Neural Computing and Applications
https://doi.org/10.1007/s00521-022-07808-y

S.I.: HUMAN-ALIGNED REINFORCEMENT LEARNING FOR AUTONOMOUS AGENTS AND ROBOTS

# Who should I trust? Cautiously learning with unreliable experts

Tamlin Love[1] · Ritesh Ajoodha[1] · Benjamin Rosman[1]

Received: 14 December 2021 / Accepted: 6 September 2022

**Abstract**
An important problem in reinforcement learning is the need for greater sample efficiency. One approach to dealing with this problem is to incorporate external information elicited from a domain expert in the learning process. Indeed, it has been shown that incorporating expert advice in the learning process can improve the rate at which an agent's policy converges. However, these approaches typically assume a single, infallible expert; learning from multiple and/or unreliable experts is considered an open problem in assisted reinforcement learning. We present CLUE (cautiously learning with unreliable experts), a framework for learning single-stage decision problems with action advice from multiple, potentially unreliable experts that augments an unassisted learning with a model of expert reliability and a Bayesian method of pooling advice to select actions during exploration. Our results show that CLUE maintains the benefits of traditional approaches when advised by reliable experts, but is robust to the presence of unreliable experts. When learning with multiple experts, CLUE is able to rank experts by their reliability and differentiate experts based on their reliability.

**Keywords** Assisted reinforcement learning · Interactive reinforcement learning · Agent teaching · Expert advice

## 1 Introduction

*Single-stage decision problems* (SSDPs), otherwise known as *contextual bandits*, are a type of reinforcement learning (RL) problem with a wide range of useful applications [1–3]. For example, consider the problem of a doctor who can observe a patient's symptoms and medical history and must prescribe the right set of treatments to improve the patient's condition. These types of problems have attracted research looking to augment the doctor with a software agent, with the long-term goal of making such diagnoses more comprehensive and widely available [4, 5].

As another example, consider the scenario of a robot frail-care assistant, tasked with monitoring its patient and assisting in daily tasks. Suppose this robot has already learned how to optimally perform each individual task (e.g. mobility assistance, dispensing medicine, etc.), but has yet to learn which tasks to perform in which situations, based on the observations it can make through its sensors. In such a scenario, it is crucial for the robot to learn which tasks to perform for given observations, as there is a great deal of risk involved should the robot perform the wrong task. For example, if the patient has slipped and fallen, the correct response might be to call for help. If the robot does not perform these tasks, serious harm could come to the patient.

In both examples, it is important for the autonomous agent to learn the problem with as few samples as possible, for a number of reasons. Primarily, these types of problems may be very complex, with a large space of possible observations and decisions. For example, the medical diagnosis problem may consist of hundreds of symptoms and treatments. Additionally, data acquisition may be difficult, either because the agent is acting in the real-world, thus potentially damaging itself and its surroundings, or because of ethical and safety issues, especially when dealing with human patients.

✉ Tamlin Love
1438243@students.wits.ac.za

Ritesh Ajoodha
ritesh.ajoodha@wits.ac.za

Benjamin Rosman
benjamin.rosman1@wits.ac.za

[1] School of Computer Science and Applied Mathematics, University of the Witwatersrand, Johannesburg, South Africa

One approach to tackling the need for sample efficiency is to incorporate external information in the learning process [6]. For example, an autonomous medical diagnosis system could be advised by a doctor who instructs the agent to prescribe certain treatments in response to certain combinations of symptoms and medical history. Given the potential complexity, it may not always be feasible to elicit all of this information before learning starts. Instead, the human advisor can advise the agent as it learns, in response to its performance. Indeed, previous work has shown that the interactive incorporation of expert advice can improve the rate at which an RL agent converges to a given performance threshold, provided that said advice is correct [7].

To increase the amount of external information available to an agent, it may be desirable to incorporate advice from multiple experts. However, this introduces its own problems when multiple experts offer conflicting advice for the same situation. Here the agent must decide which advice to follow and which to ignore. In general, expert advisers, especially humans, can give incorrect advice, either in error or through active malice [8]. Overcoming these problems has been identified as an open problem in the field of Assisted RL [6].

In order to address these issues, we present CLUE, a framework for learning SSDPs with policy advice from multiple, potentially unreliable experts. Our contributions consist of the framework itself, as well as Bayesian approaches to modelling expert reliability and pooling advice from multiple experts to facilitate decision-making. We demonstrate in a number of randomly generated SSDP environments that CLUE benefits from advice given by reliable experts, but is robust against advice given by experts that may be unreliable to some degree.

## 2 Background and related work

### 2.1 Single-stage decision problems

*Reinforcement learning* (RL) is a field of machine learning in which decision-making entities, known as *agents*, learn how to interact with an environment in order to maximise some cumulative reward signal [9]. Of the many types of RL problems, this research concerns itself with *single-stage decision problems* (SSDPs), also known as *contextual bandits* [10], with discrete states and actions. In this setting, the agent observes some *state* $s \in S$, selects some *action* $a \in A$ and receives some *reward* or *utility* $r(s, a) \in \mathbb{R}$ from the environment. Each round of observation, action selection and environment feedback is referred to as a *trial*, and each trial is independent from previous trials.

The medical diagnosis example from the previous section can be posed as an SSDP, with the set of observable symptoms and medical history forming the state space, the set of available treatments forming the action space and the reward signal being a function of the patient's health, whether or not they have experienced negative side effects, etc. The frail-care assistance robot example can also be posed as an SSDP, with each state being composed of the observations made by the robot. Unlike the medical diagnosis example, the action space here is made up of high-level strategies, rather than low-level actions such as joint angles and motor velocities. The reward may be related to the well-being of the patient.

A *policy* $\pi : S \rightarrow A$ is a function that maps each state to an action, and the goal of an agent within an SSDP is to learn the *optimal policy* $\pi^*(s)$ that maximises $EU(\pi(s)|s)$, where $EU(a|s)$ denotes the *expected utility* (i.e. *expected reward*) of choosing an action $a$ in state $s$. The expected utility function is typically not given and must be learned by the agent through its interactions with the environment. One common approach to learning this function is to maintain an *action value function* $Q(s, a) \approx EU(a|s)$, which can be updated after each trial via some weighted average of observed $r(s, a)$, such as in the update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t - Q(s_t, a_t)), \tag{1}$$

where $\alpha \in (0, 1]$ is the *step size parameter* and controls the rate at which the agent learns [9]. For stationary problems, $\alpha$ may be assigned the value of $\frac{1}{k(s,a)}$, where $k(s, a)$ is the number of times the state-action pair $(s, a)$ has been encountered. As this is always calculated at the end of a trial, $k(s, a) \geq 1$.

In order to ensure the agent's interactions with the environment are sufficient to learn a good estimate of $EU(a|s)$, a balance must be struck between acting optimally according to this estimate (so-called exploitation) and improving estimates for other state-action pairs (so-called exploration). Many different approaches exist to balance exploration and exploitation, with some common examples including epsilon greedy, adaptive greedy, explore-then-exploit (ETE) and upper confidence bound (UCB) [11, 12].

Epsilon greedy agents maintain a parameter $\epsilon \in [0, 1]$ such that with probability $\epsilon$ the agent acts randomly (exploration) and otherwise selects the action $a^*$ that maximises $Q(s, a)$ (exploitation). In practice, the value of $\epsilon$ is often decayed over time. Adaptive greedy agents maintain a parameter $z$, such that for a given state $s$, the agent exploits if $Q(s, a^*) > z$, otherwise exploring. As with $\epsilon$, $z$ is often decayed over time. ETE agents partition the entire set of trials into two phases determined by $t_b$, such that the agent explores when $t < t_b$ and exploits when $t \geq t_b$. UCB agents select the action that maximises some combination of $Q(s, a)$ and the size of a confidence interval within which that estimate lies. Using a parameter $c$ to balance exploration and exploitation, a UCB agent selects the action that maximises $Q(s,a) + c\sqrt{\frac{2\ln(t)}{N(s,a)}}$, where $N(s, a)$ denotes the number of times action $a$ has been selected for state $s$ [12].

### 2.2 Assisted reinforcement learning

Assisted reinforcement learning (ARL) is a framework that encompasses a wide range of RL methods that incorporate information external to the environment in the learning process [6]. While the focus of this paper is on contextual bandits rather than the full RL problem, the ARL framework is nevertheless applicable. Some examples of ARL approaches include RL from demonstration [13] and transfer learning in RL [14]. Of particular relevance to this work is interactive RL (IRL), in which an expert (either human of software-based) provides information to the agent during the learning process, usually as a response to the behaviour of the agent [15].

IRL methods can be classified based on the type of advice the expert gives. In *reward-shaping* approaches [16], the expert modifies the reward signal provided to the agent (e.g. by providing feedback when the agent selects certain actions). In *policy-shaping* approaches [17, 18], the expert modifies the agent's policy, typically by advising an action for a given state and having this action override the agent's policy whenever that state is encountered. Both approaches are preferred for different situations and domains. In this paper, we focus on policy shaping, as state-action advice can be more easily elicited from human experts in our domains of interest (particularly the medical diagnosis domain), requires minimal similarity between the agent and expert [7], and is more robust to infrequent and inconsistent feedback [18].

Many (but not all) approaches in ARL assume the advice to be coming from a single, infallible expert. However, this assumption does not always hold, especially when the expert is human [8]. Suboptimal advice could be the result of communication error, erroneous domain knowledge or a malicious expert. The problem of incorporating unreliable advice is especially salient when advice is retained rather than discarded after use, as erroneous advice can impact performance wherever the advice is reused [19]. Furthermore, incorporating advice from multiple experts introduces the possibility of two or more experts offering contradicting advice, requiring the agent to choose which advice is more likely to be correct [20]. The problems of incorporating advice from unreliable experts and from multiple experts are considered open questions in ARL [6].

### 2.3 Related work

Several approaches have attempted to deal with the problems of learning from multiple experts and dealing with incorrect advice. Gimelfarb, Sanner and Lee [16] tackle both problems by combining reward-shaping advice as a weighted sum of potential functions, with weights representing the belief that an expert is correct being updated as the agent learns. The decision-making rules in Sect. 3.2 are directly inspired by this Bayesian combination of advice. Griffith et al. [18] tackle the problem of incorrect advice by modelling the reliability of an expert using a static parameter $C \in (0, 1)$ when assessing whether an action is optimal. Such a model of reliability is expanded on in Sect. 3.1.

Both of the aforementioned approaches rely on reward-shaping advice and are thus incompatible with the policy-shaping advice considered in this paper. The policy reuse in Q-learning (PRQ) algorithm probabilistically selects transferred policies (or the agent's learned policy) to follow, updating the probability of selecting a policy using the reward gained following it [17]. PRQ can be used to address the problems of learning from multiple, potentially unreliable experts [19], and a comparison between PRQ and the approach presented in this paper is made in Sect. 4.3.

Other approaches that consider unreliable information (albeit with different temporalities and types of advice) include the normalised actor–critic algorithm, an RL from demonstration approach which refines an initial policy obtained from potentially imperfect demonstrations [21] and the joint learning framework for deferral to multiple experts, a classification algorithm, in which a classifier is learnt together with a deferrer which learns when to defer to one or more experts, which may have incorrect domain knowledge or biases [22].

## 3 Methodology

The aim of this paper is to devise an algorithm that can model the reliability of multiple experts and use these models to combine the action advice given by the experts to learn an optimal policy for an SSDP. To that end, we present **CLUE** (cautiously learning with unreliable experts), a framework for learning to solve SSDPs with action advice from multiple, potentially unreliable experts. High-level pseudocode is provided in Algorithm 1.

**Algorithm 1** Cautiously Learning with Unreliable Experts

```
1: procedure CLUE($D, E, N$)    ▷ $D$ = environment, $E$ = panel of experts, $N$ = number of trials
2:     for $t \in [0, ..., N-1]$ do
3:         $s_t \leftarrow$ sample_state($D$)    ▷ environment selects state
4:         $a_t \leftarrow$ act($s_t$)    ▷ agent acts, incorporating earlier advice (Section 3.2)
5:         $r_t \leftarrow$ execute_action($a_t, D$)    ▷ environment returns reward
6:         $advice_t \leftarrow$ advise($E, s_t, a_t, r_t$)    ▷ experts may offer advice (Section 4)
7:         learn($s_t, a_t, r_t$)    ▷ agent learns (Equation 1)
8:         update($s_t, a_t, r_t, advice_t$)    ▷ agent updates each model (Section 3.1)
9:     end for
10: end procedure
```

We now outline how CLUE operates and the specific contributions this work makes. CLUE involves three actors: an environment, an agent and a panel $E$ of one or more experts. The environment is a standard SSDP environment, as discussed in Sect. 2.1. Namely, for trial $t$, it samples state $s_t$, accepts action $a_t$ from the agent and returns reward $r_t$. At the end of the trial, each expert $e$ in panel $E$ receives $\langle s_t, a_t, r_t \rangle$ and may independently offer their own advice, $(s_t, a^{(e)})$ on what action the agent should have taken this trial. How and when an expert decides to offer advice differs between experts (see Sect. 4). It is worth noting here that, although we choose to have the expert give advice at the end of the trial in this work, this can occur instead at the start of a trial without requiring any change to the CLUE algorithm.

The agent is composed of three components, the first of which is a learning algorithm, which uses the information $\langle s_t, a_t, r_t \rangle$ to learn a policy, such as the action value update rule in Eq. 1. The second component, and one of the contributions of this work, is a model of the reliability of each expert (see Sect. 3.1). This model is necessary for learning which pieces of advice are to be followed and which are to be ignored. When an expert utters a piece of advice at the end of a trial, the agent uses its own information about the environment (such as a $Q$ function) to evaluate the advice and update the model (see Sect. 3.1). If the state space of a problem can be divided into $n$ "areas of expertise", $n$ models can be maintained for a single expert.

The third component, and another contribution of this work, is a decision-making process which uses the information learned by the learning algorithm and the models of each expert to select an action for a state while exploring, given any advice it has previously received for that state (see Sect. 3.2). This component is designed to augment the existing action selection algorithms that do not account for expert advice, such as those presented in Sect. 2.1.

### 3.1 Modelling experts

The first contribution we address is how an agent working within the CLUE framework models the reliability of each expert. We say that an expert is *reliable* if its advice is always *correct* (i.e. optimal), otherwise being *unreliable*. Intuitively, we can think of an expert as being unreliable to some degree. For example, an expert that offers correct advice in 95% of trials, while still unreliable, is more reliable than an expert that is always wrong. Following related work [18], we model an expert's reliability, $\rho \in [0, 1]$, as the probability of the expert giving correct advice, where $\rho = 0$ corresponds to an expert whose advice is always wrong and $\rho = 1$ corresponds to a reliable expert. We can model a probability distribution of the value of $\rho$ using a Beta distribution $Beta_\rho[\alpha, \beta]$, whose shape is determined by the parameters $\alpha, \beta > 0$ [23]. These parameters can be thought of as counts, with $\alpha$ and $\beta$ recording the number of times correct or incorrect advice was given, respectively.

At the end of trial $t$, the agent must update this distribution for each expert that gave advice for $s_t$ sometime in the past. To do this, the agent can evaluate the advice as either optimal or suboptimal, given its own information. In this work, we set $x_t = 1$ if $Q(s_t, a_t^{(e)}) = max_a Q(s, a)$, and $x_t = 0$ otherwise, where $a_t^{(e)}$ denotes the advice received from expert $e$. In order to allow for inconsistent experts (e.g. an expert whose performance degrades over time), we update the expected value $\mathbb{E}[\rho]$ using a recency-weighted moving average with weight parameter $\delta \in [0, 1]$,

$$\mathbb{E}_{t+1}[\rho] = (1 - \delta)\mathbb{E}_t[\rho] + \delta x_t, \quad (2)$$

where $\mathbb{E}_0[\rho]$ is determined by prior counts $\alpha_0$ and $\beta_0$. If the state space has been divided into $n$ "areas of expertise", across which the reliability of an expert may differ, a CLUE agent can maintain $n$ of the above models and update them separately, provided the "areas of expertise" are known to the agent (see Sect. 4.4).

## 3.2 Making decisions

We now turn our attention to the problem of how $\mathbb{E}[\rho]$ can inform the decision-making process. Suppose that, at the start of trial $t$, the agent observes state $s_t$ and recalls any advice that some subset $E_t \subseteq E$ of experts offered for state $s_t$ in trials $[0, \ldots, t-1]$. In order for the agent to be able to surpass the performance of the experts advising it, we only allow the agent to consider expert advice when exploring. Determining when the agent explores depends on the underlying action selection algorithm (see Sect. 2.1). For algorithms such as epsilon greedy, adaptive greedy and ETE, this is easily explicitly determined by their respective parameters. For UCB, the agent can be said to be exploring if $\underset{a}{\operatorname{argmax}}\, Q(s,a) \neq \underset{a}{\operatorname{argmax}}\, Q(s,a)+c\sqrt{\frac{2\ln(t)}{N(s,a)}}$.

If exploring, the agent must choose between the action suggested by the underlying action selection algorithm or between following advice it has received for $s_t$, in which case it must choose which advice to follow. If $E_t = \varnothing$, no advice has been offered, such as may happen at the beginning of the learning process, and the agent must act without advice according to its underlying action selection algorithm.

If $|E_t| \geq 1$, at least one expert has offered advice. A naïve approach may be to follow the advice of the expert with the highest expected reliability. However, this approach loses information that could be provided by consensus among experts (so-called wisdom of the crowd [24]) or by adversarial experts (whose advice is almost always wrong, thus informing the agent which actions not to take).

In order to take advantage of this information, we employ a Bayesian method of pooling advice, inspired by similar approaches in potential-based reward shaping [16] and in crowd-sourced data labelling [25]. Let $a^*$ denote the optimal action for state $s_t$ and $v_t^{(e)}$ denote the advice utterance given by expert $e$ for $s_t$, with $V_t$ denoting the set $\{v_t^{(e)} | e \in E_t\}$. Our aim, therefore, is to calculate $P(a_j = a^* | V_t)$ for each $a_j \in A$. To do this, we employ Bayes' rule as follows:

$$P(a_j = a^* | V_t) = \frac{P(V_t | a_j = a^*) P(a_j = a^*)}{\sum_{k=0}^{|A|} P(V_t | a_k = a^*) P(a_k = a^*)}. \tag{3}$$

If nothing is known about the environment prior to learning, a reasonable assumption would be to assume that each action has a uniform prior probability of being optimal. Under this assumption, Eq. 3 reduces to:

$$P(a_j = a^* | V_t) = \frac{\prod_{e \in E_t} P(v_t^{(e)} | a_j = a^*)}{\sum_{k=0}^{|A|} \prod_{e \in E_t} P(v_t^{(e)} | a_k = a^*)}, \tag{4}$$

which combines all the available advice for $s_t$ to calculate the probability of each $a_j$ being optimal. Note that, if for a particular domain one can reasonably assume a non-uniform prior distribution of $P(a = a^*)$, this distribution can be incorporated into Eq. 3 without fundamentally changing this decision-making process.

All that remains is to calculate $P(v_t^{(e)} | a_j = a^*)$. Recalling that the probability of the advice being correct is equal to $\mathbb{E}[\rho^{(e)}]$ and assuming that, if the advice is incorrect, the expert is equally likely to advise any suboptimal action, then:

$$P(v_t^{(e)} | a_k = a^*) = \begin{cases} \mathbb{E}[\rho^{(e)}] & v_t^{(e)} = a_k \\ \dfrac{1 - \mathbb{E}[\rho^{(e)}]}{|A| - 1} & v_t^{(e)} \neq a_k \end{cases} \tag{5}$$

Substituting Eq. 5 into Eq. 4, we can calculate the probability of each action in $A$ being optimal and can set $a_{best} = \arg\max_a P(a = a^* | V_t)$. In a similar approach to both epsilon greedy and probabilistic policy reuse [17], the agent selects action $a_{best}$ with probability $P(a_{best} = a^* | V_t)$, and otherwise acts as if $E_t = \varnothing$. This allows for a trade-off between following advice and exploring as normal, where the former is more likely if the agent is confident that $a_{best}$ is optimal. If the state space has been divided into $n$ "areas of expertise" and the agent knows which states belong in which area, then the agent can maintain $n$ separate models for each expert and substitute the appropriate value of $\mathbb{E}[\rho^{(e)}]$ into Eq. 5 for the state.

In the above formulations, we have assumed that the estimated $\mathbb{E}[\rho^{(e)}]$ accurately represents the underlying reliability of the expert $e$. This is not always the case, however, especially at the start of learning. Erring on the side of caution, we can compensate for the over-estimation of the reliability of particularly bad experts by introducing a threshold parameter $T \in [0, 1]$, such that if $P(a_{best} = a^* | V_t) < T$, the agent acts without advice. This approach ensures that the agent will only follow advice if it is sufficiently confident that the advice is correct.

## 4 Experiments

Having outlined the CLUE framework, we now present a number of experiments to show that **a)** when advised by at least one reliable expert, CLUE outperforms an equivalent unassisted agent, **b)** when advised by unreliable experts who are likely to give incorrect advice, CLUE asymptotically converges to the same threshold of convergence as that achieved by an equivalent unassisted agent, thereby being robust against incorrect advice, and **c)** when advised by multiple experts with different degrees of reliability, CLUE is correctly able to rank experts by their reliability and exploit this information to potentially improve performance.

To show that the performance of CLUE generalises, we run experiments on multiple, randomly generated environments. To generate these environments, we create influence diagrams (IDs) [26], whose state and action variables ($V_S$ and $V_A$) define the state- and action space, respectively, with random conditional probability distributions, utility functions and graph structures. Each variable has a binary domain $\{0, 1\}$, so that $|S| = 2^{|V_S|}$ and $|A| = 2^{|V_A|}$. To ensure that each ID represents a well-formed bandit problem, we restrict the graph structure to a directed acyclic graph. We also ensure that all state nodes are parents of action nodes (for full observability), all action nodes are parents of the reward node (so that all actions have some effect on the reward), no action nodes are descendants of another action node (so that only a single round of decision-making occurs), and that no state nodes are children of an action node (as states are observed before decision-making). Rewards are scaled between $-1$ and 1, so that results across environments are comparable. Thus the simulated environments are representative of a wide range of discrete bandit problems.

In order to control frequency and quality of advice, all experiments are conducted with simulated experts, with human expert studies lying outside the scope of this work. Our experiments are conducted around properties one might expect of a human expert, such as variance in advice quality, tolerance of poor agent performance and expertise over different regions of the state space.

Many ARL approaches limit the number of interactions between experts and agents, so as to simulate the potential cost of communication, and thus, conditions are imposed upon the expert to ensure it gives advice where it is most needed [7]. In this work, we adopt the following conditions which must be satisfied for the expert to give advice:

$$t - t' \geq \mu, \tag{6}$$

$$\sum_{t' \leq i \leq t} \frac{EU(a_i^*|s_i) - EU(a_i|s_i)}{t - t'} \geq \gamma, \tag{7}$$

where $t$ is the current time, $t'$ is the last time the expert gave advice, $a_i^*$ is the optimal action for state $s_i$, $a_i$ is the action taken by the agent at time $i$, $\mu$ is an interval parameter controlling the frequency of advice giving and $\gamma$ is a tolerance parameter controlling how tolerant the expert is of suboptimal behaviour from the agent [27].

In order to simulate reliability, each expert $e$ is controlled by a *true reliability parameter* $\rho_{true}^{(e)}$. When offering advice, the expert will advise the optimal action $a^*$ (obtained from a "ground truth" model of the environment) with probability $\rho_{true}^{(e)}$, or else will randomly advise any other action. Thus an expert with $\rho_{true}^{(e)} = 1$ is reliable, while one with $\rho_{true}^{(e)} = 0$ never advises the optimal action. An expert with $\rho_{true}^{(e)} = \frac{1}{|A|}$ therefore advises actions with uniform random probability. As CLUE models reliability only through inconsistencies in optimal and suboptimal evaluations of advice (Sect. 3.1), the number of times an action has been advised for a state, or the fact that advised actions for a state may change on subsequent visits, does not affect how the expert's reliability is modelled.

### 4.1 Panel comparisons

In this set of experiments, we compare the reward obtained in each trial by the agents advised by different panels of experts. The rewards obtained by the agents training over 80, 000 trials across 100 random environments with 10 state variables ($|S| = 1024$) and 3 action variables ($|A| = 8$) are averaged and plotted against trials. LOWESS smoothing is employed for legibility [28], with the standard deviation represented by the shaded areas. We compare the performance of each agent with three panels of experts. The first, a *Single Reliable Expert*, consists of one expert that always gives correct advice ($\rho_{true} = 1$). The second, a *Single Unreliable Expert*, consists of one expert that always gives incorrect advice ($\rho_{true} = 0$). The third, a *Varied Panel*, consists of seven experts with varying degrees of unreliability ($P_{true} = \{0, 0.1, 0.25, 0.5, 0.75, 0.9, 1\}$).

Agents tested include a *baseline agent*, which does not incorporate expert advice, a *Naïve Advice Follower* (NAF), which follows any advice it has received for a state (choosing randomly between contradicting advice) otherwise acting as the baseline agent, and CLUE, which augments the baseline agent with the framework discussed in Sect. 3 ($\alpha_0 = 1 = \beta_0$, $T = \frac{2}{|A|}$, $\delta = 0.5$). The four tested baselines are epsilon greedy ($\epsilon$ decays from 1 to 0 across 80% of trials), adaptive greedy ($z$ decays from 1 to $-1$ across 80% of trials), ETE ($t_b = 20,000$) and UCB ($c = 0.25$), all of which employ the action value update rule in Eq. 1 ($Q_0 = 0, \alpha = \frac{1}{k(s,a)}$). Results are plotted in Fig. 1.

For $\rho_{true} = 1$, both CLUE and NAF converge faster than all baselines as they quickly benefit from the optimal advice provided by the reliable expert. A demonstration of the robustness of CLUE comes when $\rho_{true} = 0$. In this scenario, NAF exclusively follows suboptimal advice and thus is unable to converge to the optimal policy. CLUE on the other hand is able to identify that the expert is unreliable and defaults to its underlying action selection algorithm, performing identically to the baselines. For the *Varied Panel*, the performance of NAF lies somewhere between the two single-expert cases, as it receives a mix of advice, and cannot discern which advice is advantageous to follow. CLUE is able to differentiate between reliable and unreliable experts and benefits from the former despite the presence of the latter. In all cases, CLUE either converges faster than the baseline when good advice is available, or otherwise converges at the same rate as the baseline.

## 4.2 Reliability estimates

To investigate the results obtained in Sect. 4.1, we plot the value of $\mathbb{E}[\rho^{(e)}]$ over time for the same expert panels. For the sake of space, we consider a single baseline (epsilon greedy) for all remaining experiments. Results are plotted in Fig. 2.

For the single-expert cases, the value of $\mathbb{E}[\rho]$ converges towards the correct value of $\rho_{true}$ (1 and 0, respectively), with the final estimates being $\mathbb{E}[\rho] = 0.995$ for the *Single Reliable Expert* and $\mathbb{E}[\rho] = 0.005$ for the *Single Unreliable Expert*. For the *Varied Panel*, each expert is correctly ranked according to their reliability and the value of $\mathbb{E}[\rho^{(e)}]$ for each expert $e$ correctly converges towards the true value of $\rho^{(e)}_{true}$, even faster than the single-expert cases. This accuracy in the estimates of reliability explains the performance obtained in the experiments in Sect. 4.1.

## 4.3 Comparison with policy reuse Q-learning

In this set of experiments, we compare CLUE with the PRQ algorithm [17], which addresses the problem of incorporating the advice of multiple, unreliable experts by probabilistically selecting which advice to follow, adjusting these probabilities using the reward obtained by following them (see Sect. 2.3). Results are compared across 100 random environments with 10, 000 trials, where $|V_S| = 7$ and $|V_A| = 3$, plotted in Fig. 3. We use hyperparameters $\tau = 0$, $\Delta\tau = 0.05$ for PRQ. In addition to the panels presented in Sect. 4.1, we also introduce a *Single Random Expert* ($\rho_{true} = 0.5$), to represent a largely unreliable expert with a higher-than-random chance of advising optimally.

The PRQ agent performs exceptionally well when a reliable expert is present, even outperforming CLUE with the *Single Reliable Expert*, as it learns to identify and follow the correct expert. However, its performance is hampered with the unreliable expert, as it takes longer for it to learn to identify and ignore the suboptimal advice. When the expert's policy is not optimal, but is nevertheless better than random (as $0.5 > \frac{1}{|A|}$), as with the *Single Random Expert*, the PRQ agent learns early on that following the expert's advice is better than its initial policy, resulting in an initial boost to performance, but is unable to surpass the performance of the expert. CLUE, on the other hand, is able to benefit from the higher-than-random chance of receiving optimal advice, but is still able to surpass the expert's performance. These comparisons further demonstrate the robustness of CLUE with less-than-ideal experts.

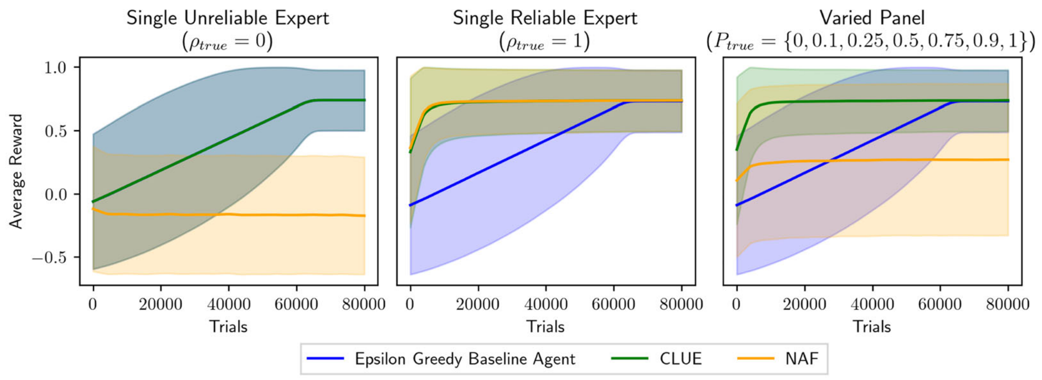
## 4.4 Experts with non-uniform reliability

In this set of experiments, we consider experts whose reliability is non-uniform over the state space. In particular, we consider the case where $S$ is divided into two "areas of expertise". For each run, states are randomly assigned to one of two sets, with uniform random probability. Each expert in a panel has two $\rho_{true}$ values, one for each set, which determine the probability of giving optimal advice. We consider four panels: a *Single Bad Expert* ($\rho_{true} = [0.2, 0]$), a *Single Good Expert* ($\rho_{true} = [1, 0.8]$), a *Single Extreme Expert* ($\rho_{true} = [1, 0]$) and *Experts with opposite areas of expertise* ($P_{true} = \{[1, 0], [0, 1]\}$). We compare CLUE with 2 models (one for each set) and CLUE with a single model (representing the case where the "areas of expertise" are not known to the agent). As in Sect. 4.1, results are averaged over 100 environments ($|V_S| = 10$, $|V_A| = 3$), with each agent training for 80,000 trials. Results are plotted in Fig. 4.

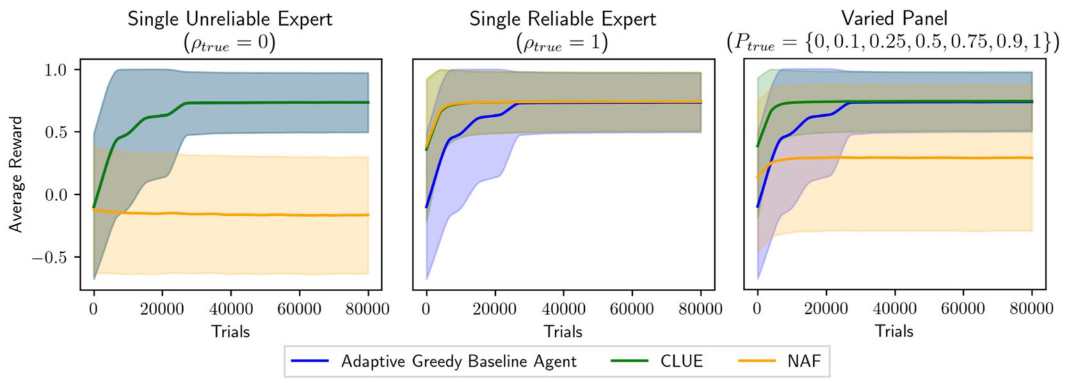
The *Single Bad* and *Single Good Expert* cases do not differ noticeably from the *Single Unreliable* and *Single Reliable Expert* cases in Sect. 4.1, with both versions of CLUE performing equally well, as the difference between the two reliabilities is not significant in either case. In the *Single Extreme Expert* case, the single-model CLUE receives only a minor boost in performance, as it is unable to sort the correct advice received in the one set of states from the incorrect advice received in the other, and thus estimates a reliability that lies somewhere between the two extremes. The two-model CLUE is able to differentiate the correct advice from the incorrect advice and is able to benefit in the states where correct advice is offered and defaulting to the baseline behaviour in the states where incorrect advice is offered.

The benefits of the two-model approach are made clear in the final panel, where two experts are reliable within their "area of expertise" and completely unreliable elsewhere. Once again, the single-model CLUE receives only a small boost to performance, but the two-model CLUE is able to identify the correct expert to follow in both sets of states and, since correct advice is available in every state, is able to achieve performance as if being advised by only a single reliable expert.

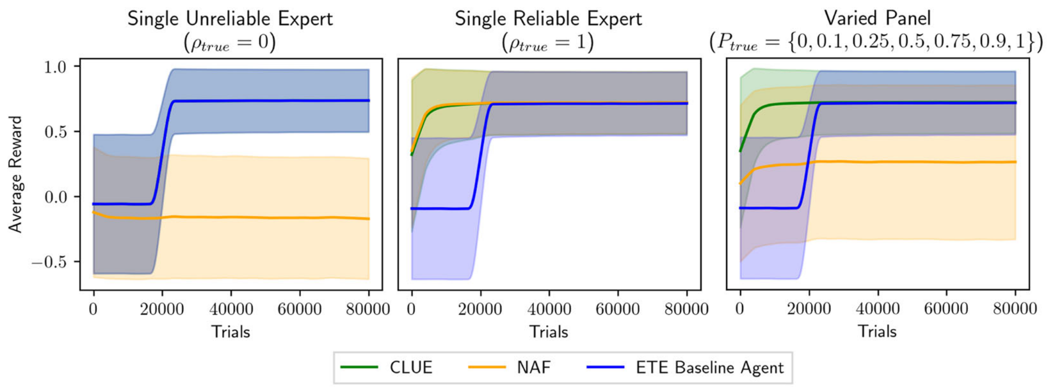
These experiments show that CLUE is able to benefit from whatever correct advice is present when it is aware of the "areas of expertise" across which the experts differ in reliability. However, the results also show that when this is not the case, as with the single-model CLUE, the agent is


Single Unreliable Expert ($\rho_{true} = 0$)
Single Reliable Expert ($\rho_{true} = 1$)
Varied Panel ($P_{true} = \{0, 0.1, 0.25, 0.5, 0.75, 0.9, 1\}$)
Average Reward
Trials
Epsilon Greedy Baseline Agent
CLUE
NAF

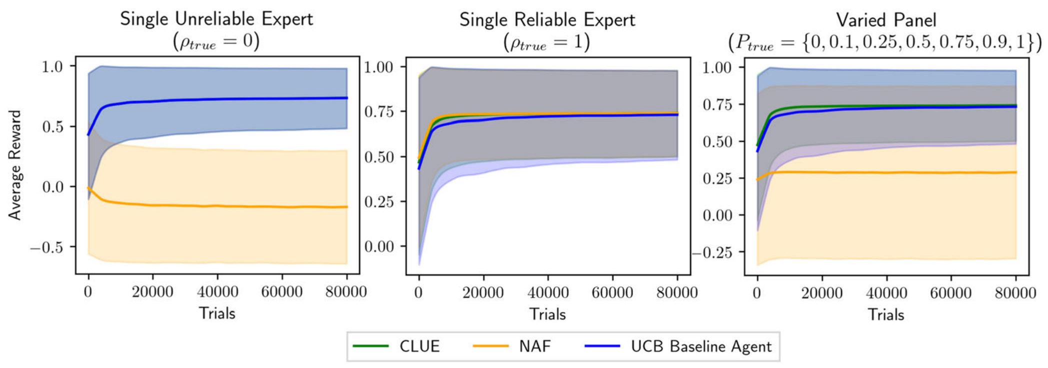
**(a)**


Single Unreliable Expert ($\rho_{true} = 0$)
Single Reliable Expert ($\rho_{true} = 1$)
Varied Panel ($P_{true} = \{0, 0.1, 0.25, 0.5, 0.75, 0.9, 1\}$)
Average Reward
Trials
Adaptive Greedy Baseline Agent
CLUE
NAF

**(b)**


Single Unreliable Expert ($\rho_{true} = 0$)
Single Reliable Expert ($\rho_{true} = 1$)
Varied Panel ($P_{true} = \{0, 0.1, 0.25, 0.5, 0.75, 0.9, 1\}$)
Average Reward
Trials
CLUE
NAF
ETE Baseline Agent

**(c)**


Single Unreliable Expert ($\rho_{true} = 0$)
Single Reliable Expert ($\rho_{true} = 1$)
Varied Panel ($P_{true} = \{0, 0.1, 0.25, 0.5, 0.75, 0.9, 1\}$)
Average Reward
Trials
CLUE
NAF
UCB Baseline Agent

**(d)**

◀**Fig. 1** Comparisons of the three panels with CLUE, NAF and a baseline agent. Baselines used are **a** epsilon greedy, **b** adaptive greedy, **c** ETE and **d** UCB. Note that CLUE and the baseline are nearly identical for $\rho_{true} = 0$

nevertheless robust to the perceived inconsistencies of the experts, defaulting to the baseline performance at worst.

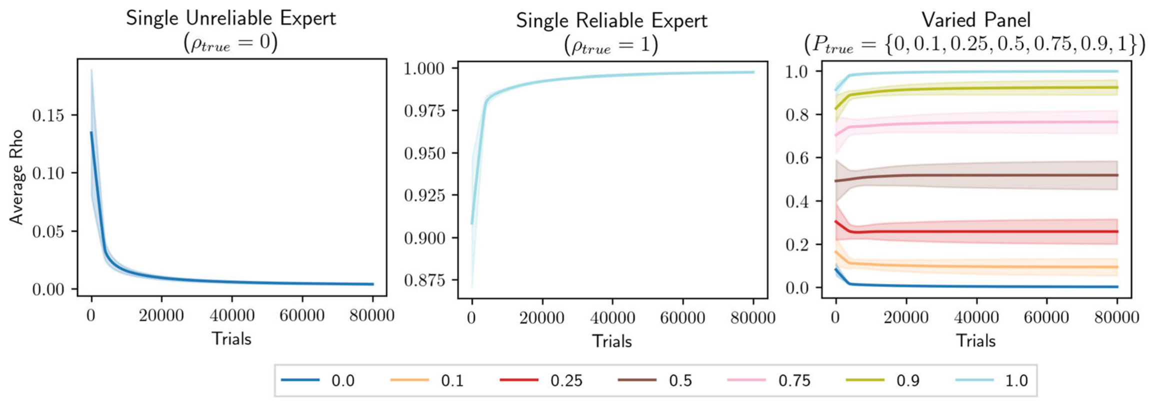


**Fig. 2** A comparison of $\mathbb{E}[\rho^{(e)}]$ for each panel with an epsilon greedy baseline. The Legend denotes the value of $\rho_{true}$

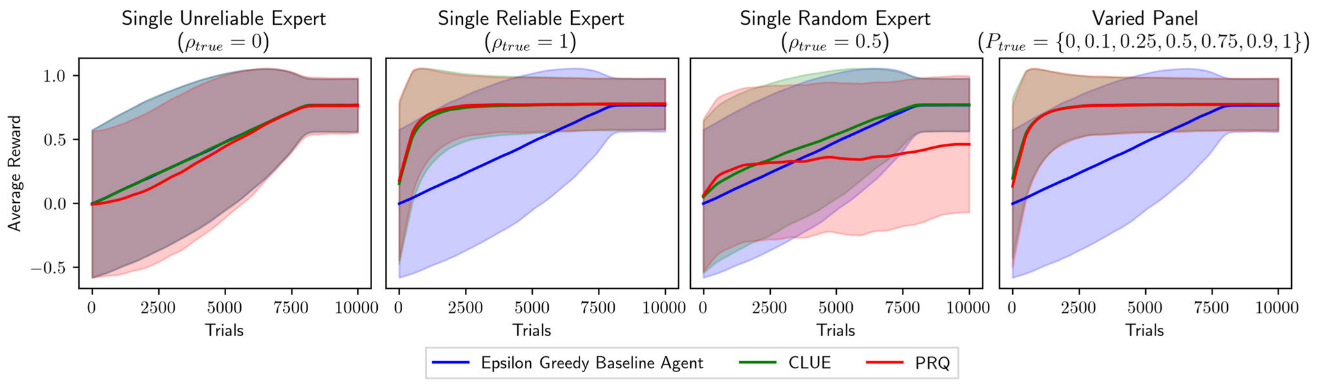


**Fig. 3** A comparison of CLUE, PRQ and an epsilon greedy baseline for each panel

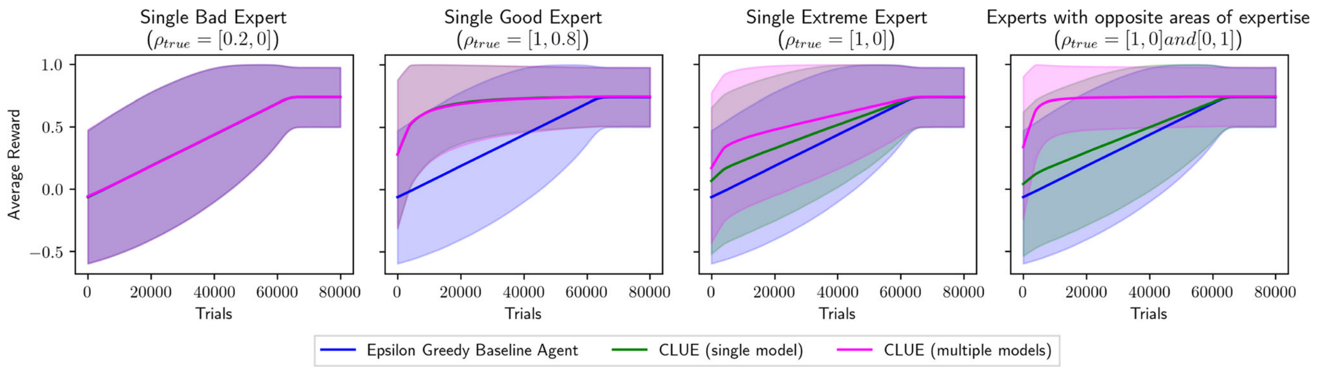


**Fig. 4** Comparison of single-model CLUE, two-model CLUE and an epsilon greedy baseline with panels of experts with two “areas of expertise”

## 5 Conclusion

This work presented the CLUE framework for learning SSDPs with policy advice from multiple, potentially unreliable experts. Our contributions consist of a method of modelling and updating reliability estimates for each expert and, using these estimates to combine policy advice to inform action selection. Our results show that CLUE maintains the benefits of traditional ARL approaches when advised by reliable experts but is robust to experts being unreliable to some degree, in both single- and multi-expert scenarios. When the expert is non-uniformly reliable across different "areas of expertise", our results show that CLUE maintains these benefits when the areas are known and is robust to the inconsistencies when they are not known. This work may allow for easier integration of external information in the learning process, ultimately contributing towards tackling more complex problems with greater sample efficiency. The explicit modelling of expert reliability allows for a more transparent decision-making process, as it can easily be ascertained why a CLUE agent did or did not follow a given piece of advice.

The experiments in this work are conducted with simulated experts and environments. This greater control over factors external to the agent (such as the frequency and quality of advice) allows for a clearer understanding of the behaviour of the CLUE agent itself. However, the real-world efficacy of CLUE can only truly be tested with real-world environments and experts (agent or human). Human experts in particular may introduce further complexities in the consistency and quality of advice. While outside the scope of this work, we consider the investigation of the performance of CLUE under real-world conditions a high priority for future work.

Other future extensions of the CLUE framework may include generalising it to the full RL problem, which may require changes to the decision-making rule (e.g. following advice for an entire episode rather than a single time step) or to the kinds of advice offered (such as rule-based advice [19] or entire policies [17]). Other extensions may seek to address how CLUE can be modified to accommodate continuous action or state spaces (another open problem in ARL [6]), or how multiple "areas of expertise" may be learned if they are not initially known.

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s00521-022-07808-y.

**Funding** The authors did not receive support from any organisation for the submitted work.

**Code availability** All code at https://github.com/tamlinlove/CLUE_SSDP.

## Declarations

**Conflict of interest** The authors have no competing interests to declare that are relevant to the content of this article.

## References

1. Li L, Chu W, Langford J, Schapire RE (2010) A contextual-bandit approach to personalized news article recommendation. In: Proceedings of the 19th international conference on World Wide Web, pp 661–670
2. Huo X, Fu F (2017) Risk-aware multi-armed bandit problem with application to portfolio selection. R Soc Open Sci 4(11):171377
3. Varatharajah Y, Berry B, Koyejo S, Iyer R (2018) A contextual-bandit-based approach for informed decision-making in clinical trials. J Environ Sci (China) Engl Ed
4. Lauritzen SL, Spiegelhalter DJ (1988) Local computations with probabilities on graphical structures and their application to expert systems. J R Stat Soc Ser B (Methodol) 50(2):157–194
5. Kao H-C, Tang K-F, Chang EY (2018) Context-aware symptom checking for disease diagnosis using hierarchical reinforcement learning. In: Thirty-second AAAI conference on artificial intelligence
6. Bignold A, Cruz F, Taylor ME, Brys T, Dazeley R, Vamplew P, Foale C (2021) A conceptual framework for externally-influenced agents: An assisted reinforcement learning review. J Ambient Intell Hum Comput 2021:1–24
7. Torrey L, Taylor M (2013) Teaching on a budget: agents advising agents in reinforcement learning. In: Proceedings of the 2013 international conference on autonomous agents and multi-agent systems, pp 1053–1060
8. Efthymiadis K, Devlin S, Kudenko D ( 2013) Overcoming erroneous domain knowledge in plan-based reward shaping. In: Proceedings of the 2013 international conference on autonomous agents and multi-agent systems, pp 1245–1246. Citeseer
9. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT Press, Cambridge
10. Langford J, Zhang T (2007) The epoch-greedy algorithm for contextual multi-armed bandits. In: Proceedings of the 20th international conference on neural information processing systems, pp 817–824. Citeseer
11. Cortes D (2018) Adapting multi-armed bandits policies to contextual bandits scenarios. arXiv preprint arXiv:1811.04383 (2018)
12. Auer P, Cesa-Bianchi N, Fischer P (2002) Finite-time analysis of the multiarmed bandit problem. Mach Learn 47(2):235–256
13. Taylor ME, Chernova S ( 2010) Integrating human demonstration and reinforcement learning: initial results in human-agent transfer. In: Proceedings of the agents learning interactively with human teachers AAMAS workshop, p 23. Citeseer
14. Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: a survey. J Mach Learn Res 10:1633–1685
15. Thomaz AL, Hoffman G, Breazeal C (2005) Real-time interactive reinforcement learning for robots. In: AAAI 2005 workshop on human comprehensible machine learning

16. Gimelfarb M, Sanner S, Lee C-G (2018) Reinforcement learning with multiple experts: A Bayesian model combination approach. Adv Neural Inf Process Syst 31:9528–9538
17. Fernández F, Veloso M (2006) Probabilistic policy reuse in a reinforcement learning agent. In: Proceedings of the fifth international joint conference on autonomous agents and multiagent systems, pp 720–727
18. Griffith S, Subramanian K, Scholz J, Isbell CL, Thomaz AL (2013) Policy shaping: integrating human feedback with reinforcement learning. Georgia Institute of Technology
19. Bignold A, Cruz F, Dazeley R, Vamplew P, Foale C (2021) Persistent rule-based interactive reinforcement learning. Neural Comput Appl 2021:1–18
20. Shelton C (2000) Balancing multiple sources of reward in reinforcement learning. Adv Neural Inf Process Syst 13:1082–1088
21. Gao Y, Xu H, Lin J, Yu F, Levine S, Darrell T (2018) Reinforcement learning from imperfect demonstrations. arXiv preprint arXiv:1802.05313
22. Keswani V, Lease M, Kenthapadi K (2021) Towards unbiased and accurate deferral to multiple experts. arXiv preprint arXiv: 2102.13004
23. Owen CEB (2008) Parameter estimation for the beta distribution. Master's thesis, Brigham Young University-Provo
24. Yi SKM, Steyvers M, Lee MD, Dry MJ (2012) The wisdom of the crowd in combinatorial problems. Cogn Sci 36(3):452–470
25. Burke P, Klein R (2020) Confident in the crowd: Bayesian inference to improve data labelling in crowdsourcing. In: 2020 International SAUPEC/RobMech/PRASA conference. IEEE, pp 1–6
26. Howard RA, Matheson JE (2005) Influence diagrams. Decis Anal 2(3):127–143
27. Innes C, Lascarides A (2019) Learning structured decision problems with unawareness. In: International conference on machine learning, pp 2941–2950
28. Cleveland WS (1981) LOWESS: a program for smoothing scatterplots by robust locally weighted regression. Am Stat 35(1):54