Deep Recurrent Policy Search for Portfolio Optimization

Duc Thien Nguyen, Desmond Cai, Shiau Hong Lim, Laura Wynter *

Abstract

We present a model-based recurrent reinforcement learning algorithm for the financial portfolio optimization problem. Due to the analytical form of the expressions, the policy gradient can be directly estimated through a stochastic computation graph without relying on Monte Carlo rollout or a separate critic. In addition, the model possesses a certain permutation invariance that leads to a form of model universality, and allows a multi-task approach for the problem, with provable error bounds. We show that the coupling of the recurrent learning with multi-task training offers faster convergence and significantly higher returns than competing methods.

1 Introduction

While deep reinforcement learning (RL) has been successful in solving many challenging sequential decision-making tasks, its successes have been mostly in domains where one has access to huge amount of training data, often through a simulator. In domains with limited real data and no access to a simulator, sequential decision making in large, continuous state-action space remains a challenging problem.

We address the problem of financial portfolio optimization, where a policy adaptively allocates a given fund into a number of assets in order to maximize the total expected return. Traditionally, learning approaches to this task have been model-based, building a predictive model for future asset prices and deriving the optimal allocation using the model predictions. Such approaches are sensitive to the quality of the chosen models, often too simplistic for the real world.

Recently, model-free deep RL has been successfully applied to this problem [Jiang *et al.*, 2017; Ye *et al.*, 2020], where one learns a policy that prescribes the next allocation given the current state. The state contains observable features, such as the price history of each asset as well as other information that may be relevant. This is therefore a task with a potentially high-dimensional continuous state-action space.

Deep RL policies, however, typically require a large amount of data to train. Most approaches that deal with continuous action spaces are based on estimating a policy gradient. Policy gradient methods either use Monte Carlo sampling (e.g. REINFORCE [Williams, 1992] or rely on learning a separate critic (e.g. A3C [Mnih *et al.*, 2016], DDPG [Lillicrap *et al.*, 2016]). The former typically has low sample-efficiency and high variance in estimating the gradient while the latter may suffer from bias in the critic.

Our approach avoids these problems by exploiting two properties that can be found in a number of sequential decision-making problems, and particularly in portfolio optimization. The first is the fact that in many practical scenarios, the actions do not impact the external state, in this case given by the price movements of the assets. In particular, the state consists of two parts:

- an internal part that stores the current asset allocation, whose transition model is known;
- an external part that contains all other market information such as the prices, whose transition model is unknown but is independent of the policy.

The reward model, taking into account the transaction costs, is also known. We show that under this setting, one can estimate the policy gradient directly without the need for Monte Carlo rollout or a separate critic, through the use of stochastic computation graphs.

Secondly, the resource allocation problem can be shown to have a certain form of permutation invariance. This property implies that one may substitute different sets of samples as input to the learning process. As instantiated on the portfolio optimization problem, it means that training a strategy on one portfolio universe can be done using samples from a different portfolio universe. We prove a bound on the error induced by this type of multi-task setting. This has a substantial benefit to the portfolio optimization problem: the number of samples available for training a deep network is increased dramatically; as such, our method is much more efficient in making use of real market data. We demonstrate empirically the considerable benefit accrued through leveraging this property.

Additionally, the property can be taken one step further to arrive at a universality feature of the policy. Specifically, leveraging both the permutation invariance and the quasi-independence across the instruments allows applying the trained policy to a portfolio universe of any size, provided that an order-preserving transformation is used.

*The authors are with IBM Research, Singapore. Emails: {Duc.Thien.Nguyen@, desmond.cai1@, shonglim@sg., lwynter@sg.}ibm.com

2 Background and related work

Our approach incorporates a partial, known model into the learning process. This is different from most existing model-based approaches – which focus on first learning an approximate transition and/or reward model, then incorporating the model into the RL process through planning or sampling [Sutton, 1991; Chua *et al.*, 2018; Hafner *et al.*, 2019]. In certain cases, the forms of the learned models (e.g. gaussian processes) are chosen such that they are amenable to computing analytical gradients [Deisenroth and Rasmussen, 2011]. In the special case of the LQR framework, a close-form optimal policy can be derived analytically.

Closest to ours is the work by [Bueno *et al.*, 2019], where stochastic computation graphs [Schulman *et al.*, 2015] are used to compute the policy gradients directly by making use of the re-parameterization trick. They empirically demonstrated their approach on a number of stochastic control tasks where the exact transition and cost models are available.

Our work is also related to the literature on multi-task learning [Sharma *et al.*, 2018; D'Eramo *et al.*, 2020]. However, existing works on multi-task learning are primarily concerned with learning of shared representations, regardless of the the similarity or dissimilarity between tasks. In our approach, we leverage multi-task learning to increase the number of available training samples and consequently improve learning performance, and we provide theoretical results to motivate our technique.

3 Multi-Task Learning

We take a multi-task learning approach to the financial portfolio optimization problem where each task corresponds to trading a particular set of assets, drawn from a large universe of assets. Our approach is based on the observation that even though the exact transition dynamics in each task is not identical, they share enough similarity such that individual assets are almost interchangeable and that one can learn a single strategy that works well in any of the tasks, making use of the combined data from all the tasks. We first provide some theoretical results that motivate this approach.

Our main result is an extension of existing results from [Lazaric *et al.*, 2012], where a finite-sample error bound is derived for the LSPI algorithm on a single task. In particular, they derive a high-probability bound on the performance difference between the final learned policy and the optimal policy. The bound is in the form $(c_1 + c_2\frac{1}{\sqrt{N}})$ where c_1 and c_2 are constants that depend on the task and the chosen feature space while N is the number of training examples. We show that as long as the tasks are ϵ-close to each other (with respect to some similarity measure), the error bound can be in the form $(c_1 + c_2\frac{1}{\sqrt{TN}} + c_3\epsilon)$, where T is the number of tasks, N the number of examples per task, and c_3 a task-dependent constant. As long as ϵ is small, we can therefore benefit from the much larger set of TN training examples.

For a measurable space with domain $\mathcal{X}$, let $\mathcal{S}(\mathcal{X})$ denote the set of probability measures over $\mathcal{X}$, and $\mathcal{B}(\mathcal{X}; L)$ denote the space of bounded measurable functions with domain $\mathcal{X}$ and bound $0 < L < \infty$. For a measure $\rho \in \mathcal{S}(\mathcal{X})$ and a measurable function $f : \mathcal{X} \to \mathbb{R}$, we define the $l_2(\rho)$-norm of f, $\|f\|_\rho$, and for a set of N points $X_1, \cdots, X_N \in \mathcal{X}$, we define the empirical norm, $\|f\|_N$ as

$$\|f\|_\rho^2 = \int f(x)^2 \rho(dx) \quad \text{and} \quad \|f\|_N^2 = \frac{1}{N}\sum_{n=1}^{N} f(X_t)^2.$$

We also define $\|f\|_\infty = \sup_{x\in\mathcal{X}} |f(x)|$ to be the supremum norm of f.

We consider a set of MDPs indexed by t. Each MDP is denoted by a tuple $\mathcal{M}_t = \langle \mathcal{X}, \mathcal{A}, R_t, P_t, \gamma \rangle$, where $\mathcal{X}$ is a common state space and is a bounded closed subset of the s-dimensional Euclidean space, $\mathcal{A}$ is a common finite action space, $R_t : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is a task specific reward function that is uniformly bounded by R_{max}, P_t is a task specific transition kernel such that $P_t(\cdot|x, a)$ is a distribution over $\mathcal{X}$ for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$, and $\gamma \in (0, 1)$ is a common discount factor. We consider deterministic policies denoted by $\pi : \mathcal{X} \to \mathcal{A}$. For a given policy π, the MDP $\mathcal{M}_t$ is reduced to a Markov chain $\mathcal{M}_t^\pi = \langle \mathcal{X}, R_t^\pi, P_t^\pi, \gamma \rangle$ with reward function $R_t^\pi(x) = R_t(x, \pi(x))$, transition kernel $P_t^\pi(\cdot|x) = P_t(\cdot|x, \pi(x))$, and stationary distribution ρ_t^π. The value function V_t^π is defined as the unique fixed-point of the Bellman operator $\mathcal{T}_t^\pi : \mathcal{B}(\mathcal{X}; V_{\text{max}} = R_{\text{max}}/(1-\gamma)) \to \mathcal{B}(\mathcal{X}; V_{\text{max}})$, which is defined by $(\mathcal{T}_t^\pi V)(x) = R_t^\pi(x) + \gamma \int_{\mathcal{X}} P_t^\pi(dy|x)V(y)$.

To approximate the value function V, we use a linear approximation architecture[1] with parameters $\alpha \in \mathbb{R}^d$ and basis functions $\varphi_i \in \mathcal{B}(\mathcal{X}; L)$ for $i = 1, \cdots, d$. We denote by $\varphi(\cdot) = (\varphi_1(\cdot), \cdots, \varphi_d(\cdot))^\mathsf{T} \in \mathbb{R}^d$ the feature vector and by $\mathcal{F}$ the linear function space spanned by the basis functions φ_i. Thus, $\mathcal{F} = \{f_\alpha \mid \alpha \in \mathbb{R}^d \text{ and } f_\alpha(\cdot) = \varphi(\cdot)^\mathsf{T}\alpha\}$.

In addition to the assumptions proposed in [Lazaric *et al.*, 2012], we extend the definition of second-order discounted-average concentrability, a smoothness property of the transition kernel originally proposed in [Antos *et al.*, 2008], and define the notion of first-order discounted-average concentrability. The latter will be used in our main result.

Assumption 1. *Given the target distribution $\sigma \in \mathcal{S}(\mathcal{X})$ and an arbitrary sequence of policies $\{\pi_m\}_{m\geq 1}$, let*

$$c_{\sigma,\mu} = \sup_{\pi_1,\ldots,\pi_m} \left\| \frac{d(\mu P^{\pi_1} \ldots P^{\pi_m})}{d\sigma} \right\|.$$

We define the first and second order discounted-average concentrability of future-state distributions as

$$C'_{\sigma,\mu} = (1-\gamma)\sum_{m\geq 0} \gamma^m c_{\sigma,\mu}(m),$$

$$C''_{\sigma,\mu} = (1-\gamma)^2 \sum_{m\geq 1} m\gamma^{m-1} c_{\sigma,\mu}(m),$$

and we assume that $C'_{\sigma,\mu}, C''_{\sigma,\mu} < \infty$.

Theorem 1. *Let $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, R, P, \gamma \rangle$ be an MDP with reward function R and transition kernel P. Denote its Bellman operator by*

$$(\mathcal{T}^\pi V)(x) = R^\pi(x) + \gamma \int_{\mathcal{X}} P^\pi(dy|x)V(y).$$

[1] Our results assume linear architecture, finite action space, and use LSPI for the purpose of deriving analytical results, our actual implementation is via non-linear architectures, continuous action space and a recurrent learning algorithm.

Given a policy π, define the Bellman difference operator between $\mathcal{M}_t$ and $\mathcal{M}$ to be $\mathcal{D}_t^\pi V = \mathcal{T}_t^\pi V - \mathcal{T}^\pi V$. Apply least squares policy iteration (LSPI) to $\mathcal{M}$, by generating, at each iteration k, a path from $\mathcal{M}$ of size n, where n satisfies Lemma 4 in [Antos et al.*, 2008]. Let $V_{-1} \in \tilde{\mathcal{F}}$ be an arbitrary initial value function, $V_0, \cdots, V_{K-1}$ ($\tilde{V}_0, \cdots, \tilde{V}_{K-1}$) be the sequence of value functions (truncated value functions) generated by LSPI after K iterations, and π_k be the greedy policy w.r.t. the truncated value function $\tilde{V}_{k-1}$. Suppose also that*

$$\|\mathcal{D}_t^\pi V^\pi\|_\mu \leq \epsilon \; \forall \; \pi, \quad \textit{and} \quad \|\mathcal{D}_t^{\pi_k} \tilde{V}_{k-1}\|_\mu \leq \epsilon \; \forall \; k.$$

Then, with probability $1-\delta$ (with respect to the random samples), we have that $\|V_t^{\pi_t^*} - V_t^{\pi_K}\|_\sigma \leq \tilde{O}(\sqrt{C''_{\sigma,\mu}} + \frac{1}{\sqrt{N}} + \epsilon\sqrt{C'_{\sigma,\mu}})$.

4 Portfolio Optimization

Consider a universe of assets $\mathcal{I}$. Let $s_{i,n}$ denote the state of asset i at time n. The state may include multiple price components, e.g. open, high, low, close, over time period n. Let $p_i(s_{i,1})$ denote the initial state distribution of asset i and assume that $s_{i,n}$ evolves according to transition probability kernel $p_i(s_{i,n+1}|s_{i,n})$. Furthermore, assume that we are given a dataset of $|\mathcal{I}|$ state trajectories, one per asset, where each trajectory $\tau_i = \{s_{i,1}, \ldots, s_{i,N}\}$ for asset i has length N.

Our goal is to solve a portfolio optimization problem over a subset of assets $\mathcal{I}_t \subseteq \mathcal{I}$. Consider t as an arbitrary indexing variable, each t will correspond to a task. We model our problem as an MDP $\mathcal{M}_t = \langle \mathcal{X}, \mathcal{A}, R_t, P_t, \gamma \rangle$. At each time step n, one observes the state $x_n = ((s_{i,n})_{i\in\mathcal{I}_t}, (w_{i,n-1})_{i\in\mathcal{I}_t})$, and takes an action $w_n = (w_{i,n})_{i\in\mathcal{I}_t}$, where $w_{i,n}$ is the share of the portfolio allocated to asset i. As such, the actions must satisfy constraints: $0 \leq w_{i,n} \leq 1$ and $\sum_{i\in\mathcal{I}_t} w_{i,n} = 1$. The probability transition kernel is a function of the asset price transition kernels. Given the data trajectories $(\tau_i)_{i\in\mathcal{I}_t}$, our objective is to maximize:

$$J_t(\mu) = \sum_{n=1}^{N} \gamma^{n-1} R_t(x_n, \mu(x_n)).$$

In settings for which N is small, it is difficult to learn a policy that generalizes well.

In practice, the number of assets $|\mathcal{I}|$ with data available is likely to be significantly greater than the number of assets $|\mathcal{I}_t|$ which the optimization is concerned with. We leverage Theorem 1 to propose an algorithm for exploiting the data available for assets $\mathcal{I} \setminus \mathcal{I}_t$. In particular, we consider a set of portfolio optimization problems, indexed by $t = 1, \ldots, T$, where each problem optimizes over a different set of assets $\mathcal{I}_t \subseteq \mathcal{I}$. We model each portfolio optimization problem t as an MDP $\mathcal{M}_t$. Instead of solving each MDP $\mathcal{M}_t$ independently, we propose to solve them jointly, by sampling subsequences of trajectories from all the MDPs. The following corollary is a consequence of Theorem 1.

Corollary 1. *Let $[T]$ be a set of similar tasks such that distance from the average MDP, given by*

$$(\mathcal{T}^\pi V)(x) = \frac{1}{T}\sum_{t=1}^{T} R_t^\pi(x) + \gamma \int_{\mathcal{X}} \frac{1}{T}\sum_{t=1}^{T} P_t^\pi(dy|x) V(y),$$

is bounded by ϵ as defined in Theorem 1. Let N be the number of samples available in each task. Then, the suboptimality of the policy for each task obtained by solving the average MDP, using NT samples from all tasks, is $\tilde{O}(1/\sqrt{NT}) + \tilde{O}(\epsilon) + C$.

5 Recurrent Portfolio Optimization

A portfolio is specified by the vector $\mathbf{w}_n = \langle w_n^i \rangle_{i=1:m}$ where n is the time period. The portfolio vector is defined in the manifold $\mathbf{S} = \{\mathbf{w} = \langle w^i \rangle : w^i \geq 0 \; \forall i, \sum_i w^i = 1\}$. Each portfolio is characterized by a set of weights $\{w_n^i\}$ representing the portion of the investment in asset i at time n. We denote $\mathbf{y}_n$ to be m-dimensional *relative price vector* whose component y_n^i is the ratio of the price at trading period n to that of the previous trading period $n-1$ for the asset i.

During time period n, a portfolio strategy can trade (buy or sell) assets to reallocate the portfolio weights from $\mathbf{w}_n$ to $\mathbf{w}'_n \in \mathbf{S}$. We consider a transaction cost c. Denote $\mathbf{c}$ the vector of transaction costs for trading each asset. We define the first component $\mathbf{c}(0) = 0$ as a no-cost option corresponding to cash and $\mathbf{c}(i) = c, \forall i > 0$. Because of transaction costs, the portfolio is reduced by a factor of $(1 - f(\mathbf{w}_n, \mathbf{w}'_n))$.

We consider the reward for reallocation $\mathbf{w}'_n$ to be defined as the logarithmic rate of return $r_n = \log(\mathbf{w}'_n \cdot \mathbf{y}_{n+1})(1 - f(\mathbf{w}_n, \mathbf{w}'_n))$. An important feature of our recurrent learning approach is that one can compute $f(\mathbf{w}_n, \mathbf{w}'_n)$ in closed form.

Our objective is to optimize a trading policy π to reallocate the portfolio weights $\mathbf{w}_n$ at every time step t to maximize a global utility function defined over the planning period. In summary, the Markov Decision Process is given by:

- The state s_n at time n comprising portfolio weights w_n.
- The trading policy π as a function of historical prices $\langle \mathbf{y}_{n-k} \rangle_{k=0:K}$ and current weights $\mathbf{w}_n$. The policy outputs a reallocation of weights $\tilde{\mathbf{w}}_n$. We use Hadamard product $\odot$ and dot product $\cdot$ to incorporate the transaction cost $\mathbf{c}$ into the portfolio rebalancing. Firstly, we *liquidify* the portfolio weight $\mathbf{w}_n$ into a *value-in-cash* weight $\mathbf{w}_n^c = \mathbf{w}_n \odot (1 - \mathbf{c})$. Let $v_n^c = \sum_i \mathbf{w}^c(i)$ be the total portfolio *value-in-cash*. We define the buy and sell quantities by $\tilde{\mathbf{w}}_n$ as follows: $\mathbf{u}_n^+ = \max(\tilde{\mathbf{w}}_n v_n^c - \mathbf{w}_n^c, 0)$; $\mathbf{u}_n^- = \max(\mathbf{w}_n^c - \tilde{\mathbf{w}}_n v_n^c, 0)$. The post-trade weight can be computed as $\bar{\mathbf{w}}_n = (\mathbf{w}_n^c + (1 - \mathbf{c}) \odot \mathbf{u}_n^+ - \mathbf{u}_n^-) \odot (\frac{1}{(1-\mathbf{c})})$, $\mathbf{w}'_n = normalize(\bar{\mathbf{w}}_n)$, in which the $normalize(\bullet)$ operator divides all components of a vector by the sum of its components. The total transaction cost can be quantified by $f(\mathbf{w}_n, \mathbf{w}'_n) = \frac{\mathbf{c}}{1-\mathbf{c}} \cdot \mathbf{u}_n^+ + \frac{\mathbf{c}}{1-\mathbf{c}} \cdot \mathbf{u}_n^-$.
- The reward for reallocation $\mathbf{w}'_n$ is defined as the logarithmic rate of return $r_n = \log(\mathbf{w}'_n \cdot \mathbf{y}_{n+1})(1 - f(\mathbf{w}_n, \mathbf{w}'_n))$.
- Given the new price $\mathbf{y}_{n+1}$, the portfolio weight at the next iteration is $\mathbf{w}_{n+1} = normalize(\mathbf{w}'_n \odot \mathbf{y}_{n+1})$.

The key observation is that in this model, all expressions are in analytic form, and as such we can define a recurrent reinforcement learning approach to directly optimize π. The models for each instrument are independent with the exception that they form a probability simplex. We exploit this feature

to obtain a universality property. Through the use of an order-preserving transformation, we can train a trading strategy on T instruments and then apply the same trading strategy to any number T' instruments. A common order-preserving transformation to apply to the strategy for T' instruments is the softmax operator. We illustrate the performance on universally-trained instances in the next section.

6 Experiments

The experiments have been run on public data from 400 of the S&P500 stocks from 2005 to 2019. Data from 2005 to 2017 was used for training while 2018 and 2019 were reserved for testing. To test the transferability amongst different tasks, we conduct 2 sets of experiments. First, we consider the target universe consisting of 8 *HighTech* stocks from [Ye *et al.*, 2020], [Ding *et al.*, 2014], namely 'GOOG', 'NVDA', 'AMZN', 'AMD', 'QCOM', 'INTC', 'MSFT', 'AAPL'. Then, to illustrate the permutation invariance in practice, we augment the training data with portfolios formed by randomly selecting other stocks from the S&P500. Second, to show the transferability of policies trained on portfolios of different sizes, we use historical data from 10 random portfolio universes for training and evaluate on 10 different random universes on the test set. We use a replay buffer and a CNN similar to that of [Jiang *et al.*, 2017]. The neural network trading policy uses the latest 30-day historical prices. We consider the transaction cost to be 0.2%.

Figure 1 illustrates two aspects of the our recurrent learning method for portfolio optimization. On the left of Fig. 1, we show how convergence varies with the multi-task approach. In particular, training on a price history of only the assets in the portfolio itself fails to converge. On the other hand, leveraging the price histories of other portfolio universes, composed of different assets converges smoothly, achieving a substantial rate of return. On the right we show another property of our method: namely that the look-ahead period of the model is flexible. We illustrate 1-, 3-, and 5-step look-ahead, in this case each corresponding to a day. The 5-step look-ahead achieves better returns than the 3-step and significantly better than 1-step. Since the model form is analytic, this comes with very little overhead in model training.

Figure 2 compares the performance of our method (called RRL in the figure) with that of standard benchmarks on the same dataset. The benchmarks include OLMAR and PAMR, two moving-average-based methods, Constant Rebalancing Portfolio (CRP), which aims to keep a constant distribution of each asset, Universal Portfolio (UP) which is a theoretically-optimal form across all possible CRP strategies, and Anticorr, which seeks to exploit anti-correlations across assets. We see that our method outperforms the benchmarks on both testing years, 2018 and 2019.

Figure 3 illustrates the universality property. We show the result of running strategies trained on a stock universe of size T tested on portfolios of size $T' \neq T$, for varying values of T and T'. In particular, we illustrate a 1-asset, a 9-asset, a 99-asset universe and finally the full 400 assets. Training is thus performed in each of these settings. Then, we test each policy on a universe of a different size, subject to a softmax operation: T': 1, 9, 99 and 400. While the 1-asset universe (i.e. allocating to a single stock and cash) performs poorly, in all other cases, we see that the universality property does indeed hold. For example, running a trading strategy on a portfolio of 9 assets can actually be improved by training on 99 (thus different) assets, as can be seen in the sub-figure 3(b), and so on, for the other T, T'. This means that the data set size that leads to the best training can be leveraged in trading strategies for portfolios of most any size.

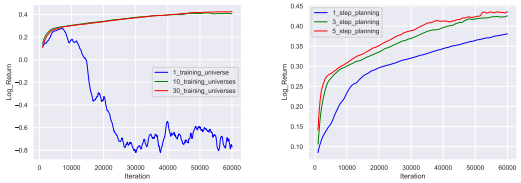


Figure 1: Comparison between different sizes of multi-task set used (left) and different numbers of training look-ahead steps (right).

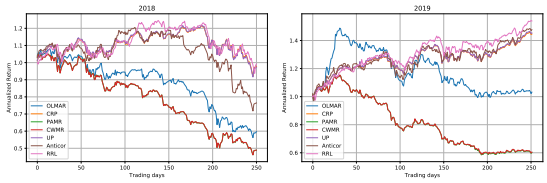
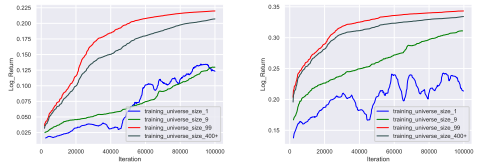
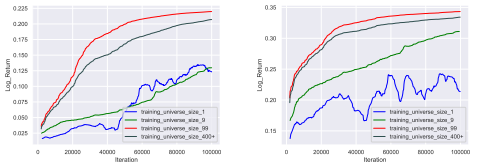


Figure 2: The recurrent learning policy compared to common baselines over two test periods, 2018 and 2019.



(a) Test on 1-asset (b) Test on 9-assets



(c) Test on 99-assets (d) Testing on 400-assets

Figure 3: Illustration of the universality property of the model: train on a portfolio universe of n assets, and test on different sizes.

7 Conclusion

We have presented a method based on a stochastic computation graph that uses recurrent reinforcement learning to derive financial portfolio trading strategies. The model has the particularity of admitting a form of permutation invariance that allows training on different sets of input, in our setting, from portfolios composed of different instruments. We bounded the loss from using this form of multi-task training. In addition, the formulation admits a universality property that allows it to be trained on a universe of m instruments and run on universes of any other size through the use of a order-preserving transformation such as softmax. Numerical results were provided on the S&P500 including against a number of common benchmarks.

References

[Antos *et al.*, 2008] András Antos, Csaba Szepesvári, and Rémi Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008.

[Bueno *et al.*, 2019] Tiago Pereira Bueno, Leliane Nunes de Barros, Denis Deratani Mauá, and Scott Sanner. Deep reactive policies for planning in stochastic nonlinear domains. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, AAAI, pages 7530–7537, 2019.

[Chua *et al.*, 2018] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 4754–4765. Curran Associates, Inc., 2018.

[Deisenroth and Rasmussen, 2011] Marc Peter Deisenroth and Carl Edward Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *In Proceedings of the International Conference on Machine Learning*, 2011.

[D'Eramo *et al.*, 2020] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. Sharing knowledge in multi-task deep reinforcement learning. In *International Conference on Learning Representations*, 2020.

[Ding *et al.*, 2014] Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. Using structured events to predict stock price movement: An empirical investigation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1415–1425, 2014.

[Hafner *et al.*, 2019] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2555–2565, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[Jiang *et al.*, 2017] Zhengyao Jiang, Dixing Xu, and Jinjun Liang. A deep reinforcement learning framework for the financial portfolio management problem, 2017.

[Lazaric *et al.*, 2012] Alessandro Lazaric, Mohammad Ghavamzadeh, and Rémi Munos. Finite-sample analysis of least-squares policy iteration. *Journal of Machine Learning Research*, 13(Oct):3041–3074, 2012.

[Lillicrap *et al.*, 2016] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2016.

[Mnih *et al.*, 2016] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[Schulman *et al.*, 2015] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 3528–3536. Curran Associates, Inc., 2015.

[Sharma *et al.*, 2018] Sahil Sharma, Ashutosh Kumar Jha, Parikshit S Hegde, and Balaraman Ravindran. Learning to multi-task by active sampling. In *International Conference on Learning Representations*, 2018.

[Sutton, 1991] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, July 1991.

[Williams, 1992] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, May 1992.

[Ye *et al.*, 2020] Yunan Ye, Hengzhi Pei, Boxin Wang, Pin-Yu Chen, Yada Zhu, Jun Xiao, and Bo Li. Reinforcement-learning based portfolio management with augmented asset movement prediction states, 2020.