=(1, 1))

(1): ReLU()

(2): Conv2d(32, 32, kernel size=(1, 2), stride=(1, 1))

(3): ReLU()

(4): Conv2d(32, 32, kernel size=(1, 2), stride=(1, 1))

(5): ReLU()

(6): Conv2d(32, 32, kernel size=(1, 2), stride=(1, 1))

(7): ReLU())

(fc): Sequential(

(0): Linear(in features=2122, out features=512, bias=True)

(1): ReLU()

(2): Linear(in features=512, out features=512, bias=True)

(3): ReLU()

(4): Linear(in features=512, out features=200, bias=True)))

```

Table 4.3: Dilated Convolutional Network with Quantile Regression - Critic

4.4.3 Results and performance

The training performance, as visualized below, is similar for different distortion parameters. There is little impact on the convergence on either Actor or Critic net.

λ	Distortion parameters for actor nets				
	0.0	0.50	0.75	1.0	1.5
DMSE	0.0070	0.0087	0.0073	0.0062	0.0074
minutes	31.009	32.699	24.261	32.359	33.571
training steps	174.42k	182.97k	138.01k	180.09k	186.61k

Table 4.4: Convergence of 10 step moving average of DMSE for actor nets. Training done on 4 GB NVIDIA GTX 1050 Ti.

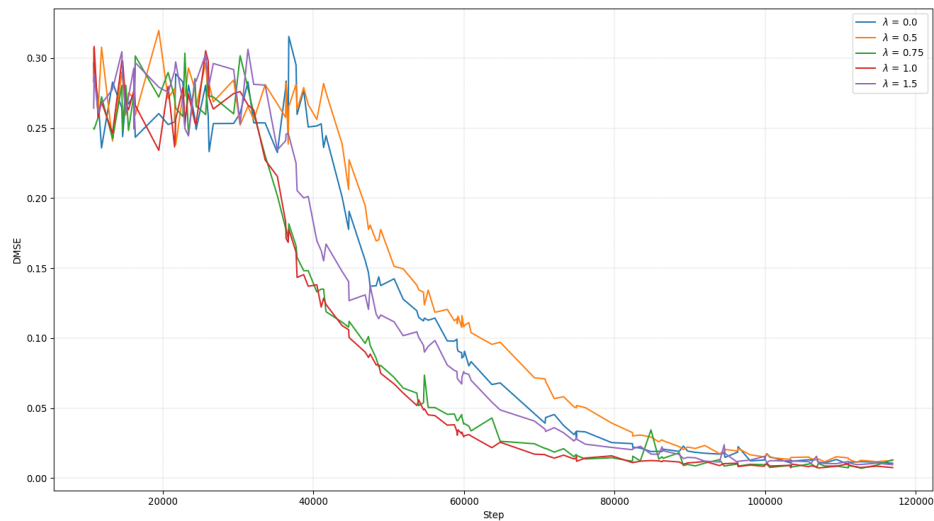


Figure 4.1: Convergence results for neural net training - Actor

Furthermore, we check the performance of the trained nets from a business perspective. If indeed the tracker can be trained so as to dominate the return of the ETF, a viable trading strategy is then to buy the tracker portfolio, and sell the underlying ETF. To compare the corresponding performance, we generate 2000 trajectories of length 100 each on such trading scenarios and compare various financial metrics to gauge the performance. As before, we start with dollar capital at time $t = 0$, the random returns are accumulated, thereby constructing a trajectory of daily profits and losses for 100. These 2000 random

trajectories are then evaluated in terms of their Sharpe Ratio, Gain-Loss ratio, Maximum Drawdown, and Acceptability indices. The following tables display the quantiles levels of these metrics on both the training and testing data. We compare the performance of running this business with different distortion parameter values. In addition, as a benchmark, we compare the performance (Sharpe Ratio, Gain Loss ratio, Acceptability Index, and Maximum Drawdown) with a business that uses Ordinary Least Squares (reg).

λ	Quantiles				
	1%	25%	50%	75%	99%
reg	-4.070	-1.787	-0.714	0.416	2.939
0.0	-3.427	-1.152	-0.110	1.035	3.485
0.5	-2.913	-0.572	0.516	1.601	4.115
0.75	-3.472	-1.054	-0.004	1.133	3.594
1.0	-3.699	-1.117	0.062	1.064	3.388
1.5	-3.389	-1.021	0.037	1.202	3.823

Table 4.5: Sharpe Ratio on training data.

λ	Quantiles				
	1%	25%	50%	75%	99%
reg	-3.955	-1.756	-0.699	0.506	3.031
0.0	-3.104	-0.683	0.256	1.198	3.429
0.5	-2.058	0.357	1.331	2.274	4.269
0.75	-3.004	-0.517	0.409	1.331	3.439
1.0	-2.512	-0.169	0.793	1.726	3.954
1.5	-2.186	0.246	1.158	2.040	4.545

Table 4.6: Sharpe Ratio on testing data.

λ	Quantiles				
	1%	25%	50%	75%	99%
reg	0.451	0.723	0.879	1.068	1.730
0.0	0.516	0.853	1.049	1.258	1.969
0.5	0.622	0.891	1.061	1.249	1.851
0.75	0.539	0.841	1.008	1.197	1.839
1.0	0.597	0.875	1.042	1.252	1.966
1.5	0.535	0.820	0.992	1.196	1.797

Table 4.7: Gain Loss ratios on training data.

λ	Quantiles				
	1%	25%	50%	75%	99%
reg	0.452	0.725	0.873	1.067	1.814
0.0	0.761	0.990	1.102	1.231	1.609
0.5	0.724	0.995	1.139	1.302	1.760
0.75	0.760	0.963	1.071	1.198	1.522
1.0	0.711	0.942	1.045	1.166	1.538
1.5	0.734	0.943	1.047	1.159	1.525

Table 4.8: Gain Loss ratios on testing data.

λ	Quantiles				
	1%	25%	50%	75%	99%
reg	1.581	3.777	5.479	7.670	13.878
0.0	2.629	5.798	8.657	12.697	26.574
0.5	2.441	4.999	7.159	10.026	18.545
0.75	2.812	5.819	8.086	11.270	22.616
1.0	2.830	5.982	8.309	11.354	21.418
1.5	2.718	5.744	8.469	12.214	23.679

Table 4.9: Maximum Drawdown on training data.

λ	Quantiles				
	1%	25%	50%	75%	99%
reg	1.558	3.694	5.311	7.528	13.709
0.0	2.102	3.818	5.067	6.731	12.082
0.5	1.746	3.107	4.199	5.613	11.405
0.75	1.873	3.502	4.667	6.174	10.717
1.0	1.671	3.099	4.013	5.382	10.158
1.5	1.570	2.916	3.760	5.005	9.661

Table 4.10: Maximum Drawdown on testing data.

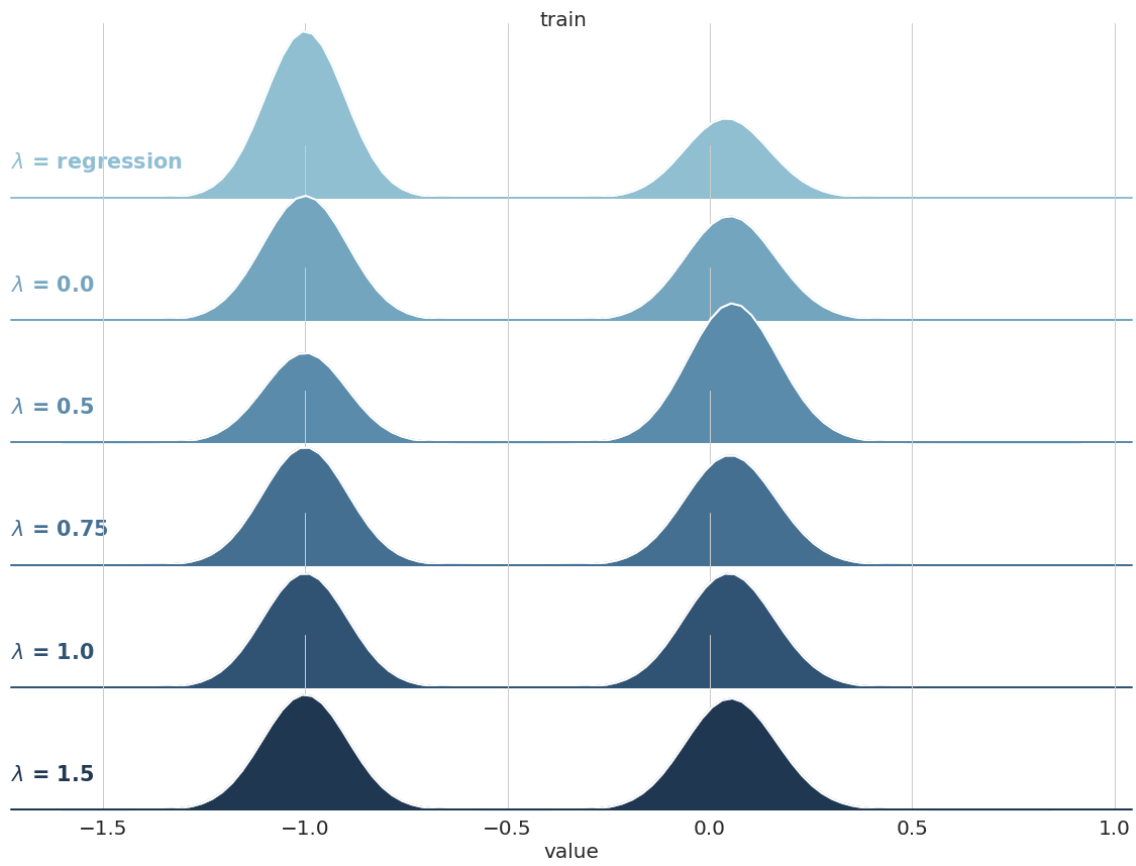
λ	Quantiles				
	1%	25%	50%	75%	99%
reg	-1.000	-1.000	-1.000	0.016	0.127
0.0	-1.000	-1.000	-1.000	0.040	0.155
0.5	-1.000	-1.000	0.019	0.064	0.177
0.75	-1.000	-1.000	-1.000	0.045	0.154
1.0	-1.000	-1.000	0.002	0.042	0.150
1.5	-1.000	-1.000	0.001	0.047	0.170

Table 4.11: Acceptability Index on training data.

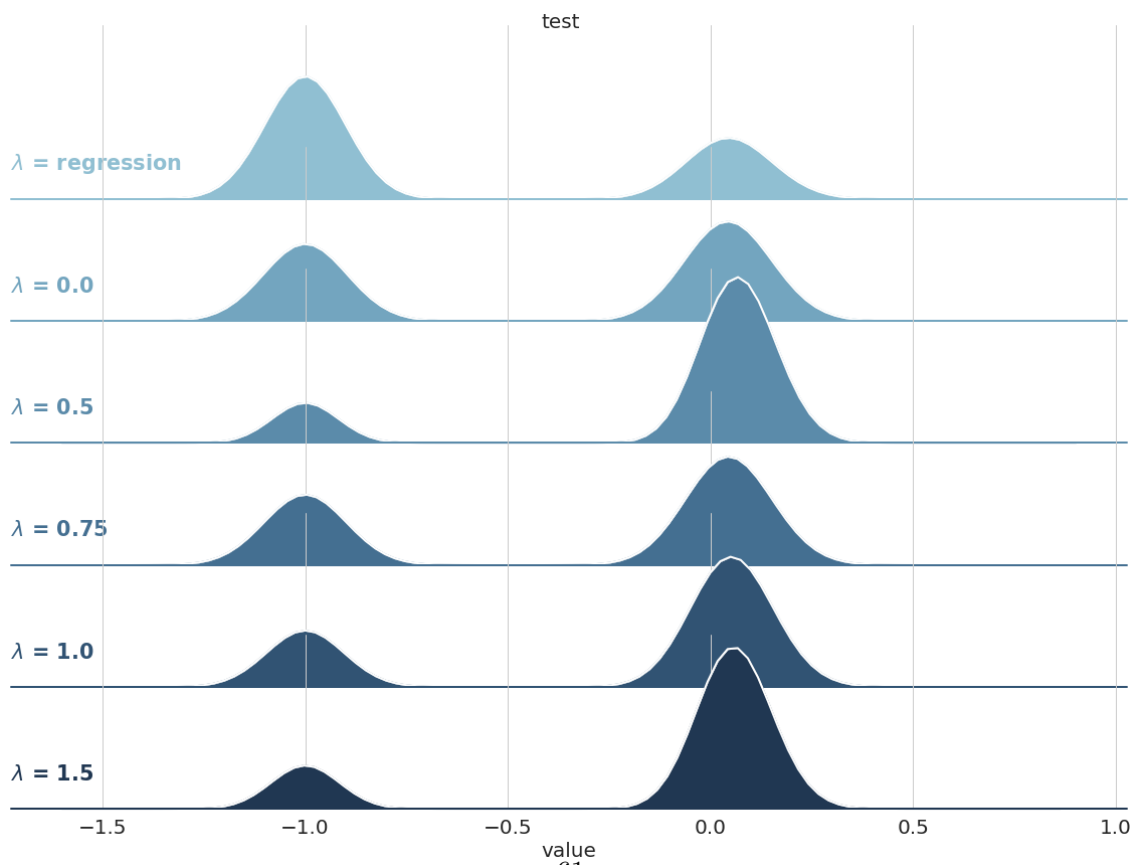
λ	Quantiles				
	1%	25%	50%	75%	99%
reg	-1.000	-1.000	-1.000	0.019	0.137
0.0	-1.000	-1.000	0.009	0.044	0.132
0.5	-1.000	0.013	0.050	0.091	0.188
0.75	-1.000	-1.000	0.015	0.049	0.133
1.0	-1.000	-1.000	0.029	0.064	0.156
1.5	-1.000	0.009	0.042	0.076	0.181

Table 4.12: Acceptability Index on testing data.

We also display the facet grids for different metrics on training and testing data with comparisons across distortion parameters. The shift rightward is visible in few of the metrics and displays a gain in performance by minimizing the distorted mean squared error vs just the mean squared error.

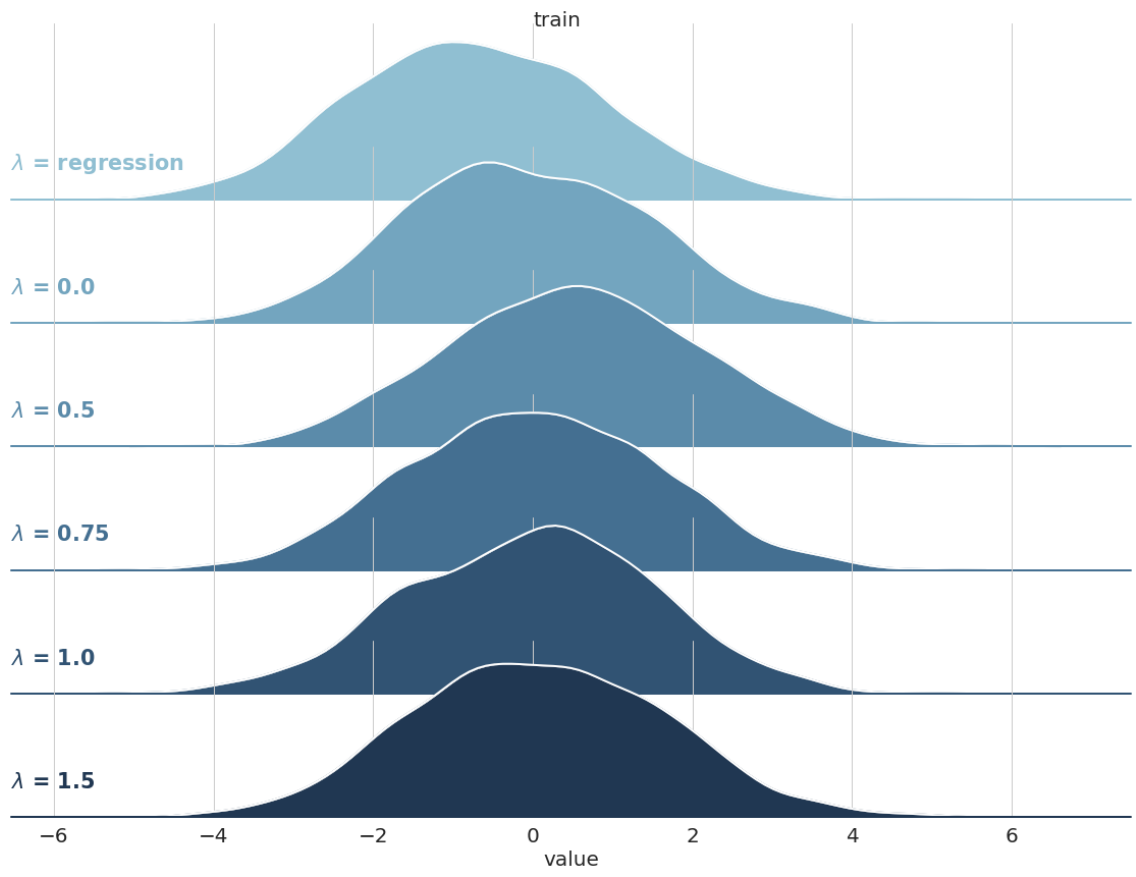


(a) Training data

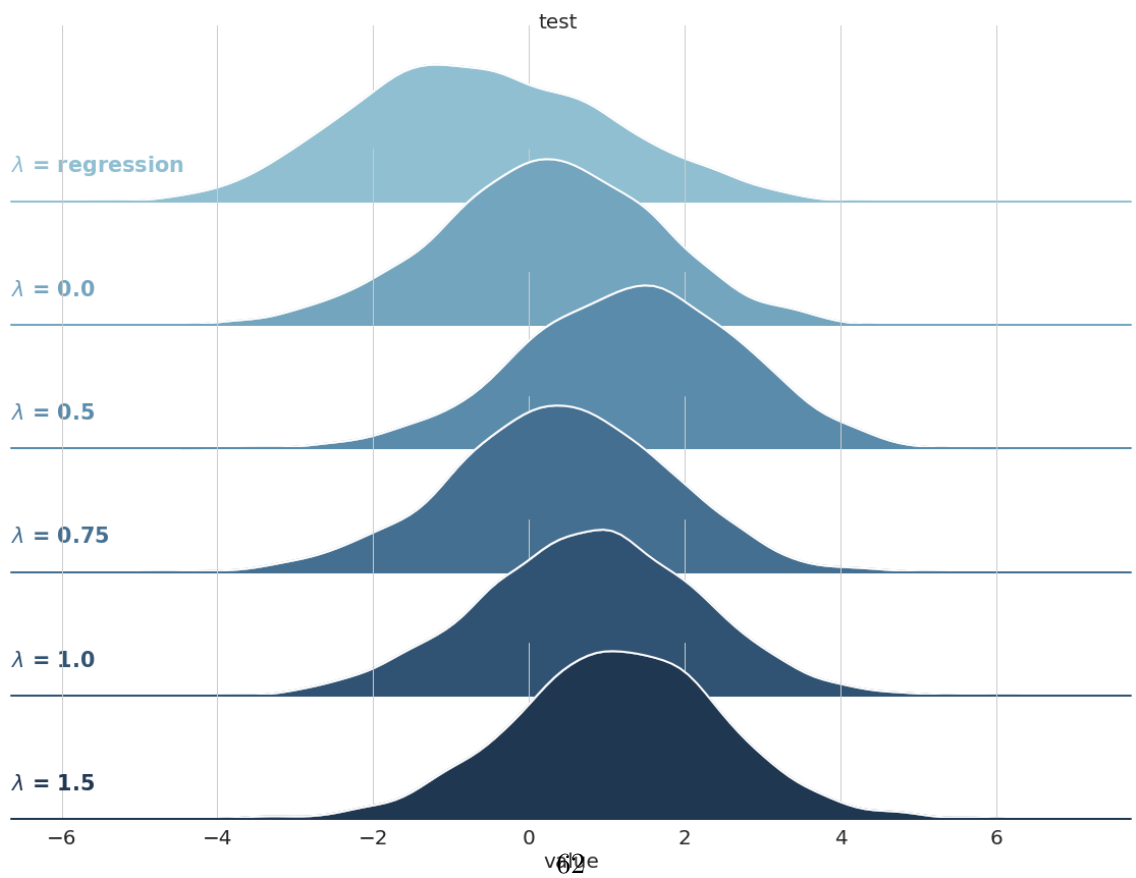


(b) Testing data

Figure 4.2: Acceptability indices comparison

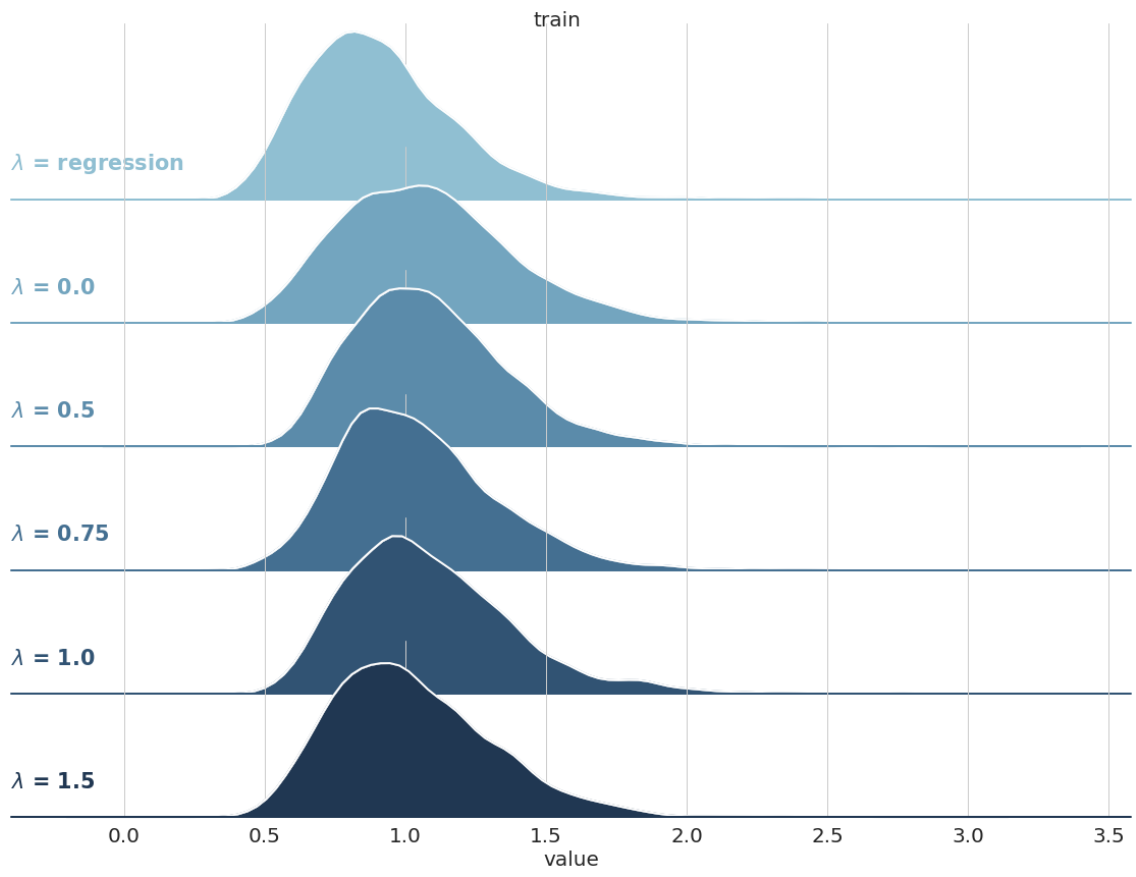


(a) Training data

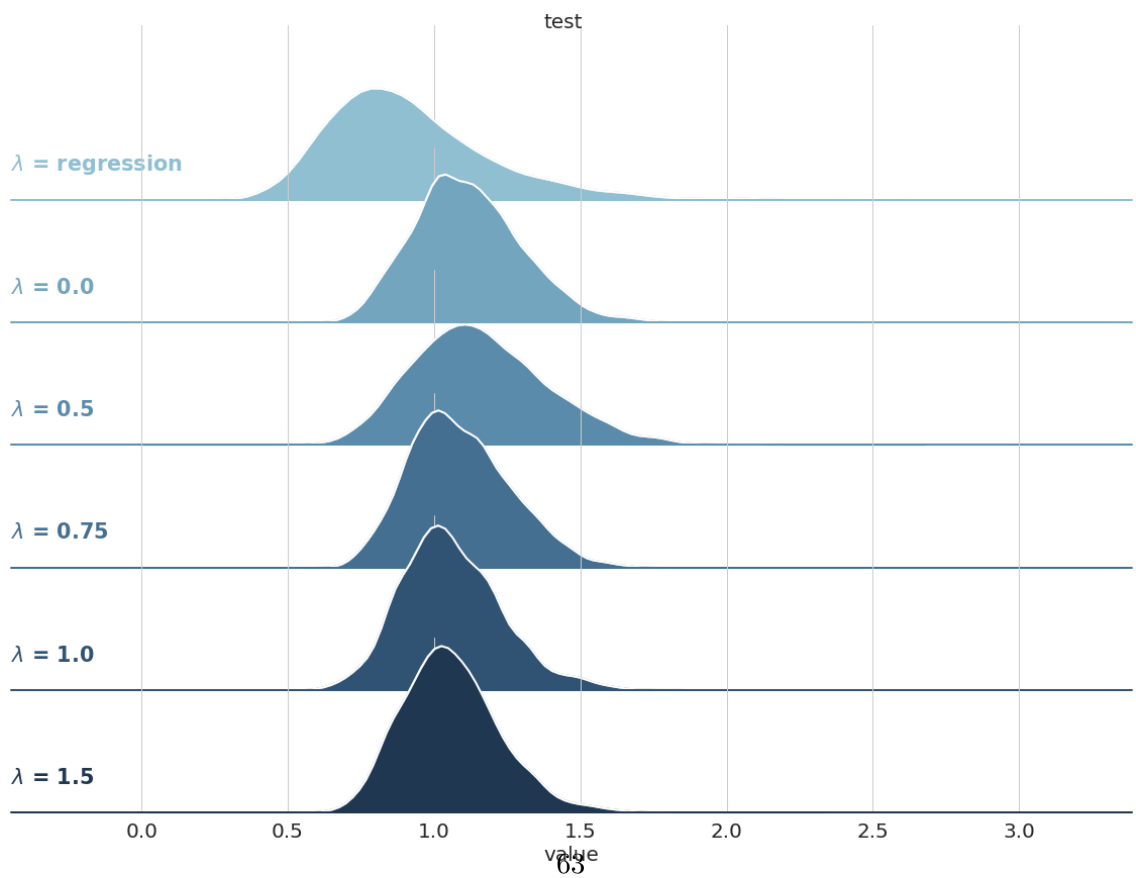


(b) Testing data

Figure 4.3: Sharpe Ratio comparison

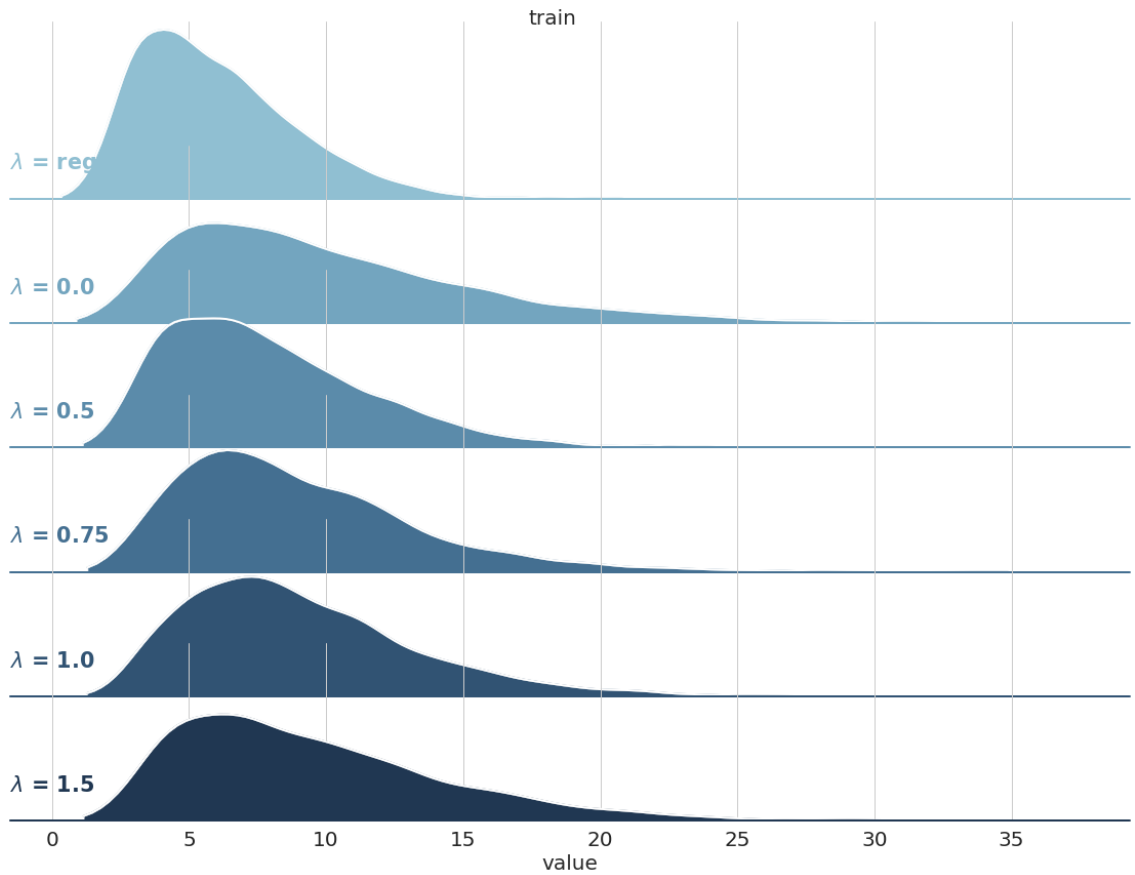


(a) Training data

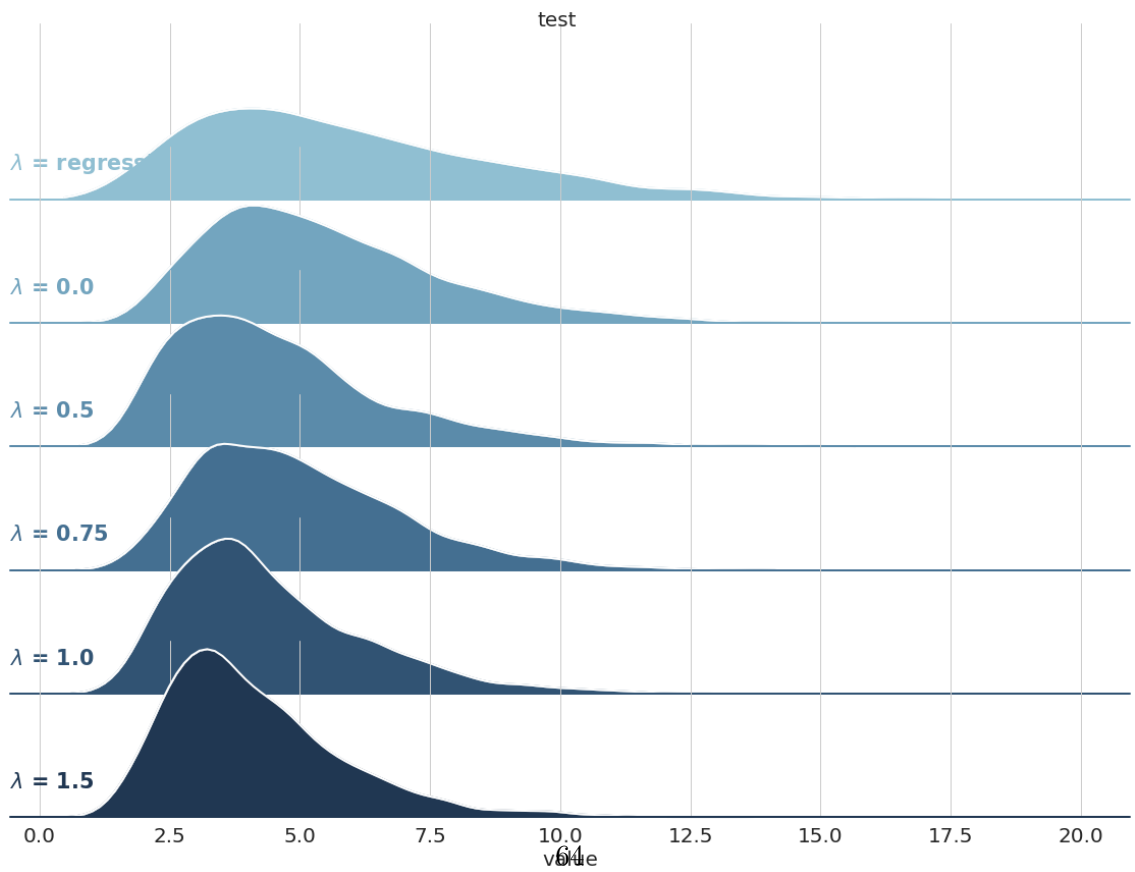


(b) Testing data

Figure 4.4: Gain-Loss ratio comparison

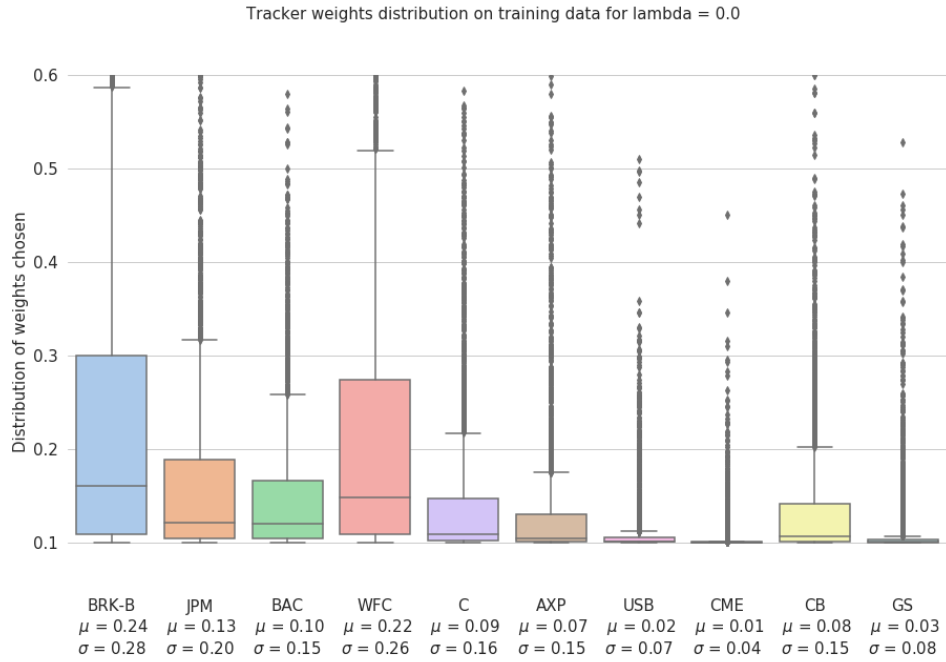


(a) Training data

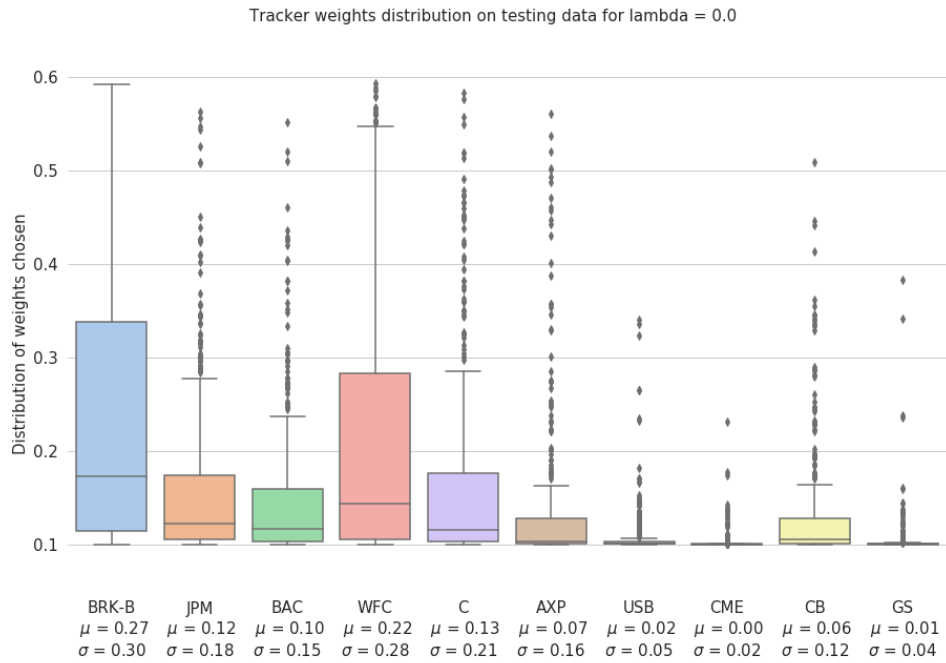


(b) Testing data

Figure 4.5: Maximum Drawdown comparison

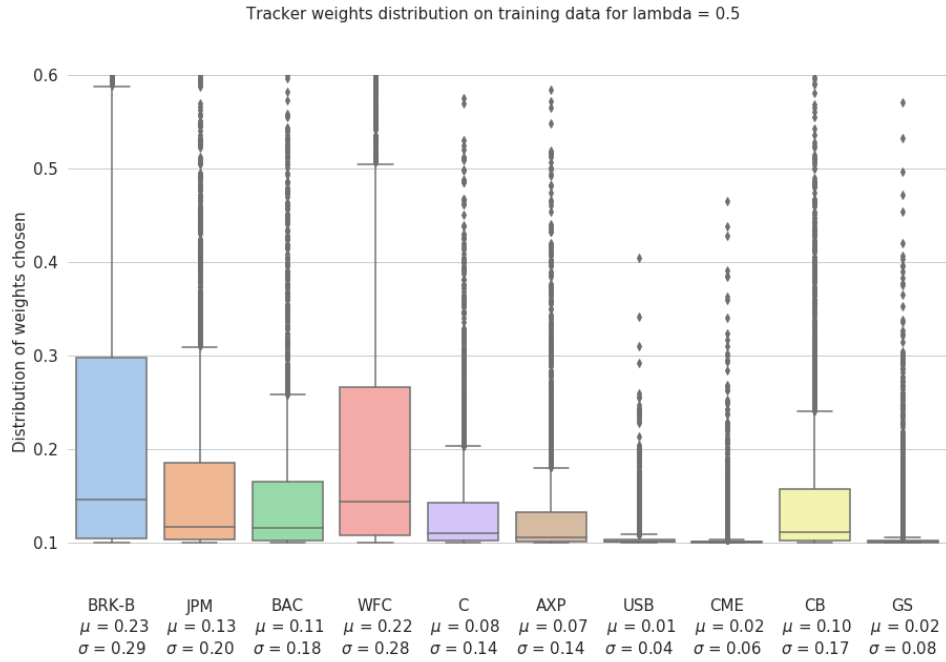


(a) Training data

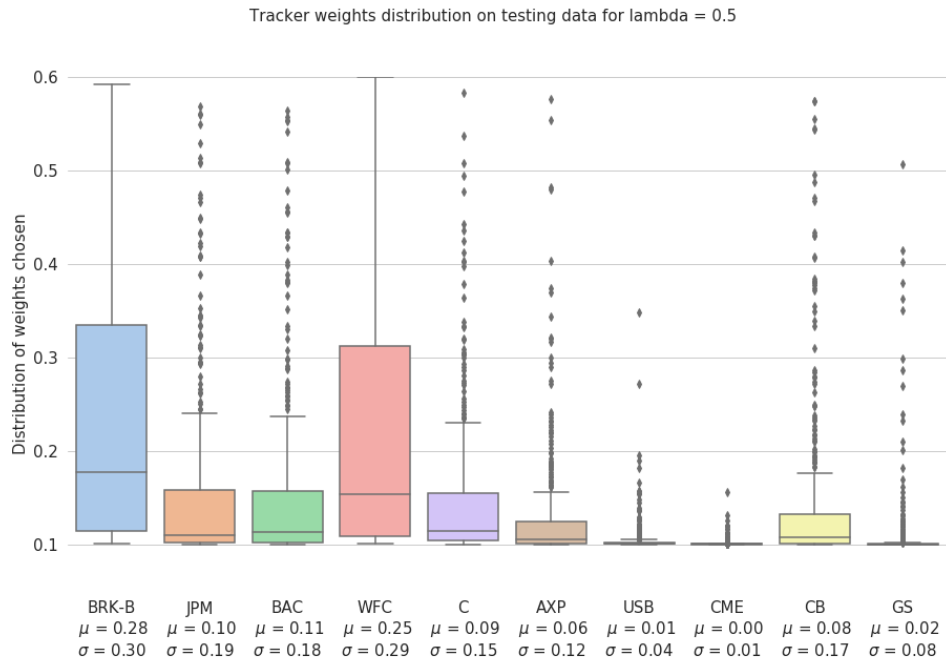


(b) Testing data

Figure 4.6: Weight distribution comparison on training and testing dataset for $\lambda = 0.0$

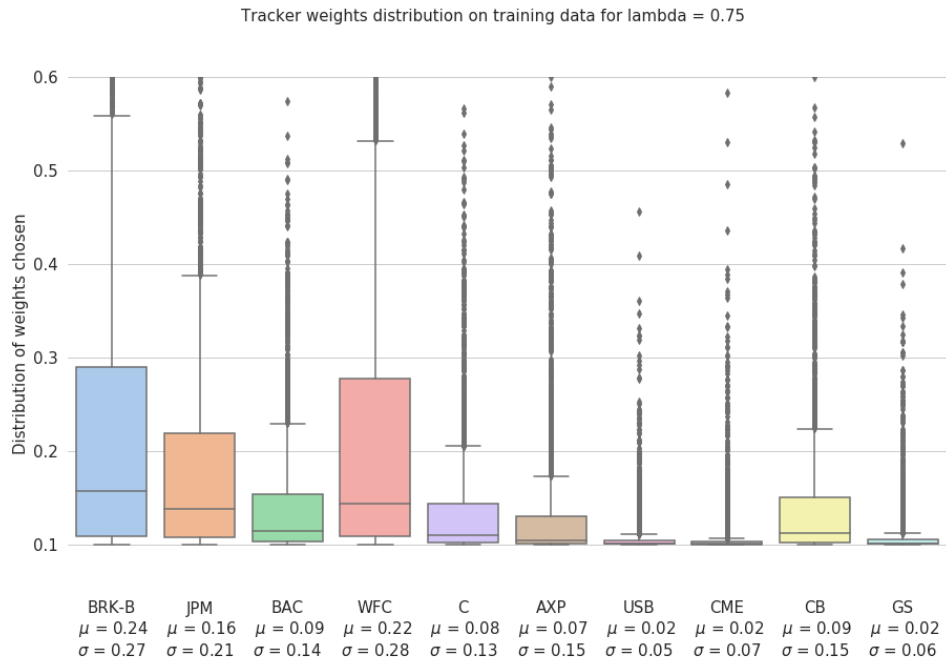


(a) Training data



(b) Testing data

Figure 4.7: Weight distribution comparison on training and testing dataset for $\lambda = 0.5$

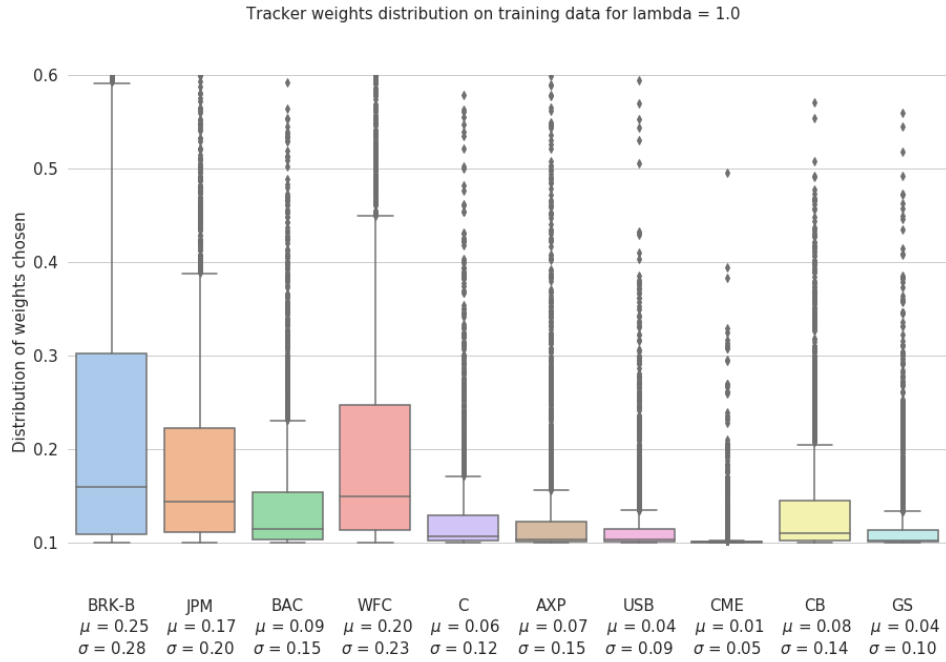


(a) Training data

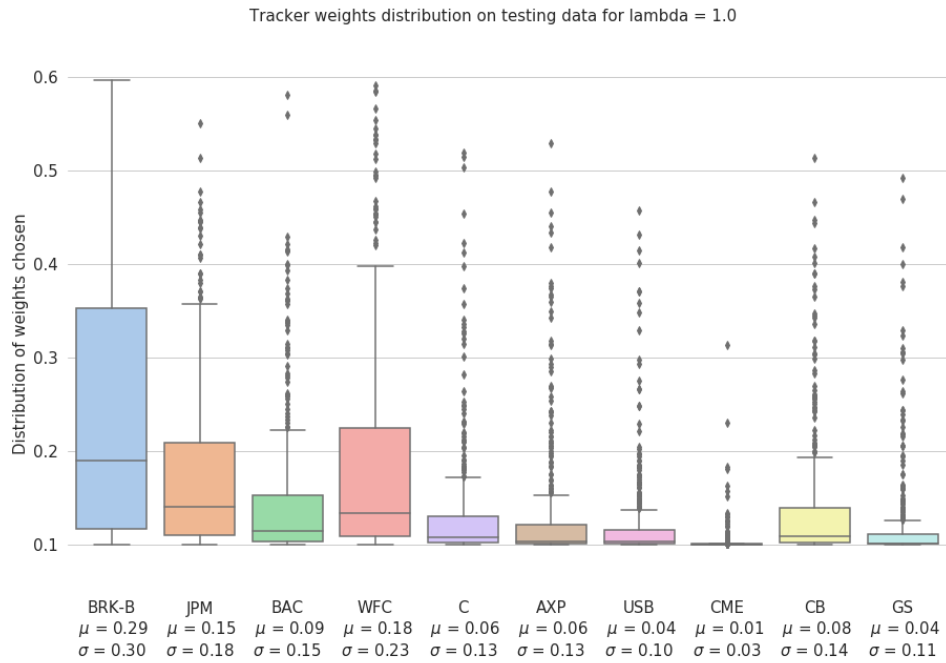


(b) Testing data

Figure 4.8: Weight distribution comparison on training and testing dataset for $\lambda = 0.75$

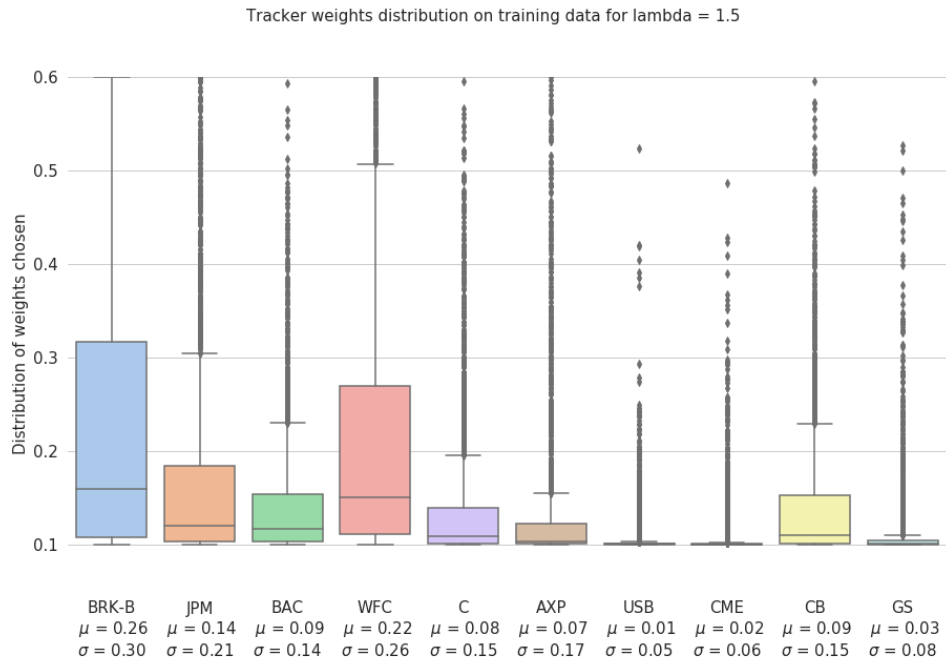


(a) Training data

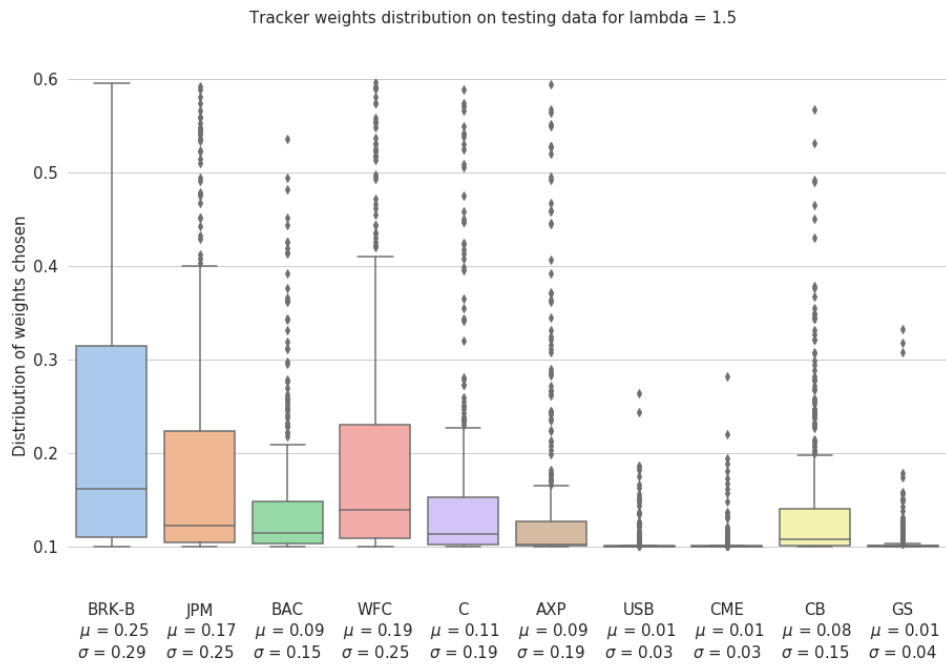


(b) Testing data

Figure 4.9: Weight distribution comparison on training and testing dataset for $\lambda = 1.0$



(a) Training data



(b) Testing data

Figure 4.10: Weight distribution comparison on training and testing dataset for $\lambda = 1.5$

Chapter 5

Conclusion and further work

The previous chapters illustrate that some methods in the deep reinforcement learning universe can be adapted to risk sensitive objective functions in conic finance. Additionally, when measured through various metrics of trading performance, it is also shown that the agent performs in a more risk averse manner as desired without sacrificing on the other metrics of performance.

The scope of the work presented here, however, is limited and is only just scratching the surface of the myriad applications of risk averse decision making in finance using reinforcement learning. With tools and theory available, some of the following projects are direct extensions of work already undertaken. One approach is to get the parameters of the underlying stochastic process of the data, and then use model based RL methods to train policies. Doing so makes use of a deep library of analytical methods to frame problems with theoretical constraints. The robustness of such methods is also proven to be sturdier. Another way to add robustness to the training under quantile regression methods would be use some variant of the quantile network architecture introduced by Dabney et al.[\[14\]](#). The advantage there is that instead of projecting distribution on a grid,

the neural net approximators can be trained to output values, as a continuous mapping, for any given quantile. Lastly, one could explore the risk sensitive universe by considering measure distortions, as introduced by Madan[3] instead of probability distortions. With theoretical background set up to distort arrival rates, and proven to be more robust distortions, one could work in a model based setting and expose the agent to measure distorted transition kernel. A heuristic approach could already be taken in a model free environment by distorting the buffer priorities in a prioritized replay buffer discussed by Schaul et al.[19]. The idea is that the training data is sampled with a probability given by a priority function. This priority function, a proxy for importance, can be distorted as discussed.

Bibliography

- [1] Madan, D., Schoutens, W. “Applied Conic Finance,” Cambridge Press, 2016.
- [2] Artzner, P., Delbaen, F., Eber, J, Heath, D. “Coherent Measures of Risk,” Mathematical Finance, Volume9, Issue3, Pages 203-228, 1999
- [3] Madan, D., Schoutens, W. “Measure Distorted Valuation for Financial Decision Making,” *preprint*
- [4] Kusuoka, S., “On law invariant coherent risk measures,” Advances in Mathematical Economics, vol 3. Springer, Tokyo, 2001.
- [5] Denuit, M., Dhaene, J., Goovaerts, M., Kaas, R. “Actuarial Theory for Dependent Risks: Measures, Orders and Models,” John Wiley & Sons, 2005.
- [6] Sutton, R. S., and Barto, A.G. “Reinforcement Learning: an Introduction,” The MIT Press, 1998.
- [7] Sutton, R. S. “Learning to predict by the methods of temporal differences,” Machine Learning, 3(1):944, 1988.
- [8] Puterman, M.L. “Markov Decision Processes: Discrete Stochastic Dynamic Programming,” Wiley, 2005.

- [9] Chang, H.S., Hu, J., Fu, M.C., Marcus, S.I. “Simulation-Based Algorithms for Markov Decision Processes,” Springer, London, 2013.
- [10] Bickel, P.J., Freedman, D.A. “Some asymptotic theory for the bootstrap,” *The Annals of Statistics*, pp.1196-1217, 1981.
- [11] Bellamare, M.G., Dabney, W., Munos, R. “A Distributional Perspective on Reinforcement Learning,” *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.
- [12] Bellamare, M.G., Dabney, W., Rowland, M., Munos, R. “Distributional Reinforcement Learning with Quantile Regression,” *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.
- [13] Koenker, R. “Quantile Regression,” Cambridge University Press, 2005
- [14] Dabney, W., Ostrovski, G., Silver, D., Munos, R. “Implicit Quantile Networks for Distributional Reinforcement Learning,” *arXiv:1806.06923*, 2018.
- [15] Silver, D., Lever, G., Hess, N., Degris, T., Wierstra, D., Riedmiller, M. “Deterministic policy gradient algorithms,” *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32, Pages I-387I-395*, 2014.
- [16] Lillicrap, T.P, Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D. “Continuous control with deep reinforcement learning,” *arXiv:1509.02971*, 2015.
- [17] Barth-Maron, G., Hoffman, M.W., Budden, D., Dabney, W., Horgan, D., TB, D., Muldal, A., Heess, N., Lillicrap, T. “Distributed Distributional Deterministic Policy Gradients,” *arXiv:1804.08617*, 2018.

- [18] Borovykh, A., Bohte, S., Oosterlee, C.W. “Conditional Time Series Forecasting with Convolutional Neural Networks,” arXiv:1703.04691, 2018.
- [19] Schaul, T., Quan, J., Antonoglou, I., Silver, D. “Prioritized Experience Replay,” arXiv:1511.05952, 2015.
- [20] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M “Playing atari with deep reinforcement learning,” arXiv:1312.5602, 2013.
- [21] Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., Silver, D. “Rainbow: Combining Improvements in Deep Reinforcement Learning,” arXiv:1710.02298, 2017.
- [22] Kingma, D.P., Ba, J. “Adam: A Method for Stochastic Optimization,” arXiv:1412.6980v9 , 2014
- [23] Gatev, E., Goetzmann, W.N., Rouwenhorst, K.G. “Pairs Trading: Performance of a Relative Value Arbitrage Rule” *Review of Financial Studies*, Volume 19, Issue 3, Pages 797-827, Fall 2006.
- [24] Elliott, R.J., van der Hoek, J., Malcolm, W.P. “Pairs trading,” *Quantitative Finance*, Vol. 5, Issue No. 3, Pages 271-276, June 2005.
- [25] Artzner, P., Delbaen, F., Eber, J, Heath, D., Ku, H. “Coherent multiperiod risk adjusted values and Bellmans principle,” *Annals of Operations Research*, Vol 152, Pages 522, 2007.
- [26] Boda, K, Filar, J.A. “Time consistent dynamic risk measures,” *Mathematical Methods of Operations Research*, Vol 63, pages 169-186, 2006

- [27] Shapiro, A. "On a time consistency concept in risk averse multistage stochastic programming," *Operations Research Letters*, Vol 37, Pages 143-147, 2009
- [28] Roorda, B., Schumacher, J.M. "Weakly time consistent concave valuations and their dual representations," *Finance and Stochastics - Springer*, Vol. 20(1), Pages 123-151, 2016