K-SHAP: Policy Clustering Algorithm for Anonymous Multi-Agent State-Action Pairs

Andrea Coletta [1] **Svitlana Vyetrenko** [2] **Tucker Balch** [2]

Abstract

Learning agent behaviors from observational data has shown to improve our understanding of their decision-making processes, advancing our ability to explain their interactions with the environment and other agents. While multiple learning techniques have been proposed in the literature, there is one particular setting that has not been explored yet: multi agent systems where agent identities remain anonymous. For instance, in financial markets labeled data that identifies market participant strategies is typically proprietary, and only the anonymous state-action pairs that result from the interaction of multiple market participants are publicly available. As a result, sequences of agent actions are not observable, restricting the applicability of existing work. In this paper, we propose a Policy Clustering algorithm, called K-SHAP, that learns to group anonymous state-action pairs according to the agent policies. We frame the problem as an Imitation Learning (IL) task, and we learn a *world-policy* able to mimic all the agent behaviors upon different environmental states. We leverage the *world-policy* to explain each anonymous observation through an additive feature attribution method called SHAP (SHapley Additive exPlanations). Finally, by clustering the explanations we show that we are able to identify different agent policies and group observations accordingly. We evaluate our approach on simulated synthetic market data and a real-world financial dataset. We show that our proposal significantly and consistently outperforms the existing methods, identifying different agent strategies.

[1]J.P. Morgan AI Research, London, UK. [2]J.P. Morgan AI Research, New York, USA. Correspondence to: Andrea Coletta <andrea.coletta@jpmchase.com>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

1. Introduction

Insect colonies, animal swarms, and human societies are examples of the complex multi-agent systems in nature. Each agent in these systems develops a behavioral strategy to take suitable actions in response to environmental changes. Animals learn different behaviors to survive and exploit available resources (e.g., food), while humans may develop complex behaviors to efficiently interact and pursue their own goals.

The study and understanding of human and animal behaviors has been a fundamental problem in both computer and behavioral sciences (Cichos *et al.*, 2020). In particular, understanding agent behaviors from observational data is essential to study, predict and simulate their behaviors (Li *et al.*, 2020; Suo *et al.*, 2021).

Existing work borrows tools from the reinforcement learning literature to learn agent strategies (Hussein *et al.*, 2017; Ho and Ermon, 2016; Song *et al.*, 2018; Fu *et al.*, 2021) using Markov Decision Processes (MDPs) as an efficient mathematical framework to formulate the problem. In particular, Inverse Reinforcement Learning (IRL) (Ng *et al.*, 2000) has been widely studied to characterize the decision-making behavior of animals and humans. IRL aims at recovering a reward function that explains the agent goal. It has been used to study worm behavioral strategies (Yamaguchi *et al.*, 2018); to model the behaviors of mice exploring a labyrinth (Ashwood *et al.*); to identify and capture the behavior of troll accounts in social networks (Luceri *et al.*, 2020). Imitation Learning (IL) (Hussein *et al.*, 2017) is another technique that has demonstrated great success in modeling agent behaviors, learning directly from their trajectories (state-action pairs). Recently, IL has shown to learn realistic driving behaviors from human demonstrations (Suo *et al.*, 2021). Finally, Hidden Markov Models (HMMs) have been extensively used to analyze temporal dynamics and model agent behaviors from observed sequential data. HMMs have been used to study different animal behaviors, including honey bees (Feldman and Balch, 2004) and mice (Jiang *et al.*, 2018).

While all these techniques assume that clear observation sequences for each agent are given, some critical domains with

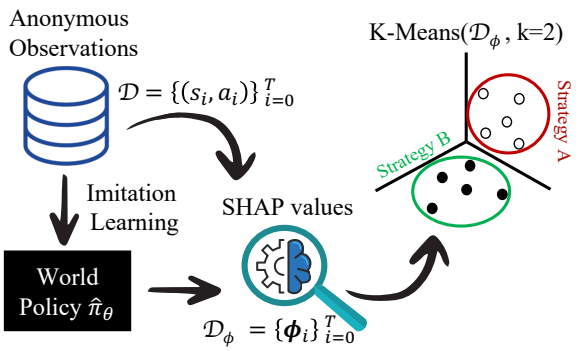

Figure 1: The three major phases of K-SHAP.

privacy concerns may provide only anonymous state-action pairs as data. For example, military operations are often anonymous to conceal strategies to opponents, especially in case of cyber-attacks or when illegal activities are operated (e.g., use of chemical or biological weapons) (Koblentz, 2019; Rid and Buchanan, 2015). In financial markets labeled data that identifies market participants is typically proprietary, and publicly available transaction data is typically anonymous (NASDAQ, 2022). However, understanding market participant strategies for the purpose of ensuring that markets are orderly and compliant with regulations is necessary (Hagströmer and Nordén, 2013; Kirilenko *et al.*, 2017; Wang *et al.*, 2021). For instance, in (Kirilenko *et al.*, 2017) the authors were able to study and demonstrate that high frequency traders did not contribute to the 2010 flash crash using audit trail transaction-level data, which reveals the identities of market participants. Nevertheless, it is most common that only the anonymous state-action pairs without any agent identifiers are publicly available from exchanges (NASDAQ, 2022).

It is well known that thousands of individual market participants can be broadly assigned to a small number of distinct behavioral strategies (Kirilenko *et al.*, 2017; Vyetrenko *et al.*, 2020). For example, multiple market agents might trade on momentum signals even though these momentum signals are of different magnitudes. Similarly, all market makers are required to place both buy and sell orders to provide liquidity in the markets, even though individual market makers might act upon different proprietary signals to do that. Previous work has used IRL with reward clustering to distinguish high frequency from other trading strategies in simulated (but not the real) markets (Yang *et al.*, 2012). However, the proposed method requires the inventory level and the labeled sequence of actions for each trader, which are usually unknown in real markets. We are not aware of any other work that would allow us to identify the individual agents or their strategies from anonymous state-action pairs.

To address the challenge of learning agent strategies from anonymous state-action pairs, in this paper we propose a novel policy-clustering method – in which we group state-action observations that belong to agents sharing the same behavior or policy. These clusters describe the different behaviors of agents, enabling further studies and analysis. In detail, we propose K-SHAP, a Policy Clustering algorithm for anonymous state-action pairs. K-SHAP comprises three major phases shown in Figure 1. First, we model the problem as an Imitation Learning task to learn a *world-policy* (Coletta *et al.*, 2022) using supervised learning over the anonymous state-action pairs. The *world-policy* is an extension of the *world-model* approach (Ha and Schmidhuber, 2018) to state-action pairs: it emulates collective agent behaviors, learning their actions (output) in response to the different environmental states (input). Then, we interpret the world-policy to explain each anonymous observation through an additive feature attribution method, namely SHAP (SHapley Additive exPlanations) (Lundberg and Lee, 2017). Finally, we show that the explanations, called SHAP values, better reveal the intrinsic clustering structure among data; and by applying a K-Means algorithm (Hartigan and Wong, 1979) in the new SHAP values space, we can group the anonymous observations into K clusters, which reflect the real agent strategies.

To support the soundness of our method, we experimentally evaluate it with market data. We first consider synthetic data from a multi-agent simulator labeled with agent IDs to serve as a ground truth (Byrd *et al.*, 2019; Liu *et al.*, 2022), then we consider real anonymous market data from NASDAQ stock exchange (NASDAQ, 2022). For real data we evaluate the *Utility* and *Silhouette*, as ground truth labels are not available. In particular, by *Utility* we mean the ability to improve a specific downstream task (i.e., supervised learning) when trained on the clusters rather than original demonstrations. We compare our proposal against state-of-art clustering algorithms, including Deep Clustering Network (DCN) (Mukherjee *et al.*, 2019) and ClusterGAN (Mukherjee *et al.*, 2019). We also adapt Σ-GIRL (Ramponi *et al.*, 2020), a Multiple-Intent IRL clustering algorithm, to anonymous state-action pairs. We show that we significantly and consistently outperform the existing work, and that K-SHAP clusters anonymous observations according to the agent behaviors. K-SHAP outperforms existing methods by a factor of 2 on all the performance metrics.

1.1. Main contributions

To the best of our knowledge, this is the first work that addresses the problem of policy clustering with anonymous state-action pairs as observations from multiple agents. We summarize the main contributions of our paper as follows:

- We formalize the k-policy clustering problem under anonymous multi-agent state-action pairs.

- We propose K-SHAP, a Policy Clustering Algorithm which leverages IL and SHAP values to identify k distinct behavioral clusters.
- We experimentally evaluate K-SHAP against existing work, showing that it achieves good performance in all performance metrics, with both synthetic and real-world data. In particular, we show that K-SHAP can group market observations according to the trader strategies, improving our understanding of market participant and their impact.

2. Preliminaries and Problem Formulation

2.1. Agent behaviors as Markov Decision Processes

We consider a multi-agent system with N autonomous agents interacting in a given environment. We adopt Markov Decision Processes (MDPs) as a natural underlying decision model for each single agent. A MDP is a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R})$ composed by: the state space $\mathcal{S}$; the action space $\mathcal{A}$; a transition function $\mathcal{P} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta(\mathcal{S})$, where $\Delta(\mathcal{S})$ denotes the probability distributions over state space $\mathcal{S}$; and a reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$.
We consider that each agent follows a policy π (i.e., behavior) to maximize the expected reward over time:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{T-1} \gamma^t \mathcal{R}(s_t, a_t)\right] \tag{1}$$

where $\gamma \in [0, 1]$ is the discounted factor that can be used to prioritize early actions. We consider a general policy $\pi : S \rightarrow A$ that describes the preferred action $a \in \mathcal{A}$ in the state $s \in \mathcal{S}$.

2.2. Agent observations

In our problem setting, we do not have direct access to the agent rewards or policies, we consider instead a more general case of an MDP without reward (MDP$\backslash\{\mathcal{R}\}$). In this setting the reward $\mathcal{R}$ is unknown, and agent behaviors are represented by a set of trajectories $\mathcal{D} = \{\xi_0, \ldots, \xi_t\}$ for each agent, where a trajectory is a sequence of state-action pairs $\xi_i = \{(s_0, a_0), \ldots, (s_k, a_k)\}$. However, we have an additional confidentiality constraint: agents act anonymously and their observations do not contain any identifier. Thus, we are not able to identify a trajectory ξ for a single agent, but observational data is anonymously gathered from all the agents into a new set $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{T}$ over a time T. Notice that the collected state-action pairs (s_i, a_i) do not contain any information about which anonymous agent generated the action a_i at state s_i, preventing the identification of state-action sequences generated by an agent. This scenario is typical of military or financial domains, where agents want to hide their strategies for tactical reasons. In particular, in financial domains, the *actions* are the orders submitted by traders according to the current financial market *state*.
Anonymous observations exponentially increase the complexity of behavioral studies as identifying an agent behavior with existing approaches is almost impracticable: considering k agents and n observations, we have $O(k^n)$ possible assignment of the observations to reconstruct the agent trajectories needed by most of the existing methodologies.

2.3. Reward functions to model agent behaviors

Given an MDP formulation, a classic approach to identify agent behaviors is to study their reward function $\mathcal{R}$. This function is the most succinct representation of the agent goals, which define their *extrinsic motivation* to act (Chentanez *et al.*, 2004). Therefore, rewards help to study agent behaviors and can be used to solve *intent-clustering* (Ramponi *et al.*, 2020; Babes *et al.*, 2011) in which we aim at identifying groups of agents sharing the same goal. Considering a MDP$\backslash\{\mathcal{R}\}$, previous work employs IRL to recover the *unknown* reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ from the agent demonstrations (Ramponi *et al.*, 2020; Yamaguchi *et al.*, 2018; Yang *et al.*, 2012). In the general case, the unknown reward function $\mathcal{R}_{\boldsymbol{\omega}}(s, a)$ can be defined as a linear combination of q weighted features: $\boldsymbol{\omega}^T \boldsymbol{\phi}(s, a)$, $\boldsymbol{\omega} \in \mathbb{R}^q$, where $\boldsymbol{\phi}(s, a)$ is the feature function and $\boldsymbol{\omega}^T$ the feature weights.
However, in IRL the problem is ill-posed: even under perfect knowledge with demonstrations from an optimal policy, there exist many solutions (reward functions) for which a given behavior is optimal (Ng *et al.*, 2000). In general, the real reward function is not identifiable (Cao *et al.*, 2021). Most important, IRL requires observations as sequences of state-action pairs for each agent, while we consider domains in which state-action pairs are anonymous.

2.4. Learning a policy to study agent behaviors

Another simple, yet effective, approach to study the behavior of an agent is to directly describe their mapping from state features to actions, as the most parsimonious description of the policy itself. Recently, IL (Hussein *et al.*, 2017) has demonstrated that it is possible to reconstruct the agent policies directly from state-action pair observations. In particular, in IL we aim at learning an optimal policy π_θ for an agent i according to its demonstrations:

$$\arg\max_{\theta} \mathbb{E}_{s \sim d^{\pi_i}, a \sim \pi_i(\cdot|s)}[log\ \pi_\theta(a|s)] \tag{2}$$

Notice that d^{π_i} denotes the distribution over states induced by agent policy π_i. Once we reconstruct the policy π_i, we directly study the behavior of the agent i by investigating how the policy maps state to actions.
In fact, for a simple policy the best explanation is often the

model π_i itself, as a parsimonious description of the agent behavior. However, models describing complex policies cannot be easily studied and explained. In such cases, we can borrow techniques from explainable AI to investigate the behavior of the model (Dwivedi *et al.*, 2022).

2.5. SHAP (SHapley Additive exPlanations)

A recent successful technique for explainability is the SHAP values method (SHapley Additive exPlanations) (Lundberg and Lee, 2017). This method belongs to the additive feature attribution methods, which are local methods designed to explain a single prediction $f(x)$ based on the input x (Ribeiro *et al.*, 2016). These methods approximate the real model f using a simpler *explanation model* g. The model g tries to guarantee that $g(x') \approx f(h_x(x'))$, where x' is a simplified input for g, and h_x is a mapping function to reconstruct the original input $x = h_x(x')$. The *explanation model* is then a linear combination of binary variables x':

$$g(x') = \phi_0 + \sum_{i=1}^{m} \phi_i \cdot x'_i \tag{3}$$

where m is the number of simplified input features, and $\phi_i \in \mathbb{R}$ measures the contribution of each feature to the model output. The sum of all feature contributions approximates the original model output $f(x)$. In particular, SHAP values method uses classic game theory to explain the model predictions. Given a set of features F the method computes the contribution ϕ_i of each feature $i \in F$ by evaluating the model with and without such a feature. To fairly account for the effects of the withholding of a feature among the others, SHAP computes the average contribution of a feature i considering the model over all the possible subsets $S \subseteq F \setminus i$:

$$\phi_i = \sum_{S \subseteq F \setminus i} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \Big[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \Big]$$

where x_S represents the input features in the subset S, and f_S is the model trained on the subset of features S or its approximation. Therefore, the SHAP values describe how the input features (state) contribute to the output (action). We denote with $\Phi \in \mathbb{R}^m$ the new space of SHAP values, and with $\boldsymbol{\phi} \in \Phi$ the SHAP values vector for an input x.

While IL and SHAP values provide a promising approach to study the behavior of an agent, IL requires a trajectory ξ, or a set of state-action pairs, for each agent to approximate the policy correctly. Therefore, we cannot directly combine these two methods in our setting, as state-action pairs are anonymous without any agent identifier. In the following sections, we will show that if we are able to learn a *world-policy* describing all the agent behaviors, SHAP values are naturally well suited to describe each observation as the market state contribution to the action, which characterizes the different trader strategies (see Section 1).

2.6. Policy-Clustering under anonymous state-action pairs

In the general policy-clustering setting we consider a set of n agents $\mathbf{A} = \{\mathrm{A}_0, \dots, \mathrm{A}_n\}$ following a finite unknown set of policies $\boldsymbol{\pi} = \{\pi_0, \dots, \pi_k\}$ such that $k < n$ (i.e., multiple agents may have the same behavior π_i). The goal of policy-clustering is to group the agents $\mathbf{A}$ into k clusters, according to their policies from the observational data $\mathcal{D} = \{\xi_0, \dots, \xi_t\}_{i=0}^{n}$. Agents in the same cluster will follow the same strategy or behavior. Unlike the general formulation, we relax the assumption of knowing agent identities and trajectories. Thus, the goal of policy-clustering under anonymous state-action pairs is to group the anonymous observational data $\mathcal{D} = \{(s_i, a_i)\}_{i=0}^{T}$ into k clusters, such that observations in the same cluster belong to agents sharing the same policy π.

3. K-SHAP Framework

This section introduces our main contribution, K-SHAP, a Policy Clustering algorithm that learns to cluster the anonymous input observations $\mathcal{D}$ into k different policies. K-SHAP is inspired by recent advancements in Imitation Learning, world policy, and explainable AI. The Algorithm comprises three major phases:

- We first frame the problem as an IL task with anonymous state-action pairs to learn a unique *world policy* able to emulate all the agent behaviors;
- We compute local explanations of the *world policy* for each state-action pair through SHAP (SHapley Additive exPlanations). The explanations describe the different agent behaviors upon each observation;
- We discuss SHAP values properties and how they facilitate the clustering. Finally, we apply a *K-Means* algorithm to group the observations into k policies.

3.1. World Policy

Hypothetically we could learn the agent policies by solving n IL problems, one for each agent; however, anonymous state-action pairs restrict this approach due to the lack of individual agent trajectories. Another approach could be a world policy $\hat{\pi}_\theta$ trained over all the observations $\mathcal{D}$.
Recent work shows that highly complex environments with multiple autonomous agents can be successfully simulated by learning a unique *world-policy* able to mimic all the agents (Ha and Schmidhuber, 2018; Coletta *et al.*, 2021; 2022). To truthfully simulate the real world and the agent

interactions, these models have to learn and impersonate multiple heterogeneous behaviors upon different input from the environment. Therefore, they inherently hold some knowledge about all the agent strategies.

Hereafter, we assume that a *world-policy* $\hat{\pi}_\theta$ is the most parsimonious model able to represent the different agent behaviors by learning their mapping from states to actions from a set of observations $\mathcal{D}$:

$$\hat{\pi}_\theta = \arg\min_\theta \sum_{(s,a)\in\mathcal{D}} l(\pi_\theta(s), a) \tag{4}$$

where l is the loss function of the learning problem.
Given the world policy, we can study its behavior over each state-action pair (s_i, a_i), to infer the different strategies adopted by the real agents, and cluster the observations accordingly. In K-SHAP we train a world policy $\hat{\pi}_\theta$ in a behavioral cloning fashion, which learns the expert policies using supervised learning.[1]

3.2. Local explanations

Explaining the *world-policy* $\hat{\pi}_\theta$ behavior under different inputs can reveal details about the strategies adopted by the real agents. K-SHAP borrows a successful technique from explainable AI, namely SHAP (SHapley Additive exPlanations) (Lundberg and Lee, 2017) to disentangle the world-policy's complexity and study its behavior.

For a demonstration (s, a) we can compute the SHAP values as a vector $\boldsymbol{\phi} \in \mathbb{R}^{|F|}$ defined as the average marginal contribution ϕ_i of each state feature $i \in F$ to the prediction $\hat{\pi}_\theta(s) \approx a$. Therefore, we can express the behavior of a real agent for an observation j as the approximated mapping from state s to action a, defined by the SHAP values vector $\boldsymbol{\phi}_j$ (which describes how each state feature i drives the agent behavior a in the state s). In general, the state s can be any representation of the environment in which the agent interacts, including images, text and tabular data, as SHAP values method can deal with complex and continuous data.

3.3. K-SHAP Algorithm

We now describe how K-SHAP combines IL and SHAP values to group the anonymous state-action pairs. The full procedure is shown in Algorithm 1.

First, K-SHAP learns the world policy using all the available observations **(line 1)** in a behavioral cloning fashion. It trains a model using supervised learning, where each state s is used as input and the related action a is considered as the output for the model. Then, by leveraging the world-policy

[1]Different training procedures or approaches can be applied to learn the *world-policy*, including adversarial training (Ho and Ermon, 2016).

Algorithm 1 K-SHAP Algorithm

Input: A set of observations $\mathcal{D} = \{(s_i, a_i)\}_{i=1}^T$, the number of clusters k, the IL loss function l
Output: Labels $\mathcal{L}$ for each observation to a given policy

1 $\hat{\pi}_\theta = \arg\min_\theta \sum_{(s_i,a_i)\in\mathcal{D}} l(\pi_\theta(s_i), a_i)$
2 $\mathcal{D}^\phi = \langle\rangle$
3 **for** $i = 0$ **to** $|\mathcal{D}|$ **do**
4 $\quad\boldsymbol{\phi} = \langle\rangle$
5 $\quad$**foreach** $j \in F$ **do**
6 $\quad\quad\boldsymbol{\phi}[j] \leftarrow \sum_{S\subseteq F\setminus j} \omega_S \left[\hat{\pi}_{S\cup\{j\}}(s_{i_{S\cup\{j\}}}) - \hat{\pi}_S(s_{i_S})\right]$
$\quad\quad$ where $\omega_S = \frac{|S|!(|F|-|S|-1)!}{|F|!}$
7 $\quad\mathcal{D}^\phi[i] \leftarrow \boldsymbol{\phi}$
8 $\mathcal{L}$ = K-Means$(\mathcal{D}^\phi, k)$
9 **return** $\mathcal{L}$

$\hat{\pi}_\theta$ and SHAP, it computes the explanations (i.e., SHAP values) for each observation $(s, a) \in \mathcal{D}$. The explanations represent the anonymous observations in a new SHAP values space $\mathcal{D}^\phi = \{\boldsymbol{\phi}_i\}_{i=0}^T$ **(lines 3-7)**. Finally, we adopt an existing clustering approach, namely K-Means, to group observations according to their SHAP values, such that observation in the same cluster will be originated from the same strategy **(line 8)**. Formally, we cluster observations by minimizing the inertia, or within-cluster sum-of-squares, of their SHAP values:

$$\sum_{j=0}^{k} \sum_{\boldsymbol{\phi}_i \in C_j} ||\boldsymbol{\phi}_i - \mu_j||^2 \tag{5}$$

where $\boldsymbol{\mu}$ are the cluster centroids, and C_j identify the observations assigned to cluster j.

We now briefly discuss two essential properties of SHAP values that we leverage to solve our k policy-clustering problem.
The first implication of the new space $\mathcal{D}^\phi$ is the ability to capture the agent behaviors as a response to the environment. In fact, SHAP values explain the world-policy $\hat{\pi}_\theta$ predictions with a linear *explanation model* g that captures state features contributions to an action a. By assuming that the world-policy mimics the different agent behaviors, the SHAP values define how the agents respond to the different input features. Therefore, the SHAP vectors $\boldsymbol{\phi}$ are similar for observations that originate from comparable behaviors — they respond similarly to environmental changes (i.e., features) – and they are close in the vector space $\mathcal{D}^\phi$. Contrarily, dissimilar behaviors have different vectors in the space.
Secondly, SHAP values offer a natural advantage in clustering problems (Lundberg *et al.*, 2018): by definition SHAP values convert all the state features into the same metric

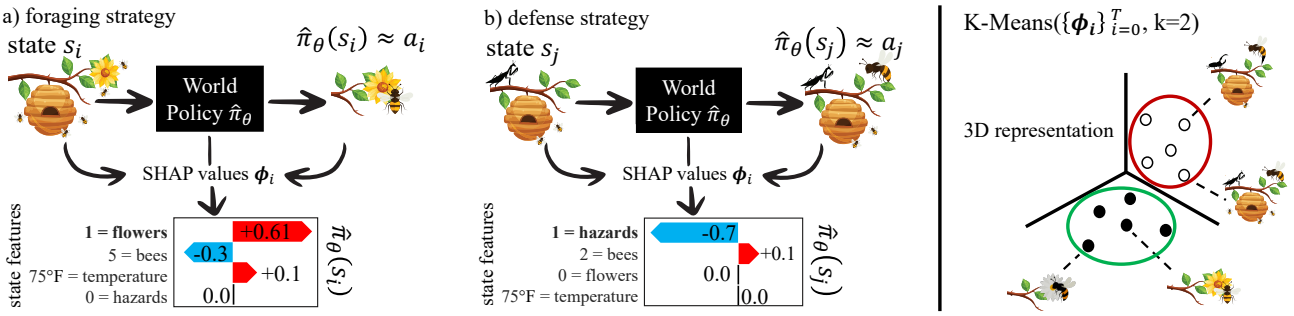

Figure 2: **Example of K-SHAP clustering.**

space. Each feature will have the same units as the model output, as the features measure their impact on the model output in the new SHAP space. This enables us to easily compare and cluster data, regardless of the different magnitude and units of the features. For example, while in Figure 2 it can be challenging to compare the temperature in Fahrenheit to the number of flowers, in the SHAP space both the features will have the same unit, representing their contribution to the action a.

K-SHAP training Our method generally requires selecting the number of clusters k as a unique hyper-parameter. When k is unknown, we adopt the classic Elbow method to optimize k according to the *distortion* or *variance* of the clusters in the SHAP values space. In Appendix D.1, we compare the clustering performance in scenarios where the value of k is either unknown or provided. In Section 4, we also show that we can use dimensionality reduction techniques like UMAP (McInnes *et al.*, 2018a) on the SHAP values space to visualize or improve clustering.
Finally, as we adopt a supervised IL technique we may need to use multiple *world-policies* in case of observations with identical states to disambiguate them and let the models learn the correct behaviors. In the case of identical states, a unique world-policy may average actions from multiple agents, with inaccurate SHAP values and clustering.

Algorithm illustration As an illustrative example, in Figure 2 we consider honey bees that act according to two different unknown behaviors (a) foraging and (b) defense. For simplicity, we describe the honey bee action $a \in [-1, 1]$ as the probability of staying close to protect the hive ($a \approx -1$) or moving to search for food ($a \approx 1$). We identify four state features describing the number of *flowers*, *bees*, *temperature*, and *hazards* close to the hive.

• In the scenario (a), we consider a foraging observation in which the honey bee observes the environment and flies close to the flower. We explain the honey behavior through the world policy using SHAP values. The SHAP values describe the honey bee behavior (i.e., search for food) as a strong contribution to the action $a = 1$ from the presence of a flower (i.e., food), and a small contribution from presence of warm weather (i.e., temperature). The presence of other bees around the hive is instead a negative contribution, as they may compete for the flower.

• In the scenario (b), we consider a defense observation in which a honey bee observes the environment and chooses to defend the hive. In this case the SHAP values describe the bee behavior as a major contribution to the action $a = -1$ from the presence of one hazard (i.e., mantis). The presence of few bees slightly reduces the need to defend the hive.

Finally, in the leftmost picture we show how we use the computed SHAP values to group the observations into $k = 2$ policies through a K-Means algorithm. The picture shows SHAP values from foraging and defense observations that lay into different regions of the space, i.e., clusters.

4. Experiments

We now experimentally evaluate K-SHAP on both synthetic and real data. We apply K-SHAP to anonymous observations from agents in a financial market (i.e., traders), and we experimentally demonstrate that we can identify state-action pair clusters dictated by the different behavioral strategies. We average the results over 20 runs, and we assume to know the number of clusters k.[2] We consider a Random Forest as the *world-policy* for K-SHAP, and we compute the SHAP values using TreeSHAP (Lundberg *et al.*, 2020). We consider two variants of our approach: K-SHAP in which we directly cluster the SHAP values; and K-SHAP ($\mathcal{Z}$) in which we cluster a 2-dimensional embedding $\mathcal{Z}$ of the SHAP values, obtained by applying UMAP (McInnes *et al.*, 2018a). UMAP is a dimensionality reduction technique similar to t-SNE (Van der Maaten and Hinton, 2008) that preserves more the global structure of the data, and requires less computational resources.

[2] In Appendix D.1, we show that K-SHAP achieves similar performance also when the number of clusters k is unknown.

Data We first use the state-of-art multi-agent market simulator ABIDES (Byrd *et al.*, 2019) to simulate synthetic market data. We would like to underline that simulated environment allows to generate state-action pairs with agent IDs, hence, providing the ground truth information for our study. We let market agents realize six distinct trading strategies: [3]

• *Market Making* (Chakraborty and Kearns, 2011) that provides liquidity to the market by placing both buy and sell orders;
• *Fundamental Trading* (Wah *et al.*, 2017) that trades according to a belief of the real stock value;
• (δ_1, δ_2)*-based Momentum Trading* (Byrd *et al.*, 2019) that trades following two momentum signals of the price (i.e., moving averages) computed over the last δ_1 and δ_2 minutes;
• *Noise Trading* (Gode and Sunder, 1993) that places orders randomly;
• *Irrational RL Trading* (Liu *et al.*, 2022) that models a sub-rational human behavior;
• *Rational RL Trading* (Liu *et al.*, 2022) that models an electronic (i.e., rational) trading algorithm.

By combining these strategies, we generate six scenarios:

• **Abides** $\boldsymbol{\pi}^3$ comprises of *Market Making*, $(12, 26)$*-based Momentum*, and *Fundamental* trading strategies, with 127 agents.
• **Abides** $\boldsymbol{\pi}^4$ comprises of *Market Making*, $(12, 26)$*-based Momentum*, *Fundamental*, and *Noise* trading strategies, with 5127 agents.
• **Abides** $\boldsymbol{\pi}^5$ comprises of *Market Making*, $(0.2, 0.4)$*-based Momentum*, $(12, 26)$*-based Momentum*, $(48, 96)$*-based Momentum*, and *Fundamental* trading strategies, with 157 agents.
• **Abides** $\boldsymbol{\pi}^6$ comprises of *Market Making*, $(0.2, 0.4)$*-based Momentum*, $(12, 26)$*-based Momentum*, $(48, 96)$*-based Momentum*, *Fundamental*, and *Noise* trading strategies, with 5157 agents.
• **RL-Agents Bubble** comprises of a *Irrational* and *Rational* RL agent, with a *Bubble* market scenario (Siegel, 2003).
• **RL-Agents Sine** comprises of a *Irrational* and *Rational* RL agent, with a cyclic market scenario.

In the last two scenarios, we model the market and train the RL agents according to the original paper (Liu *et al.*, 2022). In these two scenarios we simulate 11 days of market, while in the first four scenarios we simulate 4 days.

For real market data we consider historical data from NASDAQ stock exchange where agent IDs are not available (NASDAQ, 2022). We consider three stocks (i.e., AVXL, AINV, and ADAP) over 4 trading days from 05th to 8th Jan 2021.

[3] A detailed description of ABIDES agents is provided in Appendix B.1.

Metrics For synthetic data the ground-truth cluster labels are available, and we apply three standard metrics to evaluate the clustering: *purity score*, *Adjusted Rand Index* (ARI) (Hubert and Arabie, 1985) and *Normalized Mutual Information* (NMI) (Vinh *et al.*, 2009). The purity score ranges from 0 to 1, and it evaluates how homogeneous each cluster is (where 1 being a cluster consists of observations from a single strategy). The ARI ranges from -1 to 1, and it represents the *adjusted for chance* version of Rand index, which measures the percentage of correct cluster assignments (where 1 being a perfect clustering and 0 being a random clustering). The NMI ranges from 0 to 1 (where 1 indicated a perfect clustering), and it measures how much information is shared between the clusters and the labels, adjusted by the number of clusters.
For the historical market data where the ground truth is not available we evaluate the *Silhouette Index* (Rousseeuw, 1987) and we introduce a *Utility* metric for the clusters. The *Silhouette Index* measures the similarity of each state-action pair to its own cluster compared to other clusters. It ranges between -1 and 1, where 1 indicate the highest degree of confidence that the observation belongs to a correct cluster. The *Utility* evaluates the learning improvement when we learn a unique policy from all the state-action pairs w.r.t. learning k policies from the identified clusters. Thus, an improvement in *Utility* indicates that each cluster contains homogeneous state-action pairs in terms of strategy, as a model trained on them can predict the next action more easily. An higher *Utility* indicates a better clustering. Both the Silhouette Index and the *Utility* are the appropriate metrics to use as they can validate the cluster analysis in the absence of ground truth.[4]

Benchmarks To the best of our knowledge this is the first approach to tackle policy clustering under anonymous state-action pairs. We first compare our approach against data clustering methods, which represent a natural first attempt to cluster anonymous observations:
• We consider *K-Means* (Hartigan and Wong, 1979) to group the anonymous observations, using the state-action pairs directly (K-Means), or their 2d embedding $\mathcal{Z}$ obtained through UMAP (K-Means ($\mathcal{Z}$)).
• We consider Deep Clustering Network (DCN) (Mukherjee *et al.*, 2019) that jointly optimizes dimensionality reduction and clustering. The dimensionality reduction is accomplished via learning a deep autoencoder. We apply DCN on the anonymous state-action pairs.
• We consider ClusterGAN (Mukherjee *et al.*, 2019) that clusters by back-projecting the data to the latent-space learned by a generative adversarial network (GAN) (Goodfellow *et al.*, 2020). It introduces a mixture of one-hot and continuous variables as latent variables, which retain infor-

[4] We provide a mathematical definition of the metrics in Appendix C.1.

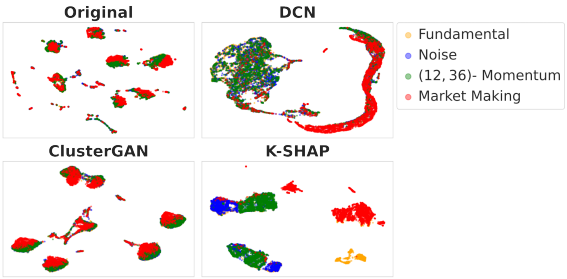

Figure 3: Two-dimensional space visualization of state-action pairs and learned clustering embedding, using UMAP. K-SHAP better reveals the intrinsic clustering structure among data.

mation about the data and reveal some intrinsic clustering structure.

• We also consider HC-MGAN (de Mello *et al.*, 2022), a Hierarchical Clustering approach using Multiple GANs to implicitly learn a latent representation of the data. In particular, this work clusters data by exploiting the fact that each GAN tends to generate data that correlates with a sub-region of the real data distribution.

Finally, we consider Σ-GIRL (Ramponi *et al.*, 2020), a recent work that solves an intention-clustering problem. This work clusters agents based on their goals in an expectation-maximization (EM) fashion, using IRL in the maximization step. It solves a problem similar to ours, but it does not consider anonymous state-action pairs. We consider a modified version called *EM K-Clustering*. Instead of a joint optimization process in which we learn the agent-cluster assignments and the reward functions through IRL, we learn the observation-cluster assignments and the policy functions through IL. We consider a Neural Network (NN) to learn the policies through IL.

4.1. Multi-Agent Synthetic Data

Table 1 investigates the algorithm performance using the simulated market data with ground truth labels. In all the scenarios, our algorithm K-SHAP outperforms existing work. In **Abides** π^3, K-SHAP outperforms other approaches of around 300% in terms of ARI and NMI, while it consistently and significantly shows the best performance in all **Abides** scenarios. In the scenarios with two RL agents, most of the approaches fail to distinguish between the two different strategies. However, when we apply K-SHAP using a 2-dimensional embedding of the SHAP values (K-SHAP ($\mathcal{Z}$)) we obtain encouraging results. It achieves 10 times better performance in terms of ARI and NMI. It is worth noticing that the purity score is less relevant with only two clusters (e.g., if we put all the observations in one cluster, we still obtain a purity score of 0.5).

Figure 3 shows a 2-dimensional visualization of the original state-action pairs against the learned clustering embedding for DCN, ClusterGAN, and K-SHAP. We omit K-Means and EM K-Clustering, as they do not use embedding representations. We consider *Abides* π^4 and the colors represent the ground-truth labels. The figure clearly shows how the original state-action pairs do not retain any natural structure to help the clustering. Moreover, ClusterGAN latent space does not efficiently cluster the observations, while DCN only partially identifies Market Makers observations. Instead, for our algorithm the SHAP values better reveal the intrinsic clustering structure among data, and only Noise Agents, which place random orders, are slightly confused with Momentum Agents.

Dataset	Algorithm	ARI	NMI	Purity
Abides π^3 (Byrd *et al.*, 2019)	K-Means	0.00±0.00	0.00±0.00	0.35±0.00
	K-Means ($\mathcal{Z}$)	0.00±0.00	0.00±0.00	0.35±0.00
	ClusterGAN (Mukherjee *et al.*, 2019)	0.11±0.10	0.14±0.14	0.33±0.25
	DCN (Yang *et al.*, 2017)	0.10±0.09	0.14±0.12	0.50±0.08
	HC-MGAN (de Mello *et al.*, 2022)	0.03±0.01	0.07±0.02	0.42±0.02
	EM K-Clustering	0.13±0.07	0.15±0.11	0.53±0.07
	K-SHAP	**0.50±0.10**	**0.56±0.05**	**0.77±0.03**
	K-SHAP ($\mathcal{Z}$)	0.23±0.00	0.25±0.00	0.60±0.00
Abides π^4 (Byrd *et al.*, 2019)	K-Means	0.00±0.00	0.00±0.00	0.27±0.01
	K-Means ($\mathcal{Z}$)	0.00±0.00	0.00±0.00	0.28±0.00
	ClusterGAN (Mukherjee *et al.*, 2019)	0.08±0.08	0.11±0.12	0.24±0.19
	DCN (Yang *et al.*, 2017)	0.04±0.06	0.06±0.04	0.33±0.07
	HC-MGAN (de Mello *et al.*, 2022)	0.04±0.01	0.07±0.01	0.36±0.02
	EM K-Clustering	0.08±0.04	0.11±0.06	0.43±0.10
	K-SHAP	**0.35±0.07**	**0.52±0.02**	**0.65±0.11**
	K-SHAP ($\mathcal{Z}$)	0.21±0.08	0.32±0.03	0.55±0.03
Abides π^5 (Byrd *et al.*, 2019)	K-Means	0.00±0.00	0.01±0.00	0.23±0.00
	K-Means ($\mathcal{Z}$)	0.01±0.00	0.01±0.00	0.24±0.00
	ClusterGAN (Mukherjee *et al.*, 2019)	0.07±0.06	0.10±0.08	0.22±0.17
	DCN (Yang *et al.*, 2017)	0.08±0.04	0.11±0.05	0.33±0.04
	HC-MGAN (de Mello *et al.*, 2022)	0.04±0.01	0.07±0.00	0.29±0.01
	EM K-Clustering	0.07±0.03	0.12±0.06	0.40±0.12
	K-SHAP	**0.22±0.08**	**0.37±0.06**	**0.51±0.10**
	K-SHAP ($\mathcal{Z}$)	0.20±0.03	0.33±0.01	0.50±0.06
Abides π^6 (Byrd *et al.*, 2019)	K-Means	0.00±0.00	0.01±0.00	0.20±0.01
	K-Means ($\mathcal{Z}$)	0.01±0.00	0.01±0.00	0.21±0.00
	ClusterGAN (Mukherjee *et al.*, 2019)	0.06±0.04	0.10±0.07	0.19±0.15
	DCN (Yang *et al.*, 2017)	0.06±0.03	0.11±0.07	0.29±0.04
	HC-MGAN (de Mello *et al.*, 2022)	0.02±0.00	0.05±0.01	0.23±0.01
	EM K-Clustering	0.07±0.03	0.12±0.04	0.38±0.15
	K-SHAP	**0.15±0.01**	**0.31±0.01**	**0.42±0.01**
	K-SHAP ($\mathcal{Z}$)	0.15±0.02	0.29±0.01	0.42±0.02
RL-Agents Bubble (Liu *et al.*, 2022)	K-Means	0.00±0.00	0.00±0.00	0.58±0.00
	K-Means ($\mathcal{Z}$)	0.00±0.00	0.00±0.00	0.58±0.00
	ClusterGAN (Mukherjee *et al.*, 2019)	0.00±0.00	0.00±0.00	0.37±0.27
	DCN (Yang *et al.*, 2017)	0.00±0.00	0.00±0.00	0.58±0.00
	HC-MGAN (de Mello *et al.*, 2022)	0.00±0.00	0.00±0.00	0.58±0.00
	EM K-Clustering	0.00±0.00	0.00±0.00	0.58±0.00
	K-SHAP	0.00±0.00	0.00±0.00	0.58±0.00
	K-SHAP ($\mathcal{Z}$)	**0.42±0.20**	**0.35±0.15**	**0.81±0.10**
RL-Agents Sine (Liu *et al.*, 2022)	K-Means	0.00±0.00	0.00±0.00	0.56±0.00
	K-Means ($\mathcal{Z}$)	0.00±0.00	0.00±0.00	0.56±0.00
	ClusterGAN (Mukherjee *et al.*, 2019)	0.00±0.00	0.00±0.00	0.39±0.28
	DCN (Yang *et al.*, 2017)	0.00±0.00	0.00±0.00	0.56±0.00
	HC-MGAN (de Mello *et al.*, 2022)	0.00±0.00	0.01±0.00	0.58±0.01
	EM K-Clustering	0.02±0.00	0.01±0.00	0.58±0.00
	K-SHAP	0.02±0.00	0.01±0.00	0.56±0.00
	K-SHAP ($\mathcal{Z}$)	**0.25±0.07**	**0.20±0.06**	**0.75±0.04**

Table 1: Multi-Agent Synthetic Data

4.2. Historical Market Data

Finally, we consider an experiment with historical market data from NASDAQ stock exchange where agent IDs are not available. We define the Utility as the percentage difference between the mean squared error (MSE) obtained by a NN trained over all the state-action pairs, and the MSE obtained by k NNs trained over the identified clusters. Each NN predicts the next action a given the current market state s, and all the NNs have the same architecture.

Stock	Algorithm	Utility	Silhouette
ADAP	K-Means	0.00±0.00	**0.23±0.01**
(NASDAQ, 2022)	K-Means ($\mathcal{Z}$)	0.06±0.01	0.00±0.01
	ClusterGAN (Mukherjee *et al.*, 2019)	0.12±0.11	0.02±0.01
	DCN (Yang *et al.*, 2017)	0.00±0.00	0.00±0.00
	EM K-Clustering	**0.47±0.02**	0.01±0.01
	K-SHAP	0.33±0.00	0.12±0.00
	K-SHAP ($\mathcal{Z}$)	0.21±0.04	0.00±0.01
AINV	K-Means	0.00±0.00	**0.24±0.00**
(NASDAQ, 2022)	K-Means ($\mathcal{Z}$)	0.05±0.02	0.00±0.00
	ClusterGAN (Mukherjee *et al.*, 2019)	0.07±0.06	0.01±0.01
	DCN (Yang *et al.*, 2017)	0.00±0.00	0.00±0.01
	EM K-Clustering	**0.43±0.02**	0.01±0.03
	K-SHAP	0.19±0.06	0.11±0.06
	K-SHAP ($\mathcal{Z}$)	0.10±0.05	0.01±0.01
AVXL	K-Means	0.00±0.01	**0.21±0.00**
(NASDAQ, 2022)	K-Means ($\mathcal{Z}$)	0.05±0.04	0.12±0.04
	ClusterGAN (Mukherjee *et al.*, 2019)	0.13±0.03	0.01±0.01
	DCN (Yang *et al.*, 2017)	0.14±0.04	0.02±0.02
	EM K-Clustering	**0.53±0.04**	0.01±0.02
	K-SHAP	0.36±0.01	0.09±0.00
	K-SHAP ($\mathcal{Z}$)	0.24±0.05	0.02±0.03

Table 2: Historical Market Data

Table 2 shows that *K-Means* always achieves the best Silhouette while the best algorithm for the Utility is *EM K-Clustering*. However, *K-Means* optimizes the clusters for within-cluster sum-of-squares which maximizes the Silhouette; while *EM K-Clustering* optimizes the clusters using k policies in an EM fashion, which indirectly maximizes the Utility metric. Both the approaches easily succeed in these metrics but they fail short in the other metric: *K-Means* has 0 Utility, and *EM K-Clustering* has 0.01 Silhouette. Instead, K-SHAP achieves a good trade-off between Silhouette and Utility, without maximizing any of them directly. In all the scenarios we consider $k = 3$, we refer to Appendix D.3 for more experiments and charts.

4.3. The impact of state features

While K-SHAP is most effective when the latent policies use multiple state features to make decisions, K-SHAP can also distinguish strategies using only few features. Given the local accuracy property (see Section 2.5), the local explanation ϕ_i for a state feature i can be both negative or positive, with different magnitudes according to the impact on the action. Therefore, policies that use the same few features, but in different ways, can still be distinguished; while with null states our approach may fail. To better study the applicability of our approach with few and no features, we created an iterated prisoner's dilemma and a one-shot prison dilemma.

First, we consider a strategy that always "betrays" and one that always "cooperates". K-SHAP achieves $Purity = 0.5$, NMI = 0 and ARI = 0 (i.e., random clusters) when the state is null. It achieves $Purity$=NMI=ARI=1 (i.e., perfect clusters) when the state is provided (i.e., time-step and previous opponent's action). In the latter scenario, with just a two-dimensional state space, the SHAP latent space is meaningful and can be used to distinguish the policies.

As a further example we consider a strategy that always "cooperates" and a strategy that flips his behavior w.r.t. the previous action. When the state is null we have random clusters. When the state is provided (i.e., time-step and previous action) K-SHAP achieves $Purity = 0.75$, $NMI = 0.35$, and $ARI = 0.25$. We introduced this example to highlight that: while the SHAP values $\boldsymbol{\phi}$ can be useful even when the strategies rely on the same few features, the world policy may underfit with such a small state space and overlapping agent strategies, resulting in worse clusters.

We conclude that the number of state features, and how the latent policies use them, is important for both the SHAP values and the world policy.

5. Conclusions and Future Work

We proposed a Policy Clustering algorithm, K-SHAP, that groups anonymous state-action pairs according to the agent strategies. We framed the problem as an IL to learn a compact representation of the agent strategies as a world-policy; we explained such a policy using SHAP; finally, we use the SHAP values to group observations according to the agent behaviors. We shown that K-SHAP consistently outperforms existing work on both synthetic and real market data. As we only access anonymous observations, the current approach works well when the actions are mostly driven by the state features: K-SHAP has not enough information to classify an action driven only by a sequence of previous actions, unless the state retains traces of the agent actions. Similarly, identical state-action pairs cannot be classified with different strategies. As future work, we envisioned to further explore and apply K-SHAP in these scenarios. Please refer to the appendix for more detailed experiments and discussions on our work impact.

Disclaimer

This paper was prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates ("J.P. Morgan"), and is not a product of the Research Department of J.P. Morgan. J.P. Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

Acknowledgements

Images are in part designed by Macrovector / Freepik.

References

Selim Amrouni, Aymeric Moulin, Jared Vann, Svitlana Vyetrenko, Tucker Balch, and Manuela Veloso. Abides-gym: gym environments for multi-agent discrete event simulation and application to financial markets. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

Zoe Ashwood, Aditi Jha, and Jonathan W Pillow. Dynamic inverse reinforcement learning for characterizing animal behavior. In *Advances in Neural Information Processing Systems*.

Monica Babes, Vukosi Marivate, Kaushik Subramanian, and Michael L Littman. Apprenticeship learning about multiple intentions. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 897–904, 2011.

Joseph B Bak-Coleman, Mark Alfano, Wolfram Barfuss, Carl T Bergstrom, Miguel A Centeno, Iain D Couzin, Jonathan F Donges, Mirta Galesic, Andrew S Gersick, Jennifer Jacquet, et al. Stewardship of global collective behavior. *Proceedings of the National Academy of Sciences*, 118(27):e2025764118, 2021.

Jean-Philippe Bouchaud, Julius Bonart, Jonathan Donier, and Martin Gould. *Trades, quotes and prices: financial markets under the microscope*. Cambridge University Press, 2018.

David Byrd, Maria Hybinette, and Tucker Hybinette Balch. Abides: Towards high-fidelity market simulation for ai research. *arXiv preprint arXiv:1904.12066*, 2019.

Haoyang Cao, Samuel Cohen, and Lukasz Szpruch. Identifiability in inverse reinforcement learning. *Advances in Neural Information Processing Systems*, 34:12362–12373, 2021.

Tanmoy Chakraborty and Michael Kearns. Market making and mean reversion. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 307–314, 2011.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. *Advances in neural information processing systems*, 17, 2004.

Frank Cichos, Kristian Gustavsson, Bernhard Mehlig, and Giovanni Volpe. Machine learning for active matter. *Nature Machine Intelligence*, 2(2):94–103, 2020.

Andrea Coletta, Matteo Prata, Michele Conti, Emanuele Mercanti, Novella Bartolini, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. Towards realistic market simulations: a generative adversarial networks approach. In *Proceedings of the Second ACM International Conference on AI in Finance*, pages 1–9, 2021.

Andrea Coletta, Aymeric Moulin, Svitlana Vyetrenko, and Tucker Balch. Learning to simulate realistic limit order book markets from data as a world agent. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 428–436, 2022.

Daniel PM de Mello, Renato M Assunçao, and Fabricio Murai. Top-down deep clustering with multi-generator gans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7770–7778, 2022.

Kshama Dwarakanath, Svitlana Vyetrenko, and Tucker Balch. Equitable marketplace mechanism design. In *Proceedings of the Third ACM International Conference on AI in Finance*, pages 232–239, 2022.

Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Omer Rana, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, et al. Explainable ai (xai): core ideas, techniques and solutions. *ACM Computing Surveys (CSUR)*, 2022.

Adam Feldman and Tucker Balch. Modeling honey bee behavior for recognition using human trainable models. In *Modeling Other Agents from Observations (Workshop at AAMAS)*, 2004.

Justin Fu, Andrea Tacchetti, Julien Perolat, and Yoram Bachrach. Evaluating strategic structures in multi-agent inverse reinforcement learning. *Journal of Artificial Intelligence Research*, 71:925–951, 2021.

Dhananjay K Gode and Shyam Sunder. Allocative efficiency of markets with zero-intelligence traders: Market as a partial substitute for individual rationality. *Journal of political economy*, 101(1):119–137, 1993.

Robert L Goldstone and Marco A Janssen. Computational models of collective behavior. *Trends in cognitive sciences*, 9(9):424–430, 2005.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. *Advances in neural information processing systems*, 31, 2018.

Björn Hagströmer and Lars Nordén. The diversity of high-frequency traders. *Journal of Financial Markets*, 16(4):741–770, 2013.

John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Zheheng Jiang, Danny Crookes, Brian D Green, Yunfeng Zhao, Haiping Ma, Ling Li, Shengping Zhang, Dacheng Tao, and Huiyu Zhou. Context-aware mouse behavior recognition using hidden markov models. *IEEE Transactions on Image Processing*, 28(3):1133–1148, 2018.

Andrei Kirilenko, Albert S Kyle, Mehrdad Samadi, and Tugkan Tuzun. The flash crash: High-frequency trading in an electronic market. *The Journal of Finance*, 72(3):967–998, 2017.

Gregory D Koblentz. Chemical-weapon use in syria: atrocities, attribution, and accountability. *The Nonproliferation Review*, 26(5-6):575–598, 2019.

Blake LeBaron. Agent-based computational finance. *Handbook of computational economics*, 2:1187–1233, 2006.

Guangyu Li, Bo Jiang, Hao Zhu, Zhengping Che, and Yan Liu. Generative attention networks for multi-agent behavioral modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7195–7202, 2020.

Penghang Liu, Kshama Dwarakanath, and Svitlana S Vyetrenko. Biased or limited: Modeling sub-rational human investors in financial markets. *arXiv preprint arXiv:2210.08569*, 2022.

Luca Luceri, Silvia Giordano, and Emilio Ferrara. Detecting troll behavior via inverse reinforcement learning: A case study of russian trolls in the 2016 us election. In *Proceedings of the international AAAI conference on web and social media*, volume 14, pages 417–427, 2020.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

Scott M Lundberg, Gabriel G Erion, and Su-In Lee. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*, 2018.

Scott M Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature machine intelligence*, 2(1):56–67, 2020.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Leland McInnes, John Healy, Nathaniel Saul, and Lukas Grossberger. Umap: Uniform manifold approximation and projection. *The Journal of Open Source Software*, 3(29):861, 2018.

Sudipto Mukherjee, Himanshu Asnani, Eugene Lin, and Sreeram Kannan. Clustergan: Latent space clustering in generative adversarial networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4610–4617, 2019.

NASDAQ. Nasdaq total view, 2022.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Giorgia Ramponi, Amarildo Likmeta, Alberto Maria Metelli, Andrea Tirinzoni, and Marcello Restelli. Truly batch model-free inverse reinforcement learning about multiple intentions. In *International Conference on Artificial Intelligence and Statistics*, pages 2359–2369. PMLR, 2020.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Why should i trust you? explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

Thomas Rid and Ben Buchanan. Attributing cyber attacks. *Journal of Strategic Studies*, 38(1-2):4–37, 2015.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Jeremy J Siegel. What is an asset price bubble? an operational definition. *European financial management*, 9(1):11–24, 2003.

Jiaming Song, Hongyu Ren, Dorsa Sadigh, and Stefano Ermon. Multi-agent generative adversarial imitation learning. *Advances in neural information processing systems*, 31, 2018.

Simon Suo, Sebastian Regalado, Sergio Casas, and Raquel Urtasun. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10400–10409, 2021.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning*, pages 1073–1080, 2009.

Svitlana Vyetrenko, David Byrd, Nick Petosa, Mahmoud Mahfouz, Danial Dervovic, Manuela Veloso, and Tucker Balch. Get real: Realism metrics for robust limit order book market simulations. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

Elaine Wah, Mason Wright, and Michael P Wellman. Welfare effects of market making in continuous double auctions. *Journal of Artificial Intelligence Research*, 59:613–650, 2017.

Xintong Wang, Christopher Hoang, Yevgeniy Vorobeychik, and Michael P. Wellman. Spoofing the limit order book: A strategic agent-based analysis. *Games*, 12(2), 2021.

Shoichiro Yamaguchi, Honda Naoki, Muneki Ikeda, Yuki Tsukada, Shunji Nakano, Ikue Mori, and Shin Ishii. Identification of animal behavioral strategies by inverse reinforcement learning. *PLoS computational biology*, 14(5):e1006122, 2018.

Steve Yang, Mark Paddrik, Roy Hayes, Andrew Todd, Andrei Kirilenko, Peter Beling, and William Scherer. Behavior based learning in identifying high frequency trading strategies. In *2012 IEEE Conference on Computational Intelligence for Financial Engineering & Economics (CIFEr)*, pages 1–8. IEEE, 2012.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pages 3861–3870. PMLR, 2017.

A. Broader Impact

Our paper introduces a first approach to understand agent strategies and their interaction in anonymized systems. In general, we believe this is an important research area to develop more accurate computational models of complex systems (Goldstone and Janssen, 2005), resulting in a better understanding of agent behaviors needed to design more suitable solutions (Bak-Coleman *et al.*, 2021).

For example, in financial markets, identifying archetypal behavioral strategies allows to construct market simulation environments that can reproduce important structural properties of the market (LeBaron, 2006). Experimentation with counterfactual scenarios in such simulators has been documented to provide significant benefits for the financial community. For instance:

- Agent-based simulation that was constructed with similar archetypal agents was used to replay May 2010 flash crash scenario – hence, promoting the understanding of the flash crash and allowing to subsequently design the policies that prevent flash crashes in the future (Kirilenko *et al.*, 2017).
- Another benefit of our approach to financial systems is the investigation of market mechanisms that can make markets more fair – it was demonstrated in simulated market environments that stock exchanges can implement the dynamic fee policy in order to enable equitability of outcomes to market participants (Dwarakanath *et al.*, 2022).
- Additionally, evaluating trading strategies or assumptions against poorly calibrated agent-based models can lead to harmful and misleading conclusions (e.g., a severe market crash in 1987 causing $1.71 trillion losses is attributed to the prevalence of simplistic market models) (Bouchaud *et al.*, 2018) - hence, it is imperative to build reliable market simulation models for the overall market stability.

B. Details of the Datasets

Here we introduce experimental details that do not fit into the main body of the paper.

In our experiments, we consider a stock market in which multiple agents (i.e., traders) interact by selling and buying shares. We consider a synthetic simulated market, which serves as ground truth, and historical market data from NASDAQ stock exchange (NASDAQ, 2022). Each state-action pair is generated by a trader in the market. The state represents the recent and ongoing stock market state (e.g., the price, volume and volatility of the stock) and the action is the agent trade (e.g., buy/sell a given number of shares at price x). In the following, we discuss in detail how we construct the synthetic datasets, and we summarize the main statistics of both synthetic and historical market data.

B.1. Multi-Agent Synthetic data

All the synthetic datasets have been generated using a state-of-art multi-agent simulator called ABIDES (Byrd *et al.*, 2019), and its OpenAI Gym extension called ABIDES-gym (Amrouni *et al.*, 2021). The simulator is written in Python3, and it is publicly available at https://github.com/jpmorganchase/abides-jpmc-public. It is a high-fidelity multi-agent market simulation used by practitioners and researchers to generate synthetic markets. The market is generated by simulating the interactions between a given set of agents, for which we can define the number, type, and strategy to simulate different markets. More details on the simulator can be found in the original ABIDES paper (Byrd *et al.*, 2019).

Agent Strategies Here we describe the agent strategies in detail:

- *Market Making* agents provide liquidity to the market by both buying and selling shares while making a profit by keeping a low net position. We use the implementation provided in the public repository of ABIDES, (i.e., `adaptive_market_maker_agent.py`) that follows the Chakraborty-Kearns 'ladder' market-making strategy (Chakraborty and Kearns, 2011), wherein the size of orders placed at each level is set as a fraction of measured transacted volume in the previous time period.
- *Noise* agents belong to the class of Zero Intelligence (ZI) agents introduced in (Gode and Sunder, 1993) to model agents that do not base their trading decisions on the knowledge of market microstructure. *Noise* agents wake up once in the trading day, and place one order randomly. Both their order direction $d \in \{buy, sell\}$, and volume $v \in [1, 100]$ are randomly sampled, while the order price is the best available on the market (i.e., near touch). We use the implementation provided in the public repository of ABIDES, i.e., `noise_agent.py`.

- *Fundamental Trading* agents belong to the class of ZI agents, but have access to an exogenous fundamental value of the stock, which represents the agent's understanding of the outside world (e.g., assets and earnings of the company). These agents trade the stock according this estimated exogenous fundamental value. They believe that if the stock is overpriced w.r.t. to the fundamental value, the price will go down, and vice versa. Therefore, they sell in the first case and buy in the other one. All the agents have a noisy observation of a simulated fundamental value of the stock, which they use to generate orders. They randomly sample the order volume $v \in [1, 100]$, while the price is the best available for 90% of the orders (i.e., near touch). For 10% of the orders, they use a more aggressive price close to the far touch. We use the implementation provided in the public repository of ABIDES, i.e., `value_agent.py`.

- (δ_1, δ_2)*-based Momentum Trading* agents trade according to a price momentum indicator of the stock. These agents consider two moving averages (MAs) of the stock price. The *short* MA is computed using the price values in the previous δ_1 minutes, while the *long* MA considers δ_2 minutes. This agent attempts to exploit extreme short-term price moves, by playing a *buy* order when the *short* MA $\delta_1 \geq$ *long* MA δ_2, and a *sell* order otherwise. The order volume $v \in [1, 100]$ is randomly sampled, and the price is the best available (i.e., near touch). We extended the implementation provided in the public repository of ABIDES, (i.e., `momentum_agent.py`) by changing the MA windows.

- *Irrational RL Trading* agents model a sub-rational human behavior. We train these RL agents according to the original paper (Liu *et al.*, 2022), in which the authors model a myopic human investor by decreasing the discount factor γ in the Bellman equation (Sutton and Barto, 2018). Therefore, as $\gamma \to 0$ the RL agent becomes more myopic and trades to maximize only a one-step reward. In the experiments the Irrational RL agents use $\gamma = 0.01$.

- *Rational RL Trading* agents (Liu *et al.*, 2022) model an electronic (i.e., rational) trading algorithm. We train these RL agents according to the original paper (Liu *et al.*, 2022), in which the authors increase the discount γ to have a fully rational agent that considers both short-term and long-term rewards. In the experiments the Rational RL agents use $\gamma = 0.99$.

Market Scenarios We combine the agent strategies to generate six different market scenarios.

• In the first four scenarios (Abides π^3, Abides π^4, Abides π^5, and Abides π^6) we simulate 4 days of synthetic market, considering for each strategy the following number of agents: 5000 Noise agents; 110 Fundamental Agents; 2 Market Making Agents; 15 (0.2,0.4)-based Momentum Agents; 15 (12,26)-based Momentum Agents; and 15 (48,96)-based Momentum Agents. The agents trade a synthetic stock priced at around 100\$. The market is simulated with nanosecond time resolution, and we consider 29 state features and 3 features for the agent actions (i.e., order depth, volume, and direction). The detailed statistics for these datasets are reported in Table 5 for Abides π^3; Table 6 for Abides π^4; Table 7 for Abides π^5 and Table 8 for Abides π^6. We use the implementation provided in the public repository of ABIDES (i.e., `rmsc04.py`) to define the agent interactions.

• The last two scenarios consider RL-based agents, which are simulated according to the two original paper (Liu *et al.*, 2022). In particular we consider a *Bubble* (Siegel, 2003) market scenario, and a market scenario in which the price follows a sine wave, with the same starting and closing price. In these scenarios, the market is simulated every minute, and the agents have 9 different actions modeled as an integer $a \in [-4, 4]$. The action $a \in \{-4, -3, -2, -1\}$ represents a buy order of size 2, where the value of a represents the order price w.r.t. the mid-price. The action $a = 0$ represents the *HOLD* action. The action $a \in \{1, 2, 3, 4\}$ represents a sell order of size 2, where the value of a represents the order price w.r.t. the mid-price. In these scenarios, we simulate 11 days of market data, we consider 20 state features and 1 integer feature for the agent actions. In Table 12 and Table 13 we summarize the dataset properties for the *Bubble* and *Sine* market scenarios, respectively.

B.2. Historical Market Data

Finally, we consider real market data from NASDAQ stock exchange (NASDAQ, 2022). In particular, we use historical market data that includes all the orders submitted to the market, without any agent identifiers. We consider three stocks (i.e., AVXL, AINV, and ADAP) over 4 days from 05th to 8th Jan 2021. The historical data contains anonymous state-action pairs at nanosecond time resolution, and we consider the same 29 state features and 3 action features as in **Abides** synthetic dataset. Table 9, Table 10, and Table 11, summarize the dataset properties for AINV, AVXL, and ADAP, respectively.

C. Algorithm and Benchmark details

• We consider *K-Means* (Hartigan and Wong, 1979) to group the anonymous observations, using the state-action pairs directly (K-Means), or their 2d embedding $\mathcal{Z}$ obtained through UMAP (K-Means ($\mathcal{Z}$)). We use K-Means implementation provided by scikit-learn (Pedregosa *et al.*, 2011), and the official UMAP implementation (McInnes *et al.*, 2018b).

• We consider Deep Clustering Network (DCN) (Mukherjee *et al.*, 2019) that jointly optimizes dimensionality reduction and clustering. The dimensionality reduction is accomplished via learning a deep autoencoder. We apply DCN on the anonymous state-action pairs: we feed both the states and the actions to the deep autoencoder that learns their encoding and reconstruction. We consider the official implementation of the paper available at https://github.com/boyangumn/DCN-New.

• We consider ClusterGAN (Mukherjee *et al.*, 2019) that clusters by back-projecting the data to the latent-space. It introduces a mixture of one-hot and continuous variables as latent variables, which retain information about the data and reveal some intrinsic clustering structure. We apply ClusterGAN to the anonymous state-action pairs. We consider the official implementation of the paper available at https://github.com/sudiptodip15/ClusterGAN.

• Finally, we consider a modified version of Σ-GIRL (Ramponi *et al.*, 2020). This work solves the intent-clustering problem in an expectation-maximization (EM) fashion, using IRL in the maximization step. Instead of a joint optimization process in which we learn the agent-cluster assignments and the reward functions through IRL, we learn the observation-cluster assignments and the policy functions through IL. In particular, we learn to assign the anonymous observations to k clusters while learning k policies through IL. We consider a Neural Network (NN) to learn the policies through IL. The NN architecture is described in Appendix C. We modified the official implementation of the paper available at https://github.com/gioramponi/sigma-girl-MIIRL.

Neural Network Architecture Here we briefly describe the architecture of the NN used in the experiments: to solve the IL task in *EM K-Clustering*; to model the world-policy in Appendix D.2; and to compute the Utility score C.1. We consider a feedforward Neural Network with 2 linear hidden layers with *Leaky ReLU* activation function, and respectively 64 and 32 neurons. After Each hidden layer we consider a 0.1-dropout layer.

World-Policy Architecture For the Random Forest world-policy we use the implementation provided by scikit-learn (Pedregosa *et al.*, 2011), where we fix the number of trees to 100 and we use mean squared error as objective. For UMAP we use the official implementation (McInnes *et al.*, 2018b), and we fix the number of neighbors observations to 15.

C.1. Metrics

• The *purity score* ranges from 0 to 1, and it evaluates how homogeneous each cluster is (where 1 being a cluster consists of observations from a single strategy). The purity is defined as follows:

$$\text{purity}(\Omega, C) = \frac{1}{N} \sum_k \max_j |\Omega_k \cap C_j|$$

where N is the number of observations, k and j are the number of clusters and strategies (ground truth), respectively. We denote with Ω the set of identified clusters, while C represents the ground truth clusters.

• The *Adjusted Rand Index* (ARI) (Hubert and Arabie, 1985) ranges from -1 to 1, and it represents the *adjusted for chance* version of Rand index, which measures the percentage of correct cluster assignments (where 1 being a perfect clustering and 0 being a random clustering). We use the ARI implementation provided by scikit-learn (Pedregosa *et al.*, 2011), and we refer to the original paper for further details (Hubert and Arabie, 1985).

• The *Normalized Mutual Information* (NMI) (Vinh *et al.*, 2009) ranges from 0 to 1 (where 1 indicated a perfect clustering), and it measures how much information is shared between the clusters and the labels, adjusted by the number of clusters. We use NMI implementation provided by scikit-learn (Pedregosa *et al.*, 2011), and we refer to the original paper for further details (Vinh *et al.*, 2009).

• The *Silhouette Index* (SI) (Rousseeuw, 1987) measures the similarity of each state-action pair to its own cluster compared to other clusters. It ranges between -1 and 1, where 1 indicate the highest degree of confidence that the observation belongs to a correct cluster. Let b_i be the mean euclidean distance between a sample i and all other points in the same cluster; and c_i

the mean distance between the sample i and all the other points in the nearest cluster, then the SI is computed as follows:

$$SI = \frac{1}{N} \sum_{i=0}^{N} \frac{c_i - b_i}{max(b_i, c_i)}$$

• The *Utility* evaluates the learning improvement when we learn a unique policy from all the state-action pairs w.r.t. learning k policies from the identified clusters. Thus, an improvement in *Utility* indicates that each cluster contains homogeneous state-action pairs in terms of strategy, as a model trained on them can predict the next action more easily. A higher utility indicates a better clustering. In particular, to evaluate the *Utility* we compare the performance of a NN trained over all the state-action pairs w.r.t. to k NNs trained on the identified clusters. Let $\epsilon_i = ||\pi(s_i) - a_i||^2$ be the error for an observation i when the action a_i is predicted by a learned policy π. We denote with $\hat{\epsilon_i}$ the error when i is predicted by a unique policy $\hat{\pi}$ trained on all the state-action pairs, while we denote with $\bar{\epsilon_i}$ the error when i is predicted by a policy $\bar{\pi}$ trained on the cluster where i belongs. Thus, the *Utility* $\mathcal{U}$ can be defined as :

$$\mathcal{U} = \frac{1}{N} \sum_{i=0}^{N} \frac{\hat{\epsilon_i} - \bar{\epsilon_i}}{\hat{\epsilon_i}}$$

D. Additional Experiments

Here we present additional experiments that do not fit into the main body of the paper.

D.1. Unknown number of clusters k

In this section, we evaluate the ability of K-SHAP to group the anonymous observations when the number of clusters k is unknown. In such a case we adopt the classic Elbow method to optimize k according to the *distortion* of the clusters in the SHAP values space. The distortion measures the mean distance of each point to its assigned cluster.

Figure 4 shows the performance of K-SHAP when the ground truth number of clusters k is given (left picture) and when k is unknown (right picture). We consider only synthetic data as we do not know the ground truth k for real market data. The picture shows that even if the performance is slightly inferior, K-SHAP maintains a similar trend and satisfactory performance when k is unknown.

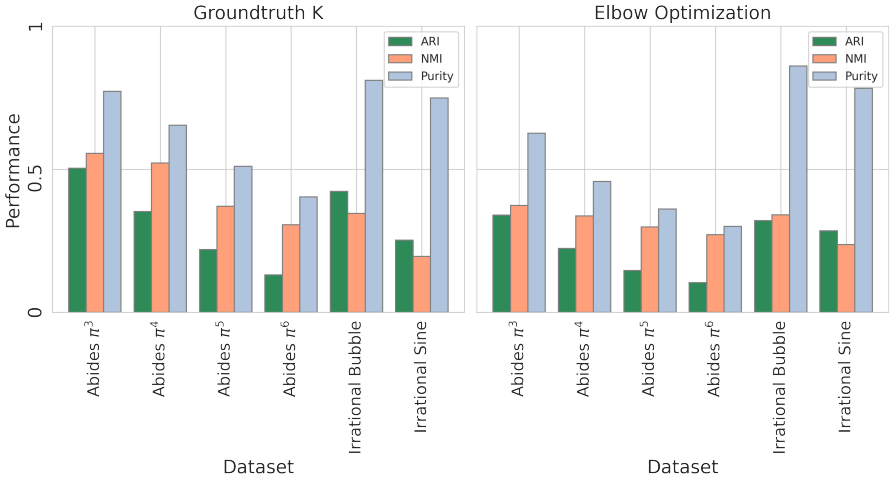

Figure 4: K-SHAP - Elbow Optimization

D.2. Contributions of the world-policy model

We now evaluate K-SHAP using three different models for the world-policy:

• Random Forest - we use the implementation provided by scikit-learn (Pedregosa *et al.*, 2011), where we fix the number of trees to 100 and we use mean squared error as objective.

• XGBoost (Chen and Guestrin, 2016) - we use the implementation provided at https://github.com/dmlc/xgboost. We fix the max depth to 100, and we use mean squared error as objective.

• Neural Network - we implement a feedforward NN with 2-hidden layers using PyTorch. The architecture details are provided in Appendix C.

For each world-policy we also consider a variant that uses a 2d embedding $\mathcal{Z}$ of the SHAP values, obtained through UMAP.

Table 3 and Table 4 show the results for the different world-policies, for both synthetic and real data. The results on synthetic data confirm that a Random Forest world-policy better suits the anonymous market observations and provides better results. However, also XGBoost achieves comparable performance, while the NN is not able to fully capture the structure of the data, achieving lower performance. The results are similar on the real historical data, even if XGBoost and NN outperform Random Forest for some stocks. In general, the higher the model accuracy in predicting the next action, given the input state, the better the SHAP values and related clustering.

Dataset	World-Policy	ARI	NMI	Purity
Abides π^3 (Byrd *et al.*, 2019)	Random Forest	**0.50±0.10**	**0.56±0.05**	**0.77±0.03**
	Random Forest ($\mathcal{Z}$)	0.23±0.00	0.25±0.00	0.60±0.00
	NN	0.03±0.02	0.05±0.01	0.42±0.02
	NN ($\mathcal{Z}$)	0.20±0.01	0.23±0.01	0.58±0.01
	XGBoost	0.48±0.01	0.55±0.01	0.76±0.00
	XGBoost ($\mathcal{Z}$)	0.34±0.06	0.41±0.07	0.69±0.06
Abides π^4 (Byrd *et al.*, 2019)	Random Forest	**0.35±0.07**	**0.52±0.02**	**0.65±0.11**
	Random Forest ($\mathcal{Z}$)	0.21±0.08	0.32±0.03	0.55±0.03
	NN	0.04±0.01	0.06±0.01	0.35±0.01
	NN ($\mathcal{Z}$)	0.16±0.04	0.19±0.05	0.44±0.03
	XGBoost	0.29±0.01	0.44±0.05	0.57±0.01
	XGBoost ($\mathcal{Z}$)	0.24±0.01	0.35±0.01	0.56±0.00
Abides π^5 (Byrd *et al.*, 2019)	Random Forest	**0.22±0.08**	**0.37±0.06**	**0.51±0.10**
	Random Forest ($\mathcal{Z}$)	0.20±0.03	0.33±0.01	0.50±0.06
	NN	0.06±0.05	0.09±0.08	0.30±0.04
	NN ($\mathcal{Z}$)	0.15±0.01	0.20±0.01	0.37±0.01
	XGBoost	0.21±0.01	0.36±0.02	0.48±0.01
	XGBoost ($\mathcal{Z}$)	0.20±0.01	0.32±0.02	0.48±0.01
Abides π^6 (Byrd *et al.*, 2019)	Random Forest	**0.15±0.01**	**0.31±0.01**	**0.42±0.01**
	Random Forest ($\mathcal{Z}$)	0.15±0.02	0.29±0.01	0.42±0.02
	NN	0.12±0.00	0.20±0.01	0.31±0.00
	NN ($\mathcal{Z}$)	0.13±0.01	0.17±0.01	0.32±0.01
	XGBoost	0.15±0.01	0.31±0.02	0.41±0.01
	XGBoost ($\mathcal{Z}$)	0.16±0.00	0.26±0.00	0.40±0.01
RL-Agents Bubble (Liu *et al.*, 2022)	Random Forest	0.00±0.00	0.00±0.00	0.58±0.00
	Random Forest ($\mathcal{Z}$)	**0.42±0.20**	**0.35±0.15**	**0.81±0.10**
	NN	0.00±0.00	0.00±0.00	0.58±0.00
	NN ($\mathcal{Z}$)	0.09±0.07	0.08±0.07	0.64±0.06
	XGBoost	0.00±0.00	0.00±0.00	0.58±0.00
	XGBoost ($\mathcal{Z}$)	0.00±0.00	0.00±0.00	0.58±0.00
RL-Agents Sine (Liu *et al.*, 2022)	Random Forest	0.02±0.00	0.01±0.00	0.56±0.00
	Random Forest ($\mathcal{Z}$)	**0.25±0.07**	**0.20±0.06**	**0.75±0.04**
	NN	0.00±0.00	0.00±0.00	0.56±0.00
	NN ($\mathcal{Z}$)	0.05±0.00	0.04±0.00	0.62±0.00
	XGBoost	0.02±0.00	0.01±0.00	0.56±0.00
	XGBoost ($\mathcal{Z}$)	0.01±0.00	0.00±0.00	0.56±0.00

Table 3: World Policies - Multi-Agent Synthetic Data

Stock	World-Policy	Utility	Silhouette
ADAP (NASDAQ, 2022)	Random Forest	**0.33±0.00**	0.12±0.00
	Random Forest ($\mathcal{Z}$)	0.21±0.04	0.00±0.01
	NN	0.01±0.00	**0.22±0.01**
	NN ($\mathcal{Z}$)	0.00±0.00	0.01±0.01
	XGBoost	0.33±0.01	0.11±0.00
	XGBoost ($\mathcal{Z}$)	0.28±0.05	0.01±0.01
AINV (NASDAQ, 2022)	Random Forest	0.19±0.06	**0.11±0.06**
	Random Forest ($\mathcal{Z}$)	0.10±0.05	0.01±0.01
	NN	0.00±0.00	0.06±0.02
	NN ($\mathcal{Z}$)	0.00±0.00	0.03±0.05
	XGBoost	**0.28±0.00**	0.09±0.00
	XGBoost ($\mathcal{Z}$)	0.21±0.02	0.01±0.01
AVXL (NASDAQ, 2022)	Random Forest	**0.36±0.01**	**0.09±0.00**
	Random Forest ($\mathcal{Z}$)	0.28±0.06	0.04±0.05
	NN	0.00±0.00	0.05±0.00
	NN ($\mathcal{Z}$)	0.00±0.00	0.05±0.06
	XGBoost	**0.36±0.01**	0.08±0.00
	XGBoost ($\mathcal{Z}$)	0.36±0.02	0.01±0.01

Table 4: World Policies - Historical Market Data

D.3. Historical market data at varying of number of clusters k

In this section, we evaluate the *Utility* and *Silhouette* when we vary the number of clusters k from 2 to 6. We use historical market data for AVXL stock. For clarity of the presentation, we remove from the picture the variants of K-Means and K-SHAP that use a dimensionality reduction technique, namely K-Means ($\mathcal{Z}$) and K-SHAP ($\mathcal{Z}$).

Figure 5 confirms the trend shown in the paper, and K-SHAP still represents a good trade-off between Utility and Silhouette. For $k = 2$ K-SHAP groups the state-action pairs mostly into buy and sell orders. These two clusters improve the Silhouette, as the orders in each cluster are more homogeneous, but the Utility is close to 0. In fact, a cluster containing only buy (or sell) observations does not highlight any particular strategy.

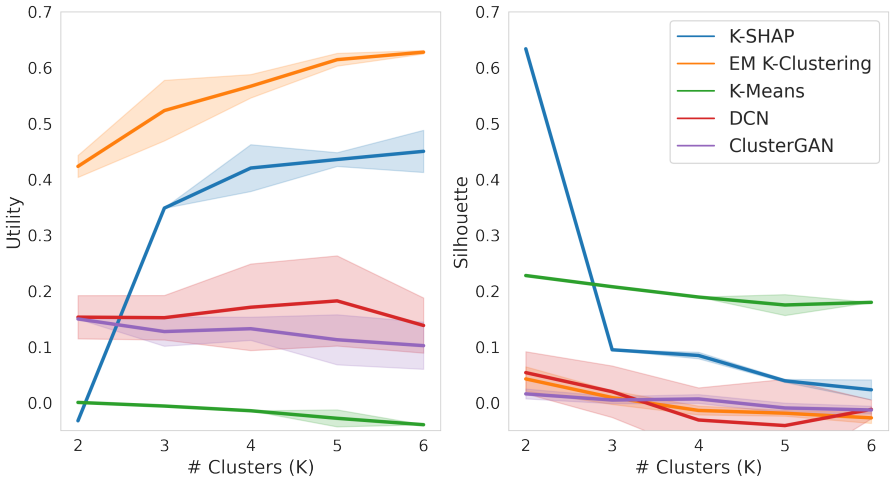

Figure 5: Historical Market data AVXL at varying of nr. clusters k.

Feature	mean	min	max	Type
Spread	2.346	1.000	67.000	Int
Vol. Imbalance Lev.1	0.500	0.001	0.999	Float
Vol. Imbalance Lev.2	0.507	0.002	0.998	Float
Vol. Imbalance Lev.5	0.513	0.007	0.997	Float
Exec. Vol. Imbalance 1min	0.462	0.000	1.000	Float
Exec. Vol. Imbalance 5min	0.463	0.187	0.820	Float
Exec. Vol. Imbalance 12min	0.463	0.255	0.765	Float
Exec. Vol. Imbalance 26min	0.463	0.289	0.725	Float
Price return 1min	-0.000	-0.002	0.002	Float
Price return 5min	-0.000	-0.003	0.004	Float
Price return 12min	-0.000	-0.003	0.006	Float
Price return 26min	-0.000	-0.005	0.007	Float
Price MA 12s	99.966	99.547	100.821	Float
Price MA 26s	99.966	99.532	100.822	Float
Price MA 60s	99.966	99.562	100.821	Float
Price MA 5min	99.966	99.586	100.814	Float
Price MA 12min	99.966	99.599	100.804	Float
Price MA 26min	99.967	99.644	100.793	Float
Price MA 48min	99.970	99.759	100.754	Float
Price MA 1h36min	99.976	99.832	100.567	Float
Spread MA 12s	1.599	1.000	49.083	Float
Spread MA 26s	1.594	1.000	57.192	Float
Spread MA 60s	1.595	1.000	39.950	Float
Spread MA 5min	1.601	1.000	21.200	Float
Spread MA 12min	1.606	1.000	13.250	Float
Spread MA 26min	1.611	1.000	7.885	Float
Price MA Diff 12s-26s	0.501	0.000	1.000	Float
Price MA Diff 12min-26min	0.392	0.000	1.000	Float
Price MA Diff 48min-1h36min	0.395	0.000	1.000	Float
Order Size	41.634	1.000	100.000	Int
Order depth	9.217	-23.000	91.000	Int
Order Direction	0.495	0.000	1.000	Bin

Table 5: Summary and Description of Abides π^3

Feature	mean	min	max	Type
Spread	2.647	1.000	67.000	Int
Vol. Imbalance Lev.1	0.503	0.001	0.999	Float
Vol. Imbalance Lev.2	0.511	0.002	0.998	Float
Vol. Imbalance Lev.5	0.517	0.007	0.997	Float
Exec. Vol. Imbalance 1min	0.464	0.000	1.000	Float
Exec. Vol. Imbalance 5min	0.464	0.187	0.820	Float
Exec. Vol. Imbalance 12min	0.464	0.255	0.765	Float
Exec. Vol. Imbalance 26min	0.463	0.289	0.725	Float
Price return 1min	-0.000	-0.003	0.002	Float
Price return 5min	0.000	-0.004	0.004	Float
Price return 12min	0.000	-0.004	0.006	Float
Price return 26min	0.000	-0.005	0.007	Float
Price MA 12s	99.970	99.534	100.821	Float
Price MA 26s	99.970	99.527	100.822	Float
Price MA 60s	99.970	99.562	100.821	Float
Price MA 5min	99.970	99.586	100.814	Float
Price MA 12min	99.970	99.599	100.804	Float
Price MA 26min	99.969	99.644	100.793	Float
Price MA 48min	99.971	99.759	100.754	Float
Price MA 1h36min	99.978	99.832	100.567	Float
Spread MA 12s	1.638	1.000	49.083	Float
Spread MA 26s	1.627	1.000	57.192	Float
Spread MA 60s	1.626	1.000	39.950	Float
Spread MA 5min	1.632	1.000	21.200	Float
Spread MA 12min	1.641	1.000	13.250	Float
Spread MA 26min	1.661	1.000	7.885	Float
Price MA Diff 12s-26s	0.512	0.000	1.000	Float
Price MA Diff 12min-26min	0.406	0.000	1.000	Float
Price MA Diff 48min-1h36min	0.395	0.000	1.000	Float
Order Size	43.883	1.000	100.000	Int
Order depth	6.441	-23.000	91.000	Int
Order Direction	0.502	0.000	1.000	Bin

Table 6: Summary and Description of Abides π^4

Feature	mean	min	max	Type
Spread	2.779	1.000	67.000	Int
Vol. Imbalance Lev.1	0.507	0.001	0.999	Float
Vol. Imbalance Lev.2	0.509	0.004	0.998	Float
Vol. Imbalance Lev.5	0.515	0.005	0.997	Float
Exec. Vol. Imbalance 1min	0.464	0.002	0.988	Float
Exec. Vol. Imbalance 5min	0.462	0.187	0.819	Float
Exec. Vol. Imbalance 12min	0.463	0.255	0.765	Float
Exec. Vol. Imbalance 26min	0.461	0.290	0.725	Float
Price return 1min	-0.000	-0.002	0.002	Float
Price return 5min	-0.000	-0.003	0.004	Float
Price return 12min	-0.000	-0.003	0.006	Float
Price return 26min	-0.000	-0.005	0.007	Float
Price MA 12s	99.964	99.564	100.821	Float
Price MA 26s	99.964	99.547	100.822	Float
Price MA 60s	99.964	99.577	100.821	Float
Price MA 5min	99.964	99.586	100.813	Float
Price MA 12min	99.964	99.599	100.804	Float
Price MA 26min	99.965	99.644	100.793	Float
Price MA 48min	99.968	99.759	100.754	Float
Price MA 1h36min	99.975	99.832	100.567	Float
Spread MA 12s	1.617	1.000	46.250	Float
Spread MA 26s	1.593	1.000	57.192	Float
Spread MA 60s	1.590	1.000	33.100	Float
Spread MA 5min	1.612	1.000	21.200	Float
Spread MA 12min	1.614	1.000	13.250	Float
Spread MA 26min	1.604	1.000	7.885	Float
Price MA Diff 12s-26s	0.506	0.000	1.000	Float
Price MA Diff 12min-26min	0.397	0.000	1.000	Float
Price MA Diff 48min-1h36min	0.374	0.000	1.000	Float
Order Size	42.638	1.000	100.000	Int
Order depth	4.688	-23.000	91.000	Int
Order Direction	0.486	0.000	1.000	Bin

Table 7: Summary and Description of Abides π^5

Feature	mean	min	max	Type
Spread	2.904	1.000	67.000	Int
Vol. Imbalance Lev.1	0.507	0.001	0.999	Float
Vol. Imbalance Lev.2	0.511	0.004	0.998	Float
Vol. Imbalance Lev.5	0.516	0.005	0.997	Float
Exec. Vol. Imbalance 1min	0.465	0.002	1.000	Float
Exec. Vol. Imbalance 5min	0.463	0.187	0.819	Float
Exec. Vol. Imbalance 12min	0.463	0.255	0.765	Float
Exec. Vol. Imbalance 26min	0.462	0.290	0.725	Float
Price return 1min	-0.000	-0.003	0.002	Float
Price return 5min	0.000	-0.004	0.004	Float
Price return 12min	0.000	-0.004	0.006	Float
Price return 26min	-0.000	-0.005	0.007	Float
Price MA 12s	99.967	99.534	100.821	Float
Price MA 26s	99.967	99.527	100.822	Float
Price MA 60s	99.967	99.576	100.821	Float
Price MA 5min	99.968	99.586	100.813	Float
Price MA 12min	99.967	99.599	100.804	Float
Price MA 26min	99.967	99.644	100.793	Float
Price MA 48min	99.969	99.759	100.754	Float
Price MA 1h36min	99.976	99.832	100.567	Float
Spread MA 12s	1.635	1.000	46.250	Float
Spread MA 26s	1.614	1.000	57.192	Float
Spread MA 60s	1.608	1.000	33.100	Float
Spread MA 5min	1.634	1.000	21.200	Float
Spread MA 12min	1.644	1.000	13.250	Float
Spread MA 26min	1.643	1.000	7.885	Float
Price MA Diff 12s-26s	0.514	0.000	1.000	Float
Price MA Diff 12min-26min	0.408	0.000	1.000	Float
Price MA Diff 48min-1h36min	0.380	0.000	1.000	Float
Order Size	44.081	1.000	100.000	Int
Order depth	3.617	-23.000	91.000	Int
Order Direction	0.489	0.000	1.000	Bin

Table 8: Summary and Description of Abides π^6

Feature	mean	min	max	Type
Spread	2.719	1.000	12.000	Int
Vol. Imbalance Lev.1	0.477	0.001	0.999	Float
Vol. Imbalance Lev.2	0.488	0.008	0.997	Float
Vol. Imbalance Lev.5	0.482	0.070	0.938	Float
Exec. Vol. Imbalance 1min	0.615	0.000	1.000	Float
Exec. Vol. Imbalance 5min	0.663	0.000	1.000	Float
Exec. Vol. Imbalance 12min	0.671	0.000	1.000	Float
Exec. Vol. Imbalance 26min	0.650	0.034	1.000	Float
Price return 1min	0.000	-0.008	0.011	Float
Price return 5min	0.000	-0.014	0.012	Float
Price return 12min	0.000	-0.022	0.016	Float
Price return 26min	0.000	-0.030	0.021	Float
Price MA 12s	11.258	10.416	12.131	Float
Price MA 26s	11.257	10.419	12.133	Float
Price MA 60s	11.257	10.421	12.129	Float
Price MA 5min	11.257	10.430	12.131	Float
Price MA 12min	11.256	10.452	12.198	Float
Price MA 26min	11.256	10.465	12.252	Float
Price MA 48min	11.261	10.482	12.375	Float
Price MA 1h36min	10.380	10.492	11.471	Float
Spread MA 12s	2.506	1.000	10.333	Float
Spread MA 26s	2.517	1.000	9.846	Float
Spread MA 60s	2.536	1.000	8.300	Float
Spread MA 5min	2.664	1.000	9.400	Float
Spread MA 12min	2.802	1.000	11.500	Float
Spread MA 26min	3.035	1.154	18.154	Float
Price MA Diff 12s-26s	0.582	0.000	1.000	Float
Price MA Diff 12min-26min	0.516	0.000	1.000	Float
Price MA Diff 48min-1h36min	0.452	0.000	1.000	Float
Order Size	105.494	1.000	300.000	Int
Order depth	0.655	-4.000	74.000	Int
Order Direction	0.550	0.000	1.000	Bin

Table 9: Summary and Description of AINV stock data

Feature	mean	min	max	Type
Spread	3.038	1.000	12.000	Int
Vol. Imbalance Lev.1	0.479	0.000	1.000	Float
Vol. Imbalance Lev.2	0.478	0.001	0.999	Float
Vol. Imbalance Lev.5	0.479	0.048	0.974	Float
Exec. Vol. Imbalance 1min	0.536	0.000	1.000	Float
Exec. Vol. Imbalance 5min	0.574	0.000	1.000	Float
Exec. Vol. Imbalance 12min	0.584	0.000	1.000	Float
Exec. Vol. Imbalance 26min	0.569	0.010	0.972	Float
Price return 1min	-0.000	-0.016	0.011	Float
Price return 5min	-0.000	-0.022	0.016	Float
Price return 12min	-0.001	-0.030	0.021	Float
Price return 26min	-0.002	-0.041	0.033	Float
Price MA 12s	5.560	5.132	6.020	Float
Price MA 26s	5.561	5.140	6.017	Float
Price MA 60s	5.561	5.140	6.015	Float
Price MA 5min	5.560	5.143	6.015	Float
Price MA 12min	5.561	5.145	6.010	Float
Price MA 26min	5.564	5.153	6.009	Float
Price MA 48min	5.569	5.160	5.997	Float
Price MA 1h36min	5.574	5.179	6.026	Float
Spread MA 12s	2.804	1.000	11.917	Float
Spread MA 26s	2.835	1.000	15.192	Float
Spread MA 60s	2.887	1.000	16.783	Float
Spread MA 5min	3.067	1.000	16.000	Float
Spread MA 12min	3.216	1.167	13.417	Float
Spread MA 26min	3.374	1.308	15.885	Float
Price MA Diff 12s-26s	0.547	0.000	1.000	Float
Price MA Diff 12min-26min	0.465	0.000	1.000	Float
Price MA Diff 48min-1h36min	0.517	0.000	1.000	Float
Order Size	149.949	1.000	600.000	Int
Order depth	1.599	-4.000	94.000	Int
Order Direction	0.497	0.000	1.000	Bin

Table 10: Summary and Description of AVXL stock data

Feature	mean	min	max	Type
Spread	2.254	1.000	26.000	Int
Vol. Imbalance Lev.1	0.459	0.000	1.000	Float
Vol. Imbalance Lev.2	0.462	0.002	0.998	Float
Vol. Imbalance Lev.5	0.475	0.068	0.967	Float
Exec. Vol. Imbalance 1min	0.568	0.000	1.000	Float
Exec. Vol. Imbalance 5min	0.582	0.000	1.000	Float
Exec. Vol. Imbalance 12min	0.574	0.000	1.000	Float
Exec. Vol. Imbalance 26min	0.565	0.004	1.000	Float
Price return 1min	0.001	-0.027	0.025	Float
Price return 5min	0.002	-0.038	0.053	Float
Price return 12min	0.004	-0.054	0.056	Float
Price return 26min	0.007	-0.069	0.077	Float
Price MA 12s	5.994	5.260	6.405	Float
Price MA 26s	5.993	5.260	6.404	Float
Price MA 60s	5.992	5.260	6.401	Float
Price MA 5min	5.990	5.281	6.393	Float
Price MA 12min	5.984	5.329	6.375	Float
Price MA 26min	5.974	5.373	6.391	Float
Price MA 48min	5.964	5.411	6.407	Float
Price MA 1h36min	5.953	5.448	6.433	Float
Spread MA 12s	2.193	1.000	23.667	Float
Spread MA 26s	2.199	1.000	26.192	Float
Spread MA 60s	2.238	1.000	28.350	Float
Spread MA 5min	2.397	1.000	25.200	Float
Spread MA 12min	2.688	1.000	27.333	Float
Spread MA 26min	3.098	1.000	18.077	Float
Price MA Diff 12s-26s	0.609	0.000	1.000	Float
Price MA Diff 12min-26min	0.605	0.000	1.000	Float
Price MA Diff 48min-1h36min	0.560	0.000	1.000	Float
Order Size	156.239	1.000	600.000	Int
Order depth	1.069	-4.000	50.000	Int
Order Direction	0.546	0.000	1.000	Bin

Table 11: Summary and Description of ADAP stock data

Feature	mean	min	max	Type
Price Best Ask	99.893	98.858	100.352	Float
Price Best Bid	99.892	98.857	100.350	Float
Vol Best Ask	51.604	0.000	1443.000	Int
Vol Best Bid	43.314	0.000	1000.000	Int
Exec. Vol Ask 1min	3.504	0.000	400.000	Int
Exec. Vol Bid 1min	3.800	0.000	298.000	Int
Book depth Ask	14.420	0.000	1378.000	Int
Book depth Bid	15.873	0.000	1342.000	Int
Last Exec Price	99.884	98.857	100.302	Float
Volatility 30 min	72.411	0.000	608.599	Float
Spread	1.013	0.000	2.000	Int
Price	99.892	98.857	100.350	Float
Norm. Price MA 1min	1.000	0.992	1.011	Float
Norm. Price MA 2min	1.000	0.992	1.012	Float
Norm. Price MA 3min	1.000	0.988	1.012	Float
Norm. Price MA 5min	1.000	0.987	1.012	Float
Norm. Price MA 6min	1.000	0.987	1.012	Float
Norm. Price MA 10min	1.000	0.987	1.013	Float
Norm. Price MA 15min	1.000	0.988	1.013	Float
Norm. Price MA 30min	1.000	0.987	1.013	Float
Action	0.319	-4.000	4.000	Int

Table 12: Summary and Description of RL-Agents Bubble

Feature	mean	min	max	Type
Price Best Ask	100.088	98.932	101.217	Float
Price Best Bid	100.087	98.931	101.217	Float
Vol Best Ask	45.171	0.000	500.000	Int
Vol Best Bid	45.905	0.000	776.000	Int
Exec. Vol Ask 1min	3.199	0.000	113.000	Int
Exec. Vol Bid 1min	2.453	0.000	200.000	Int
Book depth Ask	116.077	0.000	3664.000	Int
Book depth Bid	71.810	0.000	1396.000	Int
Last Exec Price	100.057	98.931	101.024	Float
Volatility 30 min	276.131	0.000	744.593	Float
Spread	1.001	0.000	2.000	Int
Price	100.087	98.931	101.217	Float
Norm. Price MA 1min	1.000	0.990	2.002	Float
Norm. Price MA 2min	1.001	0.990	2.002	Float
Norm. Price MA 3min	1.001	0.989	2.002	Float
Norm. Price MA 5min	1.001	0.987	2.002	Float
Norm. Price MA 6min	1.001	0.987	2.002	Float
Norm. Price MA 10min	1.000	0.985	2.002	Float
Norm. Price MA 15min	1.001	0.986	2.002	Float
Norm. Price MA 30min	1.000	0.986	2.002	Float
Action	0.095	-4.000	4.000	Int

Table 13: Summary and Description of RL-Agents Sine