Journal of Artificial Intelligence Research 80 (2024) 27-86 Submitted 08/2023; published 05/2024

Estimating Agent Skill in Continuous Action Domains

Christopher Archibald ARCHIBALD@CS.BYU.EDU
Brigham Young University, Provo, UT 84604 USA

Delma Nieves-Rivera DIN7@MSSTATE.EDU
Mississippi State University, Starkville, MS 93444 USA

Abstract

Actions in most real-world continuous domains cannot be executed exactly. An agent's performance in these domains is influenced by two critical factors: the ability to select effective actions (decision-making skill), and how precisely it can execute those selected actions (execution skill). This article addresses the problem of estimating the execution and decision-making skill of an agent, given observations. Several execution skill estimation methods are presented, each of which utilize different information from the observations and make assumptions about the agent's decision-making ability. A final novel method forgoes these assumptions about decision-making and instead estimates the execution and decision-making skills simultaneously under a single Bayesian framework. Experimental results in several domains evaluate the estimation accuracy of the estimators, especially focusing on how robust they are as agents and their decision-making methods are varied. These results demonstrate that reasoning about both types of skill together significantly improves the robustness and accuracy of execution skill estimation. A case study is presented using the proposed methods to estimate the skill of Major League Baseball pitchers, demonstrating how these methods can be applied to real-world data sources.

1. Introduction

Many real-world physical settings and activities require an agent to both *select* and *execute* continuous actions. It is generally impossible to execute selected actions with perfect precision, resulting in some amount of execution error which varies among agents. Successful agents need to appropriately account for their execution noise when selecting actions.

Consider as an example the game of darts. In this game metal darts are thrown at a circular dartboard, scoring points depending on which region of the dartboard is hit. A dart-thrower must first *make a decision*, selecting a point on the dartboard at which to aim. Second, the dart-thrower must *execute*, throwing the dart towards the intended target. Both decision-making and execution are important to the overall success of a dart-throwing agent. Other continuous domains with these properties include robotics settings, where angles and velocities are selected from continuous ranges; human settings, where a human either moves itself or causes other objects to move, again in continuous space; and computational settings, where abstract toy domains or simulations of games with continuous action spaces are modeled and used as test-beds for algorithms and approaches.

This article is focused on the task of evaluating the decision-making and execution capabilities of an agent observed acting in a domain. We present several methods for estimating these capabilities, previously termed *skills* (Archibald, Altman, & Shoham, 2010). We are interested in this problem for the potential impact it has in helping model and analyze agents in adversarial continuous domains, as well as the potential applications for assessing and giving feedback on human skill levels in many real-world settings; including sports, law enforcement, and elderly assistance. As the use of AI proliferates, there will be a desire to utilize AI methods to assist humans with their decision-making in many of these important domains, which provide unique challenges. One requirement for an AI agent to provide personalized assistance to humans in settings is that the AI must have an accurate model of that human's execution skill, since target action choices and quality depend inherently on an agent's ability to accurately execute an action. The techniques presented in this article enable the estimation of the skill level of individuals from observations, and are an essential component of any AI-assisted decision-making system in these domains.

The various methods presented in this article differ in the information that they leverage, the assumptions that they make, and the types of agents and situations that they can be applied to. Most of the methods apply Bayesian reasoning to the skill estimation problems. Each of the presented methods will be experimentally evaluated in a variety of domains and using a diverse set of agents.

The remainder of the article proceeds as follows: in Section 1.1, relevant related work is discussed. Section 2 precisely introduces the problem addressed in this article. Sections 3 through 7 introduce the methods that are the focus of this work. Section 8 introduces the domains, agents, and procedures that will be used to experimentally evaluate the methods. Section 9 presents and discusses the results of the experiments. Section 10 demonstrates how these methods can be applied to data from human agents in Major League Baseball. Finally, we conclude in Section 11 and give directions for future work.

This article extends two previous conference papers on this topic (Archibald & Nieves-Rivera, 2018, 2019). These papers introduced the execution skill estimation problem and proposed two solutions to this problem. For completeness, those methods are presented in Sections 3 and 5. The method described in Section 5 was also used in another conference paper to estimate the execution skill of baseball pitchers as a subcomponent of a larger game-theoretic pitching strategy system (Melville et al., 2023). This article introduces the decision-making skill estimation problem in continuous action domains and two methods that address both problems simultaneously (Section 6). Additionally, this article contributes adaptations of all methods to sequential domains (Section 7). All of the experimental results, including the application of the OR and JEEDS methods to baseball data in Section 10, are also contributions of this article.

1.1 Background

The problem of estimating agent skill has many connections to prior work in several areas. These areas include execution uncertainty, opponent modeling, imperfect decision-making, commonly called *bounded rationality*, execution skill estimation, reinforcement learning, and explainable AI. The topic of skill in games has been investigated by several researchers (Larkey et al., 1997; Dreef et al., 2002; Borm & Genugten, 2001; Dreef et al., 2004), although skill, as these works define it, refers to a characteristic of the game itself and is meant to indicate how much the outcome of a game relies on the player decisions as opposed to random chance.

The Bayesian reasoning techniques we apply to this problem have been well-established in the artificial intelligence literature and can be found in many textbooks, notably (Russell & Norvig, 2009) and (Thrun, Burgard, & Fox, 2005). The notions of execution skill and strategic skill were first introduced and investigated in the domain of computational pool, where agents with varying strategic and execution skills were evaluated to investigate the interactions of these two notions (Archibald et al., 2010). Extensive previous work in computational pool has largely focused on two main challenges (Greenspan et al., 2008). The first was creating an accurate and efficient physics simulator of billiards (Greenspan, 2005, 2006). The second was using this simulator to computationally select a good action, given an execution noise level (Archibald et al., 2016; Landry et al., 2015; Smith, 2007).

1.1.1 Execution Uncertainty

Research in multiple areas has investigated the topic of execution uncertainty. Among them, we can find general games (Bowling & Veloso, 2004; Archibald & Shoham, 2011), where agents cannot execute actions with certainty, and security games (Yin et al., 2011; Jiang et al., 2013), where it cannot always be assumed that actions undertaken by the security agents will be executed as planned. Auction settings (Van Valkenhoef et al., 2010), when goods, services, or payments may fail to occur as desired, are also an example.

1.1.2 Opponent Modeling

Opponent modeling is the problem of estimating the properties of an opponent (Nashed & Zilberstein, 2022). Much previous work on this topic has been done in imperfect information games like poker (Billings et al., 1998; Bard et al., 2015, 2013; Davis et al., 2014), but this work focuses on strategic characteristics and limitations of the opponents, and the domains do not include execution uncertainty. Examples of additional areas where opponent modeling work has been explored include general multi-agent systems (Carmel & Markovitch, 1995), real-time strategy games (Schadd et al., 2007), and n-player games (Sturtevant et al., 2006).

1.1.3 Bounded Rationality

To reason about agents with less than perfect decision-making ability, we utilize notions of bounded rationality (Simon, 1972). While a rich literature investigates many different aspects of bounded rationality, the proposed method focuses on one particular model, which has been called the *logit* or *softmax* model (Rosenfeld & Kraus, 2018; Ling et al., 2019). In this model, given a rationality parameter λ, action i with corresponding utility U_i is selected by the agent with probability $P(i) = e^{\lambda U_i}/\sum_j e^{\lambda U_j}$. As $\lambda \to 0$, the distribution over actions approaches uniform random, while as $\lambda \to \infty$ it approaches the deterministic selection of the action with maximum utility.

This model was notably used in the definition of quantal response equilibrium (McKelvey & Palfrey, 1995), which has in turn been used to model human decision-making in many game-theoretic scenarios (Yang et al., 2011; Batzilis et al., 2019; Rosenfeld & Kraus, 2018).

Other work has used this model to estimate λ rationality parameters for human agents in a discrete information-gathering game (Ling et al., 2019) or to vary the difficulty level of AI agents in games (Wu et al., 2019). One of our proposed methods (see Section 6) uses this softmax model internally, producing an estimate of the λ-rationality parameter for the observed agent.

The inference of this parameter was also done in (Guo & Gmytrasiewicz, 2011), given observations of an agent acting. They focused on discrete action settings where the expected value the observed agent would obtain for any action was known. Their contribution was to demonstrate a suitable conjugate prior, which can be used in the Bayesian update to maintain beliefs about the observed agent's λ parameter. They also showed that with a continuous set of possible opponent agent types, no such conjugate prior may be possible and numerical approximations are necessary. In our work we utilize numerical approximations since our setting has both continuous actions and a continuous set of agent types, meaning their conjugate prior methods are not applicable.

1.1.4 Estimating Execution Skill

Two previous papers focused on execution skill estimation in darts, given a sequence of scores resulting from dart throws and the corresponding intended target for each throw (Tibshirani et al., 2011; Miller & Archibald, 2021). This skill estimate can then be used to determine the ideal target location for each individual agent. The main difficulty faced is that score information does not include the exact location where a dart landed, and some scores do not even disambiguate dartboard regions. For example, a score of 12 could result from hitting the 12 region, the double 6 region, or the triple 4 region, which are in very different places on the dartboard.

The first paper required all of the input dart throws to be aimed at the center of the dartboard (Tibshirani et al., 2011), which means that the estimation procedure cannot be used during the normal course of most dart games. This limitation was overcome by the second paper using Monte-Carlo methods (Miller & Archibald, 2021). Our current work differs from these two papers in that we assume the intended action is unknown while the exact executed action is observed.

1.1.5 Reinforcement Learning

We model the environment as a *Markov Decision Process* (MDP), which is a general model also used in reinforcement learning (RL). RL typically approaches components of an MDP as unknown, which must be learned through experience with the environment. A general RL approach to the domains we approach in this paper might include the execution error distribution of the agent as a part of the probabilistic transition function of the MDP. Thus, one could view the problem of estimating an agent's execution skill as learning a part of the transition function of an MDP, where the other part of the transition function would correspond to the underlying dynamics of the environment, given a specific executed action.

Consider the task of determining a policy for an agent with its own execution skill, given an existing policy for another agent with a different execution skill. Since the environment portion of the transition function is the same for both agents, there should be a chance of reusing some of the information encoded in the existing policy. There has been work in RL that has focused on reusing policies between different closely-related tasks (Fernández & Veloso, 2013; Szita, Takács, & Lorincz, 2003), and researchers have focused on metrics which can capture the similarity between two MDPs (García, Visús, & Fernández, 2022). This general area is called *transfer learning*, and surveys of the relevant literature can be found (Taylor & Stone, 2009; Zhuang, Qi, Duan, Xi, Zhu, Zhu, Xiong, & He, 2020). We hypothesize that, where the notion of execution skill applies, the ability to separate an agent's execution skill from the environment transition function could be leveraged to improve transfer learning and more quickly determine good policies for new agents.

1.1.6 Explainable AI

The area of explainable AI, including explainable AI planning, focuses on helping humans who interact with an AI planning system to understand the "why" behind decisions and recommendations made by the AI system. Much work has been done on these topics (Saeed & Omlin, 2023; Chakraborti, Sreedharan, & Kambhampati, 2020). For an AI system providing decision advice to human agents in domains with human execution error, explanations which included information about execution skill might be especially helpful. This is especially true in sequential planning domains, where the value of being in certain states depends inherently on the execution skill of the agent.

2. Problem Statement

Necessary notation, terminology, and models are introduced in this section.

2.1 Environments and Agents

This article focuses on domains with continuous action spaces that can be modeled as Markov Decision Processes (MDPs). These MDPs consist of a set of states S, a continuous set of actions $A \subseteq \mathbb{R}^n$, a reward function $R : S \times A \mapsto \mathbb{R}$ and a transition function P which specifies, for any state and action combination, a distribution over possible next states.

An *agent* is the entity that selects actions and receives rewards in a given MDP. We consider agents to consist of both a decision-making component and an execution component. An agent's decision-making component specifies how the agent determines which action to attempt in a given state. These actions are referred to as *intended* or *target* actions. An agent's execution component specifies how accurately the agent can execute a given target action. This component is represented by a distribution over random perturbations referred to as χ. Samples from χ are drawn and added to the different target actions before they are executed. The *execution skill* of an execution component will refer to the standard deviation of the distribution χ. The space of possible execution skill levels (standard deviations of execution skill distribution χ) will be referred to as Σ, with $\sigma \in \Sigma$ referring to a specific execution skill level. An execution noise distribution corresponding to execution skill level σ will be denoted as χ^σ.

An agent's decision-making component can be thought of as specifying a distribution over target actions for a given state. There is no limit to the form or function of the decision-

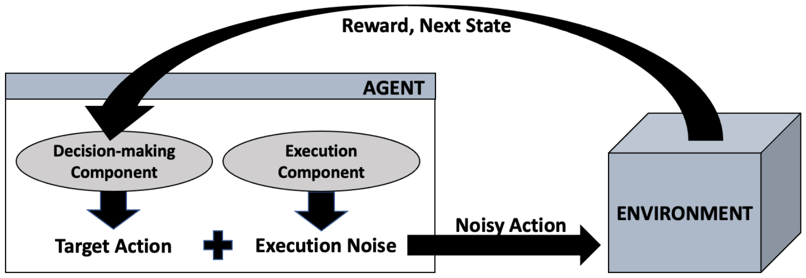


Figure 1: Agent Environment Interaction

making component. It can work in any manner, as long as it produces the desired distribution, which could have all the probability mass on a single target action. The *skill* of the decision-making component indicates how effective the intended actions are, with respect to the execution skill of the agent. This is represented by a single rationality parameter λ that will be estimated by the proposed model. The space of possible decision-making skill levels (rationality parameters) will be represented as Λ, with $\lambda \in \Lambda$ referring to one possible decision-making skill level. Section 8.3.2 discusses how the decision-making skill estimates are evaluated for the various agents.

Given a state in which to act, an agent first selects a target action and then attempts to execute that action, resulting in a perturbed action actually being executed. From the actual executed action, a reward and the next state are obtained. The model described above is portrayed visually in Figure 1.

2.1.1 Assumptions

In this work, the underlying assumptions about the model components are the same as have been used in previous work (Archibald & Nieves-Rivera, 2018, 2019). First, χ is assumed to be zero-mean and thus its most impactful characteristic is its standard deviation. Second, χ is assumed to be independent of the current state and intended action. Finally, χ is assumed to be known to the decision-making component. In other words, an agent is assumed to have correct beliefs about its own execution precision. We also assume that the execution skill and decision-making components of an agent do not change while the agent is being observed. Of course, in many settings, some of these assumptions might not apply: for instance, χ might vary depending on the state and intended action, the agent might have incorrect beliefs about χ, or the agent's skill might improve or worsen over time, but addressing these situations is left for future work as motivated by specific applications.

In this work, we assume that executed actions, received rewards, and state transitions are all observable and can be leveraged by the skill estimation techniques we propose. Crucially, we assume that target actions are not observed.

For simplicity of exposition, we first focus on MDPs where an agent's action choice does not impact the next state, or equivalently, where the agent faces a random sequence of states and the executed action only impacts the immediate reward obtained. Adaptations of the proposed approaches for general MDPs where action choices impact the sequence of states and rewards, which we call sequential domains, are presented in Section 7.

2.2 The Skill Estimation Problem

The problems of interest can now be precisely stated. The *skill estimation* problem is to estimate the parameters of an agent's execution and decision-making skill given observations of that agent acting in an MDP. The observations include only the executed action; intended actions are not observed. The skill estimation problem consists of two subproblems, the *execution skill estimation* problem (Archibald & Nieves-Rivera, 2018, 2019), and the *decision-making skill estimation* problem. The formulation of the decision-making skill estimation problem for continuous action domains is a novel contribution of this work. We show that considering both problems simultaneously yields significant advantages over only focusing on them in isolation.

The methods presented in this paper utilize the observed data in an online fashion, producing an estimate of the observed agent's skill level after each new observation. Each method leverages different information available in the observations. The first two methods, described in Sections 3 and 5, focus on execution skill estimation and utilize observations of the rewards received by the agent and the actions executed by the agent, respectively. In Section 6, we show how both reward and action information can be used to estimate both the execution and decision-making skill of the agent. In particular, this final method has the best overall performance in our experimental domains.

2.3 Baseline Execution Skill Estimation: True Noise Method

Before moving on, we first note that if the intended action were observed at each time step k, the execution skill estimation problem becomes easily solvable. In this case, the true execution error ϵ_k could be determined by looking at the difference between intended and executed actions. σ could then be estimated directly after n observations as the sample standard deviation of this vector of n ϵ values. This method, which we call the *True Noise Method* (*TN*) (Archibald & Nieves-Rivera, 2018), cannot be used in practice as it relies on unavailable information, but is introduced as a baseline for comparison since it can be computed during controlled experiments.

2.4 Toy Domain: 1D-Darts

To streamline the presentation of the skill estimation methods, some results in the *one-dimensional darts*, or 1D-Darts, domain will be reported after each method is described. 1D-Darts is a toy domain with a one-dimensional continuous action space that was introduced in (Archibald & Nieves-Rivera, 2018). It is fully described in Section 8, but in short, agents face a randomly-generated reward function defined on a one-dimensional action space. They observe the reward function and select an action. Gaussian noise is added to this action and the agent receives the reward corresponding to the noisy action. An agent's execution skill is the standard deviation of this Gaussian noise. An agent's decision-making skill refers to the effectiveness of the method that is used to determine which action to attempt, given the reward function and their execution skill. Full descriptions of all experimental details can be found in Section 8.

3. Reward Based Execution Skill Estimation

Our first execution skill estimation method focuses on the rewards obtained by the agent, which are part of each observation. This method is called the *Observed Reward* (*OR*) method (Archibald & Nieves-Rivera, 2018). The reward included with each observation is a single sample from the distribution of rewards that the agent could receive from that state, which is induced by the agent's execution error.

Given knowledge of the structure of the state and reward function, we can determine, for different possible execution skill levels σ^i, the maximum expected reward $V^*_{\sigma^i}$ that an agent with that execution skill level could receive in an observed state. As observations are processed at each time step k, the mean maximum expected reward $\overline{V^*_{\sigma^i}}$ for each hypothesis σ^i can be computed as the sample mean of $V^*_{\sigma^i}$ across all observed states. As actual rewards obtained by the agent are observed, the sample mean of the observed rewards can be similarly computed. We call this value the *mean observed reward*. At time step k, the OR method predicts σ using the mean observed reward and all of the $\overline{V^*_{\sigma^i}}$ values. This is done using linear interpolation on the $\overline{V^*_{\sigma^i}}$ values to obtain an estimate of the execution skill level that would result in the mean observed reward.

If the decision-making of the acting agent is perfectly rational, or in other words, the agent always selects the action with the maximum expected reward, then this is equivalent to estimating the mean maximum expected reward from a set of samples drawn from the true distribution. In this case, the estimate will converge to the true mean maximum expected reward by the law of large numbers as the number of observations approaches infinity. Thus, for perfectly rational agents, the OR estimate will also converge to the correct value, limited only by the resolution of our set of hypothesis execution skill levels. For agents who are not perfectly rational, the OR method is expected to work less well.

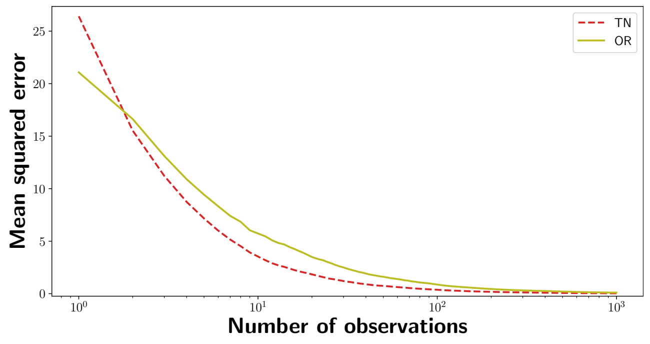


Figure 2: MSE of Execution Skill Estimates on Rational Agents in 1D-Darts

3.1 Observed Reward Performance

Figure 2 shows the MSE for the TN and OR methods over 1,000 observations, averaged across 3,892 experiments in 1D-Darts (see Table 2 in the appendix). In this case, the agent is perfectly rational, which matches the assumption of the OR method. It can be seen that, as the number of observations increases, both of these methods work, with OR converging slightly slower than TN to an MSE of 0.

While the OR method is able to estimate the execution skill level of an agent with perfectly rational decision-making, it does not make use of all the information in the observations. In particular, the OR method ignores the action that is actually executed by the agent. The exploration of our subsequent methods is motivated by the following questions: Can execution skill estimation be improved by utilizing information from the agent's executed actions, instead of simply the obtained rewards? Additionally, are there methods that might work outside of the assumption of perfectly rational decision-making?

4. Bayesian Skill Estimation Framework

This section introduces the Bayesian framework that forms the foundation of the remaining skill estimation methods. In Section 5 this framework will be used to estimate execution skill under some assumptions about the agent's decision-making skill. It will also be expanded to address both execution and decision-making skill estimation in Section 6. At its core, the Bayesian framework approaches skill estimation as probabilistic inference in a Bayesian network with random variables corresponding to various problem components. The random variables that will be used within the framework are:

- Σ, a random variable representing the execution skill level of the agent. This is an unobserved random variable that is not directly influenced by anything else. Execution skill estimation methods will perform inference to update a probability distribution over possible execution skill levels, given the observations. σ will be used to indicate a specific execution skill level for an agent, which is assumed to be constant for an agent throughout an experiment.
- Λ, a random variable corresponding to the decision-making skill of the agent. This is also unobserved and is not influenced by anything else, as it is considered an inherent characteristic of an agent. Decision-making skill estimation methods will infer a distribution over possible decision-making skill levels, given observations. λ will be used to refer to a specific decision-making skill level, which is also assumed to be constant throughout an experiment for a given agent.
- T, a random variable indicating the target, or intended action, for a given observation. This random variable is also not directly observed, but can be influenced by the execution and decision-making skill random variables Σ and Λ. A target action for a given observation will be referred to as t. The influence of the current state on the target action will be left implicit.
- X, a random variable indicating the action that is actually executed and observed. A specific executed action will be referred to as x. This random variable is directly

influenced by the target action random variable T, as well as the execution skill random variable Σ. $P(x \mid t, \sigma)$ indicates the true conditional distribution over executed actions given an intended action t and an execution skill level σ, and will directly correspond to the execution noise distribution χ^{σ}, centered on t.

The methods that utilize this Bayesian framework will each specify a concrete Bayesian network constructed from these random variables. These networks are shown in Figures 3 and 6. Each method will start with a prior distribution over the set of possible skill levels the agent could have. The Bayesian network corresponding to each method, along with chosen prior and conditional distributions, will be used to derive an update rule which will specify the posterior probability distribution over skill levels, given information from a single observation. The methods will repeatedly use this single update in an online manner, using the posterior after each update as the prior for the next. We now describe the conditional probability distributions that will be used, corresponding to how skill is modeled.

4.1 Modeling Execution Skill

In this work the Gaussian probability density function (ϕ) given in Equation 1 is used to specify $P(x \mid t, \sigma)$, the distribution over executed actions, given a target action and execution skill level. Another distribution could be substituted where motivated. Due to the symmetry of this choice of execution distribution $P(x \mid t, \sigma) = P(t \mid x, \sigma)$, since $\phi(x; t, \sigma) = \phi(t; x, \sigma)$.

$$P(x \mid t, \sigma) = \phi(x; t, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-t)^2}{2\sigma^2}} \tag{1}$$

4.2 Modeling Decision-making Skill

To model imperfect decision-making, the decision-making distribution shown in Equation 2 is used[1]. It utilizes the softmax bounded rationality model to specify a distribution over target actions t, given an execution skill level σ and a decision-making skill level λ. In this model, as $\lambda \to \infty$, an agent's decision-making skill increases.

$$P(t \mid \sigma, \lambda) = \frac{e^{\lambda V(t,\sigma)}}{\sum_{t' \in A} e^{\lambda V(t',\sigma)}} \tag{2}$$

The function $V : A \times \Sigma \mapsto \mathbb{R}$ is used as shorthand, where $V(a, \sigma) = \mathbb{E}_{\varepsilon \sim \chi^{\sigma}}[R(s, a + \varepsilon)]$, indicating the expected reward for an action in state s, given an execution skill level.

4.3 Representation and Estimates

The probability distributions over skill levels will be represented using a discrete distribution over a set of hypothesis skill levels. Two different methods for producing a single numerical skill level estimate from this distribution have been explored. The first method is the *Maximum a posteriori Skill* (MS) estimate, which selects the skill hypothesis with the highest posterior probability. The second method is the *expected skill* (ES) estimate,

1. This equation is presented in the discrete form, since in practice, the methods utilize a fine-grained discretization of the action space.

which takes into account each hypothesis and its corresponding posterior probability. For execution skill estimation, previous work has established that the ES estimate is superior, and so all execution skill estimation results in this article will only be reported for the ES estimate. Results for the MS estimate are included in the appendix. Since this article is the first to explore decision-making skill estimation, both estimates will be evaluated on that task, and will be denoted by an -ES or -MS algorithm suffix respectively. Exact equations for producing each of these estimates for the proposed skill estimation methods are presented in their respective sections.

5. Action Based Execution Skill Estimation

The first application of the Bayesian framework will be to estimate execution skill using information from the executed action in each observation. For this task, we utilize the Bayesian network shown in Figure 3, which includes the random variables Σ, T, and X. The figure uses plate notation, which indicates that there are distinct T and X random variables corresponding to each time step, while there is a single Σ random variable. The arrows indicate the probabilistic influence that these different random variables have on each other. The execution skill level Σ influences the target action T, and the final executed action X depends on both Σ and T. The X random variable is filled in, indicating that it is observed. We call the execution skill estimation method that uses this network the *Action based eXecution skill Estimation* method, or AXE.

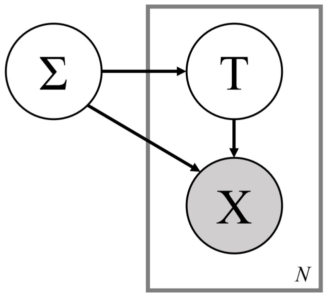


Figure 3: Bayesian Network for AXE method

5.1 AXE Derivation

The main goal of AXE is to estimate the probability of an execution skill level σ_i given the observed executed actions $x_{1:n}$, using Bayes rule as follows:

$$P(\sigma_i \mid x_{1:n}) \propto P(x_n \mid \sigma_i) P(\sigma_i \mid x_{1:n-1})$$

We will derive an update which will be used repeatedly, computing new beliefs $P(\sigma_i \mid x_{1:n})$ after observation n from the previous beliefs $P(\sigma_i \mid x_{1:n-1})$.

The conditional distribution $P(x_n \mid \sigma_i)$ does not correspond directly to any components of the model that has been described. Using the dependencies encoded in the Bayesian network of Figure 3, this can be rewritten in terms of modeled concepts as:

$$P(x_n \mid \sigma_i) = \int_{t \in T} P(t \mid \sigma_i) P(x_n \mid t, \sigma_i)$$

Here the first component $P(t \mid \sigma_i)$ represents the agent's decision-making component and specifies the probability the agent will select t as a target action given its execution skill level is σ_i. $P(x_n \mid t, \sigma_i)$ is the distribution over executed actions, given a target and execution skill level, which is the execution noise distribution χ^{σ}, centered on t. The question now becomes, how can the decision-making distribution $P(t \mid \sigma_i)$ be represented and computed? Different choices will lead to different variants of the AXE approach.

5.2 Rational AXE Variant

As discussed in Section 3, the OR method assumes that the decision-making component is perfectly rational. This same assumption can be represented within AXE by having $P(t \mid \sigma_i)$ be a distribution with all of the probability mass on a single optimal target action for each σ_i. This optimal target action will be denoted by t_{σ_i}. This results in $P(x_n \mid \sigma_i) = P(x_n \mid t_{\sigma_i}, \sigma_i)$. This method is expected to be extremely brittle, as any decision-making deviation from t_{σ_i} by the agent would lead to incorrect estimates.

5.3 Focal Actions AXE Variant

In many settings, we cannot assume that the agent (and estimator) will always be able to perfectly compute and select the optimal action. How can AXE be applied when we cannot assume that the agent being observed is a completely rational decision-maker? We now present a variant of AXE addressing this issue.

First, we move past the assumption that the optimal action for a given state and execution skill level can be precisely calculated. Instead, we will assume that, for a given state, a set of focal actions F is present. This set will consist of actions that are expected to be chosen by the agent, either because they are good actions for agents with no execution error or because they might be known to typically generate high, but not necessarily optimal, rewards. Moreover, it is assumed that this set of actions can be quickly generated without necessarily taking into account their robustness with respect to execution noise. We will assume only that the decision-making distribution has no preference for any specific focal action, selecting from among them uniformly at random. If the number of focal actions in the set F is m, this yields $P(x_n \mid \sigma_i) = \frac{1}{m} \sum_{t \in F} P(x_n \mid t, \sigma_i)$.

To additionally account for the case that an agent with imperfect decision-making skill might not even select one of the focal actions, an additional uniform distribution over all actions can also be included. $P(x_n \mid \sigma_i)$ can be constructed as a mixture of β times the focal action distribution and $(1-\beta)$ times a uniform distribution over all actions. β values near 1 should work better for agents and domains that frequently select focal actions, while lower β values should be preferable when domains and agents make it harder to specify a good focal action set to which many intended actions belong. If W is the width of the action space, then the full update can be written as follows:

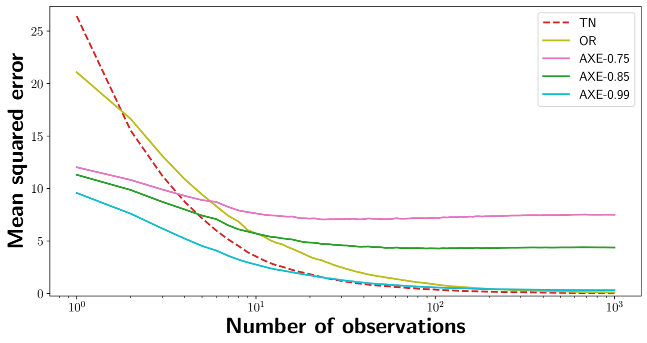


Figure 4: MSE of Execution Skill Estimates on Rational Agents in 1D-Darts

$$P(\sigma_i \mid x_{1:n}) \propto P(\sigma_i \mid x_{1:n-1}) \left[\beta \left(\frac{1}{m} \sum_{t \in F} P(x_n \mid t, \sigma_i) \right) + \frac{(1-\beta)}{W} \right] \tag{3}$$

In the remainder of this work, AXE will refer to the use of the update in Equation 3 (Archibald & Nieves-Rivera, 2019)[2].

5.4 Estimation for AXE

AXE maintains a single distribution over execution skill levels, from which the MS and ES estimates are produced using Equations 4 and 5.

$$\tilde{\sigma}_{MS} = \underset{\sigma^i}{\arg\max}\, P(\sigma^i \mid x_{1:n}) \tag{4}$$

$$\tilde{\sigma}_{ES} = \sum_i \sigma^i P(\sigma^i \mid x_{1:n}) \tag{5}$$

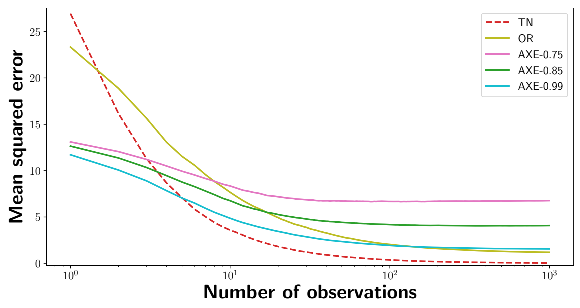
5.5 AXE Performance

Once again, we use 1D-Darts to briefly discuss the performance of the proposed AXE method, which is first compared with the OR and TN methods. Experiment details can be found in Section 8. Figure 4 compares AXE with three different β values to OR and TN on perfectly rational agents. AXE-0.99 has significantly better performance than OR with fewer observations, converging around an order of magnitude faster. This shows that leveraging action information, as AXE does, is more efficient than relying only on reward information, as OR does, when estimating the execution skill of a rational agent in 1D-Darts. However, the performance of AXE-0.85 and AXE-0.75 are not competitive with OR. We see that AXE can have different performance with different β values and so this must be fine-tuned.

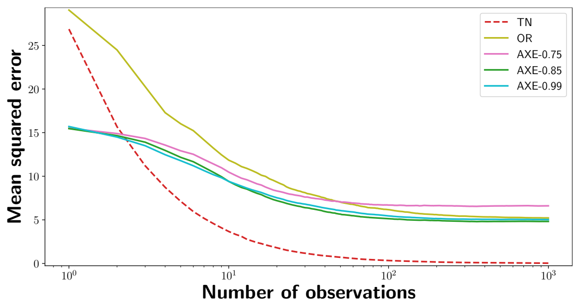
Figure 5 evaluates the effectiveness of AXE and OR at handling deviations from perfectly rational action choices. The Softmax agent, described in full in Section 8.2, assigns probabili-

2. This method was called *The Bayesian Approach* (TBA) in this previous work.

ties to each action based on how close to optimal that action is. The $\lambda_{\%}$ value, described in Section 8.3.2, basically gives an indication of how close the action selection is to optimal. $\lambda_{\%}$ of 100% indicates a perfectly rational agent, while a value of 0% indicates a uniform random agent. The estimates from the AXE-0.99 and OR methods do not converge to an MSE of 0 within 1000 observations on these less rational agents. AXE-0.85 and AXE-0.75 do even worse. The final error of OR and AXE-0.99 for agents with $\lambda_{\%} \in [25\%, 100\%]$ (Figure 5a) is not egregiously high, but as the decision-making of the agents gets worse, their final error increases (Figure 5b). The performance of AXE-0.85 is now about the same as AXE-0.99 while AXE-0.75 is still worse. Thus, the quality of the execution skill estimates produced by OR and AXE are impacted by the rationality of an agent's decision-making, as well as the chosen β parameter.



(a) Softmax Agents with $\lambda_{\%} \in$ [25%, 100%]



(b) Softmax Agents with $\lambda_{\%} \in$ [0%, 25%]

Figure 5: MSE of Execution Skill Estimates in 1D-Darts

AXE can work well in settings where a set of focal actions can be generated for each state, and when agents are behaving with nearly perfect rationality. It has been demonstrated to work well for estimating the execution skill of a top computational billiards agent (Archibald & Nieves-Rivera, 2019; Archibald, Altman, & Shoham, 2009), and has also been used for estimating the execution skill of professional baseball pitchers (Melville et al., 2023), which will be explored in more detail in Section 10. However, the applicability of AXE is somewhat limited by the fact that it requires two things to be specified: the set of focal actions and the β value. These will certainly vary per domain and also could vary per agent. Additionally, as agents exhibit less rational decision-making, the assumptions of AXE will cause its estimates to be less and less useful. The methods presented next utilize both reward and action information to reason about both the execution and decision-making skill of the agent, overcoming these limitations.

6. Joint Estimation of Decision-making and Execution Skill

In this section, the Bayesian framework introduced in Section 4 is used to estimate agent skill using information about both the executed actions and the rewards available in each state. This is done by including the random variable Λ in the Bayesian network shown in Figure 6. Λ, which represents the decision-making skill of the agent, allows the method to reason about the influence of Λ on the conditional distribution of T, the target action. This conditional distribution depends on the reward associated with each possible target action. As discussed previously, agents with higher λ values will assign more probability to target actions with higher expected rewards. Again, the dependence of target action t on each state is left implicit. The skill estimation problem can be precisely phrased in this context as correctly computing $P(\sigma, \lambda \mid x_{1:n})$, the beliefs over the joint space of the agent's execution and decision-making skill combinations after n observations.

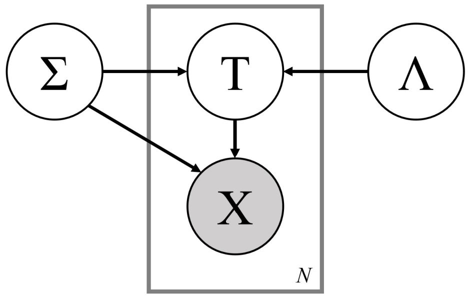


Figure 6: Bayesian Network for Joint Skill Estimation

6.1 Why Joint Estimation?

Correct beliefs about an observed agent's decision-making and execution skills cannot be obtained by treating each independently, since they are conditionally dependent given observations of executed actions. This can be seen by their relationship in the Bayesian network of Figure 6. In other words $P(\sigma, \lambda \mid x_{1:n}) \neq P(\sigma \mid x_{1:n})P(\lambda \mid x_{1:n})$. Estimating only one of these parameters requires assumptions to be made about the other, as was done by OR and AXE. Our method will maintain a joint distribution over both skill parameters, from which specific estimates can be calculated when needed.

6.2 Joint Skill Estimation

The proposed *Joint Estimation of Execution and Decision-making Skill* (JEEDS) method can now be completely described. The JEEDS method will compute the joint distribution $P(\sigma, \lambda \mid x_{1:n})$ after every newly observed executed action x_n. The derivation will specify this computation for a particular σ_i and λ_j skill combination in terms of the components that have previously been defined, as well as the joint distribution from the previous time step. It begins with the desired quantity: $P(\sigma_i, \lambda_j \mid x_{1:n})$. By Bayes Rule, we have:

$$P(\sigma_i, \lambda_j | x_{1:n}) \propto P(x_n \mid \sigma_i, \lambda_j) P(\sigma_i, \lambda_j \mid x_{1:n-1}) \tag{6}$$

$P(\sigma_i, \lambda_j \mid x_{1:n-1})$ is the distribution from the previous time step. At $n = 1$ a prior distribution will be used. The reduction of $P(x_n \mid \sigma_i, \lambda_j)$ on the right-hand side of Equation 6 to known components is guided by the structure of the Bayesian network in Figure 6.

$$\begin{aligned} P(x_n \mid \sigma_i, \lambda_j) &= \sum_{t \in A} P(x_n, t \mid \sigma_i, \lambda_j) \\ &= \sum_{t \in A} P(t \mid \sigma_i, \lambda_j) P(x_n \mid t, \sigma_i) \end{aligned} \tag{7}$$

All expressions now correspond to the skill models described earlier. Using the conditional distributions specified in Equations 1 and 2 and inserting the result into Equation 6 produces the complete JEEDS update.

$$\begin{aligned} P(\sigma_i, \lambda_j | x_{1:n}) &\propto P(\sigma_i, \lambda_j \,|\, x_{1:n-1}) \sum_{t \in A} P(t \,|\, \sigma_i, \lambda_j) P(x_n \,|\, t, \sigma_i) \\ &= P(\sigma_i, \lambda_j \,|\, x_{1:n-1}) \sum_{t \in A} \frac{e^{\lambda_j V(t,\sigma_i)} \phi(x_n; t, \sigma_i)}{\sum_{t' \in A} e^{\lambda_j V(t',\sigma_i)}} \\ &= \frac{P(\sigma_i, \lambda_j \,|\, x_{1:n-1})}{\sum_{t' \in A} e^{\lambda_j V(t',\sigma_i)}} \sum_{t \in A} e^{\lambda_j V(t,\sigma_i)} \phi(t; x_n, \sigma_i) \end{aligned} \tag{8}$$

After performing this update for every joint skill combination, the resulting belief is normalized, ensuring a valid probability distribution is maintained.

6.3 Marginal Skill Estimation

One potential limitation of the JEEDS method is the space required to store the joint skill beliefs, which scales quadratically with the number of hypothesis skill levels, as a probability is stored for every possible combination of decision-making and execution skill parameters. As discussed in Section 6.1, the conditional dependence between execution and decision-making skills introduced by the observations makes this necessary. The *Marginal Estimation of Execution and Decision-making Skill* (MEEDS) method is introduced to evaluate the importance of joint beliefs. The MEEDS method consists of two beliefs that are stored independently, one for each type of skill, requiring only linear space. The current beliefs about one skill are used in the update of the other. The derivation for MEEDS, which is similar to that for JEEDS, is omitted, but yields the updates given in Equation 9 (for execution skill) and Equation 10 (for decision-making skill). The MEEDS and JEEDS methods will be compared experimentally to investigate the importance of joint skill estimation.

$$P(\sigma_i \,|\, x_{1:n}) \propto P(\sigma_i \,|\, x_{1:n-1}) \sum_{\lambda_j \in \Lambda} P(\lambda_j) \sum_{t \in A} \left(\frac{\phi(t; x_n, \sigma_i) e^{\lambda_j V(t,\sigma_i)}}{\sum_{t' \in A} e^{\lambda_j V(t',\sigma_i)}} \right) \tag{9}$$

$$P(\lambda_j \,|\, x_{1:n}) \propto P(\lambda_j \,|\, x_{1:n-1}) \sum_{\sigma_i \in \Sigma} P(\sigma_i) \sum_{t \in A} \left(\frac{\phi(t; x_n, \sigma_i) e^{\lambda_j V(t,\sigma_i)}}{\sum_{t' \in A} e^{\lambda_j V(t',\sigma_i)}} \right) \tag{10}$$

6.4 Estimation for JEEDS/MEEDS

JEEDS maintains a joint distribution over both skills $P(\sigma_i, \lambda_j \,|\, x_{1:n})$. The two types of estimates (MS and ES) are produced as follows:

$$\tilde{\sigma}_{MS} = \underset{\sigma_i}{\operatorname{argmax}} \sum_{\lambda_j \in \Lambda} P(\sigma_i, \lambda_j \,|\, x_{1:n}) \tag{11}$$

$$\tilde{\lambda}_{MS} = \underset{\lambda_j}{\arg\max} \sum_{\sigma_i \in \Sigma} P(\sigma_i, \lambda_j \mid x_{1:n}) \tag{12}$$

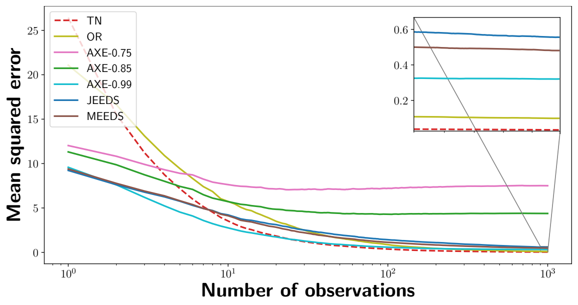
$$\tilde{\sigma}_{ES} = \sum_{\sigma_i \in \Sigma} \sum_{\lambda_j \in \Lambda} P(\sigma_i, \lambda_j \mid x_{1:n}) \sigma_i \tag{13}$$

$$\tilde{\lambda}_{ES} = \sum_{\lambda_j \in \Lambda} \sum_{\sigma_i \in \Sigma} P(\sigma_i, \lambda_j \mid x_{1:n}) \lambda_j \tag{14}$$

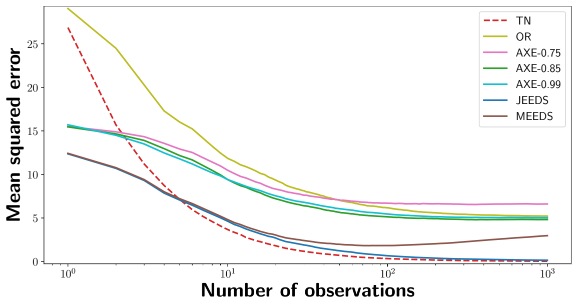
MEEDS maintains one distribution for each skill type, from which estimates are produced. These are produced in the same manner as for the AXE method (Figures 4 and 5) The estimates of the λ rationality parameter are produced in the same way, simply substituting λ for σ in the those equations.

6.5 JEEDS/MEEDS Performance

Figure 7 shows the performance of the JEEDS and MEEDS estimators along with the previous methods, again in the domain of 1D-Darts. Figure 7a focuses on perfectly rational agents, and JEEDS and MEEDS methods both converge to a MSE near 0, but do so more slowly than AXE-0.99. This is to be expected, as these two methods both need time to learn from the observations that the agent they are observing is perfectly rational, while AXE-0.99 has that assumption built in. However, both new methods significantly outperform the other AXE variants, showing the advantage of not having to specify a β value.



(a) Rational Agent



(b) Softmax Agents with $\lambda_{\%} \in [0\%, 25\%]$

Figure 7: MSE of Execution Skill Estimates in 1D-Darts

The real test of these new methods is on less rational agents. Figure 7b shows the performance on Softmax agents with $\lambda_{\%} \in [0\%, 25\%]$. JEEDS converges to a MSE near zero on these agents, and appears to be unaffected by their reduced rationality! All of the other methods, including MEEDS, cannot accurately determine the execution skill of these agents within 1000 observations. This shows the impact of the way JEEDS reasons about both execution and decision-making skill. Complete experimental results for 1D-Darts and other experimental domains will be presented and discussed in Section 9.

7. Skill Estimation in Sequential Domains

In the presentation so far, the focus has been on domains where an agent's goal is to maximize immediate reward, as actions have no impact on the future states that the agent will visit. However, many settings are what we will call *sequential*, meaning that an agent's actions give immediate reward *and* influence which future states are visited. In these domains, the agent's goal is to maximize the total reward gained over the entire episode. Some actions might get a lot of immediate reward, but not lead to favorable future states and so they may not be good actions for the agent to select.

As an example, consider the variant of darts played by professionals: 501-Darts. In this game, the competitors start out with a score of 501 and take turns throwing a set of three darts at a dart board. Each competitor's score is reduced by the total point value of the regions where the three darts land. The game is over when a player has a score of exactly 0. Additionally, the final dart throw that causes the score to reach 0 must land in a double-scoring region or the player busts, meaning the player's set is over and they return to their score at the beginning of the set. The player also busts if a throw causes their score to go to exactly 1 or below 0. It is clear that simply maximizing the score of each individual dart throw is not a darts player's true goal. Instead, they are trying to reach a score of 0 in as few dart throws as possible. We can model this as the player receiving a reward of -1 for each dart throw, and then trying to maximize the total reward gained. How can the previously presented methods be modified to work in sequential domains like this? Adaptations for each of the previous techniques will be presented in turn.

One concept that will be utilized by several of these adaptations is that of the *optimal action-value function* corresponding to an execution skill level σ, which will be denoted by $Q^\sigma : S \times A \mapsto \mathbb{R}$. This function specifies the future expected reward an agent should be able to receive, given that they start in state s and take action a, followed by acting perfectly with respect to the optimal action-value function for their execution skill in future states they reach. The optimal value function for each hypothesis execution skill level σ is defined as the solution to Equation 15, computed using value iteration separately for each σ.

$$Q^\sigma(s,a) = \mathbb{E}_{\varepsilon\sim\chi^\sigma}\left[R(s,a+\varepsilon) + \sum_{s'\in S} P(s' \mid s, a+\varepsilon)\left[\max_{a'\in A} Q^\sigma(s',a')\right]\right] \tag{15}$$

7.1 OR in Sequential Domains

The observed reward (OR) method described in Section 3 tracks the average reward gained by the agent per state visited and compares that to a computed maximum for each execution skill level. In sequential domains, OR will instead track the average total reward gained by the agent *per episode*. The naïve method of doing this is to track the total reward gained during an episode and to update the stored agent average at the end of each episode, essentially performing Monte-Carlo value estimation. We call this method *OR-FullGame*. One downside to this method is that the skill estimate for the observed agent will only be updated once an entire episode is finished. In some domains, this might happen rarely or not at all.

Another approach then, is to update the average total episode reward *during* each episode. Until an episode is completed, the accumulated reward for the current episode will be considered an estimate of the total episode reward. This estimate will be averaged in with the total rewards already observed from completed episodes. As more and more episodes are completed, the estimated current episode reward will have less and less impact on the overall average total episode reward and, thus, on the execution skill estimate. This method will be referred to as *OR-MidGame*.

In each case, the average episode reward for perfectly rational agents with an array of execution skill levels will be precomputed, and the observed average episode reward (either FullGame or Midgame) will be used to interpolate into this set to determine the current execution skill estimate for the agent. These average episode rewards will be computed under the assumption that a perfectly rational agent with execution skill σ will select an action that maximizes $Q^{\sigma}(s, a)$ when making a decision in state s.

7.2 AXE in Sequential Domains

The AXE method does not explicitly utilize reward information, and so no adaptation to its update equation is required. One possible modification is to adjust the set of focal points that are used. This set could vary based on the state an agent is in and/or the execution skill level that is being evaluated. Changes to the focal points would be motivated and undertaken using domain-specific knowledge.

7.3 JEEDS/MEEDS in Sequential Domains

The JEEDS/MEEDS methods use the reward information at a fundamental level and so need to be adjusted to work in sequential domains. The modification is to substitute $Q^{\sigma}(s, a)$ for $R(s, a)$ when computing the update for execution skill level σ. This adaptation requires knowledge of the action-value function for each hypothesis execution skill level, but these can be precomputed and loaded when needed by the estimators.

8. Experimental Setup

The estimation methods presented in this article were experimentally evaluated using observa-tions from several different types of agents acting in multiple different domains. The agents were selected to cover a variety of decision-making approaches. This section describes the experimental domains, agents, and other experimental details. The results of the experiments are presented and discussed in Section 9.

8.1 Experimental Domains

The following domains were chosen for the evaluation of the proposed methods. They range from a toy domain to more real-world settings. These were selected to enable each method's estimates to be compared to a ground truth in order to determine the effectiveness and accuracy of each method.

8.1.1 One-dimensional Darts

One-dimensional darts (1D-Darts), was designed as a simple domain for investigation of execution and decision-making skill (Archibald & Nieves-Rivera, 2018, 2019). In this work we use a slightly modified version as follows: There is a one-dimensional continuous action space with a reward function defined on it. The reward is 0 for actions outside $[-10, 10]$ while inside that region the reward alternates between the values of 1 and 2. The alternation of the reward function is randomly generated for each new state that the agent faces. An example of a reward function in our 1D-Darts domain can be seen in Figure 8.

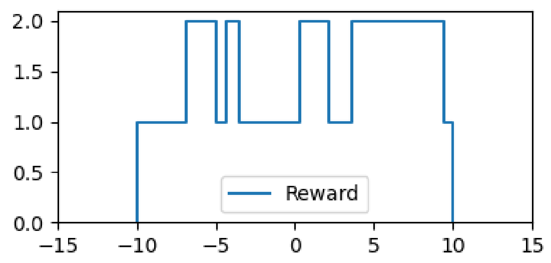
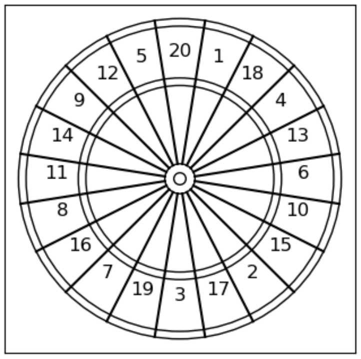


Figure 8: Example 1D-Darts Reward Function

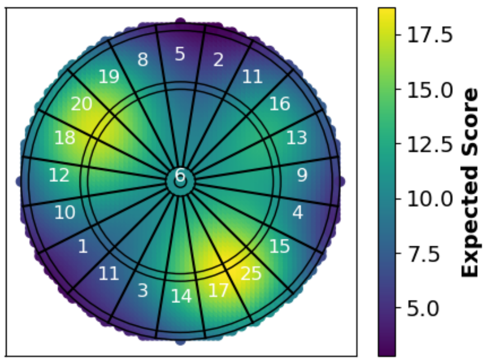
8.1.2 Two-dimensional Darts

Our *two-dimensional darts* (2D-Darts) domain is a variant of the traditional game of darts. Darts has been used previously for execution skill estimation given knowledge of target actions (Tibshirani et al., 2011; Miller & Archibald, 2021). In darts, projectiles are thrown at a circular target area with a radius of 170 mm. The target area is divided into twenty slices, each labeled with the base reward obtained for landing a dart in that slice. The traditional game of darts uses a dartboard with one fixed configuration of these base rewards, shown in Figure 9a. The circular band around the edge of the dartboard is the double-score region, while the band about halfway to the edge of the board is the triple-score region. Darts landing in these regions receive $2\times/3\times$ the base reward for the slice, respectively. The center of the board consists of a small circular region surrounded immediately by another band. These two regions, called double and single bull, respectively, are not considered a part of any slice but are instead collectively called the *bullseye*. In the traditional game, a dart landing in the double bull receives a reward of 50, while a dart landing in the single bull receives a reward of 25. Actions landing outside the board receive zero reward.

For 2D-Darts the base rewards are randomly shuffled in each new state that the agent faces. The base reward for the bullseye (25) is also shuffled with the rest of the slice base rewards. This yields more interesting and challenging states, giving a more robust evaluation of an agent's decision-making and execution components. Figure 9b shows an example 2D-Dart state with shuffled base rewards. Each location on the dartboard is colored with the expected reward an agent would receive if they select that location as their intended target, assuming an execution skill level of $\sigma = 39.375$ mm. There are multiple areas with high rewards, allowing for variety in agent action selection.



(a) Traditional Dartboard Configuration, used in 201-Darts Domain



(b) Example Shuffled 2D-Darts State showing expected scores for $\sigma = 39.375$ mm

Figure 9: Dart Board Examples

8.1.3 201-Darts

201-Darts is a sequential domain adapted for a single agent from the traditional game of 501 darts. Agents face the same dartboard as in 2D-Darts but the base rewards for each slice always stay the same as the traditional configuration shown in Figure 9a. The state of the game is an agent's current score, which begins at 201. On each turn, the agent throws a dart at the dartboard and receives a reward of -1. The agent's score is reduced by the value of the region where the dart landed. An episode finishes when the agent's score reaches exactly zero. The dart which reaches a score of 0 must land in a double-score region. Otherwise, the agent's score is reset to the previous score. If an agent's score is negative or exactly 1, then it is also reset to the previous score. The agent's goal is to maximize their total reward, or equivalently, to reach a score of 0 in as few darts as possible.

8.2 Agents

Four agents, each with a different decision-making component, were used to evaluate the proposed skill estimation methods. The *Rational* agent always selects an optimal, expected-reward-maximizing action for its execution skill level. The other agents' imperfect decision-making components are each parameterized by a single parameter. The *Flip* agent employs an ϵ-greedy strategy, either selecting an optimal action w.p. $\lambda_f \in [0, 1]$ or an action uniformly at random. The *Softmax* agent uses $\lambda_s \in [0, \infty]$ and Equation 2 to probabilistically select an action. The *Deceptive* agent selects the action obtaining $\lambda_d \in [0, 1]$ times the maximum possible expected reward that is farthest from an optimal action. It does this with the idea of hiding its execution skill, while controlling the amount of reward that it is sacrificing. This agent acts suboptimally with respect to the reward in the environment, but its action selection is not random. Instead, it is essentially considering an additional reward signal as it makes a decision: the distance between its executed action and an optimal action. As a result, this agent is expected to be the most challenging for the estimators, since the estimators are ignorant as to the agent's true motives. Each of the non-rational agents places a positive probability on actions that are not the rational, expected reward-maximizing action in a given state. In each case, a higher λ indicates a more rational agent. The set of possible target actions each agent can select is limited to locations on the dartboard for 2D-Darts and 201-Darts, and in the interval $[-10, 10]$ for 1D-Darts.

8.3 Experimental Procedure

The experiments were conducted by repeating the following process in each domain. First, a random execution noise level was generated, which was used by all agents. Second, a decision-making skill level was randomly chosen for each agent from their respective range of rationality parameters. The agents then faced the same sequence of environment states, except for in 201-Darts, where the initial state is always the same and the state sequence depends on the actions executed by the agent. Each agent selected a target action, execution noise was drawn from the agent's execution noise distribution and added to the target action, and the final executed action was observed by each of the estimation methods. Each estimation method produced a skill estimate after each observation. Each experiment consisted of 1,000 observations. Table 2 in the appendix presents the total number of experiments obtained for each domain and type of agent. Note that fewer experiments were used for the Rational agent because only a single rationality parameter had to be covered.

The action space for all agents was discretized with a resolution of 0.01 (1D-Darts) and 5.0 mm (2D-Darts, 201-Darts). For 201-Darts, the Q^σ function for each hypothesis execution skill level was precomputed using value iteration. Expected action rewards were computed by convolving the state's reward (1D-Darts, 2D-Darts) or Q^σ (201-Darts) function with the agent's execution noise distribution using that same resolution.

For 1D-Darts the execution error of agents was drawn from a single-dimensional Gaussian distribution with standard deviation σ. For 2D-Darts and 201-Darts a symmetric two-dimensional Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ was used, where $\boldsymbol{\mu} = [0, 0]$ and $\boldsymbol{\Sigma} = \sigma I_2$, leaving a single execution skill parameter to be estimated. The range of execution skill noise levels was $[0.5, 4.5]$ (1D-Darts) and $[3.0, 150.5]$ mm (2D-Darts, 201-Darts). The agent rationality parameter ranges were: $\lambda_f, \lambda_d \in [0.0, 1.0]$ (for all domains), $\lambda_s \in [0.001, 100.0]$ (1D-Darts) and $\lambda_s \in [0.001, 32.0]$ (2D-Darts, 201-Darts). All estimation methods used 17 (1D-Darts) and 33 (2D-Darts, 201-Darts) hypothesis skill levels for execution skill, whereas 33 were used for decision-making skill on all domains. The decision-making skill level hypotheses were logarithmically distributed to capture a more even distribution over possible $\lambda_{\%}$ values (see Section 8.3.2). All beliefs were initialized to be uniform over the space of possible skill parameters.

For the AXE method, the set of focal actions in a state consisted of the optimal actions for the set of execution skill hypotheses. Experiments were conducted with each $\beta \in \{0.50, 0.75, 0.85, 0.90, 0.95, 0.99\}$. The accuracy of the estimates varied depending on the β value used. The β value which produced the minimum MSE (averaged across all experiments) after 1000 observations, varied between domains and agents. Results are presented with three different β values $(0.75, 0.85, 0.99)$, to give a sense of how performance is impacted as the agent type and domain are changed. Plots presenting the performance of AXE with all β values can be found in the appendix.

8.3.1 Evaluating Execution Skill

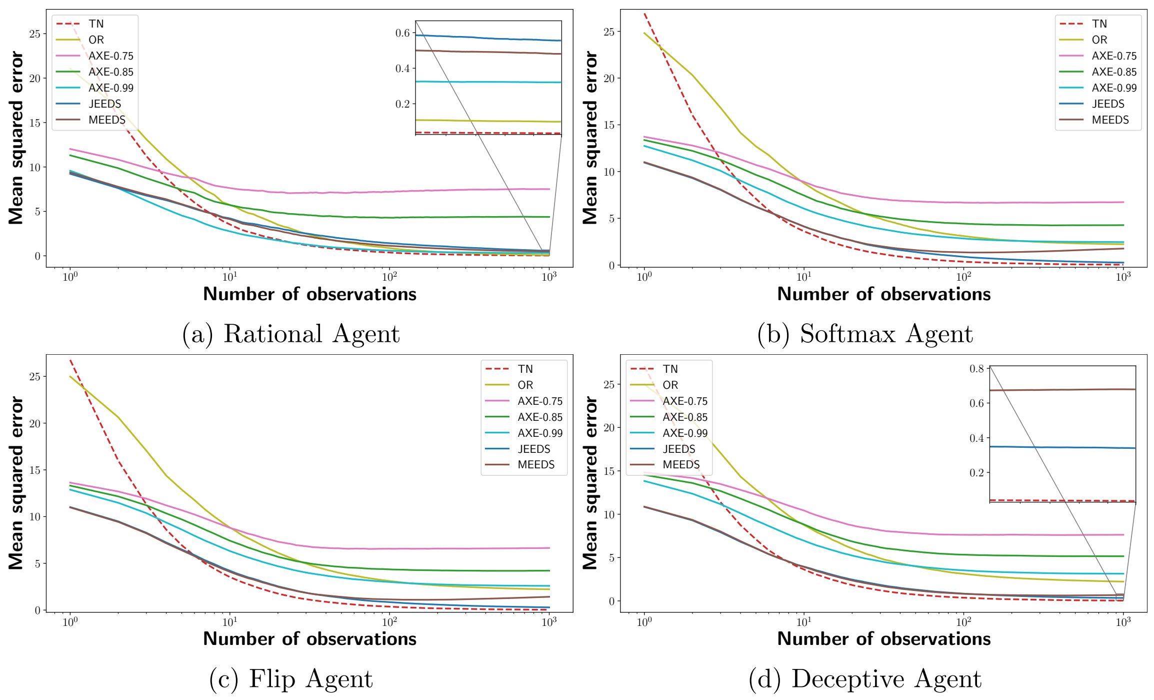
As all of the estimation methods produce execution skill estimates of σ for the observed agent on the same scale, these estimates are evaluated by comparing the estimates with the true σ values for the agents. The results generally report the mean squared error (MSE) of these estimates for each estimator.

8.3.2 Evaluating Decision-making Skill

One major challenge in evaluating the proposed JEEDS/MEEDS decision-making skill estimators is measuring the accuracy of the estimates on all of the types of agents. As just discussed, this is not an issue for execution skill, since the standard deviation of each agent's execution skill distribution can be compared to the estimate straightforwardly. For decision-making skill the situation is different. The JEEDS/MEEDS methods produce an estimate of the λ_s rationality parameter, which only has a well-defined meaning in the context of the Softmax decision-making distribution. To evaluate the accuracy of this estimate on the other agents, we need a way to map between the different rationality parameters. To our knowledge, no previous notion of an *inter-model* rationality parameter that could address this issue has been proposed.

To this end, we utilize a novel measure of rationality that can be calculated from an agent's selected target actions and their true σ value, both of which are known during the experiments. We define for a given state: r_M, the maximum expected reward possible, r_U, the expected reward obtainable by randomizing uniformly over actions, and r_A the expected reward for the agent's selected target action, where all expectations are with respect to the execution noise distribution χ^σ. If an agent's decision-making distribution over target actions is known, r_A can be the expected reward of that entire distribution, instead of simply the single target action. The *rationality percentage* $\lambda_\% = \frac{r_A - r_U}{r_M - r_U}$ indicates where an agent lies between uniform random and maximum expected reward. For a perfectly rational agent, $\lambda_\% = 1$, while as $\lambda_\%$ decreases, the rationality of the agent also decreases. An agent with $\lambda_\% = 0$ is obtaining the same expected reward they would by selecting actions uniformly at random. $\lambda_\%$ can be negative when selected actions are worse than uniform random.

During the experiments, the average $\lambda_\%$ for each agent was computed across all observations. The estimated λ_s value from the JEEDS/MEEDS methods was then converted to a $\lambda_\%$ value using a function that was fit to a separate dataset of the Softmax agent acting in each domain. This conversion function takes a λ_s value and the estimated σ value and outputs an approximately equivalent $\lambda_\%$ value. For more specific details regarding this conversion process, please see the appendix. This converted $\lambda_\%$ value was then compared to the $\lambda_\%$ value computed for the observed agent and the results generally report the MSE of these values. The execution skill estimate σ is utilized during the conversion because the $\lambda_\%$ for the Softmax agent depends not only on the difference between r_M and r_U, but also on how much the expected reward varies across different actions. $\lambda_\%$ can be higher if a single action is best by a wide margin, and lower if many actions yield similar expected reward. The



(a) Rational Agent

(b) Softmax Agent

(c) Flip Agent

(d) Deceptive Agent

Figure 10: MSE of Execution Skill Estimates in 1D-Darts

agent's execution skill level influences how much the rewards vary. With little execution noise, actions can have more varied expected rewards, whereas with a lot of execution noise, many actions have similar rewards. The exploration of a rationality parameter for which the λ_s conversion is independent of execution skill is left as future work.

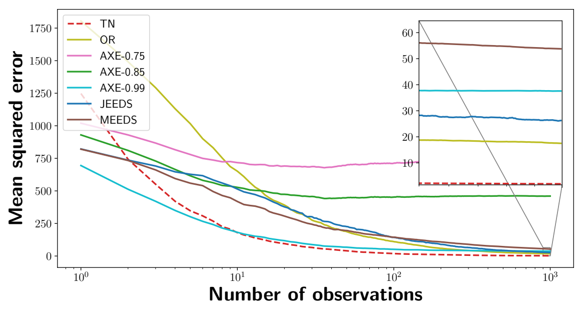
9. Experimental Results

We now present the results of the experimental evaluation of the proposed methods. Complete tables of final errors and confidence intervals for each figure are included in the appendix. We begin in Sections 9.1 and 9.2 by analyzing the effectiveness of the estimators for each type of skill in isolation, followed in Section 9.3 by an exploration of the interaction between the two types of skill and the performance of the estimators.

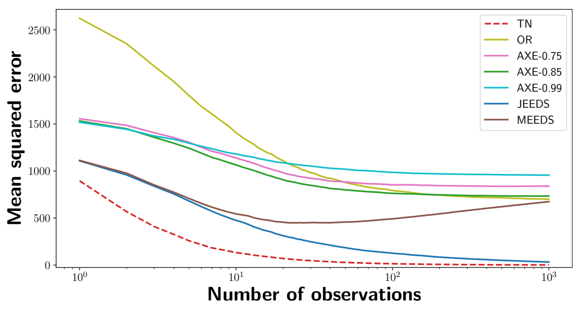
9.1 Estimating Execution Skill

We start with an investigation of the accuracy of the execution skill estimators. Figures 10 (1D-Darts), 11 (2D-Darts) and 12 (201-Darts) show the average MSE of the different execution skill estimation methods as a function of the number of observations for each of the four agents. This average MSE is computed across all experiments, execution skill levels, and decision-making skill levels and shows the general performance of each method.

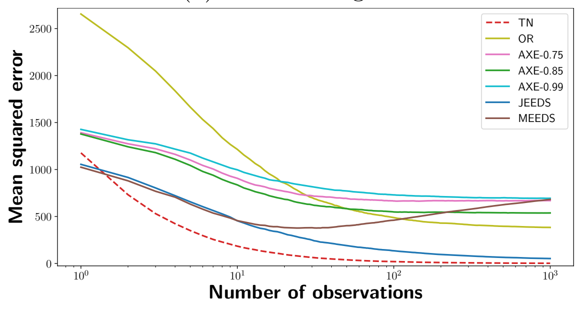
We begin by examining estimator performance on the Rational agent, shown in Subfigure (a) for each domain. Each estimation method works well on this agent in all domains, having



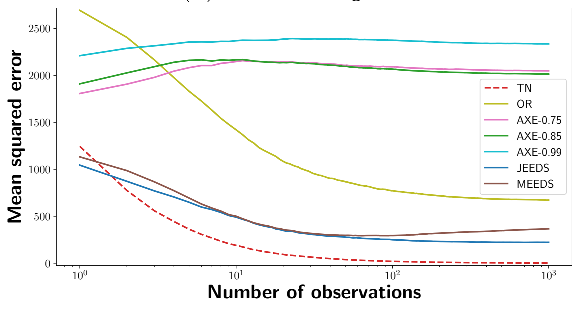
(a) Rational Agent



(b) Softmax Agent



(c) Flip Agent



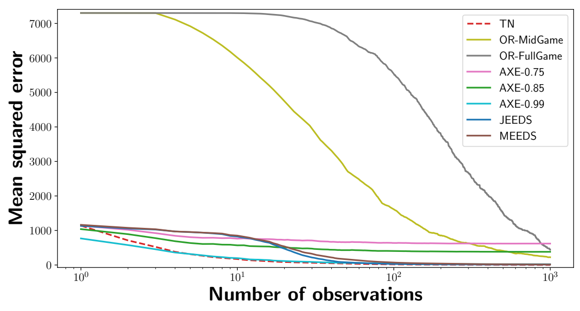
(d) Deceptive Agent

Figure 11: MSE of Execution Skill Estimates in 2D-Darts

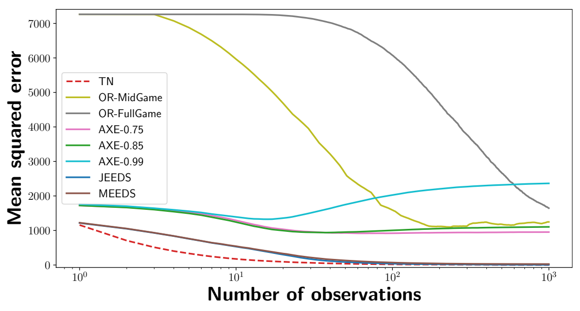
MSE converge to near 0. For the AXE method, this good performance only occurs when $\beta = 0.99$, while the other β values result in worse MSE values. This is one shortcoming of the AXE method, that the β value can impact the performance. In the 201-Darts domain, the two OR methods converge much slower than the other methods, but reach low error after all 1000 observations. Of the two OR variants, OR-MidGame converges more quickly in 201-Darts. In all three domains, the AXE-0.99 estimator has the lowest error with fewer than 100 observations. When the agent is perfectly rational, we are able to more easily predict the actions that they will take, and thus AXE works very well with the correct β value. This confirms findings from previous work on the effectiveness of the AXE method (Archibald & Nieves-Rivera, 2019). When agents are rational, each of the methods can be used to estimate execution skill, given enough observations.

The situation gets a lot more interesting as we start considering agents who are not perfectly rational. For the Softmax agent in 1D-Darts (Figure 10b) and 2D-Darts (Figure 11c), the only method that converges to a MSE near 0 is JEEDS. All of the other methods still have a significantly higher MSE after all 1000 observations. In 201-Darts (Figure 12c), the JEEDS and MEEDS methods are both able to successfully converge to very low MSE, while the OR and AXE methods do not work very well.

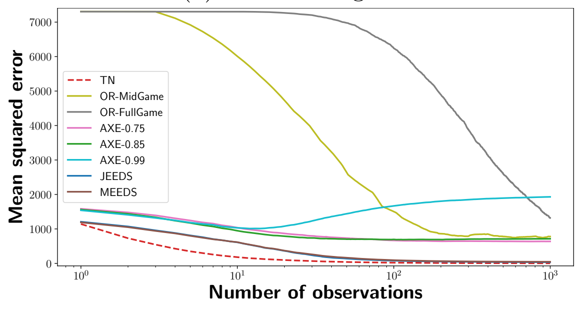
The results for the Flip agent (Figures 10c, 11b, and 12b) reveal the same successful estimators in each domain. This shows that the success of the JEEDS estimator at estimating execution skill for agents of limited rationality is not dependent on the estimator's rationality model matching the actual method used by the agent to select actions.



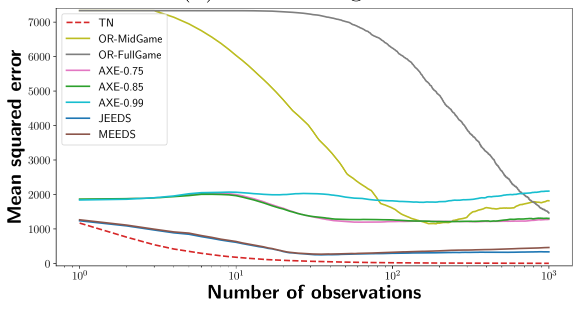
(a) Rational Agent



(b) Softmax Agent



(c) Flip Agent



(d) Deceptive Agent

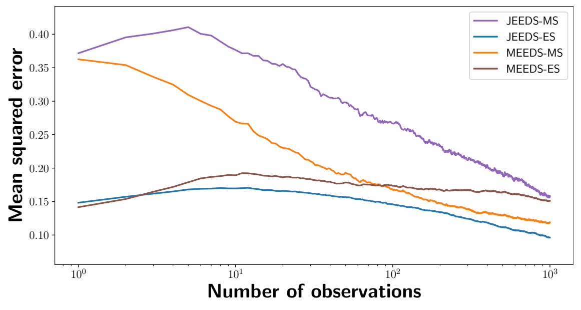
Figure 12: MSE of Execution Skill Estimates in 201-Darts

The final agent is the Deceptive agent. In the 1D-Domain, JEEDS works well (Figure 10d), but in the other two domains (Figures 11d and 12d) it results in a final MSE larger than it produces for the other agents. In both 2D-Darts and 201-Darts, the best performing estimator is JEEDS, followed by MEEDS, with the other estimators performing much worse. Interestingly, the AXE methods with all β values perform the poorest of all the methods by a large margin in the 2D-Darts domain. The Deceptive agent intentionally takes actions that are far away from the best actions to take in a state, and this appears to greatly confuse the AXE approach, since it relies on the distance between the observed action and the focal actions. The way that the Deceptive agent selects its actions results in an agent that all of the estimators have trouble estimating. JEEDS does the best at handling this deceptive behavior, but performs worse than with the other agents. This shows that the way non-rational behavior is being modeled struggles to handle the deliberate deception of this agent.

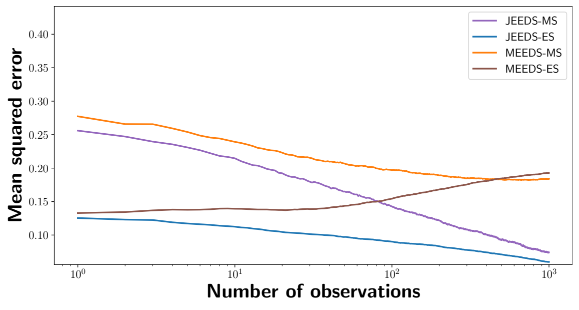
All together, these results show that the methods can produce accurate estimates when agents are known to be rational, or very close to rational. However, when agents have low decision-making skill, reasoning about both execution and decision-making skill together, as JEEDS does, is essential to producing robust and accurate execution skill estimates.

9.2 Estimating Decision-making Skill

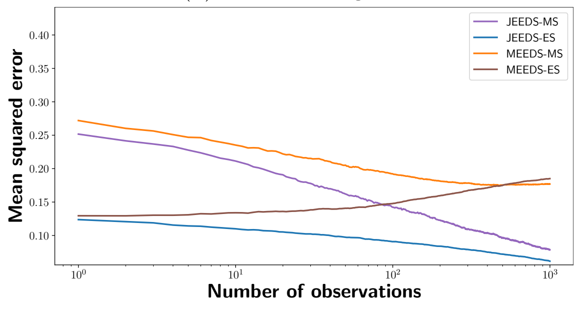
Figures 13 (1D-Darts), 14 (2D-Darts) and 15 (201-Darts) show the average MSE of the JEEDS and MEEDS decision-making estimates across all agents and experiments, where decision-making error is calculated as described in Section 8.3.2. As these are the first



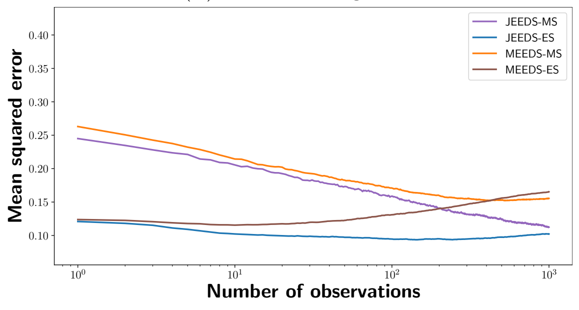
(a) Rational Agent



(b) Softmax Agent



(c) Flip Agent



(d) Deceptive Agent

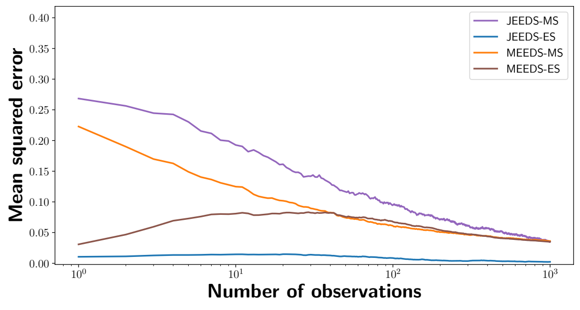
Figure 13: MSE of Decision-making Skill Estimates in 1D-Darts

methods for this task, there are no previous methods to compare against. As discussed earlier, in these results we report both the ES and MS estimates.

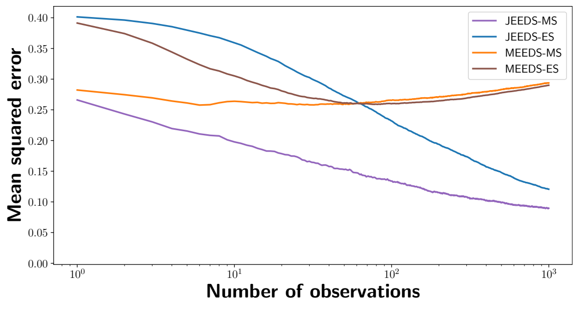
All estimators are able to achieve low MSE in all domains for rational agents. For all of the non-rational agents agents in 1D-Darts and 2D-Darts, the JEEDS methods work better than the MEEDS methods. The MEEDS estimates do not seem to get more accurate with more observations, while the JEEDS methods do improve over time. In 201-Darts, all of the decision-making skill estimates reach near zero MSE after 1000 observations for the Softmax and Flip agents, while for the Deceptive agent, none of the estimates achieve near zero MSE in 1000 observations. For JEEDS, the ES estimate outperforms the MS estimate with all the agents in 1D-Darts and 2D-Darts and for the Rational and Deceptive agents in 2D-Darts. For the Softmax and Flip agents in 2D-Darts the MS estimate is better. In 201-Darts both estimates converge to the same MSE after all 1000 observations.

Across all of the experiments, it appears that JEEDS generally achieves less error than MEEDS, coverging to low error in cases where MEEDS doesn't. One case where the methods seem to work well is when the agents are highly rational. The 201-Darts domain also appears to be an easier domain for estimating decision-making skill for all but the Deceptive agent. Producing accurate decision-making skill estimates across a range of agent types and skill levels is a challenging problem, and the JEEDS method is not quite as accurate at this task as it is at estimating execution skill across a wide range of decision-making skills.

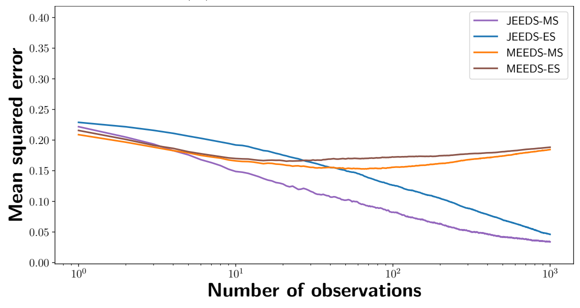
While reasoning about decision-making and execution skill jointly is essential for estimating execution skill accurately for agents with limited rationality, our experimental results indi-



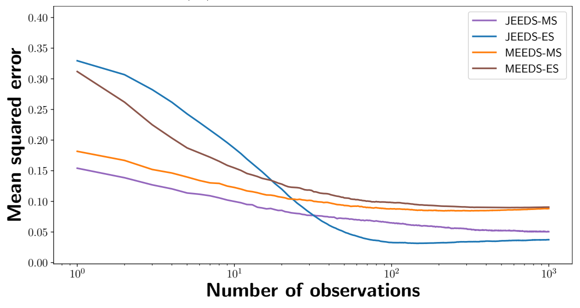
(a) Rational Agent



(b) Softmax Agent



(c) Flip Agent



(d) Deceptive Agent

Figure 14: MSE of Decision-making Skill Estimates in 2D-Darts

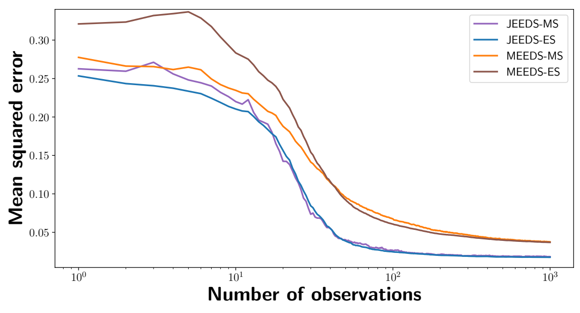
cate that the decision-making skill estimates produced by these methods are, on average, not quite as reliable as the execution skill estimates. In particular, as we explore in the next section, it seems that decision-making skill estimates are more negatively impacted by limited execution skill than execution skill estimates are negatively impacted by limited decision-making skill.

9.3 Interaction Between Skills

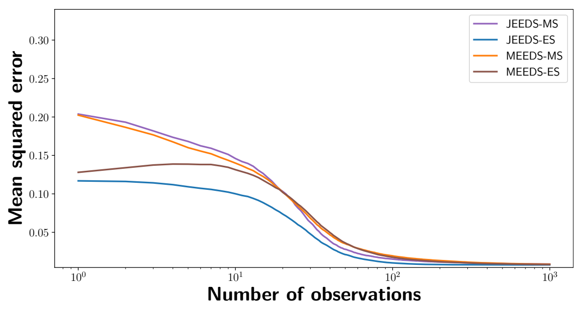
Having investigated the accuracy of the different skill estimators, averaged across all different values for the other skill, we now explore the interactions between these two skill types and how they influence the accuracy of the JEEDS estimator. JEEDS was selected because it performed the best overall for estimating both skill types across all agents and rationality levels. For consistency, JEEDS will utilize the ES estimate for both execution skill and decision-making skill estimates. The analysis in this section focuses on the Softmax agents. Results for the Flip and Deceptive agents yield similar conclusions and can be found in the appendix.

We first explore the estimation of skill over time, but now the Softmax agents are bucketed by skill level. Figure 16 shows the average MSE of the JEEDS skill estimates, averaged across all Softmax agents in each bucket. Each row shows the results for a different domain. Each graph indicates the skill ranges for the buckets.

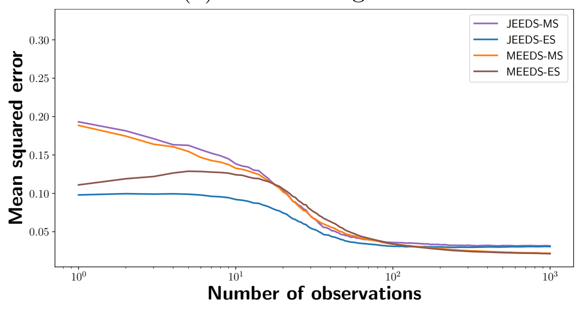
For the JEEDS execution skill estimates, shown in the left column, the MSE curves do not display huge variations between the different buckets, especially once enough observations



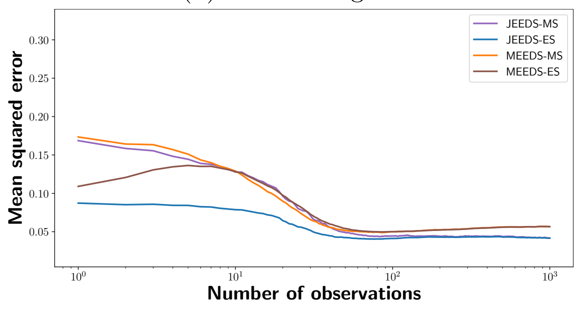
(a) Rational Agent



(b) Softmax Agent



(c) Flip Agent

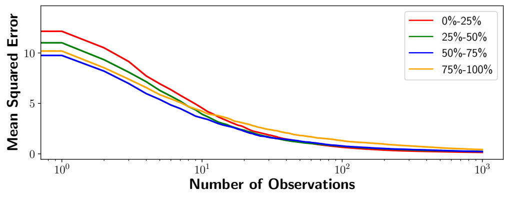


(d) Deceptive Agent

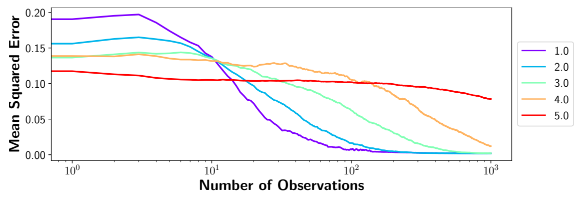
Figure 15: MSE of Decision-making Skill Estimates in 201-Darts

have been made. This was expected, as the figures showing average performance across all buckets demonstrated good convergence, as was discussed in Section 9.1. The buckets do exhibit minor differences, especially with fewer than 100 observations. These minor differences vary from domain to domain. Interestingly, the two groups with the slowest convergence in 1D-Darts and 2D-Darts are the agents on the extremes of the rationality spectrum. For 201-Darts, the slowest convergence is for the two highest decision-making skill buckets. The reasons why the MSE converges exactly as it does probably depend on the agent, the domain, and the bucket sizes and ranges. More lower-level insights will be explored shortly through different figures, but for the present we conclude that the execution skill estimates of the JEEDS estimator are accurate on agents of all skill levels, given enough observations. The decision-making skill level of an agent does have a minor impact on the exact convergence speed of the JEEDS estimator, but this impact is not enough to result in major differences in final estimation accuracy.

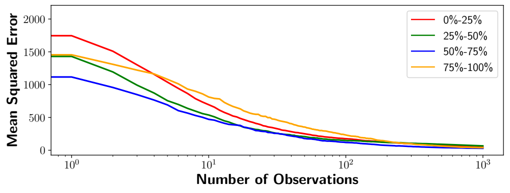
We now turn to the JEEDS decision-making skill estimates, shown in the right column of Figure 16. The agent buckets now correspond to execution skill ranges, with smaller numbers corresponding to less execution error and thus more skilled agents. It can be immediately seen that there is far more variation among bucket curves than there was in the execution skill MSE curves in the left column. Thus, an immediate conclusion is that the execution skill level of an agent has a huge impact on the ability of the JEEDS estimator to accurately estimate decision-making skill. In all domains, these estimates are more accurate when the agents are the most skilled at execution, in the smaller σ buckets. In 1D-Darts and 2D-Darts, the estimates get less and less accurate as the agents' execution error increases.



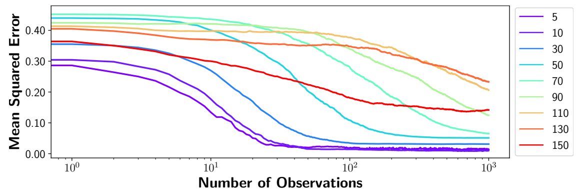
(a) 1D-Darts Execution Skill MSE



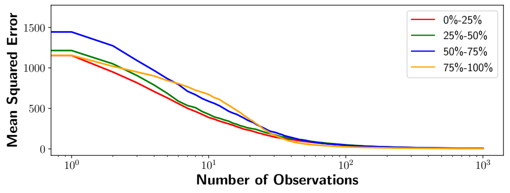
(b) 1D-Darts Decision-making Skill MSE



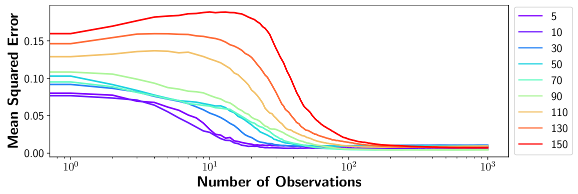
(c) 2D-Darts Execution Skill MSE



(d) 2D-Darts Decision-making Skill MSE



(e) 201-Darts Execution Skill MSE



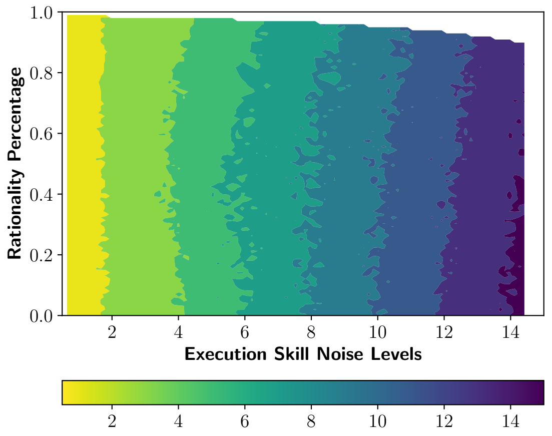
(f) 201-Darts Decision-making Skill MSE

Figure 16: JEEDS Skill Estimation Bucketed MSE on Softmax Agents

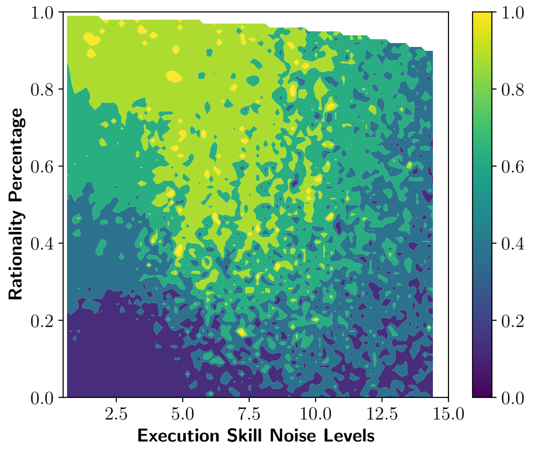
201-Darts provides a slightly different perspective, as the estimates are similarly accurate for all the buckets after 1000 observations. However, with fewer than 100 observations the same finding from the other domains holds: the more execution error an agent has, the less accurate the estimate of their decision-making skill estimate is.

Next, we investigate the impact of the interaction between the skills on estimation accuracy in a more qualitative way. Figure 17 shows a visualization of the average final skill estimates produced by the JEEDS estimator on Softmax agents across the entire joint space of skill combinations. Similar figures for other agents and estimators are included in the appendix. These visualizations are linear interpolations of the estimates, computed using the `scipy.interpolate.griddata` function with the `rescale` option enabled, from all experiments with the JEEDS estimator and Softmax agents. There are two subfigures for each experimental domain, showing the interpolated average final execution skill estimates and average final decision-making skill estimates respectively. The skill estimates are shown as colors, and each subfigure has a colorbar that shows the color scale used. Perfect estimation of skill would show up as an entire subfigure looking like a stretched copy of the colorbar.

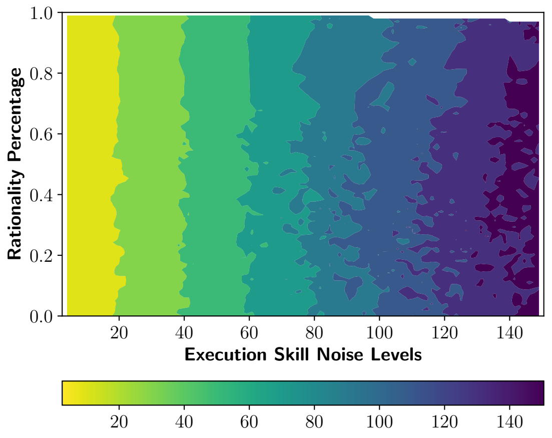
Focusing first on the execution skill plots in the left column, we see clear vertical bands of different colors that line up nicely with the colorbar. This shows again that the execution skill estimates are generally accurate across all decision-making skill levels, with a few small pockets of noise and error. In the 1D-Darts and 2D-Darts domains, the vertical color bands make a slight shift to the right as the decision-making skill is increased. This shows that as



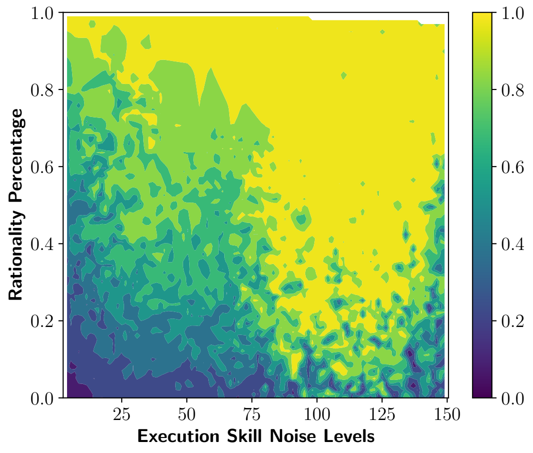
(a) Execution Skill Estimates in 1D-Darts



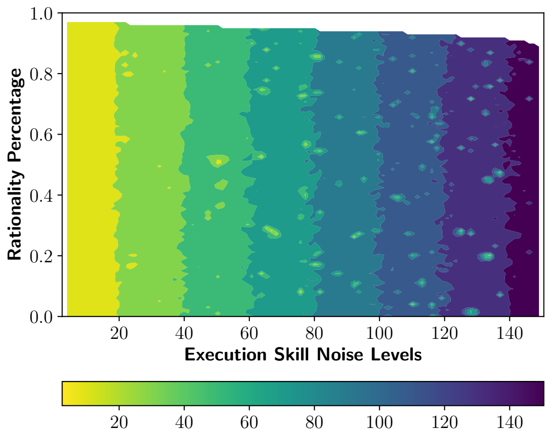
(b) Dec-making Skill Estimates in 1D-Darts



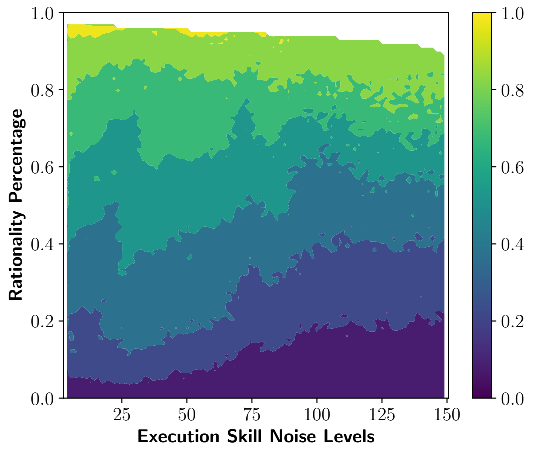
(c) Execution Skill Estimates in 2D-Darts



(d) Dec-making Skill Estimates in 2D-Darts



(e) Execution Skill Estimates in 201-Darts



(f) Dec-making Skill Estimates in 201-Darts

Figure 17: Interpolated JEEDS Estimates on Softmax Agents

an agent's decision-making skill increases, the estimate of their execution skill can sometimes be a bit lower than it would have been if they had less decision-making skill. Overall, the left column of Figure 17 visually demonstrates the ability of the JEEDS estimator to provide accurate execution skill estimates across the entire spectrum of skill level combinations.

The interpolated decision-making skill estimates in the right column of Figure 17 show a more nuanced view of the performance of the JEEDS estimator. At a high level, there are vast regions of the joint parameter space where the estimates are very inaccurate. However, in each domain, there are regions where the estimates at least exhibit the correct progression from bad to good decision-making skill. In 1D-Darts (Figure 17b) we can see that for low execution noise levels (roughly 0.0-5.0) the decision-making skill estimates could be used at a high level to distinguish between agents at different decision-making skill levels, since the estimates change in the correct direction as skill increases. When agents have more noise in their actions, and thus have less execution skill, the decision-making skill estimates produced by JEEDS look closer and closer to random. A similar phenomenon can be observed in 2D-Darts (Figure 17d), where at execution noise levels above 100 mm the decision-making skill estimates are very inaccurate, while at execution noise levels lower than that the estimates show a reasonable progression. In both 1D-Darts and 2D-Darts when the execution noise is very close to zero, the decision-making skill estimates are quite close to matching what they should be. Interestingly, in the 201-Darts domain, the execution skill of the agent does not appear to have a big impact on the decision-making skill estimates produced, as they look fairly consistent across all execution noise levels.

To summarize, in all three experimental domains JEEDS can accurately estimate execution skill across the entire range of decision-making skill levels that were included in our experiments. However, decision-making skill is much harder to estimate accurately. The estimates are most accurate across all domains when execution noise is very close to 0. With slightly more execution noise the estimates produced by JEEDS can correctly, but crudely, group or order agents by decision-making skill. In 1D-Darts and 2D-Darts even this rough agent ordering is not possible when agents have a significant amount of execution noise. We conclude that in these domains high execution noise overwhelms and masks any decision-making skill information contained in the agents' executed actions.

10. An Application: Pitcher Execution Skill Estimation in Baseball

The previous experimental domains allowed us to evaluate the accuracy of the proposed skill estimation methods since the true skill of the experimental agents was known. We now apply the proposed methods to the task of estimating agent skill on data from a real-world domain, where the true skill level of agents is not known. This is done in the domain of Major League Baseball (MLB). In the sport of baseball, pitchers throw a baseball towards a batter who is standing at home plate, with the goal of getting them out. Batters, on the other hand, are trying to hit the ball into play with a bat, allowing them time to run around the bases. Batters receive a *strike* if they swing and hit the ball out of play, swing and miss, or if they decide not to swing but the pitch crosses through the strike zone. They receive a *ball* if they do not swing and the pitch misses the strike zone. The strike zone is shown as the rectangle in Figures 18 and 19. It extends vertically roughly from a batter's knees to their armpits and is horizontally limited by the width of home plate. After four balls the batter is allowed to advance to first base, while if they receive three strikes, then they are out[3]. It is perhaps obvious that the execution skill of a pitcher has a huge impact on their success and the target locations and pitching strategy they should employ. For present purposes we consider a pitcher's execution skill to be their accuracy at having their pitch pass through an intended location in or near the strike zone.

10.1 Baseball Pitching Data

Rich baseball data has been collected for many years. Data about baseball pitches is especially detailed, including pitch velocity, movement, spin rate, location where the pitch left the pitcher's hand, and location where the pitch crossed home plate. We accessed this data using `pybaseball` (LeDoux, 2017), a Python package that provides access to baseball data collected from different public sources. Matching the assumptions made in this paper, the pitch data does not include the intended pitch location, as this is not observed. Each pitch has a specific *type* which is chosen from a discrete set of types by the pitcher. Example pitch types are fastball, change-up, and breaking ball. These types differ in how the pitcher holds and throws the ball and have different characteristics. The main differences are in how fast the pitch goes and how much the ball curves. Each pitcher typically has a small number of pitch types that they have honed and can utilize in games. The type for each pitch is included in the data. Our goal is to use the proposed methods to estimate the execution skill of MLB pitchers on specific pitch types.

10.2 Evaluating Skill Estimation in Baseball

As previously noted, with baseball data the true skill level of each pitcher is not known. This means that we cannot exactly measure the accuracy of the skill estimates the methods produce. In order to give some evaluation of the reasonableness of the estimates, a high-level comparison will be performed with another statistic that is generally used to describe a pitcher's command, or ability to hit their intended target. This statistic is the number of walks (BB) a pitcher issues per inning that they pitch (IP), which is called BB/IP. Pitchers with good control generally do not issue a lot of walks, as these are typically undesirable outcomes resulting from not being able to consistently hit the strike zone.

The proposed methods will be evaluated by their ability to generate pitcher rankings that are consistent with the rankings of pitchers by BB/IP. Of course, the rankings will not be identical, as there will not be a perfect correlation between a pitcher's execution skill level and their BB/IP. But this will give us some idea that the methods are producing reasonable estimates.

To make this analysis the most clear, we first ranked all of the pitchers in MLB by BB/IP during the 2021 season. We then included in our experiment the top ten and bottom ten pitchers according to this metric. We will consider the estimates produced by our methods to be reasonable if they can frequently distinguish between these two groups in the rankings that they produce. This inclusion of the pitchers who appear to be at the extreme ends

3. For more complete rules of baseball, see: https://www.rulesofsport.com/sports/baseball.html

of the control spectrum will give the execution skill estimation methods the best chance of showing reasonable differentiation consistent with the BB/IP statistic.

Since different pitchers have different pitch types in their repertoires, we cannot use a single type across all pitchers. In fact, for the chosen 20 pitchers, there was not a single pitch type that all of the pitchers had a significant number of in the data. Additionally, some pitch types differ wildly in how much they move and how hard they might be to control. However, the different pitch types can be categorized into three broad classes: fastballs, breaking balls, and off-speed pitches. To have the best chance of comparing pitchers as fairly as possible, in these results we focus on the class of fastballs, which is composed of three types: four-seam fastball, two-seam fastball, and cutter. All of the pitchers we chose have data on at least one of the three types of fastball pitches. We will report each pitcher's fastball skill levels, which will be their skill level averaged across whichever of the three fastball pitch types they have enough of in the data.

10.3 Application Details

We now describe the additional details which were necessary to apply the proposed methods to MLB data.

10.3.1 Expected Utility Function

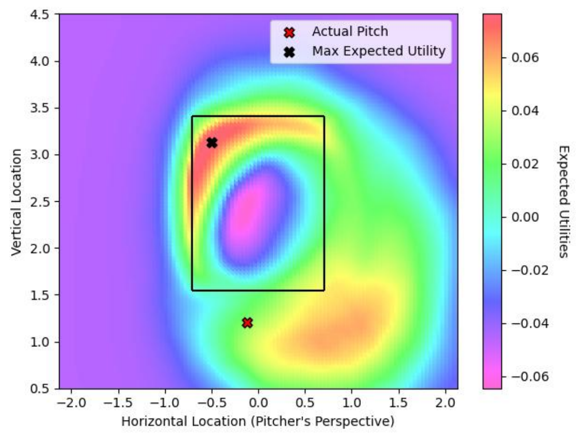
The OR and JEEDS methods both explicitly utilize reward information. For the baseball setting, this means we require a reward function defined over pitch locations for a pitcher facing a given batter. To this end we utilize the OptimusPitch (OP) model proposed in (Melville et al., 2023). OP is trained to produce the expected utility for a completely described pitch against a specific batter, given the game context of the at-bat. To generate a raw expected utility for each different location a pitch could cross home plate, OP is queried once for each possible pitch location. The location information about the pitch is modified, but the other data (velocity, spin-rate, etc.), is left unchanged. This gives us a grid of expected utility for the pitcher, much like a dartboard. An example raw expected utility grid is shown in Figure 18a. The remainder of Figure 18 shows this utility convolved with different execution noise levels, demonstrating how the optimal aiming point depends on execution skill.

10.3.2 Focal points for AXE

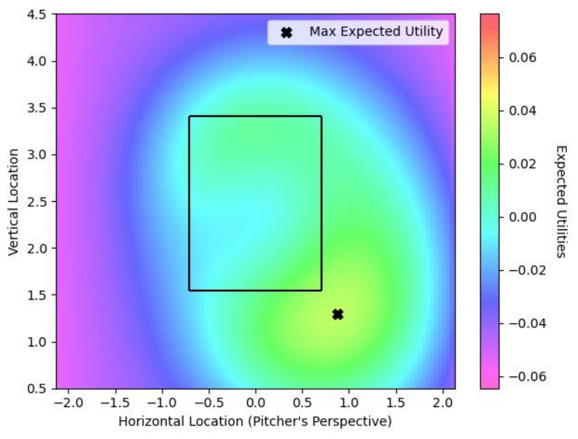
The set of focal actions required by AXE is shown in Figure 19 and was hand-designed for the baseball domain. The 21 points include 5 in the strike zone, 8 on the edge of the strike zone, and 8 just outside the strike zone. Additionally, for each pitch, the location yielding the maximum expected utility for each execution skill level hypothesis was included, as long as it was at least 2 inches from any focal point already in the set.

10.4 Experimental Procedure

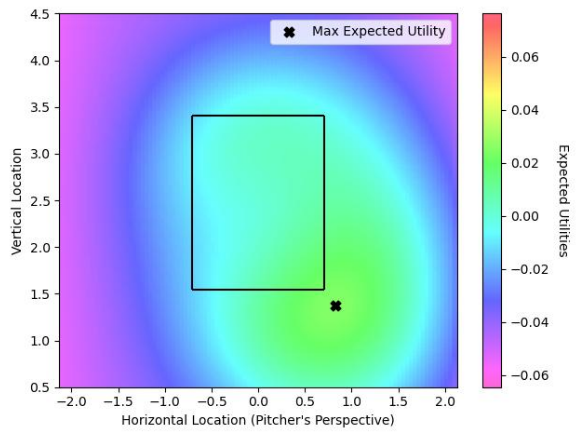
The experiments were conducted as follows. For a given pitcher and fastball pitch type, the most recent 1000 pitches from the 2021 season were obtained from the data. When fewer were available, all available pitches were used. If any pitcher had fewer than 80 pitches of a



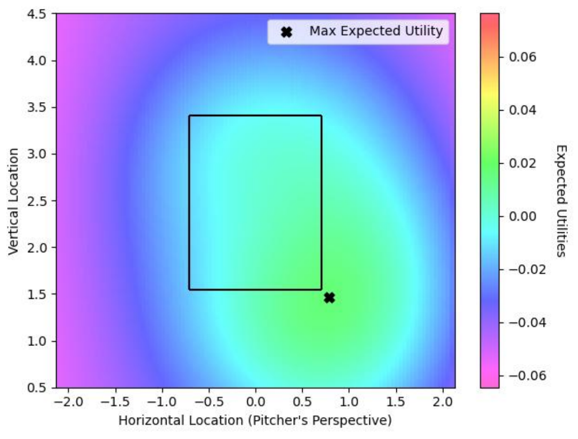
(a) Raw Expected Utility (i.e. $\sigma = 0.0$)



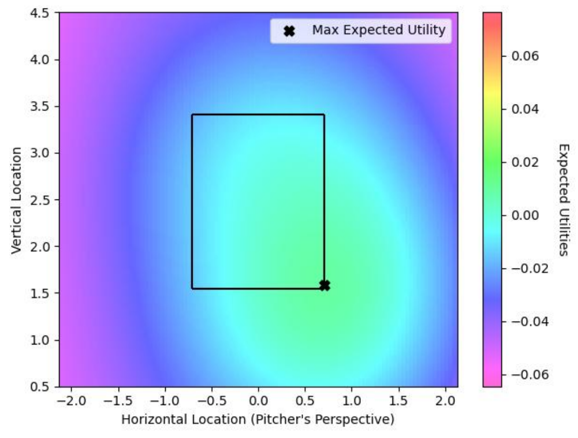
(b) $\sigma = 0.50$



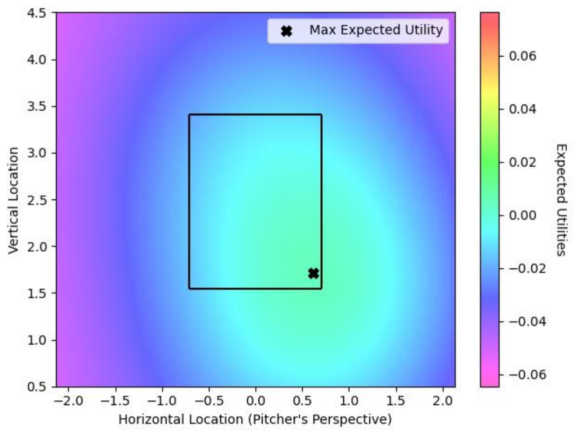
(c) $\sigma = 0.63$



(d) $\sigma = 0.75$



(e) $\sigma = 0.88$



(f) $\sigma = 1.0$

Figure 18: Expected utility maps for different execution skill levels on a sample change-up pitch from Joe Mantiply facing Joey Bart. Colors indicate the expected utility of each target location with the given σ. The black X indicates the optimal target location for the pitch, which varies as σ changes.

given type, that pitch type for that pitcher was not included in the experimental results, so as not to include high variance estimates. For each pitch, the raw expected utility grid was generated using the OP model. This utility grid and the observed location of the actual pitch were passed into all of the estimation methods. The results are composed from the final skill estimates, determined after processing all relevant data.

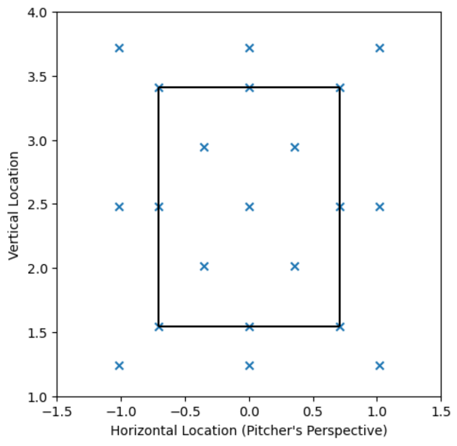


Figure 19: AXE focal points for MLB data. The black box indicates the strike zone.

10.4.1 Experiment Details

All methods modeled a player's execution error as noise added to the intended action, where this noise is drawn from a symmetric two-dimensional Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = [0, 0]$ and $\boldsymbol{\Sigma} = \sigma I_2$. Thus, the methods estimate a separate σ for each pitcher-pitch type combination, which we consider to be the player's execution skill.

66 hypothesis skill levels were used by the methods for both execution and decision-making skills. The execution skill hypotheses were distributed between 0.17 ft ($\sim$ 2 inches) and 2.81 ft ($\sim$ 33.7 inches). An execution skill of 2.81 ft was selected as the upper limit since a pitcher is able to hit the strike zone only 20% of the time with this execution skill level when aiming in the middle. Out of the 66 execution skill level hypotheses, 60 were distributed between 0.17 ft and 1.0 ft, and the remaining 6 were distributed between 1.0 ft and 2.81 ft. This was done because the best pitch location can vary for execution skill hypotheses smaller than 1 ft, whereas they generally stay the same, near the middle of the strike zone, for those greater than 1 ft. The decision-making skill levels hypotheses were distributed as described in Section 8.3.

The set of possible pitch locations ranges from -2.13 ft to 2.13 ft horizontally and from -2.5 ft to 6.6 ft vertically[4]. The pitch locations were discretized with a resolution of 0.5 inches in both dimensions. These ranges were selected as they include at least 95% of all the observed pitch locations available in the data from the 2021 MLB season. Any pitch location outside this region was assigned the minimum value from the corresponding utility grid.

4. A negative vertical pitch location corresponds with the ball hitting the ground before reaching home plate

All β values in the set $\{0.5, 0.75, 0.85, 0.9, 0.95, 0.99\}$ were used for the AXE method in the experiments. The different β values all produced very similar estimates. The results presented in the next section are with $\beta = 0.95$, which was a good β value in the darts experiments for agents with good decision-making skill. This choice was made under the assumption that professional baseball pitchers are highly incentivized to make good decisions about where to aim, and we expect most pitchers to be fairly competent at that task.

10.5 Results

Table 1 shows the average fastball execution skill estimates produced by the OR, AXE, and JEEDS methods for the pitchers in the experiment. In addition, the table also shows the decision-making skill estimate produced by JEEDS for all of the pitchers. The cells in each column are shaded corresponding to their rank from best (lowest $\tilde{\sigma}$ or highest $\tilde{\lambda}$) to worst. The ranking of each pitcher within each column is shown to the left of their estimate.

The OR and JEEDS execution skill estimation methods are able to perfectly differentiate between the top-10 and bottom-10 groups. Each of these estimators ranks the top-10 pitchers by BP/IP as more skilled at execution than the bottom-10. The AXE method has two pitchers from each group that are ranked in the other group, but overall still seems to concur with most of the top-10/bottom-10 categorizations. The fact that the skill estimation methods are so consistent with the BP/IP rankings is evidence that the execution skill estimates are reasonable. It should be noted that the estimators came to these conclusions without knowing about walks and/or innings. The methods' inputs are simply the expected utility map for each pitch along with the pitch's actual location. Thus, we conclude that the execution skill estimates are reasonable for MLB pitchers.

The right column shows the decision-making skill estimate $\tilde{\lambda}$ produced by the JEEDS estimator. While there is no existing baseball statistic that we believe correlates with this skill estimate, it is notable that the ranking derived from these estimates is quite different from the BP/IP rankings (higher $\tilde{\lambda}$ indicates more rational decision-making). In fact, six of the top-10 pitchers according to decision-making skill (shown in order in the $\tilde{\lambda}$ column on the right) are found in the bottom-10 of the BP/IP rankings. To get a sense of how these $\tilde{\lambda}$ values correlate with our $\lambda_{\%}$ rationality measure, the rightmost column shows an estimate of the $\lambda_{\%}$ value corresponding to each of the estimated $\tilde{\lambda}$ values.

Each pitcher's $\tilde{\lambda}_{\%}$ value estimate was computed as follows. For each pitcher and pitch type from the experiment, fifty random pitches were chosen. The corresponding raw expected utility map, as a function of pitch location, was generated for each pitch. Each raw expected utility map was then convolved with the execution noise corresponding to the final JEEDS estimate for that pitcher and pitch type, resulting in a map showing the expected utility for the pitcher to aim the pitch in each possible location, assuming the estimate of their execution noise was accurate. From this map, the expected utility of the best possible target location (r_M) and a target location selected uniformly at random (r_U) were computed. This was compared to the expected utility the pitcher would get, assuming they selected a target pitch location according to Equation 2 with their estimated λ parameter, denoted as r_λ. The estimated rationality percentage for the pitch was then computed as $\frac{r_\lambda - r_U}{r_M - r_U}$, as described in Section 8.3.2. The values computed in this way from each of the sampled pitches were

Pitcher	BP/IP		$\tilde{\sigma}$ (ft)						$\tilde{\lambda}$			
			OR		AXE		JEEDS		JEEDS		$\tilde{\lambda}_{\%}$	
Joe Mantiply	1	0.084	5	0.73	1	0.52	2	0.64	19	216.1	15	91.3
Chris Martin	2	0.091	3	0.70	5	0.55	4	0.66	8	487.1	14	91.3
Jacob deGrom	3	0.124	7	0.75	4	0.55	5	0.67	12	398.9	7	95.0
Corey Kluber	4	0.128	10	0.78	14	0.64	10	0.73	14	340.1	18	90.5
Emmanuel Clase	5	0.139	9	0.78	6	0.56	7	0.71	18	222.6	19	90.1
Aaron Nola	6	0.146	2	0.70	3	0.54	1	0.64	10	437.3	10	93.2
Ross Stripling	7	0.149	4	0.72	8	0.61	6	0.70	9	467.9	11	92.7
Austin Pruitt	8	0.149	8	0.77	9	0.61	9	0.73	11	426.2	9	93.6
George Kirby	9	0.151	1	0.67	2	0.53	3	0.65	7	542.9	6	95.3
Lance Lynn	10	0.156	6	0.74	15	0.64	8	0.72	13	342.2	17	91.0
Trevor Richards	11	0.556	18	0.86	19	0.71	12	0.75	20	191.7	20	86.1
Caleb Smith	12	0.556	17	0.86	17	0.67	16	0.78	15	278.2	16	91.2
Ryne Stanek	13	0.577	14	0.82	13	0.64	14	0.78	1	1213.8	1	98.5
Jordan Hicks	14	0.580	13	0.82	12	0.63	13	0.77	17	267.9	13	91.4
Yusei Kikuchi	15	0.586	15	0.84	16	0.64	18	0.81	4	803.2	8	94.7
Austin Davis	16	0.589	12	0.80	7	0.61	11	0.75	16	275.0	12	91.7
Joan Adon	17	0.603	11	0.80	10	0.61	15	0.78	6	590.8	5	95.8
Tucker Davidson	18	0.673	16	0.85	11	0.62	17	0.80	5	677.6	4	96.6
Jake Diekman	19	0.736	19	0.86	18	0.67	19	0.84	2	1203.0	2	98.3
Tanner Scott	20	0.741	20	0.99	20	0.76	20	0.97	3	1181.9	3	97.4

Table 1: Skill estimates for top-10 and bottom-10 MLB pitchers according to BP/IP

averaged for each pitch type. The $\tilde{\lambda}_{\%}$ value shown in the rightmost column was then computed as the average across the pitch types included in the experiment for each pitcher.

All of the pitchers have rationality percentages between 86.1 and 98.5% and so the differences in their decision-making skill (λ) estimates do not correspond to gross differences in their effectiveness. In the other experimental domains, these rationality percentages all correspond with highly rational agents. Our general conclusion is that none of the pitchers appear to especially struggle with this portion of their game, although we hypothesize that this information might still be helpful for coaches. This is because the coaching techniques to help pitchers improve decision-making might be very different than those used to improve other aspects of pitching. Even these small differences in decision-making ability might be impactful over the course of an entire MLB season or player's career.

These skill estimates have the potential to impact baseball in several ways. First, accurate execution skill estimates can be embedded in larger decision-making aids for pitchers, to help them make informed decisions about where to aim their pitches. An example of this was the game-theoretic model of the interaction between batter and pitcher which computed an equilibrium target location and pitch type for a pitcher facing a batter (Melville et al., 2023). This model crucially required an accurate idea of how precisely each pitcher could hit their intended target. Other possible uses include being able to identify pitchers who have the execution skill to succeed, but might need to make better decisions. This information could also be useful for batters in helping them understand the accuracy of the pitchers they face and what types of strategies they might employ as a result. Baseball is one example of a real-world setting where the proposed execution and decision-making skill estimation methods can be used to improve our understanding of agent performance and our ability to help people make better decisions and experience more success in their activities.

11. Conclusion and Future Work

Execution and decision-making skill constitute two important characteristics of agents in many continuous domains. In this article, we presented several skill estimation methods, each of which utilizes different information which can be observed in the environment to produce an estimate. These methods were experimentally evaluated in several computational domains with a variety of agents, as well as on Major League Baseball pitching data. The OR method relies only on the reward obtained by the agent, while AXE focuses on executed actions. The best-performing method across the experiments was JEEDS, which uses information about actions and rewards to jointly reason about execution and decision-making together. This joint reasoning allowed it to accurately estimate execution skill even when agents made decisions that were far from perfectly rational, cases where the other methods perform poorly. This novel method also does not require the specification of a β parameter and a set of focal points, as are needed for AXE. Reasoning about decision-making skill significantly improves the robustness and accuracy of execution skill estimation across a wide range of agent rationality levels and types. This unique combination of robustness and accuracy mean JEEDS is clearly the best method to apply to new settings, especially when there is uncertainty about the rationality of the observed agents.

In the future, there are several possible extensions of this work that could be explored. First, it would be interesting to relax the assumption of the decision-making component having correct beliefs about the execution distribution χ. It is not obvious that humans, for example, always know their own execution error. The baseball results could be used and explored more deeply by baseball teams to improve their evaluation and coaching of pitchers. The estimation methods could be used to evaluate human or robotic agent skill in real-world tasks of interest in addition to baseball. Some of these novel domains might necessitate the use of different error distributions than were used in this work, or the estimation of execution skill in higher dimensions. In many domains, it is expected that agents' skill levels might change over time. We plan to adapt the JEEDS method to produce time-varying skill estimates which can be used to track agent skill levels in settings where this applies. Finally, it would be interesting to explore if other components of an MDP, like the reward or transition functions, can be estimated given knowledge of an agent's execution and decision-making skill levels.

Appendix A

A. Rationality Percentage

In this section, we detail the process of converting from a λ_s estimate to a $\lambda_{\%}$ estimate. As mentioned in the text, this conversion relies on a separate dataset of the Softmax agent acting in the domain. We first describe how this dataset was created, and then how it is used in the conversion function.

A.1 Conversion Dataset Generation

The following process was followed to create a dataset for the rationality parameter conversion. First, a Softmax agent was created for each combination of execution skill and decision-making skill from the following sets:

- 1D-Darts: 6 different execution noises [0.5, 1.0, 2.0, 3.0, 4.0, 5.0] and 100 λ values, distributed logarithmically between 0.00001 and 100.
- 2D-Darts & 201-Darts: 33 different execution noises distributed evenly between 2.5 and 150.5 and 100 λ values, distributed logarithmically between 0.00001 and 32.

Each of these agents played the same 1000 states (1D-Darts) or 50 states (2D-Darts), or the 201 possible states (201-Darts), and the average $\lambda_{\%}$ for each agent was computed. The dataset then stores all of this information, which says for each execution noise and decision-making skill parameter combination that was tried, what the corresponding average $\lambda_{\%}$ obtained was.

A.2 Conversion Process

While estimating the skill of an agent, the JEEDS method produces a λ_s estimate of that agent's rationality parameter, as well as an estimate of that agent's execution noise parameter, σ_s. To convert this λ_s value to a $\lambda_{\%}$ value, we interpolate from the conversion dataset as follows:

- Find the execution noise levels present in the dataset that are just below (σ_b) and just above (σ_a) σ_s. If σ_s is greater than all (or less than all) we only use the one that it is closest to.
- For each of those σ values, interpolate using the λ_s estimate on the dataset to get a $\lambda_{\%}$ estimate.
- Then interpolate on those estimates using the σ_s estimate. If using only one σ from the dataset, then simply return the corresponding interpolated $\lambda_{\%}$ estimate from the previous step.

Figure 20 displays the data in the 1D-Darts dataset, where each line corresponds to a different execution noise level, which is shown in the legend. The x-axis shows the rationality parameter used by the Softmax agent (λ_s), while the y-axis shows the rationality percentage obtained by that agent ($\lambda_{\%}$). Figures 21 and 22 show the same thing for the 2D-Darts and 201-Darts datasets respectively. These figures help visualize the rationality parameter conversion that is necessary for evaluating the decision-making skill estimates.

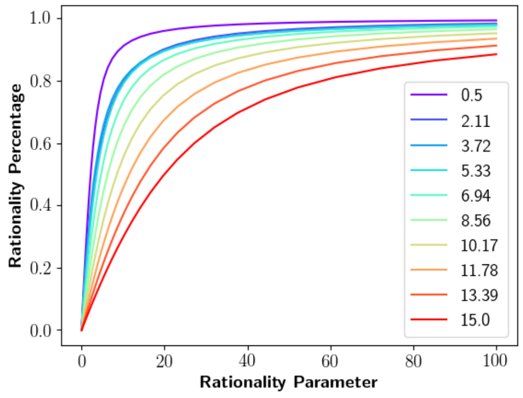


Figure 20: 1D-Darts Conversion Dataset for Softmax Agents

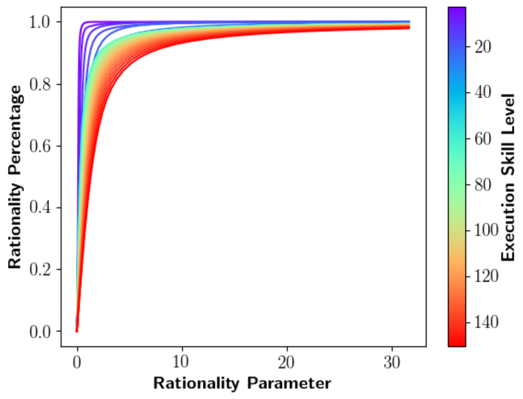


Figure 21: 2D-Darts Conversion Dataset for Softmax Agents

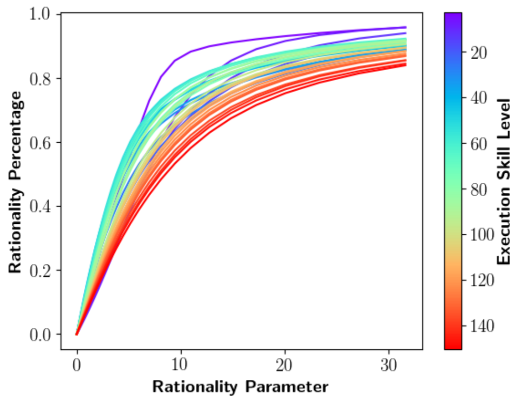


Figure 22: 201-Darts Conversion Dataset for Softmax Agents

B. Experiments Performed (For Each Domain and Agent Type)

Domain	Agents			
	Rational	Softmax	Flip	Deceptive
1D-Darts	3,892	14,681	15,136	14,914
2D-Darts	1,860	7,368	5,556	5,494
201-Darts	1,216	13,803	3,603	3,507

Table 2: Experiments performed

C. Results: Confidence Intervals

In this section, we include tables that give the exact final MSE and 95% confidence interval upper and lower bounds for the data used to generate the figures presented in the paper. This information can be used to determine which differences between methods, after all observations, are significant at a confidence level of 95%.

C.1 Data Tables Corresponding to Figure 2

Methods	MSE	$\pm$
OR	0.0992	0.0059

Table 3: Rational Agents in 1D-Darts

C.2 Data Tables Corresponding to Figure 4

Methods	MSE	±
AXE-Beta0.75	7.5056	0.2447
AXE-Beta0.85	4.3800	0.1446
AXE-Beta0.99	0.3207	0.0114
OR	0.0992	0.0059

Table 4: Rational Agents in 1D-Darts

C.3 Data Tables Corresponding to Figure 5

Methods	MSE	±
AXE-Beta0.75	6.77	0.13
AXE-Beta0.85	4.08	0.08
AXE-Beta0.99	1.56	0.09
OR	1.19	0.03

Table 5: Softmax Agents with $\lambda_{\%} \in [25\%, 75\%]$ in 1D-Darts

Methods	MSE	±
AXE-Beta0.75	6.61	0.22
OR	5.22	0.14
AXE-Beta0.99	5.03	0.15
AXE-Beta0.85	4.82	0.14

Table 6: Softmax Agents with $\lambda_{\%} \in [0\%, 25\%]$ in 1D-Darts

C.4 Data Tables Corresponding to Figure 7

Methods	MSE	±
AXE-Beta0.75	7.51	0.24
AXE-Beta0.85	4.38	0.14
JEEDS	0.56	0.02
MEEDS	0.48	0.02
AXE-Beta0.99	0.32	0.01
OR	0.0992	0.0059

Table 7: Rational Agents in 1D-Darts

Methods	MSE	±
AXE-Beta0.75	6.61	0.22
OR	5.22	0.14
AXE-Beta0.99	5.03	0.15
AXE-Beta0.85	4.82	0.14
MEEDS	2.97	0.15
JEEDS	0.15	0.01

Table 8: Softmax Agents with $\lambda_{\%} \in [0\%, 25\%]$ in 1D-Darts

C.5 Data Tables Corresponding to Figure 10

Methods	MSE	±
AXE-Beta0.75	7.5056	0.2447
AXE-Beta0.85	4.3800	0.1446
JEEDS	0.5559	0.0235
MEEDS	0.4808	0.0224
AXE-Beta0.99	0.3207	0.0114
OR	0.0992	0.0059

Table 9: Rational Agents in 1D-Darts

Methods	MSE	±
AXE-Beta0.75	6.7271	0.1094
AXE-Beta0.85	4.2689	0.0729
AXE-Beta0.99	2.4524	0.0742
OR	2.1784	2.2788
MEEDS	1.7590	0.0541
JEEDS	0.2608	0.0069

Table 10: Softmax Agents in 1D-Darts

Methods	MSE	±
AXE-Beta0.75	6.6423	0.1072
AXE-Beta0.85	4.2142	0.0704
AXE-Beta0.99	2.5907	0.0733
OR	2.2252	0.0484
MEEDS	1.4185	0.0440
JEEDS	0.2880	0.0072

Table 11: Flip Agents in 1D-Darts

Methods	MSE	±
AXE-Beta0.75	7.6259	0.1268
AXE-Beta0.85	5.1526	0.1048
AXE-Beta0.99	3.1355	0.1096
OR	2.2255	0.0485
MEEDS	0.6789	0.0189
JEEDS	0.3398	0.0079

Table 12: Deceptive Agents in 1D-Darts

C.6 Data Tables Corresponding to Figure 11

Methods	MSE	±
AXE-Beta0.75	731.9400	36.6308
AXE-Beta0.85	460.2289	23.5640
MEEDS	53.7908	5.5968
AXE-Beta0.99	37.5875	1.8285
JEEDS	26.2581	3.6996
OR	17.5040	1.3548

Table 13: Rational Agents in 2D-Darts

Methods	MSE	±
AXE-Beta0.99	955.7333	35.3907
AXE-Beta0.75	838.5062	22.9607
AXE-Beta0.85	734.1883	24.8714
OR	700.8745	25.4861
MEEDS	674.2374	28.1585
JEEDS	32.98170	1.7022

Table 14: Softmax Agents in 2D-Darts

Methods	MSE	±
AXE-Beta0.99	693.8648	30.5871
MEEDS	681.4206	31.6763
AXE-Beta0.75	669.0564	21.1769
AXE-Beta0.85	537.5876	20.7917
OR	383.4875	14.2179
JEEDS	52.3452	2.2400

Table 15: Flip Agents in 2D-Darts

Methods	MSE	±
AXE-Beta0.99	2335.4008	97.6855
AXE-Beta0.75	2048.2245	88.5945
AXE-Beta0.85	2014.2396	91.9437
OR	671.7718	26.2501
MEEDS	366.3712	20.7320
JEEDS	222.2695	6.4422

Table 16: Deceptive Agents in 2D-Darts

C.7 Data Tables Corresponding to Figure 12

Methods	MSE	±
AXE-Beta0.75	620.7358	41.3294
OR-FullGame	442.7862	102.2378
AXE-Beta0.85	382.9585	26.2363
OR-MidGame	225.0919	26.7429
AXE-Beta0.99	22.4947	1.7685
MEEDS	15.7327	1.2648
JEEDS	11.4587	1.1036

Table 17: Rational Agents in 201-Darts

Methods	MSE	±
AXE-Beta0.99	2364.9953	67.8775
OR-FullGame	1647.3625	56.7145
OR-MidGame	1248.4901	40.6826
AXE-Beta0.85	1104.4084	35.9062
AXE-Beta0.75	954.1515	26.8332
MEEDS	27.2867	1.2093
JEEDS	5.1639	0.1448

Table 18: Softmax Agents in 201-Darts

Methods	MSE	±
AXE-Beta0.99	1929.5418	95.2748
AXE-Beta0.99	1929.5418	95.2748
OR-FullGame	1318.4880	102.4277
OR-MidGame	777.411400	49.7180
AXE-Beta0.85	713.8930	48.0338
AXE-Beta0.75	638.2259	35.7792
MEEDS	46.8949	2.9871
JEEDS	34.4753	1.2912

Table 19: Flip Agents in 201-Darts

Methods	MSE	±
AXE-Beta0.99	2096.7184	159.4918
OR-MidGame	1817.1322	132.2493
OR-FullGame	1464.9147	111.359
AXE-Beta0.85	1305.8166	132.6981
AXE-Beta0.75	1277.4170	126.5356
MEEDS	460.7001	36.6836
JEEDS	334.1817	30.0620

Table 20: Deceptive Agents in 201-Darts

C.8 Data Tables Corresponding to Figure 13

Methods	MSE	±
JEEDS-MS	0.1587	0.0084
MEEDS-ES	0.1513	0.0092
MEEDS-MS	0.1188	0.0075
JEEDS-ES	0.0962	0.0056

Table 21: Rational Agents in 1D-Darts

Methods	MSE	±
MEEDS-ES	0.1928	0.0037
MEEDS-MS	0.1840	0.0036
JEEDS-MS	0.0742	0.0023
JEEDS-ES	0.0598	0.0017

Table 22: Softmax Agents in 1D-Darts

Methods	MSE	±
MEEDS-ES	0.1851	0.0037
MEEDS-MS	0.1769	0.0036
JEEDS-MS	0.0786	0.0023
JEEDS-ES	0.0615	0.0017

Table 23: Flip Agents in 1D-Darts

Methods	MSE	±
MEEDS-ES	0.1653	0.0035
MEEDS-MS	0.1554	0.0033
JEEDS-MS	0.1125	0.0029
JEEDS-ES	0.1023	0.0025

Table 24: Deceptive Agents in 1D-Darts

C.9 Data Tables Corresponding to Figure 14

Methods	MSE	±
MEEDS-MS	0.0362	0.0033
JEEDS-MS	0.0354	0.0048
MEEDS-ES	0.0350	0.0033
JEEDS-ES	0.0024	0.0010

Table 25: Rational Agents in 2D-Darts

Methods	MSE	±
MEEDS-MS	0.2939	0.0076
MEEDS-ES	0.2900	0.0076
JEEDS-ES	0.1207	0.0040
JEEDS-MS	0.0897	0.0029

Table 26: Softmax Agents in 2D-Darts

Methods	MSE	±
MEEDS-ES	0.1883	0.0062
MEEDS-MS	0.1846	0.0061
JEEDS-ES	0.0462	0.0021
JEEDS-MS	0.0339	0.0017

Table 27: Flip Agents in 2D-Darts

Methods	MSE	±
MEEDS-ES	0.0906	0.0040
MEEDS-MS	0.0883	0.0039
JEEDS-MS	0.0505	0.0019
JEEDS-ES	0.0375	0.0014

Table 28: Deceptive Agents in 2D-Darts

C.10 Data Tables Corresponding to Figure 15

Methods	MSE	±
MEEDS-MS	0.0376	0.0024
MEEDS-ES	0.0369	0.0023
JEEDS-MS	0.0183	0.0016
JEEDS-ES	0.0176	0.0014

Table 29: Rational Agents in 201-Darts

Methods	MSE	±
JEEDS-MS	0.0087	0.0002
MEEDS-MS	0.0085	0.0003
MEEDS-ES	0.0081	0.0003
JEEDS-ES	0.0077	0.0002

Table 30: Softmax Agents in 201-Darts

Methods	MSE	±
JEEDS-MS	0.0316	0.0012
JEEDS-ES	0.0307	0.0012
MEEDS-MS	0.0218	0.0010
MEEDS-ES	0.0215	0.0010

Table 31: Flip Agents in 201-Darts

Methods	MSE	±
MEEDS-MS	0.0566	0.0025
MEEDS-ES	0.0565	0.0025
JEEDS-MS	0.0417	0.0026
JEEDS-ES	0.0416	0.0026

Table 32: Deceptive Agents in 201-Darts

C.11 Data Tables Corresponding to Figure 16 (Subfigures a,c, & e)

Bucket	MSE	±
1.00	0.1050	0.0099
0.50	0.0593	0.0057
0.75	0.0575	0.0059
0.25	0.0390	0.0041

Table 33: Softmax Agents in 1D-Darts

Bucket	MSE	±
0.50	66.1640	4.7134
0.25	43.7318	2.6902
1.00	39.4370	3.4167
0.75	29.2267	2.7780

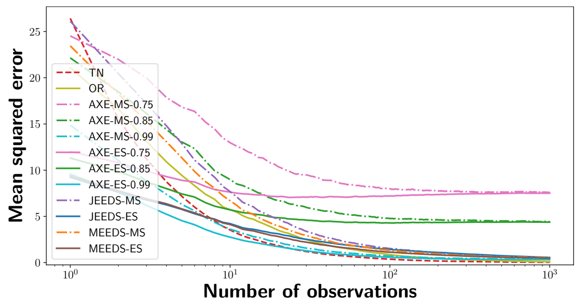
Table 34: Softmax Agents in 2D-Darts

Bucket	MSE	±
0.50	6.4254	0.4105
0.75	6.2464	0.3831
0.25	5.2013	0.2705
1.00	3.9918	0.1944

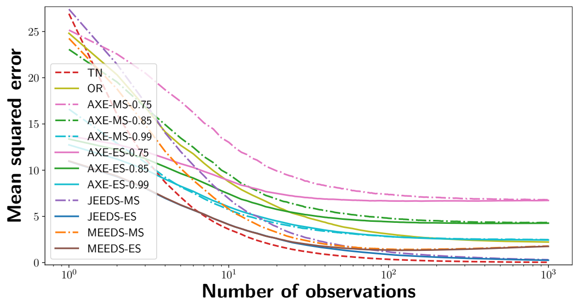
Table 35: Softmax Agent in 201-Darts

D. Average MSE of all Execution Skill Estimation Methods

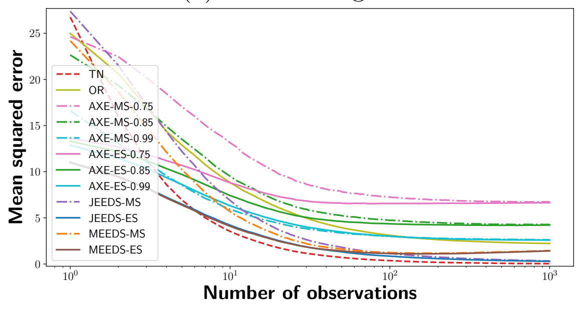
The following figures present the average MSE for all the execution skill estimation methods, including the AXE-MS and JEEDS-MS/MEEDS-MS, for the different domains.



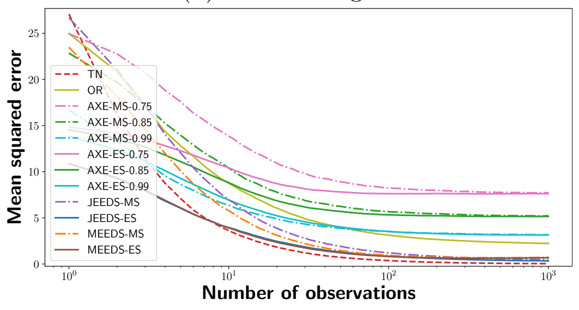
(a) Rational Agent



(b) Softmax Agent

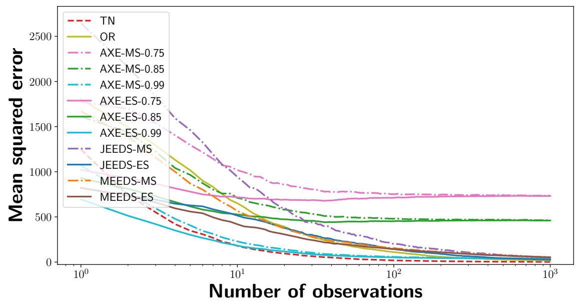


(c) Flip Agent

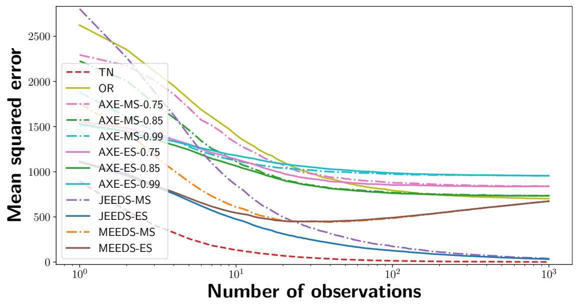


(d) Deceptive Agent

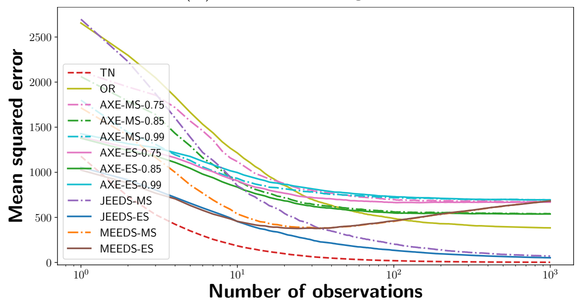
Figure 23: MSE of Execution Skill Estimates 1D-Darts



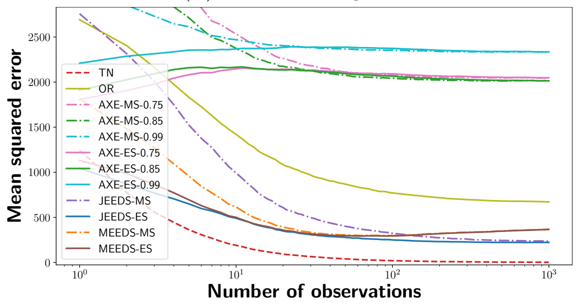
(a) Rational Agent



(b) Softmax Agent

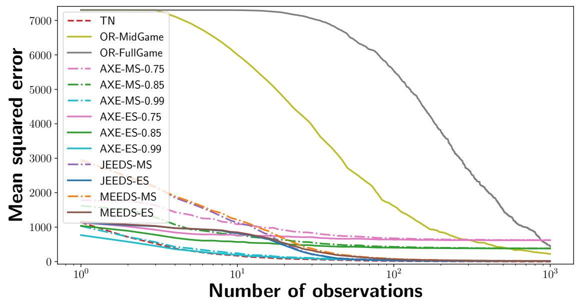


(c) Flip Agent

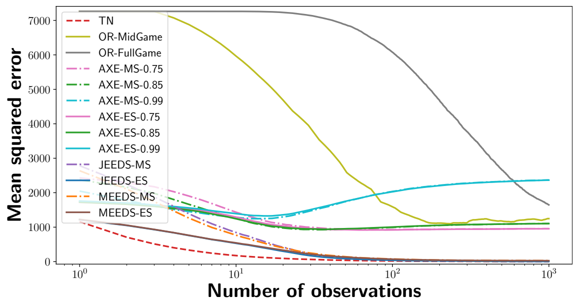


(d) Deceptive Agent

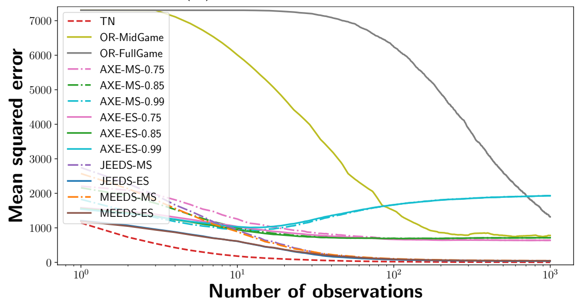
Figure 24: MSE of Execution Skill Estimates in 2D-Darts



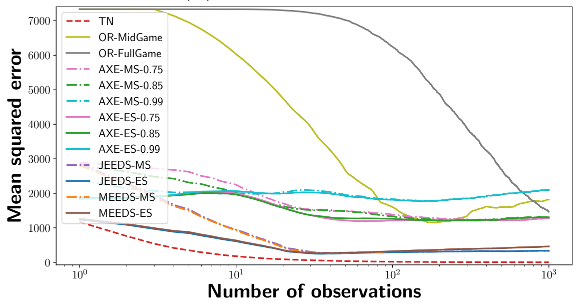
(a) Rational Agent



(b) Softmax Agent



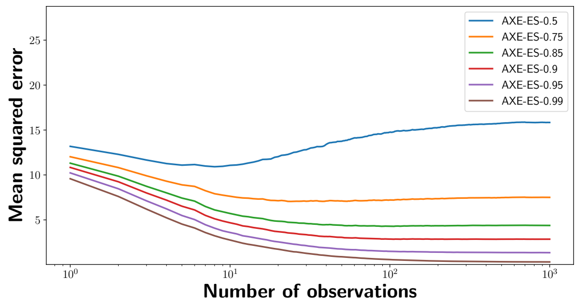
(c) Flip Agent



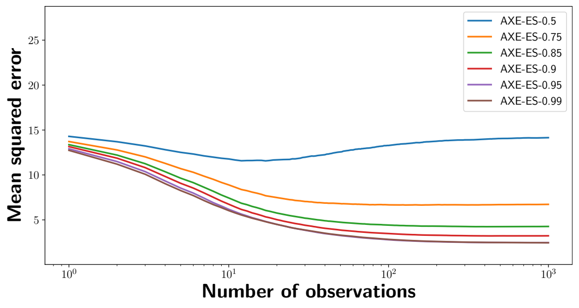
(d) Deceptive Agent

Figure 25: MSE of Execution Skill Estimates in 201-Darts

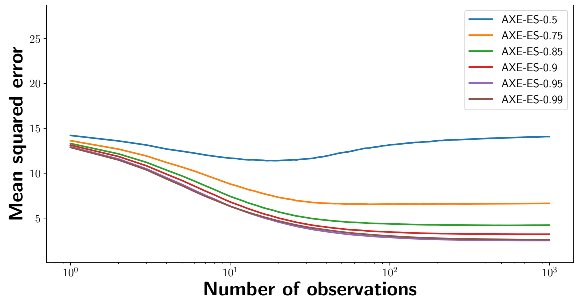
E. Comparison of the Performance of Different β values for AXE



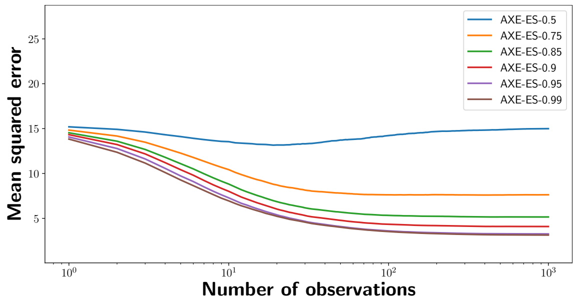
(a) Rational Agent



(b) Softmax Agent

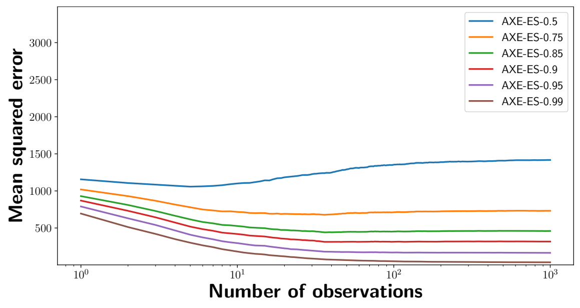


(c) Flip Agent

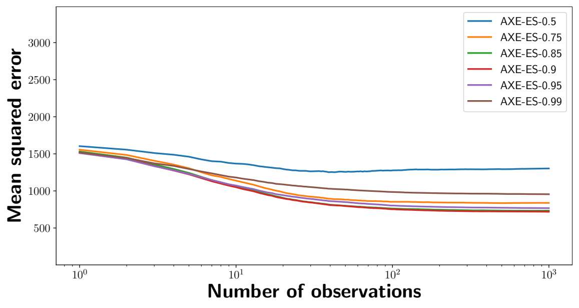


(d) Deceptive Agent

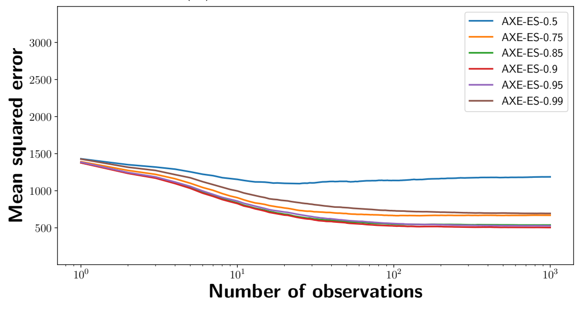
Figure 26: MSE of Execution Skill Estimates 1D-Darts



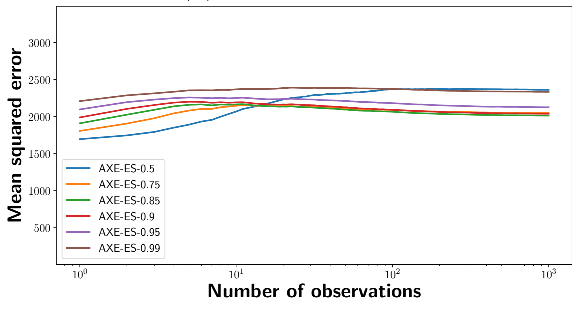
(a) Rational Agent



(b) Softmax Agent

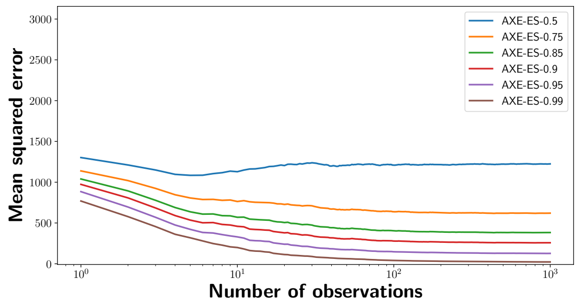


(c) Flip Agent

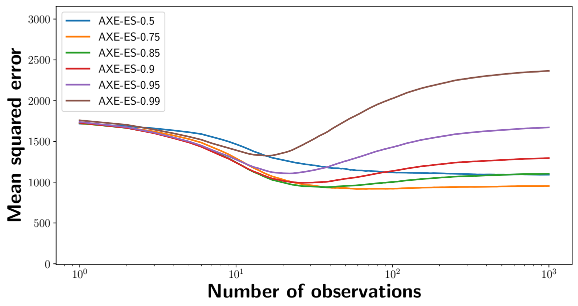


(d) Deceptive Agent

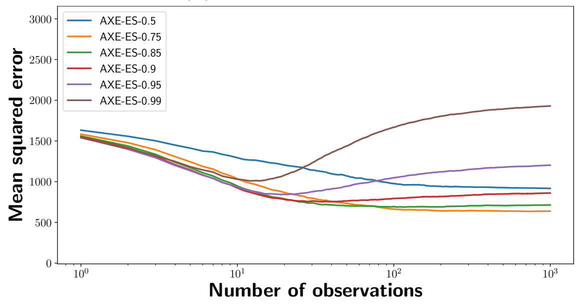
Figure 27: MSE of Execution Skill Estimates in 2D-Darts



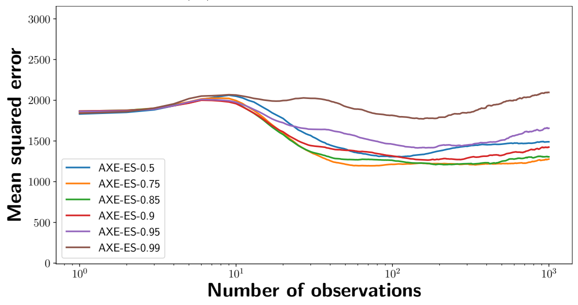
(a) Rational Agent



(b) Softmax Agent



(c) Flip Agent

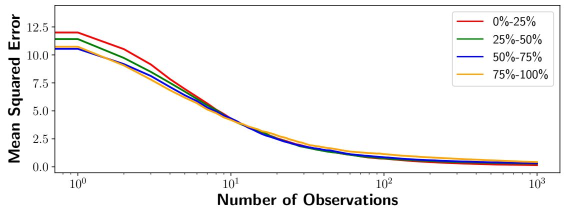


(d) Deceptive Agent

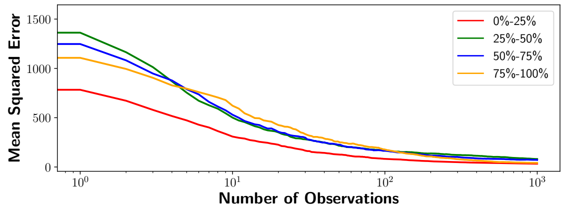
Figure 28: MSE of Execution Skill Estimates in 201-Darts

F. JEEDS Execution Skill Estimates Grouped By Decision-making Skill Buckets - Other Non-Rational Agents

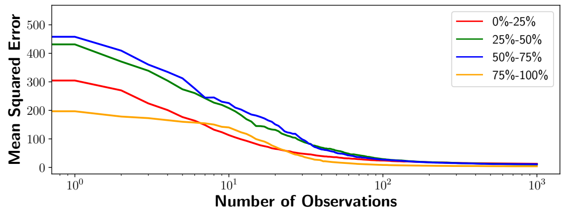
F.1 Flip Agents



(a) 1D-Darts

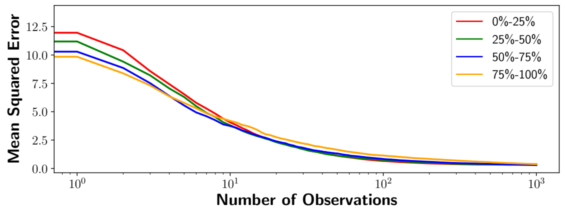


(b) 2D-Darts

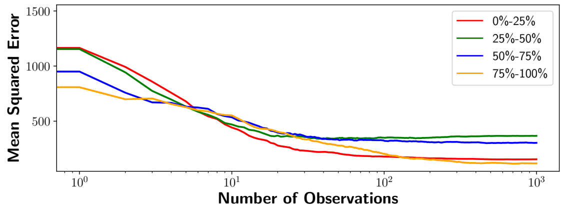


(c) 201-Darts

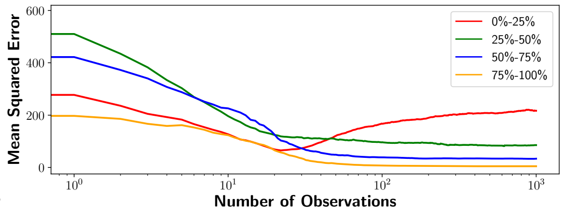
F.2 Deceptive Agents



(a) 1D-Darts



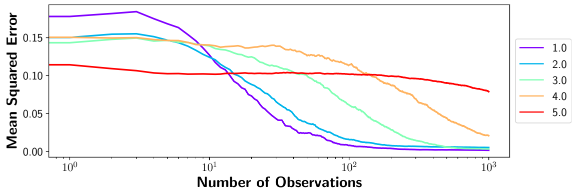
(b) 2D-Darts



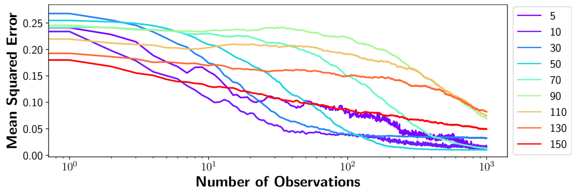
(c) 201-Darts

G. JEEDS Decision-making Skill Estimates Grouped By Execution Skill Buckets - Other Non-Rational Agents

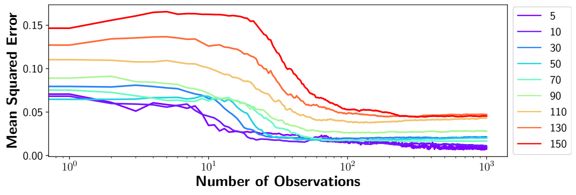
G.1 Flip Agents



(a) 1D-Darts

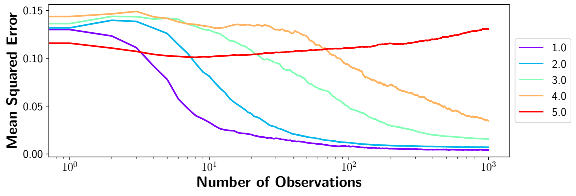


(b) 2D-Darts

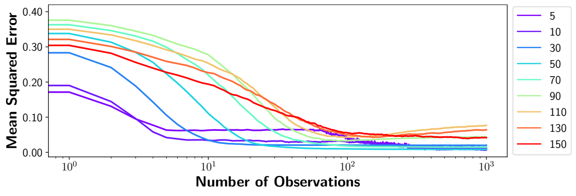


(c) 201-Darts

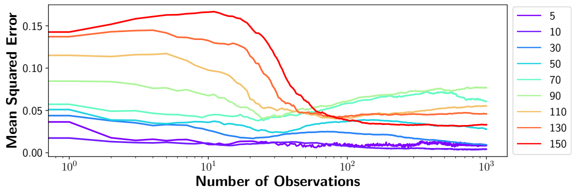
G.2 Deceptive Agents



(a) 1D-Darts



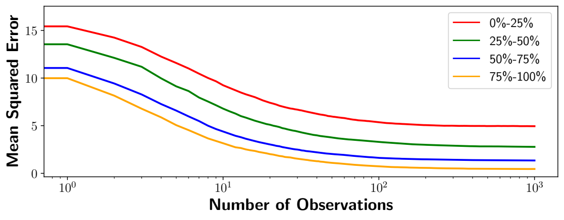
(b) 2D-Darts



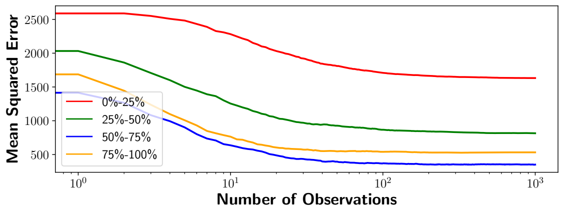
(c) 201-Darts

H. AXE Execution Skill Estimates Grouped By Decision-making Skill Buckets - All Agents

H.1 Softmax Agents

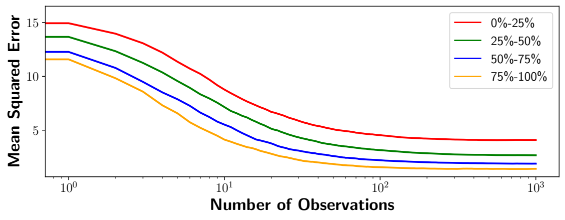


(a) 1D-Darts

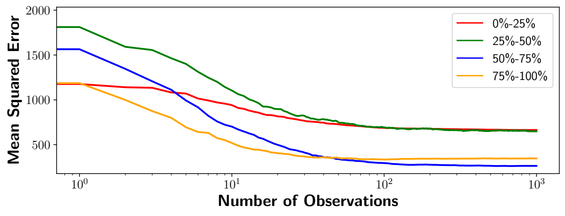


(b) 2D-Darts

H.2 Flip Agents

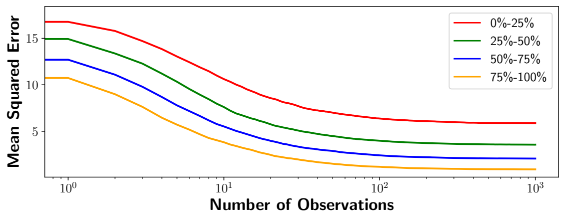


(a) 1D-Darts

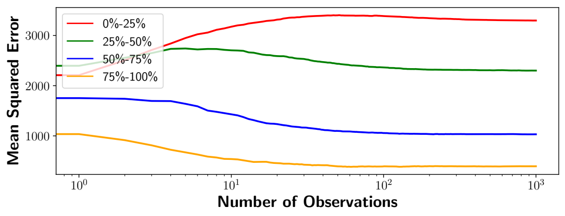


(b) 2D-Darts

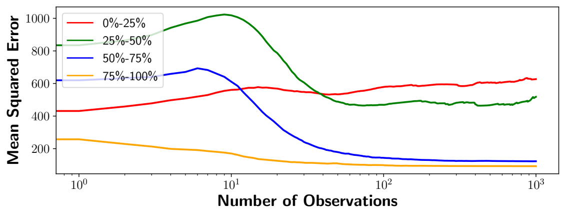
H.3 Deceptive Agents



(a) 1D-Darts



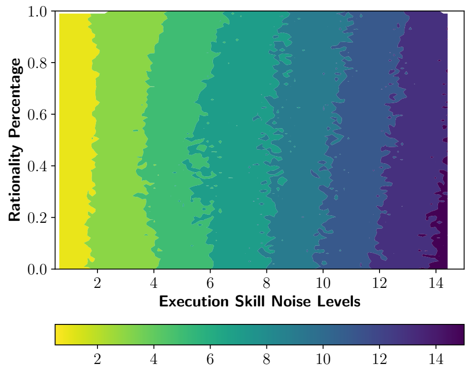
(b) 2D-Darts



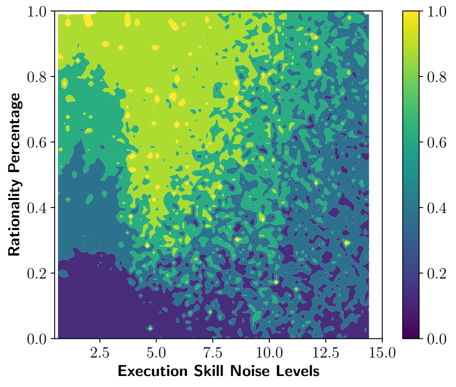
(c) 201-Darts

I. JEEDS Estimates - Other Non-Rational Agents

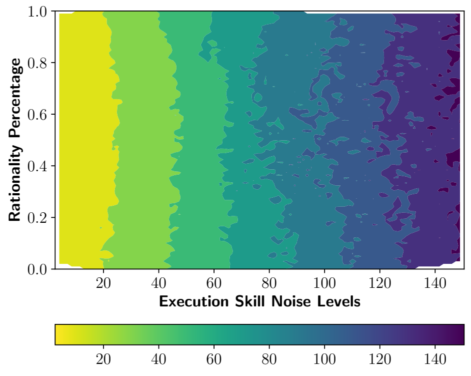
I.1 Flip Agents



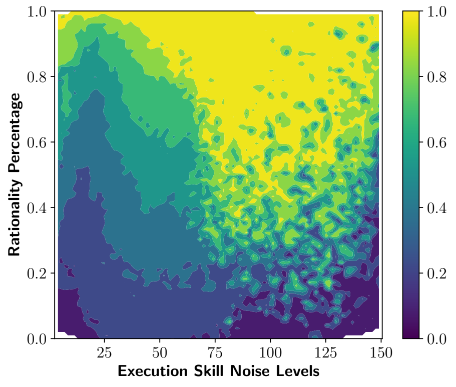
(a) Execution Skill Estimates - 1D



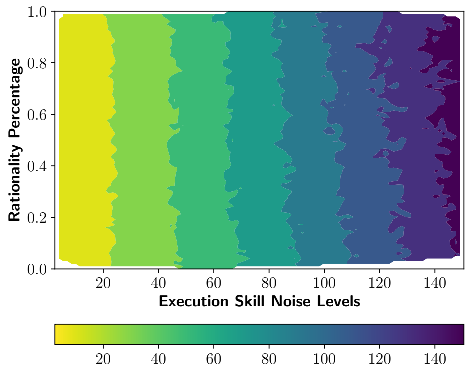
(b) Dec-making Skill Estimates - 1D



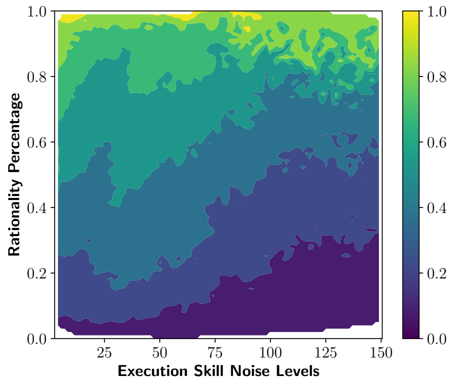
(c) Execution Skill Estimates - 2D



(d) Dec-making Skill Estimates - 2D



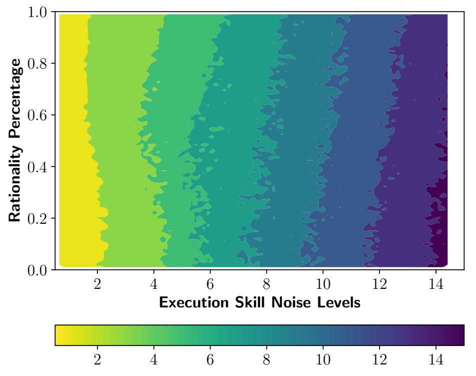
(e) Execution Skill Estimates - 201 Darts



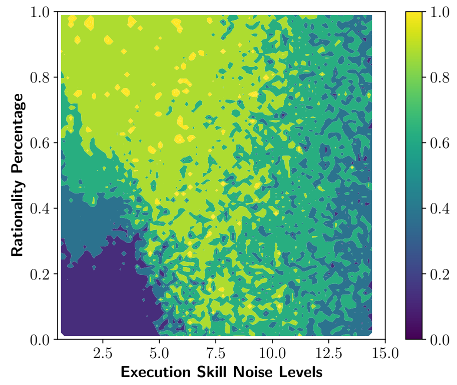
(f) Dec-making Skill Estimates - 201 Darts

Figure 38: JEEDS Estimates on Flip Agents

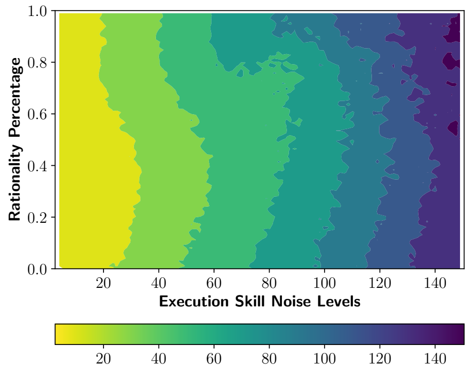
I.2 Deceptive Agents



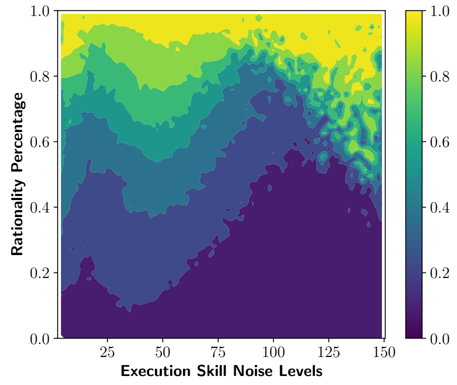
(a) Execution Skill Estimates - 1D



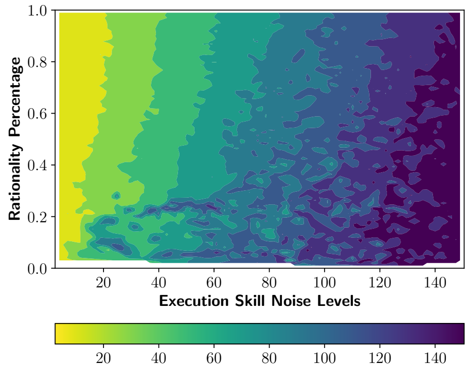
(b) Dec-making Skill Estimates - 1D



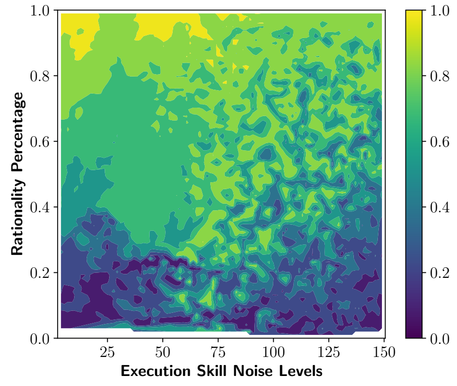
(c) Execution Skill Estimates - 2D



(d) Dec-making Skill Estimates - 2D



(e) Execution Skill Estimates - 201 Darts

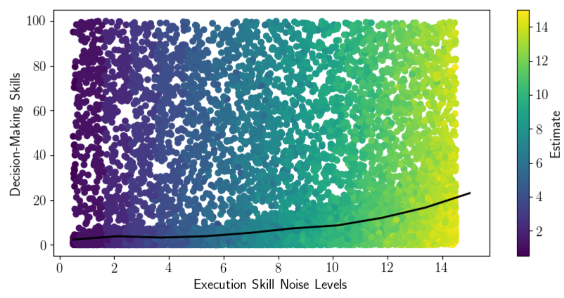


(f) Dec-making Skill Estimates - 201 Darts

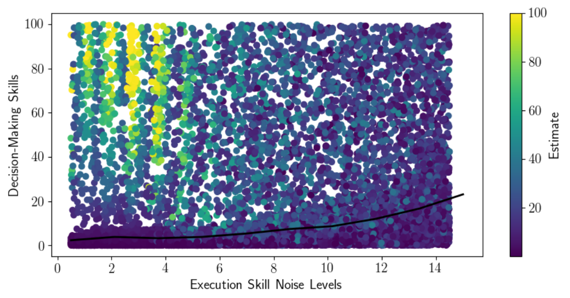
Figure 39: JEEDS Estimates on Deceptive Agents

J. JEEDS Estimates: Decision-Making Skill vs. Rationality Percentage

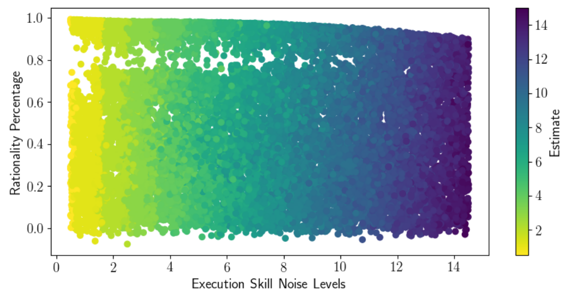
The following figures present the final JEEDS execution skill estimates (left column) and JEEDS decision-making skill (right column) across the different parameter combinations used for the experiments for a Softmax agent in each domain. For each figure, the top row presents the decision-making skill parameters (λ_s in this case), whereas the bottom row presents the rationality percentage ($\lambda_{\%}$). The black line on the top row represents where the 50% rationality percentage falls.



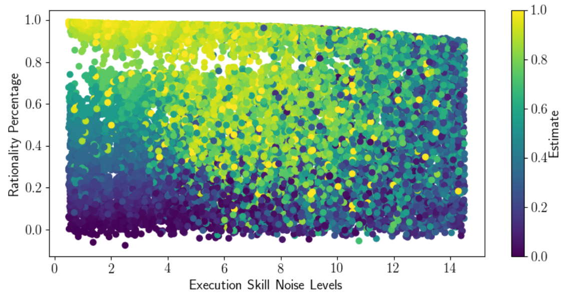
(a) Execution Skill Estimates



(b) Dec-Making Skill Estimates

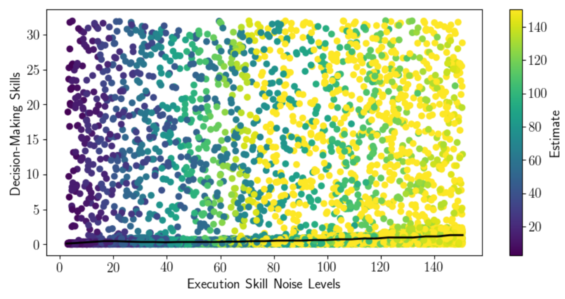


(c) Execution Skill Estimates

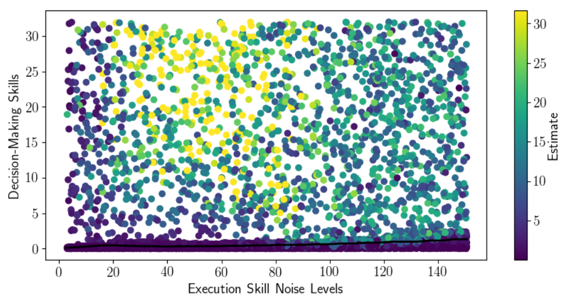


(d) Rationality Percentage Estimates

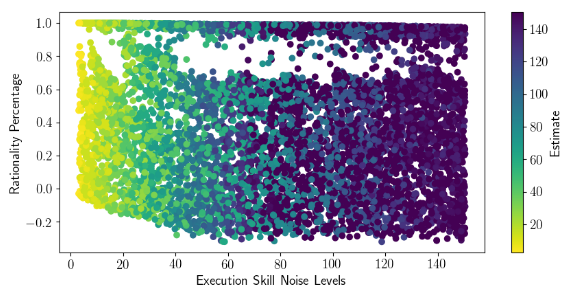
Figure 40: JEEDS Estimates for the Softmax Agent in 1D-Darts



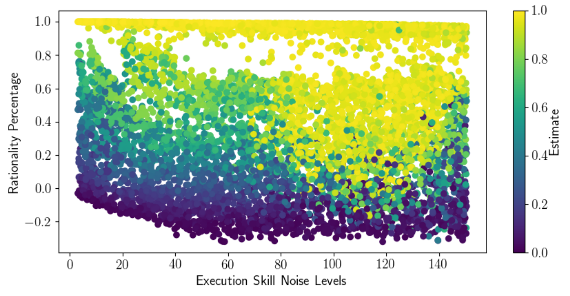
(a) Execution Skill Estimates



(b) Dec-Making Skill Estimates

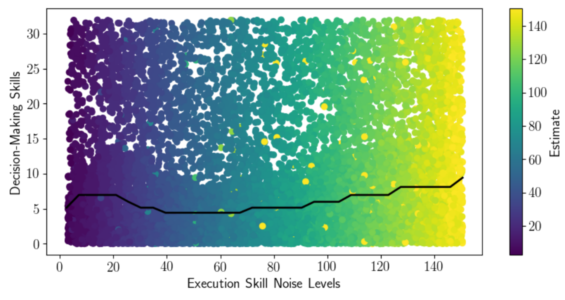


(c) Execution Skill Estimates

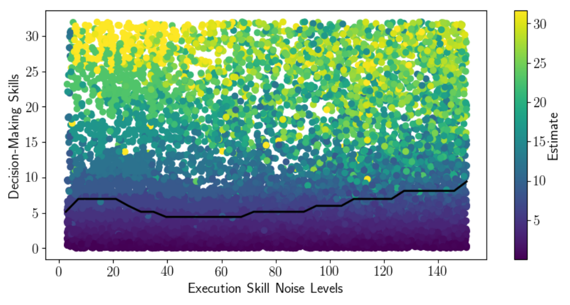


(d) Rationality Percentage Estimates

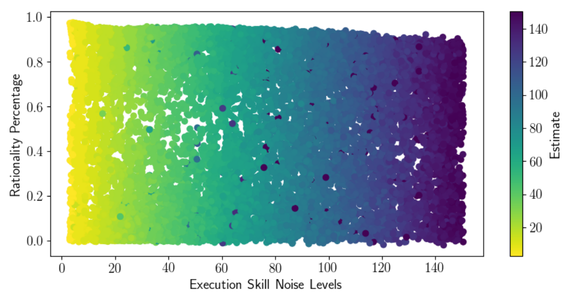
Figure 41: JEEDS Estimates for the Softmax Agent in 2D-Darts



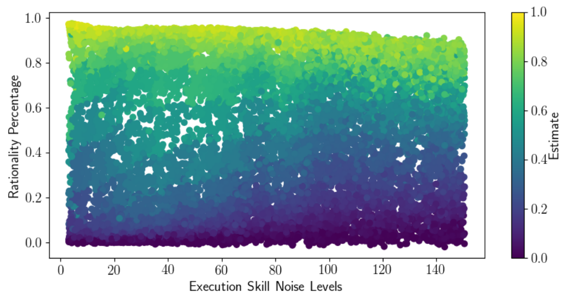
(a) Execution Skill Estimates



(b) Dec-Making Skill Estimates



(c) Execution Skill Estimates



(d) Rationality Percentage Estimates

Figure 42: JEEDS Estimates for the Softmax Agent in 201-Darts

References

Archibald, C., Altman, A., & Shoham, Y. (2009). Analysis of a winning computational billiards player. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1377–1382.

Archibald, C., Altman, A., & Shoham, Y. (2010). Success, strategy and skill: an experimental study. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1089–1096. International Foundation for Autonomous Agents and Multiagent Systems.

Archibald, C., Altman, A., & Shoham, Y. (2016). A distributed agent for computational pool. *IEEE Transactions on Computational Intelligence and AI in Games*, *8*(2), 190–202.

Archibald, C., & Nieves-Rivera, D. (2018). Execution skill estimation. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '18, pp. 1859–1861, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Archibald, C., & Nieves-Rivera, D. (2019). Bayesian execution skill estimation. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 6014–6021.

Archibald, C., & Shoham, Y. (2011). Hustling in repeated zero-sum games with imperfect execution. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*, IJCAI'11, p. 31–36. AAAI Press.

Bard, N., Johanson, M., Burch, N., & Bowling, M. (2013). Online implicit agent modelling. In *Proceedings of the Twelfth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 255–262.

Bard, N., Nicholas, D., Szepesvari, C., & Bowling, M. (2015). Decision-theoretic clustering of strategies. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*. To Appear.

Batzilis, D., Jaffe, S., Levitt, S., List, J. A., & Picel, J. (2019). Behavior in strategic settings: Evidence from a million rock-paper-scissors games. *Games*, *10*(2), 18.

Billings, D., Papp, D., Schaeffer, J., & Szafron, D. (1998). Opponent modeling in poker. In *AAAI/IAAI*, pp. 493–499.

Borm, P., & Genugten, B. (2001). On a relative measure of skill for games with chance elements. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, *9*(1), 91–114.

Bowling, M., & Veloso, M. (2004). Existence of multiagent equilibria with limited agents. *Journal of Artificial Intelligence Research*, *22*, 353–384. A previous version appeared as a CMU Technical Report, CMU-CS-02-104.

Carmel, D., & Markovitch, S. (1995). Opponent modeling in multi-agent systems. In *International Joint Conference on Artificial Intelligence*, pp. 40–52. Springer.

Chakraborti, T., Sreedharan, S., & Kambhampati, S. (2020). The emerging landscape of explainable automated planning & decision making. In Bessiere, C. (Ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 4803–4811. International Joint Conferences on Artificial Intelligence Organization. Survey track.

Davis, T., Burch, N., & Bowling, M. (2014). Using response functions to measure strategy strength. In *Proceedings of the Twenty-Eighth Conference on Artificial Intelligence (AAAI)*, pp. 630–636.

Dreef, M., Borm, P., & Genugten, B. v. d. (2002). On strategy and relative skill in poker. Discussion paper 59, Tilburg University, Center for Economic Research.

Dreef, M., Borm, P., & van der Genugten, B. (2004). A new relative skill measure for games with chance elements. *Managerial and Decision Economics*, *25*(5), 255–264.

Fernández, F., & Veloso, M. (2013). Learning domain structure through probabilistic policy reuse in reinforcement learning. *Progress in Artificial Intelligence*, *2*, 13–27.

García, J., Visús, Á., & Fernández, F. (2022). A taxonomy for similarity metrics between markov decision processes. *Machine Learning*, *111*(11), 4217–4247.

Greenspan, M., Lam, J., Leckie, W., Godard, M., Zaidi, I., Anderson, K., Dupuis, D., & Jordan, S. (2008). Toward a competitive pool playing robot. *IEEE Computer Magazine*, *41*(1), 46–53.

Greenspan, M. (2005). UofA wins the pool tournament. *International Computer Games Association Journal*, *28*(3), 191–193.

Greenspan, M. (2006). PickPocket wins the pool tournament. *International Computer Games Association Journal*, *29*(3), 153–156.

Guo, Q., & Gmytrasiewicz, P. (2011). Modeling bounded rationality of agents during interactions. In *Workshops at the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Jiang, A. X., Yin, Z., Zhang, C., Tambe, M., & Kraus, S. (2013). Game-theoretic randomization for security patrolling with dynamic execution uncertainty. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pp. 207–214. International Foundation for Autonomous Agents and Multiagent Systems.

Landry, J. F., Dussault, J. P., & Beaudry, E. (2015). A straight approach to planning for 14.1 billiards. *IEEE Transactions on Computational Intelligence and AI in Games*, *PP*(99), 1–1.

Larkey, P., Kadane, J. B., Austin, R., & Zamirm, S. (1997). Skill in games. *Management Science*, *43*(5), 596–609.

LeDoux, J. (2017). Introducing pybaseball: an open source package for baseball data analysis. https://jamesrledoux.com/projects/open-source/introducing-pybaseball/. Accessed: 04-04-2023.

Ling, C. K., Fang, F., & Kolter, J. Z. (2019). Large scale learning of agent rationality in two-player zero-sum games. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 6104–6111.

McKelvey, R. D., & Palfrey, T. R. (1995). Quantal response equilibria for normal form games. *Games and Economic Behavior*, *10*(1), 6–38.

Melville, W., Melville, J., Dawson, T., Nieves-Rivera, D., Archibald, C., & Grimsman, D. (2023). A game theoretical approach to optimal pitch sequencing. In *MIT Sloan Sports Analytics Conference*.

Miller, T., & Archibald, C. (2021). Monte carlo skill estimation for darts. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1–8.

Nashed, S., & Zilberstein, S. (2022). A survey of opponent modeling in adversarial domains. *Journal of Artificial Intelligence Research*, *73*, 277–327.

Rosenfeld, A., & Kraus, S. (2018). Predicting human decision-making: From prediction to action. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *12*(1), 1–150.

Russell, S., & Norvig, P. (2009). *Artificial Intelligence: A Modern Approach* (3rd edition). Prentice Hall Press, Upper Saddle River, NJ, USA.

Saeed, W., & Omlin, C. (2023). Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, *263*, 110273.

Schadd, F., Bakkes, S., & Spronck, P. (2007). Opponent modeling in real-time strategy games.. In *GAMEON*, pp. 61–70.

Simon, H. A. (1972). Theories of bounded rationality. *Decision and organization*, *1*(1), 161–176.

Smith, M. (2007). PickPocket: A computer billiards shark. *Artificial Intelligence*, *171*, 1069–1091.

Sturtevant, N., Zinkevich, M., & Bowling, M. (2006). ProbMaxn: Opponent modeling in n-player games. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence (AAAI)*, pp. 1057–1063.

Szita, I., Takács, B., & Lorincz, A. (2003). ε–mdps: Learning in varying environments.. *Journal of Machine Learning Research*, *3*(1).

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey.. *Journal of Machine Learning Research*, *10*(7).

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. MIT Press, Cambridge, MA.

Tibshirani, R. J., Price, A., & Taylor, J. (2011). A statistician plays darts. *Journal of the Royal Statistical Society. Series A (Statistics in Society)*, *174*(1), 213–226.

Van Valkenhoef, G., Ramchurn, S. D., Vytelingum, P., Jennings, N. R., & Verbrugge, R. (2010). Continuous double auctions with execution uncertainty. In *Agent-Mediated Electronic Commerce. Designing Trading Strategies and Mechanisms for Electronic Markets*, pp. 226–241. Springer.

Wu, I.-C., Wu, T.-R., Liu, A.-J., Guei, H., & Wei, T. (2019). On strength adjustment for mcts-based programs. *Proceedings of the AAAI Conference on Artificial Intelligence*, *33*(01), 1222–1229.

Yang, R., Kiekintveld, C., Ordonez, F., Tambe, M., & John, R. (2011). Improved computational models of human behavior in security games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pp. 1155–1156. International Foundation for Autonomous Agents and Multiagent Systems.

Yin, Z., Jain, M., Tambe, M., & Ordónez, F. (2011). Risk-averse strategies for security games with execution and observational uncertainty.. In *AAAI*.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., & He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.