Imitation Learning for Financial Applications

Sven Goluža*, Tessa Bauman†, Tomislav Kovačević‡ and Zvonko Kostanjčar§,
University of Zagreb, Faculty of Electrical Engineering and Computing
Laboratory for Financial and Risk Analytics
Unska 3, 10000 Zagreb, Croatia
*sven.goluza@fer.hr, †tessa.bauman@fer.hr, ‡tomislav.kovacevic@fer.hr and §zvonko.kostanjcar@fer.hr

***Abstract*—Algorithms for solving decision-making problems under uncertainty are often employed in complex environments such as financial markets. Reinforcement learning (RL), a mathematical framework for sequential decision-making, is widely used in many financial decision-making problems, such as portfolio optimization, optimal execution, and market making. However, RL methods require a manual design of the reward function, which can be challenging in noisy market environments. Moreover, in such environments, it is often easier to demonstrate the desired behavior rather than manually engineer it. These problems can be solved using imitation learning (IL) algorithms, which extract knowledge from expert demonstrations by directly imitating the desired behavior or learning the expert's reward function. This paper introduces the foundational methods in IL, outlines the differences with familiar frameworks, and provides a survey of the current IL applications in finance.**

***Keywords*—imitation learning, reinforcement learning, quantitative finance, machine learning**

I. Introduction

With the advances in information technology and the ever-increasing amount of data, the number of potential applications using artificially-intelligent agents capable of mimicking human behavior and making decisions has increased rapidly in the financial industry. A popular approach to learning autonomous behavior is reinforcement learning (RL) [1]. RL agents learn by interacting with the environment through trial and error learning and receiving feedback for their actions, thus improving their behavior using their experience. Despite the recent success in many fields, especially with advances in deep reinforcement learning [2], [3], RL algorithms have shown shortcomings in complex and dynamic environments such as financial markets. Learning by interaction in the context of markets usually involves historical data, which is highly stochastic and noisy. Under these circumstances, data-driven methods frequently fail to generalize and are often less effective than experts. Moreover, manually designing a reward function proves to be a sensitive issue as it may require a significant amount of experiments and feature engineering [4]. All of the mentioned problems have elevated the study of imitation learning (IL) [5].

Imitation learning (also known as apprenticeship learning or learning from demonstrations) deals with learning behaviors from expert demonstrations. Historically, it is an interdisciplinary research area with multiple existing taxonomies stemming from robotics, dynamic systems, and other engineering-based methods [6]. It has been applied to control domain problems such as navigation [7] and robotic manipulation [8]. Progress in high-dimensional computation and machine learning, particularly deep learning, has given rise to potential applications of AI-based IL methods in the financial industry.

Traditionally, the two main branches of study within IL are behavioral cloning [9] and inverse reinforcement learning [10]. Behavioral cloning (BC) methods learn to perform a task by directly replicating observed expert behavior. They are closely related to a traditional supervised learning setting as they learn a mapping from features (states) to labels (actions). On the other hand, inverse reinforcement learning (IRL) methods learn a reward function from the observed expert behavior, leveraging the learned reward function later on through RL. However, due to a rise in research on generative adversarial networks [11], and their connection to IRL [12], a new IL branch based on the generative adversarial imitation learning algorithm [13] started to develop. Given that this algorithm does not belong to either BC or IRL (as it follows the IRL structure but does not recover the reward function), researchers nowadays tend to form an additional IL branch often referred to as adversarial imitation learning (AIL) [14]. Even though IRL and AIL are closely related, this paper will follow the newly formed taxonomy of three separate branches.

II. Reinforcement learning

Reinforcement learning is a machine learning approach for learning optimal decisions by interacting with an environment in a trial and error fashion. The environment dynamics are described using Markov Decision Processes (MDPs). Formally, an MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{R}, p_0 \rangle$:

- $\mathcal{S}$ is a set of states,
- $\mathcal{A}$ is a set of actions,
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto [0, 1]$ is a transition probability matrix,
- $\gamma \in [0, 1]$ is a discount factor,
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ is a reward function,
- p_0 is an initial state distribution.

At each step t, the agent makes an action A_t based on the environment's current state observation S_t. Performing an action A_t in state S_t determines the next state S_{t+1} and the reward R_{t+1}. This iterative process is repeated, and sequences of states, actions, and rewards are acquired $(S_0, A_0, R_1, S_1, A_1, R_2, ..., S_t, A_t, R_{t+1})$. The

state-action sequence is referred to as a trajectory. MDP states have Markov property, meaning that the next state only depends on the current state and the current action, not on the history of all the states and actions that were taken before.

An agent's behavior is determined by its policy π, which can either be a deterministic or stochastic function. RL is based on the reward hypothesis, which states that all goals can be described by the maximization of cumulative reward, which is mathematically expressed as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

The agent's primary goal is to find a policy π^* that maximizes the expected return $\mathbb{E}[G_t]$, i.e., the expected cumulative reward over all future time steps, for each given state:

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}[G_t | S_t = s], \forall s \in \mathcal{S}.$$

Any data-driven method well suited to solving problems formulated in terms of an MDP and yielding an optimal policy can be considered an RL method. Numerous RL methods are broadly categorized into value-based, policy-based, and actor-critic. Another categorization is based on the use of model dynamics and transition probabilities during training: model-free and model-based methods. Further information on RL algorithms can be found in [1].

RL and IL algorithms handle decision-making tasks and differ from supervised learning in terms of sequential and non-i.i.d data. As discussed in Section I, IL is a useful tool for dealing with the limitations of RL in complex environments. It is a common assumption that having prior knowledge about a problem is more effective than learning solely through trial and error. IL methods incorporate prior knowledge about a specific problem by leveraging experts and collecting their demonstrations as a set of trajectories. IL approach assumes this is a more effective way of addressing behavioral learning, as it's often easier to demonstrate the optimal behavior for a task than mathematically formulate it.

III. Imitation Learning

Imitation learning is an approach for learning the desired behavior from expert demonstrations. This approach is helpful in scenarios where it is more practical for an expert to demonstrate the ideal behavior rather than compute the optimal policy or define the reward function manually. An expert can either be a human or any agent capable of optimally performing a task.

A. Behavioral Cloning

Behavioral cloning methods are direct IL methods that learn a mapping from states to actions without recovering the reward function [9]. These methods are formulated as a supervised learning problem where the dataset is a set of state-action pairs $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^N$ obtained from the expert, and the goal is to find the optimal policy π^* [15]. The policy refers to a regressor or classifier here. BC methods are efficient because of their ability to learn without interacting with the environment, as they are not formulated as an MDP.

Algorithm 1: Dataset Aggregation (DAgger)

1: Initialize $\mathcal{D} \leftarrow \emptyset$, initialize $\hat{\pi}_1$ to any policy in Π
2: **for** iteration $i = 1$ to N **do**:
3: Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$
4: Sample T-step trajectories using π_i
5: Get dataset $\mathcal{D}_i = \{(s, \pi^*(s))\}$ of visited states by π_i and actions given by expert
6: Aggregate datasets: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_i$
7: Train classifier $\hat{\pi}_{i+1}$ on $\mathcal{D}$
8: **endfor**
9: **return** best $\hat{\pi}_i$ on validation set

There are some tradeoffs, however, as supervised learning methods assume actions in the expert's trajectories are i.i.d. BC methods violate that property because actions (labels) influence future states (features) and thus influence future data distribution. This distribution mismatch problem leads to compounding errors when the agent finds himself in a state he has never been to before. Ideally, at each step t, the agent finds himself in a state drawn from the state distribution of expert trajectories ($s_t \sim p_{\pi^*}(s_t)$). In the setup described above, the agent will find himself in the state distribution resulting from taking action induced by the learned policy in the previous step ($s_t \sim p_{\pi_\theta}(s_t)$). A supervised approach to imitation learning disables the agent from learning to recover from failures and fails to generalize to unseen scenarios.

BC methods tend to solve this problem by changing $p_{\pi_\theta}(s_t)$ using data augmentation techniques. While using standard data augmentation techniques such as synthetic data and expanded state spaces helps, most BC methods build on specific data augmentation technique which interactively queries the experts in additional data points resulting from applying the current policy [16]. The idea is to start from an expert policy and iteratively and slowly replace it with the learned policy. Using this approach, the agent is retraining under the new state distribution after each policy change, and thus BC algorithms reduce imitation problem to supervised learning.

Dataset Aggregation (DAgger) [16] is an iterative BC algorithm that learns a deterministic policy using the dataset aggregation technique. It is shown in detail in Algorithm 1. This data-efficient approach combines the presence of the expert with expanding the space of possible trajectories to improve performance in unseen scenarios and recover from failure. The main drawbacks of the algorithm are frequent expert querying and the computational burden of iteratively updating the agent's policy. Later research papers such as [17]–[19] were proposed to improve DAgger.

B. Inverse Reinforcement Learning

Inverse reinforcement learning methods are indirect IL methods that learn a reward function from expert demonstrations [20]. As the reward hypothesis states that all goals can be described by the reward function formulation, RL assumes that the reward function, not policy, is the most robust and transferable definition of the learning task [21]. IRL methods describe the environment using an MDP, and the input to an IRL method consists of a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, p_0, \mathcal{D} \rangle$, where $\mathcal{D} = \{\tau_1, \tau_2, ..., \tau_n\}$ is a set of expert demonstrations, and τ_i represents a trajectory. The goal is to recover the reward function $\mathcal{R}$. All IRL methods follow the same iterative template, which allows for the simultaneous improvement of the learned policy and the learned reward function [20]:

1) Collect expert demonstrations $\mathcal{D} \sim \pi^*$.
2) Initialize the reward function r_w.
3) Learn the policy π^L by using RL with respect to the current r_w.
4) Update parameters w to minimize the divergence between the expert policy π^* and the learned policy π^L.
5) Repeat the previous two steps until divergence is reduced to the desired level.

The problem of determining the reward function is ill-posed because a policy can be optimal for multiple reward functions [4]. One of the first IRL algorithms which addressed the solution ambiguity is presented in [21]. The authors assume the reward function is linear in parameters and use the concept of feature expectations to resolve the ambiguity. Feature expectations are the expected value of state (and/or action) features, defined in the reward function, accumulated over the trajectory. Furthermore, they are estimated by sampling trajectories and used to express the value of a policy. The authors find a policy π^L and its corresponding r_w such that the difference between the value of the expert policy π^* and π^L is smaller than a predefined margin ϵ. IRL methods that optimize different margins are referred to as the maximum margin methods [4], [21], [22]. Minimizing the margin based on feature expectations is a heuristic approach and therefore has significant drawbacks when expert demonstrations are suboptimal. This leads to the case where no single reward function makes the expert policy both optimal and better than all the other policies.

To tackle these issues, [23] introduced the method based on the probabilistic model of behavior. This method chooses a distribution $p(\tau|w)$ that maximizes the entropy among the distributions that match the feature expectations of the demonstrator. Maximum entropy method builds on maximum margin methods and alleviates some of their fallbacks. However, this method still requires solving the full RL problem for optimal policy in the inner loop, carefully designing features to impose the structure on the reward function, and a model of the environment. To apply these methods in real-world settings, especially in finance, the problem of learning the expressive reward function under unknown dynamics needs to be solved.

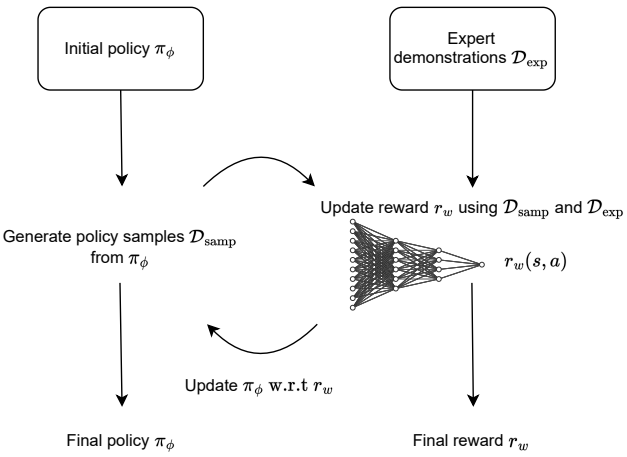

Fig. 1: Guided cost learning (GCL) algorithm

Guided cost learning (GCL) algorithm was proposed in [24], combining sample-efficient handling of unknown dynamics with neural network representation of reward function to handle existing problems in the literature. The visual guide to the algorithm is given in Fig. 1. Later research like [25]–[27] explored ideas such as ranking the expert demonstrations and integrating IRL methods with other methods such as self-supervised learning.

Most of the problems found in BC and IRL methods (robustness, unknown dynamics, suboptimal demonstrations) have been tackled with generative adversarial imitation learning (GAIL) [13], which combines the robustness of IRL and the efficiency of BC to improve IL methods further.

C. Adversarial Imitation Learning

The connection between IRL and GANs was presented in [12], where authors conclude that sample x represents a trajectory τ, generator G represents an agent's policy $\pi^L \sim p(\tau)$, and discriminator D is used as a proxy for reward [12].

GAIL algorithm [13] leverages the mentioned connection between IRL and GANs by training a policy that reproduces the expert behavior and a discriminator that distinguishes trajectories induced by the agent's policy from trajectories demonstrated by the expert. GAIL fits a parameterized policy π^L_θ, and a discriminator network $D_w : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$. The discriminator network represents the cost function, the negative of the reward function. The iterative algorithm is shown in detail in Algorithm 2. At the end of an iterative process, GAIL does not yield the reward function and, as such, can not be considered an IRL method. Ever since the introduction of GAIL, the research community has made further proposals on building upon it and applying it to real-world scenarios [28]–[33].

Algorithm 2: Generative Adversarial Imitation Learning (GAIL)

1: **Input:** expert trajectories $\tau_{\text{exp}} \sim \pi^*$, initial policy and discriminator parameters θ_0, w_0;
2: **for** iteration $i = 1$ to N **do**:
3: Sample trajectories $\tau_i \sim \pi^L_{\theta_i}$
4: Update discriminator parameters from w_i to w_{i+1} with gradient

$$\mathbb{E}_{\tau_i}[\nabla_w \log(D_w(s,a))] + \mathbb{E}_{\tau_{\text{exp}}}[\nabla_w \log(1 - D_w(s,a))]$$

5: Update policy parameters from θ_i to θ_{i+1}, using the TRPO rule with cost function $\log(D_{w_{i+1}}(s,a))$
6: **endfor**
7: **return** optimized policy parameters θ

IV. Applications in Finance

Imitation learning applications in finance can be broadly classified into identifying investment behavior and learning investment strategies. Important mechanisms that drive financial markets, such as price discovery, market liquidity, and order flow, can be explained by studying investment behavior and evaluating its economic impact. Those studies can also be of interest to market regulators and operators to counter possible fraudulent practices and maintain the transparency and functioning of financial markets. The other group of IL applications in finance considers the problem of investment strategy design across different assets and periods. They do so by observing trading history data from experts, which may be human (such as fund managers or prop traders) or an oracle with complete information (access to past and future data). Although the ML literature has recently become more prominent in finance, the applications of IL in finance have yet to be studied in depth [34].

One of the first applications was the identification of high-frequency trading strategies based on order book data [35]. The authors collected up to a month of order book audit trail data from the CME exchange for the E-Mini S&P 500 index futures. They modeled trader behavior as an MDP and used order book trails as demonstration data to infer traders' policies and reward functions. The state space was defined by inventory positions and order volume imbalance at the first five levels of the limit order book (LOB). Action space was defined by three actions (placing a limit order (LMT), canceling an existing limit order (CXL), and placing a market order (MKT)) directed at any of the ten buckets that LOB was divided into. The authors used model-based IRL methods which estimated the state transition probability matrix. Reward functions offered useful features for supervised learning, such as the classification of traders, and unsupervised learning, such as the categorization of traders. They introduced the gaussian process IRL (GPIRL) method to learn traders' reward functions and compared it to linear IRL (LNIRL) from [4] in terms of trading strategy identification. They used an SVM classifier to identify traders based on reward functions recovered from mentioned IRL methods and showed that GPIRL outperforms LNIRL. The authors showed that IRL methods are superior to statistical approaches, as they aim to learn the objectives of traders in various market conditions. That makes IRL methods more informative and robust and provides a solid foundation for identifying the behavioral patterns of traders.

In contrast to previous research that relied on labeled market participant data, [36] aimed to model traders' behavior using only unlabeled market data. The authors proposed a neural network-based multi-modal IL model which performs latent segmentation of stock trading strategies by their reward function. Each segment represents an individual strategy and has its reward function. An IL model is defined for each segment and trained to predict which trading segment was most likely to have submitted a particular order at a particular time. Finally, the model predicts order submission probability by combining outputs from each segment. State space features included price series and order book features at ten levels above and below the mid-market price. Action space consisted of order type (LMT, MKT, CXL), discretized order price, and volume. Their proposed model outperformed other IL benchmarks, such as GAIL, in terms of accuracy on historical order book data obtained from the Tokyo Stock Exchange. Thus, it provides a meaningful interpretation of traders' behavior. The limitation of their study is that the segment classification ability depends significantly on the formulation of the reward function.

In [37], the authors designed an investor sentiment trading system using IRL. They modeled the financial market dynamics as MDP and treated investor sentiment as actions in different market states. They extracted the reward function using GPIRL and leveraged the learned reward function in supervised learning to construct a trading strategy. The hypothesis was that the market sentiment reward function could be an effective feature space for predicting future financial market trends as it filters out irrelevant news sentiment signals. News sentiment from the Thomson Reuters news analytics database was used to proxy investor sentiment toward the U.S. market. State variables included discretized log return and volatility to describe the financial market conditions. At the same time, action space consisted of investor sentiment shocks to the market, categorized into three categories: positive, neutral, and negative. From an academic standpoint, this research offers a novel method for uncovering the underlying connection between the financial market and investor sentiment. From a practical standpoint, this research shows that an IRL-based trading strategy is superior to benchmark strategies based on S&P 500 index and market-based ETFs, as well as a few other news sentiment-based strategies.

In [38], a trading strategy combining BC and RL was proposed. The authors modeled the market, consisting of minute-level frequency data, as a partially observable MDP (POMDP). They employed a recurrent deterministic

policy gradient (RDPG) model for solving the POMDP. Observation space features consisted of cumulative account profits, price series, and technical indicators. The action space was defined as a continuous probability vector corresponding to taking the trading positions, but actions (short/long) are discretized to $\{-1, 1\}$. Differential Sharpe ratio was used as the reward function. The model was improved with BC by using the dual thrust strategy as an oracle, having access to all historical data. This IL technique enhances the trading agent's financial domain knowledge. BC was added to the model by incorporating two separate modules. Firstly, the agent was pre-trained using demonstrations from the dual thrust strategy, and secondly, the intra-day oracle was introduced. The oracle takes a long position at the lowest price and a short position at the highest price in a given day. Its actions were incorporated into the policy gradient to reduce the inefficient exploration phase. The model was backtested on IF and IC futures data, representative of stock index futures in China, and outperformed other benchmarks (long-only, short-only, and DDPG) in terms of Sharpe ratio and total return. The authors demonstrated that the BC modules significantly enhance the model's profitability metrics through an ablation study, revealing the impact of each component on performance. The final study demonstrates the model's ability to generalize by evaluating its performance on markets different from those it was trained on.

Goal-based wealth management problem (optimization of retirement plans or target-dated funds) was explored in [39] with IRL and RL. G-learner, the algorithm based on G-learning (probabilistic extension of Q-learning), was used in the RL wealth management problem. In contrast, the GIRL (G-learning IRL) model was used to infer the rewards of financial agents. The state space was defined as the vector of dollar values of asset positions and the action space as the vector of dollar-valued changes in asset positions. The authors defined a parameterized reward function that imposes a penalty for the underperformance of a portfolio relative to a target, along with regularization terms. The GIRL algorithm has been used to infer the reward function parameters, given a history of positions in an investment portfolio and an agent's allocation decisions. GIRL imitates the G-learner by minimizing a loss function over the state-action trajectories generated by the G-learner. Simulated models of equity returns were utilized to showcase the application of the G-learner and GIRL algorithms in goal-based wealth management. The results show that the G-Learner outperforms the benchmark equally-weighted portfolio strategy and that GIRL's learned parameters are close to the original G-learner parameters.

A combination of IRL and RL for optimal asset allocation was proposed in [40]. The IRL model learned the investment strategies of portfolio managers using their trading histories, and the RL model was used to improve those strategies. The T-REX algorithm [27] is utilized in the IRL step, incorporating the portfolio performance ranking of managers into its optimization process and eliminating the assumption of an optimal or near-optimal policy in the demonstrated trading history. The state and action vectors are the same as in [39]. The reward function is a quadratic function of the state and action, consisting of only four parameters due to the limited data available in portfolio management applications. It incorporates penalty for under-performance of the traded portfolio relative to its moving target, transaction costs, and trades constraint. The recovered reward function was used in the G-learner algorithm to further optimize the investment policy. The dataset included 18 mutual funds' data with monthly holdings, trades, and cashflows, but due to data constraints, all stocks within each portfolio have been grouped by industry sector. Experiments showed that the combination of IRL and RL outperforms the majority of fund managers. Additionally, inferred reward parameters can be used to group different trading strategies and gain a deeper understanding of fund managers' actions.

In [41], the authors introduced an RL framework for optimal order execution. They improved noisy and imperfect data in their RL framework by using the learning-based oracle teacher with access to all historical data. Model over-fitting was prevented by training universal strategy on various instruments. State space consisted of the elapsed time, remaining inventory to execute, price, and volume of each time step. Action corresponded to the discrete proportion of the target order, and the reward function consisted of trading profitability and market impact penalty. Two agents were being trained: a student and a teacher. The teacher agent, who has access to perfect information, learns a policy that acts as an oracle. This policy then guides the learning of the student agent, which is the agent used in real-time order execution. Finally, the student agent is tested and compared to benchmarks in the China A-shares market using a dataset that includes minute-level market information. Experiments showed that the universal IL model outperforms all the other models and that teacher guidance significantly boosts the model's performance.

V. Conclusion

Machine learning methods, combined with growth in computational power and the amount of data, have enabled significant development of imitation learning in the last decade. IL algorithms aim to learn an optimal policy given expert demonstrations, and they were shown to be effective in financial applications. IL methods were found to outperform other statistical approaches in identifying investment behavior. IL also learned investment strategies across different assets and periods, outperforming baselines such as market indices and machine learning based models. These presented methods can serve as a future benchmark in developing IL models. Selection of a labeling algorithm as an oracle, investors' suboptimal behavior, insufficient training data, and feature representation are some of the open IL research points in quantitative finance.

ACKNOWLEDGMENT

This work was supported in part by the Croatian Science Foundation under Project 5241, and European Regional Development Fund under Grant KK.01.1.1.01.0009 (DATACROSS).

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[3] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.*, "Mastering the game of go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[4] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proceedings of the Seventeenth International Conference on Machine Learning*, vol. 0, 2000.

[5] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[6] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters *et al.*, "An algorithmic perspective on imitation learning," *Foundations and Trends® in Robotics*, vol. 7, no. 1-2, pp. 1–179, 2018.

[7] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 4693–4700.

[8] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun, "Survey of imitation learning for robotic manipulation," *International Journal of Intelligent Robotics and Applications*, vol. 3, pp. 362–369, 2019.

[9] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine Intelligence 15*, 1995.

[10] S. Russell, "Learning agents for uncertain environments," in *Proceedings of the Annual ACM Conference on Computational Learning Theory*, 1998, pp. 101–103.

[11] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[12] C. Finn, P. Christiano, P. Abbeel, and S. Levine, "A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models," *arXiv preprint arXiv:1611.03852*, 2016.

[13] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016.

[14] B. Zheng, S. Verma, J. Zhou, I. W. Tsang, and F. Chen, "Imitation learning: Progress, taxonomies and challenges," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–16, 2022.

[15] S. Ross and D. Bagnell, "Efficient reductions for imitation learning," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2010, pp. 661–668.

[16] S. Ross, G. J. Gordon, and J. A. Bagnell, "No-regret reductions for imitation learning and structured prediction," *Aistats*, vol. 15, pp. 627–635, 2011. [Online]. Available: http://proceedings.mlr.press/v15/ross11a/ross11a.pdf

[17] S. Ross and J. A. Bagnell, "Reinforcement and imitation learning via interactive no-regret learning," *arXiv preprint arXiv:1406.5979*, 2014.

[18] H. Le, A. Kang, Y. Yue, and P. Carr, "Smooth imitation learning for online sequence prediction," in *International Conference on Machine Learning*. PMLR, 2016, pp. 680–688.

[19] M. Laskey, J. Lee, R. Fox, A. Dragan, and K. Goldberg, "Dart: Noise injection for robust imitation learning," in *Conference on robot learning*. PMLR, 2017, pp. 143–156.

[20] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, p. 103500, 2021.

[21] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," *Proceedings, Twenty-First International Conference on Machine Learning, ICML 2004*, pp. 1–8, 2004.

[22] N. D. Ratliff, J. Andrew Bagnell, and M. A. Zinkevic, "Maximum margin planning," in *ACM International Conference Proceeding Series*, vol. 148, 2006, pp. 729–736.

[23] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey, "Maximum entropy inverse reinforcement learning," in *Proceedings of the National Conference on Artificial Intelligence*, vol. 3, 2008.

[24] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *33rd International Conference on Machine Learning, ICML 2016*, vol. 1, 2016, pp. 95–107.

[25] A. Nair, D. Chen, P. Agrawal, P. Isola, P. Abbeel, J. Malik, and S. Levine, "Combining self-supervised learning and imitation for vision-based rope manipulation," in *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2017, pp. 2146–2153.

[26] S. Reddy, A. D. Dragan, and S. Levine, "SQIL: Imitation Learning via Reinforcement Learning with Sparse Rewards," *arXiv preprint arXiv:1905.11108*, may 2019. [Online]. Available: https://arxiv.org/abs/1905.11108v3

[27] D. S. Brown, W. Goo, P. Nagarajan, and S. Niekum, "Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations," in *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, 2019.

[28] J. Fu, K. Luo, and S. Levine, "Learning Robust Rewards with Adversarial Inverse Reinforcement Learning," *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*, oct 2017. [Online]. Available: https://arxiv.org/abs/1710.11248v2

[29] Y. Li, J. Song, and S. Ermon, "Infogail: Interpretable imitation learning from visual demonstrations," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[30] Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess, "Robust imitation of diverse behaviors," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[31] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel, "Goal-conditioned imitation learning," in *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[32] W. Sun, A. Vemula, B. Boots, and D. Bagnell, "Provably efficient imitation learning from observation alone," in *International conference on machine learning*. PMLR, 2019, pp. 6036–6045.

[33] G. Zuo, K. Chen, J. Lu, and X. Huang, "Deterministic generative adversarial imitation learning," *Neurocomputing*, vol. 388, pp. 60–69, 2020.

[34] M. F. Dixon, I. Halperin, and P. Bilokon, *Machine learning in Finance*. Springer, 2020, vol. 1170.

[35] S. Y. Yang, Q. Qiao, P. A. Beling, W. T. Scherer, and A. A. Kirilenko, "Gaussian process-based algorithmic trading strategy identification," *Quantitative Finance*, vol. 15, no. 10, pp. 1683–1703, 2015.

[36] I. Maeda, D. DeGraw, M. Kitano, H. Matsushima, K. Izumi, H. Sakaji, and A. Kato, "Latent segmentation of stock trading strategies using multi-modal imitation learning," *Journal of Risk and Financial Management*, vol. 13, no. 11, p. 250, 2020.

[37] S. Y. Yang, Y. Yu, and S. Almahdi, "An investor sentiment reward-based trading system using gaussian inverse reinforcement learning algorithm," *Expert Systems with Applications*, vol. 114, pp. 388–401, 2018.

[38] Y. Liu, Q. Liu, H. Zhao, Z. Pan, and C. Liu, "Adaptive quantitative trading: An imitative deep reinforcement learning approach," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 02, 2020, pp. 2128–2135.

[39] M. Dixon and I. Halperin, "G-learner and girl: Goal based wealth management with reinforcement learning," *arXiv preprint arXiv:2002.10990*, 2020.

[40] I. Halperin, J. Liu, and X. Zhang, "Combining reinforcement learning and inverse reinforcement learning for asset allocation recommendations," *arXiv preprint arXiv:2201.01874*, 2022.

[41] Y. Fang, K. Ren, W. Liu, D. Zhou, W. Zhang, and Y. Yu, "Imitation from learning-based oracle for universal order execution in quantitative finance," 2021, unpublished.