Expertise Combination with Spectral Clustering in Inverse Reinforcement Learning

Johannes Vallikivi

4th Year Project Report
Artificial Intelligence and Mathematics
School of Informatics
University of Edinburgh

2022

Abstract

Already present in early childhood, humans have a high capacity to learn from each other by observation (Tomasello, 2009). Being able to model this behavior is an active area of artificial intelligence research. This project explores the question of how can an agent combine their own expertise with what they have learnt from observing another, possibly suboptimal agent in order to perform a task more optimally. Expertise combination with spectral clustering as a method for achieving the combination of the knowledge of two agent is proposed which divides the state space of a simulated environment into regions to aid in choosing in what state is it useful to abide by which agent's behavior. It is shown that the proposed method of expertise combination is efficient in gridworld environments for creating new policies that outperform the original agents. Ultimately, it is demonstrated that incorporating expertise combination within the inverse reinforcement learning loop is an efficient tactic to learn from agents that do not directly optimise for the target task.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Johannes Vallikivi*)

Acknowledgements

I want to thank my supervisor Dr. J. Michael Herrmann for his guidance, encouragement, and support. My sincere thanks also go out to Billy I. Lyons who helped me build up the initial momentum to tackle this problem.

To my friends, family, and Johanna, thank you for being there for me.

Table of Contents

Chapter 1

Introduction

Training agents to successfully complete tasks in both complex virtual and real life environments is notoriously difficult. There is a plethora of different approaches to tackling this problem with one of the most popular being reinforcement learning (RL) (Sutton and Barto, 2018). However, it is difficult to successfully apply RL (Kakade, 2003) and the need for being able to model learning by observation has arisen (Taylor, Suay, and Chernova, 2011).

This project explores the question of how can an agent in an environment, with only partial knowledge of its task, learn from the demonstrations of another agent's behavior and, most importantly, how can it combine this learnt knowledge with its own expertise to better succeed at completing its task. These questions are motivated by examples from real life. For one, humans not only learn from interacting with their physical surroundings, but they also learn by observing others.

When we see others performing a task we haven't performed before, we often will use these observations to imitate others when we have to perform that task ourselves. This process of imitation is inherently very complex. When observing behaviour, we have the chance of taking on that same behavior when it is useful for us. In order to successfully imitate behavior, we need to decide and understand several things such as:

1. How to imitate given what we have seen?
2. When do we have enough observations to be successful in imitation?
3. Will this imitation be useful for our own task?
4. What parts of the behavior seen in our observations do we want to imitate?

Answering these questions could be very useful for solving various problems in artificial intelligence such as issues in cooperative reinforcement learning or the problem of adapting to highly dynamic environments. For example, it has been noted that in cooperative settings, young children track the behavior and *intentionality* of others and are able to quickly adapt their own behavior to complete a cooperative task when one of their peers is unable to complete their own part of that task (Tomasello, 2009). Having the capacity to track the behavior of others and adapt to changing environments by

using what we have learnt from our observations is very important as it could mean that we would not have to learn from scratch each time the environment or task changes. Also, avoiding learning from scratch could mean performing less exploratory actions that could be detrimental to the agent or the environment.

In this report, an overview of the background to this problem is given in Chapter 2 and the contributions of this project are listed in the following chapter. The full problem statement is presented in Chapter 4 and the question of learning the policy of another agent using inverse reinforcement learning is explored in Chapter 5. The approach and results of combining expertise are covered in Chapter 6 and in the following chapter the overall approach is discussed.

Chapter 2

Background

2.1 Reinforcement Learning

Reinforcement learning (RL) is a popular subfield of machine learning (ML) where agents are trained in dynamic environments to perform different tasks. Inspired by how learning takes place in nature, the RL training procedure seeks to reward the agent's actions that are favourable and punish the actions that should be avoided, whereby over time, with sufficient feedback from the environment, the agent should learn to behave optimally.

In order to mathematically formalise the problem the agent is solving, it is most common to use the concept of Markov decision processes (MDPs) in RL (Sutton and Barto, 2018).

A Markov decision process (MDP) is most often described as a 5-tuple

$$\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle,$$

where $\mathcal{S}$ is the set of all possible states of the environment and $\mathcal{A}$ is the set of all possible actions that can be taken in the environment. $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is the transition matrix whereby $\mathcal{T}(S_0, A_0, S_1)$ is the probability that taking an action A_0 in state S_0 will result in the transition to state S_1. The reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ gives the expected reward of a state transition, so for example $\mathcal{R}(S_1, A_1, S_2)$ is the expected reward given a transition from state S_1 to state S_2 by taking an action A_1. The discount rate $\gamma \in [0,1]$ specifies how important future rewards are. For example, the discounted sum of rewards for the transition sequence $S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} S_2 \xrightarrow{A_2} S_3$ is given by $\sum_{0 \leq i \leq 2} \gamma^i \mathcal{R}(S_i, A_i, S_{i+1})$.

A reinforcement learning agent is usually tied to its policy $\pi : \mathcal{A} \times \mathcal{S} \to [0,1]$ which describes the desired behavior of the agent in the environment. A policy $\pi(a|s)$ specifies the probability of the agent choosing an action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$ and is the mathematical definition of the notion of intended behavior as mentioned above. To *solve* a Markov decision process means for the agent to learn an optimal policy π^*. If the policy is *optimal* then it is said that it is maximising the expected discounted sum of rewards given any state.

In practice, there are several approaches to solving a given MDP. For example, if the whole MDP-tuple is accessible, *value iteration* can be used, which seeks to find the optimal value of different states whereby the value $V_\pi : \mathcal{S} \to \mathbb{R}$ of a state seeks to represent the average discounted sum of rewards of starting from a state and continuing with a selected policy π. Another popular approach is temporal-difference learning (Sutton, 1988) with a popular flavour being Q-learning (Watkins, 1989), which seeks to learn the action-value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ which is analogous to the state-value function V but instead of just assuming the knowledge of the current state it also assumes knowledge of the following action within that state.

Most of modern reinforcement learning builds upon the ideas from its inception by using neural networks to estimate different value functions. For example, Deep Q-networks were developed by the company DeepMind to enhance Q-learning by using deep neural networks to approximate the action-value function Q (Mnih et al., 2015).

However, reinforcement learning often makes multiple assumptions about the environment, some of which cannot be met in practical scenarios. One of these assumptions is the knowledge of the reward function $\mathcal{R}$ which is difficult to design to yield optimal results (Russell, 1998). For these reasons, there are whole subfields in RL research that deal with estimating the reward function of an environment, like inverse reinforcement learning (IRL), or directly deal with learning the policy from behaviour demonstrations from experts, like imitation learning (IL) (Hussein et al., 2017).

2.2 Inverse Reinforcement Learning

Training a reinforcement learning agent requires knowledge of the reward function of the environment. However, it is often the case that a reward function is hard to define. Learning a reward function instead of an optimal policy is what inverse reinforcement learning (IRL) aims to solve. In other words, while an RL algorithm knows the reward function that maps states to rewards and tries to find an optimal way to behave, an IRL algorithm knows about expert behavior but tries to find the reward function.

The motivation for this is clear — while we might have a lot of examples of behavior, we may find it hard to describe what the behavior is optimising for and hence miss out on a learning opportunity. For example, we have a large amount of examples of animal behavior but often we are unsure what they are optimising for. So using IRL to model animal behavior could be beneficial (Russell, 1998). Another important motivation is that using IRL we can also use the learned reward functions to train new experts in perturbed environments. This allows for creating apprentices of the original experts which are also adaptive to changed settings. Contrasting to imitation learning methods, which directly learn the policy of the expert, Russel proposed that a policy is not as effective and thus not as robust a representation of agent behavior as is the the reward function. This would imply that slightly perturbing the environment and using the learned policy may not be as successful in transferring optimal behavior across environments as directly retraining on the estimated reward function inside the perturbed environment.

Similarly to RL, it is common to model the behavior of an expert in IRL as the solution,

i.e. policy, for an Markov decision process $\mathcal{M}_{\setminus\mathcal{R}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma \rangle$ for which we do not know the reward function $\mathcal{R}$. The usual way the IRL problem is approached is by an iterative process whereby the reward function $\mathcal{R}'$ is guessed, using a finite collection of n demonstrations $\mathcal{D} = \{\mathcal{P}_i\}_{0 \leq i < n}$ where a path or trajectory $\mathcal{P}_i$ is a collection of state-action pairs in the environment, to create, by extension, a guess of the underlying MDP $\mathcal{M}' = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}', \gamma \rangle$. This MDP is then used to train an agent with which its learned policy is compared with the expert's policy. Over time, as the guess of the reward function $\mathcal{R}'$ is changed to minimise differences in the policies, the reward function should, by design, become similar to the true reward map $\mathcal{R}$. This approach, however, has multiple flaws, like the assumption of expert optimality (Piot, Geist, and Pietquin, 2017) and the issue of reward identifiability where multiple different reward functions can be used to describe a set of observations (Russell, 1998).

In this project, maximum entropy inverse reinforcement learning (MEIRL) (Ziebart et al., 2008) was used to learn optimal rewards and policies. MEIRL, in a cycle, performs gradient ascent on the predicted reward function whereby the loss gradient is the difference between the state visitation frequency of the observed demonstrations and the expected state visitation frequency of the policy that optimises the guessed reward function. The state visitation frequency of demonstrations $\mathcal{D}$ is $\nu_{\mathcal{D}} : \mathcal{S} \to [0, 1]$ where $\nu_{\mathcal{D}}(s)$ is the statistical frequency of visits to state $s \in \mathcal{S}$ given the demonstrations $\mathcal{D}$. This cycle is stopped when the reward function updates become sufficiently small given some preset threshold.

2.3 Learning from suboptimal experts

For inverse reinforcement learning, finding valuable demonstrations from an expert optimised to doing a task represented by a reward function $\mathcal{R}$ is problematic for a few reasons. For one, these demonstrations could be of humans interacting with the physical environment and therefore noisy and inaccurate at times. Also, it could be the case that we only have demonstrations from experts optimised for other tasks which have incomplete, albeit valuable overlap with what we would like to optimise for. Since these demonstrations can still be useful in human learning, it is important to explore how to incorporate them in learning models.

There are several research directions in this area. For example, there has been research in using demonstrations from suboptimal experts to achieve better performance than the experts using inverse reinforcement learning, imitation learning, or apprenticeship learning. However, this research often requires access to demonstrations that are close-to-optimal in order for the models to have any success. For example, the TRAIL model (Yang, Levine, and Nachum, 2022), which computes a latent action space by pretraining on a set of offline, suboptimal demonstrations to become more sample-efficient when observing new, optimal demonstrations, still requires that the offline demonstrations are near-optimal.

There has also been research in ranking demonstrations which allows extrapolation techniques to yield better policies than the experts performing demonstrations (Brown, Goo, Nagarajan, et al., 2019 and Brown, Goo, and Niekum, 2019). While these models

can work with highly suboptimal trajectories, the momentum that the rankings provide towards the extrapolation of better policies cannot produce optimal behavior that have a inherently different structure to it compared to the demonstrations. Also, automatic noise injection as a method for automatically generating ranked demonstrations cannot accurately describe the whole set of suboptimal behavior. A simple scenario where this fails could be, for example, when an agent is performing a completely different, irrelevant task in a certain portion of the state space.

2.4 Combining expertise

Combining expertise, or multiple policies has been studied in recent works in a reinforcement learning context and most of them fall under the hierarchical reinforcement learning research area (Pateria et al., 2021). For example, the Option Keyboard approach where an agent learns how to combine skills by associating them with *pseudo-rewards* and thus performs combination in the space of intentions has shown promise (Barreto et al., 2019). However to combine these skills, another layer of reinforcement learning has to be used.

Along similar lines, it has been shown how demonstrations from suboptimal experts can be used to achieve better exploration of the complex state-action space (Jeong et al., 2021). Also, policy reuse is another research topic that uses previously learned policies to perform exploration and where a similarity function is defined for policies which intends to weigh policies based on how probable it is that using them for learning the new task will be profitable (Fernández and Veloso, 2006). Both of these research directions describe a kind of expertise combination. However, similarly to hierarchical RL models, these methods arrive at expertise combination through reinforcement learning.

In this project, unlike other research, reinforcement learning is not used in the policy combination step. Instead, it is shown how spectral clustering can be used by the learner agent to guide the combination of its own policy π_L with the estimated policy π_E of the suboptimal expert such that at each state $s \in \mathcal{S}$, the learner chooses to use either its own policy $\pi_L(\mathcal{A}|s)$ or the policy $\pi_E(\mathcal{A}|s)$.

2.4.1 Clustering in Expertise Combination

In order to avoid expensive calculations, such as running an RL model as an additional layer on top of multiple policies in order to produce a combined policy, clustering is explored in this project as the basis for expertise combination. Graph clustering aims to create a partition of the nodes of a graph so as to describe the structure and hence connectedness of its nodes and subgraphs. This is useful, for example, in analysing the behavior of agents moving between different states (nodes). Spectral clustering is known to be a highly efficient method to cluster or partition a graph, since it can use the eigendecomposition of the Laplacian of the graph similarty matrix (such as the adjacency matrix) to group nodes together (Von Luxburg, 2007).

This technique does, however, have the downside of requiring symmetric affinity matrices and hence it does not work with directed graphs. Digraphs are relevant in the

MDP setting where there may be asymmetries in the movement between any two states. The 'DEcomposition into DIrectional COMponents' (DEDICOM) model (Harshman, 1978) which allows for the clustering of directed graphs is a proposed alternative in analysing agent trajectories or policies, for example in computer games (Bauckhage et al., 2014).

Clustering has been used before with success in imitation learning for trying to handle demonstrations which could represent varying intentions or tasks. The Expectation-Maximation (EM) trajectory clustering algorithm which iteratively clusters trajectories by their intentions given any IRL method has been shown to be successful (Babes et al., 2011). This approach is useful in differentiating between demonstrations and thus allows to solve multiple, simpler IRL problems which result in rewards and policies which represent different intentions.

Behavior clustering has also been used in later research, with advances in using non-parametric methods to alleviate the need for specifying a target number of clusters beforehand (Choi and Kim, 2012, and Rajasekaran, Zhang, and Fu, 2017). However, these approaches do not aid in selecting useful portions of two policies to combine for solving problems in multi-task environments.

Chapter 3

Contributions

The contributions of this project are listed as follows:

1. Providing an overview of the problem.
2. Programming experiments to validate the proposed approach of policy combination using spectral clustering.
3. Demonstrating the efficiency of the proposed approach in an IRL setting.

The included project materials contain all of the code that was used in this project. Wherever not stated otherwise in the code base, I have been the sole author. Parts of the code around inverse reinforcement learning and the gridworld environment have been authored either wholly or partially by Billy I. Lyons and have been stated clearly within the code base. Also, open-source libraries have been used and there are a few sections of open-sourced code that have been adapted for this project – these adaptations have also been stated clearly within the code base.

Chapter 4

Problem Statement

This project explores the following scenario. Let there be an MDP without a reward function $\mathcal{M}_{\setminus \mathcal{R}} = \langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \gamma \rangle$, an unknown task (i.e. reward function) $\mathcal{R}$, a demonstrating suboptimal expert agent A_E with a policy π_E optimised to maximise reward $\mathcal{R}_E \neq \mathcal{R}$, and a (learner) agent A_L equipped with 1) a policy π_L optimised to maximise reward $\mathcal{R}_L \neq \mathcal{R}$ and 2) access to an oracle $O_{\mathcal{R}}$ which gives an ordering of policies whereby for some $\pi_1, \pi_2 \in \Pi$, the space of all policies in $\mathcal{M}_{\setminus \mathcal{R}}$, $\pi_1 \geq \pi_2$ means π_1 is better or as suited for completing the task given by $\mathcal{R}$ compared to π_2. How can A_L learn of the expertise π'_E of A_E by receiving a demonstration D^t_E from A_E at every time step t and combine this knowledge of A_E with its own experience to achieve a policy π which is better than π_E and π_L in completing the task $\mathcal{R}$ (that is, $\pi \geq \pi_E$ and $\pi \geq \pi_L$)?

Policy ordering oracle

As in Brown, Goo, Nagarajan, et al., 2019, in this project, instead of making known $\mathcal{R}$, a similar assumption is made by making known an ordering $O_{\mathcal{R}}$ whereby for some $\pi_1, \pi_2 \in \Pi$, $\pi_1 \geq \pi_2$ means π_1 is better or as suited for completing the task given by $\mathcal{R}$ compared to π_2, and $\pi_2 \geq \pi_1$ means π_2 is better suited or as suited to completing the task given by $\mathcal{R}$ compared to π_1. This ordering will be assumed to be equivalent to a function that compares the average rewards of two policies where the average reward of a policy is given by the average discounted sum of rewards starting at each possible initial state in the MDP.

The reason why the assumption is made of the existence of an ordering function is because it is a more generic approach than evaluating policies directly and thus puts less restrictions on the learning model compared to assuming access to the true reward function $\mathcal{R}$. This means the proposed model can be incorporated in RL settings, but also in settings where true reward is not known. This ordering could be made available, for example, by including human feedback or having some calculation of policy average discounted return based on some given heuristics.

State and action space

For this project, the following further assumption is made of the state space $\mathcal{S}$ and action space $\mathcal{A}$ being discrete. It should be noted, however, that all of the solutions explored in this work can be extended to continuous state and action spaces through multiple tactics such as discretisation using basis functions.

Chapter 5

Inverse Reinforcement Learning in Gridworld

5.1 Gridworld dynamics

This project uses the gridworld environment to test out the strategy of learning suboptimal expert rewards and policies and using them for expertise combination. A gridworld is a discrete-state environment with 4 discrete actions consisting of moving left, right, up, or down to an adjacent cell. The gridworld is subject to noise w whereby if the agent chooses an action then it has a w probability of going to a random adjacent cell. For example, if $w = 0.3$ and the agent chooses to take the action to go right, then there is a $0.7 + \frac{0.3}{4}$ probability for the agent to end up in the cell adjacent to the right, cell, a $\frac{0.3}{4}$ probability to end up in the cell above and likewise for the cells below and to the left. Also, it should be noted, that the gridworld environments do not wrap agent positions so, for example, moving right in a right-most cell means staying in place.

Figure 5.1 shows a 5×5 gridword with the optimal policy. The darker the arrows, the larger the agent's preference in taking the action to move in the respective direction. In Figure 5.2 is an example of a 7×7 gridworld with the optimal policy. It can be seen that starting off from any of the first three columns (from the left) of cells, the optimal policy directs the agent to the bottom-left cell and starting off from any of the last three columns of cells the optimal policy directs the agent to the bottom-right cell.

5.2 Maximum Entropy Inverse Reinforcement Learning in Gridworld

The maximum entropy inverse reinforcement learning model (Ziebart et al., 2008) with stochastic gradient ascent was used in this project to estimate the reward and policy of agents from demonstrations. An example estimation from 300 demonstrations of the reward function and policy from Figure 5.2 can be seen in Figure 5.4. While the estimated policy learnt from the estimated reward function using value iteration shows great resemblance, clearly the estimated reward function is quite different to the original.

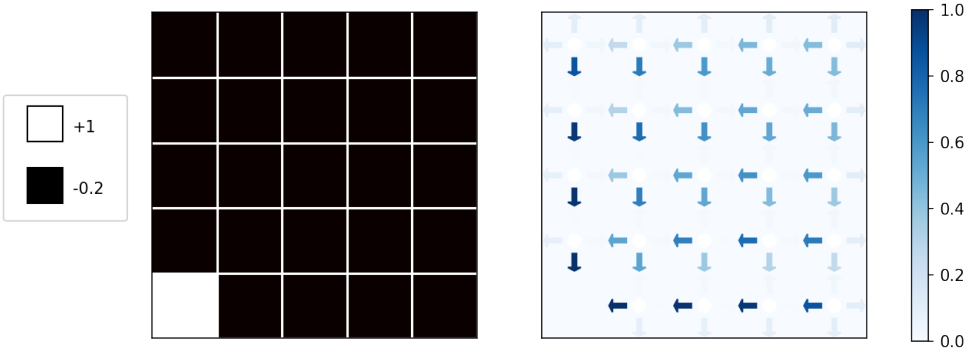

Figure 5.1: Example: Rewards (left) and optimal policy (right) for a 5×5 gridworld with transition noise $w = 0.04$ and discount $\gamma = 0.98$. Rewards are received upon arrival to any cell and the colour of the arrows denote the probability of choosing the action in the direction of the arrow given the current cell.

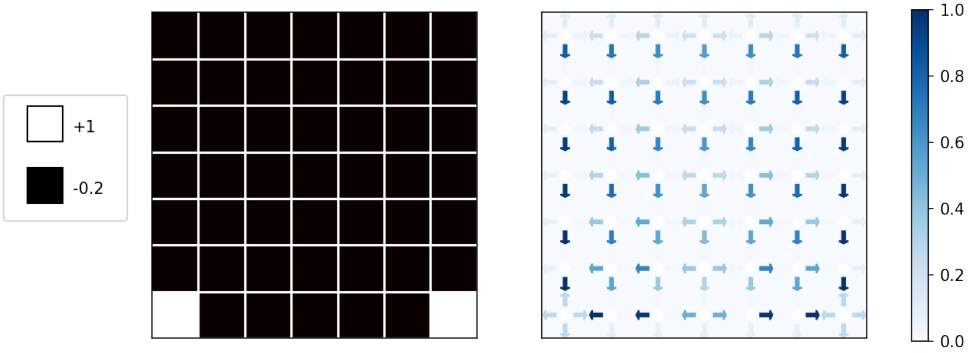

Figure 5.2: Example: Rewards (left) and optimal policy (right) for a 7×7 gridworld with transition noise $w = 0.04$ and discount $\gamma = 0.98$. Rewards are received upon arrival to any cell and the colour of the arrows denote the probability of choosing the action in the direction of the arrow given the current cell.

This is explained by the reward identifiability issue in IRL whereby multiple reward functions can explain the policy of an agent (Russell, 1998).

For both reinforcement learning and inverse reinforcement learning models used in this project, the value iteration algorithm from Sutton and Barto, 2018 was used to calculate the value function given an MDP. The resulting policy was calculated by taking the softmax of the Q-value calculated from the value function.

5.2.1 Estimating Convergence

Since the learning agent does not have access to the true reward function in IRL models, it is important to know after how many demonstrations can the IRL process yield a sufficiently accurate estimation of the expert's policy. One way to do this could be to measure convergence of the value function after each new batch of demonstrations arrives.

Figure 5.3 (top) shows the means and standard deviations of convergence of the true average discounted return of the estimated expert with demonstrations from the example policy shown in Figure 5.1. For the 7×7 gridworld, Figure 5.4 (top) shows the same for the example policy shown in Figure 5.2. The figures also contain the means and standard deviations of the convergence metric. For this experiment, the convergence metric was the Kendall rank correlation coefficient whereby each value was calculated between the new and previous estimation of the value function after each new batch of demonstrations was provided. As can be seen from the figures by visual inspection, the correlation coefficient has a high correlation between the true reward and could indeed be used in a tactic to estimate when the IRL agent has observed sufficiently many demonstrations.

The reason as to why a rank correlation coefficient was used as a comparison instead of, say the root mean squared error between the value functions, is because the multiple value functions can describe the same policy. For example a slight linear perturbation of a value function will have minimal impact, if none, on the policy – in this case, the rank correlation coefficient will be small between the two value functions, but the Euclidean norm would be large.

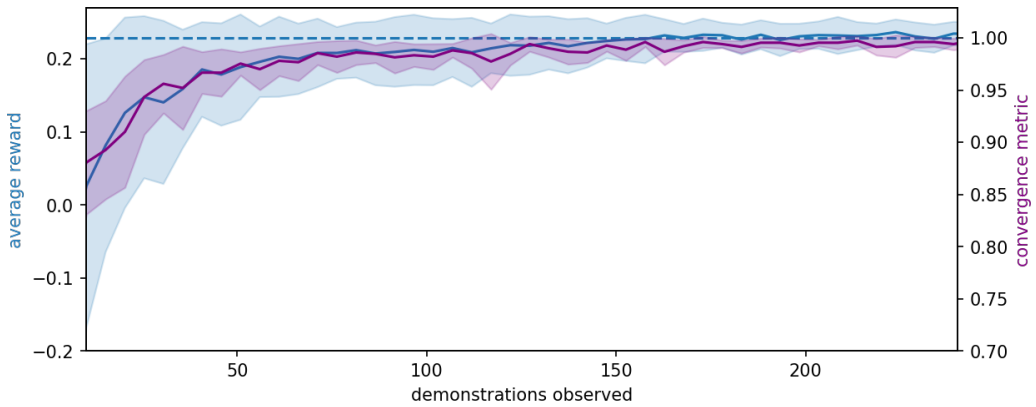

Figure 5.3: Average and standard deviation of estimated agent's true reward (blue) given 10 experiments of performing Maximum Entropy Inverse Reinforcement Learning by receiving demonstrations in a 5×5 gridworld, average and standard deviation of the Kendall rank correlation coefficient (purple) between last two estimated value functions. The average true reward for the actual policy of the agent is given a blue dashed line.

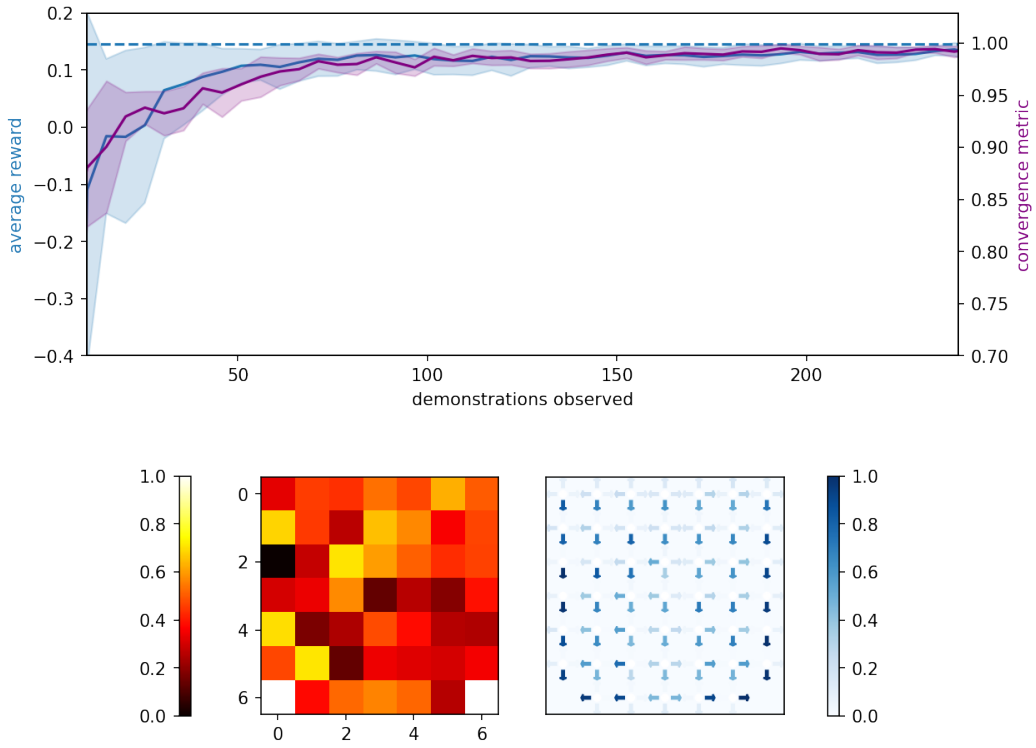

Figure 5.4: Top: average and standard deviation of estimated agent's true reward (blue) given 10 experiments of performing Maximum Entropy Inverse Reinforcement Learning by receiving demonstrations in a 7×7 gridworld, average and standard deviation of the Kendall rank correlation coefficient (purple) between last two estimated value functions. The average true reward for the actual policy of the agent is given a blue dashed line. Bottom left: example of an estimated reward function (normalised to the range $[0, 1]$) of the expert at 300 demonstrations received. Bottom right: respective estimated policy of the expert at 300 demonstrations received.

Chapter 6

Expertise Combination

As part of the research question of this project, given two policies π_1 and π_2, a policy π is seeked such that given some state $s \in \mathcal{S}$ there is either $\pi(s) = \pi_1(s)$ or $\pi(s) = \pi_2(s)$ whereby π is better than the two initial policies. This effectively means that any state, the agent using policy π will either behave like an agent with policy π_1 or an agent with policy π_2. However, we can see that if we have a state space of size 100 (for example, in a 10×10 gridworld), then by brute force, we would have to cover 2^{100} different policies to find the best combination of expertise from π_1 and π_2.

Consider the set of states $S_1 \subseteq \mathcal{S}$ where $\pi(s) = \pi_1(s)$ if and only if $s \in S_1$ and $\pi(s) = \pi_2(s)$ if and only if $s \notin S_1$. Let us define the policy combination function

$$C : \mathcal{P}(\mathcal{S}) \times \Pi \times \Pi \to \Pi$$

with $C(S_1, \pi_1, \pi_2)(s,:) = \pi_1(s,:)$ whenever $s \in S_1$ and otherwise $C(S_1, \pi_1, \pi_2)(s,:) = \pi_2(s,:)$. Since policies describe trajectories across multiple states and hence expertise should not be viewed on a per-state basis, it is useful to realise that S_1 and therefore $\mathcal{S} \setminus S_1$ most probably contains some states that are adjacent to one another as deemed by the transition matrix $\mathcal{T}$. This is useful when we are looking to significantly reduce the search space for a good combined policy π.

To this end, the idea of finding a set of clusters $\mathbf{P}_\mathcal{S}$ of the state space $\mathcal{S}$ is explored such that:

1. $\forall K_1, K_2 \in \mathbf{P}_\mathcal{S}, K_1 \cap K_2 = \emptyset$
2. $\bigcup \mathbf{P}_\mathcal{S} = \mathcal{S}$
3. $\forall K \in \mathbf{P}_\mathcal{S}$, K is connected in $\mathcal{S}$
4. $|\mathbf{P}_\mathcal{S}| < |\mathcal{S}|$

Then a subset $P \subseteq \mathbf{P}_\mathcal{S}$ of the clustering is seeked such that the policy $\pi = C(\bigcup P, \pi_1, \pi_2)$ would be an adequate combination of expertise, i.e. $\pi \geq \pi_1$ and $\pi \geq \pi_2$.

6.1 Clustering the state space

In order to find connected clusters of the state space, the problem of graph clustering is visited. It is important to note that in the case of an MDP, the state space $\mathcal{S}$, combined with the transition matrix $\mathcal{T}$, yields a homogeneous Markov chain if we sum over possible actions $\mathcal{A}$ and assume a policy π to represent state-action probabilities. Note that the choice of π could, for example, be the uniform distribution over actions, regardless of state. Hence, we can convert the MDP into a weighted directed graph, the representation of a Markov chain, and consider the clustering problem as a graph clustering problem.

This yields another large benefit aside from reducing the search space. The benefit comes from the fact that, for clustering the state space, spectral clustering can be used, instead of more complex clustering methods, such as performing trajectory-analysis to compute a high-dimensional latent space which encodes a distance metric between each and every state. Spectral clustering can efficiently cluster graphs given adjacency matrices which is why defining one is the following topic.

It must be noted however, that there have been examples (such as by Bauckhage et al., 2014), where the requirement of symmetry in the adjacency matrix is detrimental to spectral clustering results. For that reason, the clustering with the DEDICOM model was attempted. However, this yielded unsatisfactory results. For one, clusters were not always path-connected, which was one of the requirements in project. This could be due to the high connectivity imposed by the transition matrix in gridworld environments.

6.1.1 Defining an adjacency matrix

First, the weighted adjacency matrix $A_w : \mathcal{S} \times \mathcal{S} \to \mathbb{R}^+$ is defined as

$$A_w(s_0, s_1) = \sum_{a \in \mathcal{A}} w(s_0, a)\mathcal{T}(s_0, a, s_1)$$

which, for some $s_0, s_1 \in \mathcal{S}$, is the probability of going from s_0 to s_1 given a weighting $w : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. For spectral clustering this adjacency matrix is then averaged with its transpose to create a symmetric affinity matrix.

The reason as to why a linear weighting was be introduced in the calculation was to make possible the incorporation of information from the policies π_1 and π_2 which we are looking to combine. This means that clustering could be performed which is not only aware of the system dynamics (the transition matrix) but also of the preferences set out in the policies we are combining.

To make the role of the weighting more intuitive we can consider cases where for some state $s \in \mathcal{S}$ and $a \in \mathcal{A}$ we have that $w(s,a)$ is low in one case and high in the other. In the former case, if $w(s,a)$ is small, then the "connectedness" between s and the states where a can take the agent is smaller than for the latter case, that is when $w(s,a)$ is large. In other words, if we consider that $\mathcal{T}(s,a,s') > 0$ for some $s' \in \mathcal{S}$, then as $w(s,a)$ increases there is a increasingly larger chance that s and s' end up in the same cluster.

In the experiments, several values for w were tried. The first of which was a uniform weighting to compare against the results of values w which depend on π_1 and π_2. That

is, the uniform weighting was defined asas $w_{\text{uni}} = \frac{1}{|\mathcal{A}|}$ given any $s \in \mathcal{S}$ and $a \in \mathcal{A}$. The resulting adjacency matrix $A_{w_{\text{uni}}}$ can be thought as representative of the Markov chain for the agent in the MDP who prefers all actions equally.

The policies π_1 and π_2 were incorporated into the other values of w that were tested. In order for the clustering algorithm not to have any bias towards one policy or the other and hence consider both policies equally, preference was not subjected to either π_1 or π_2 within w. In other words, the roles of π_1 and π_2 were defined to be symmetric. The first weighting that was tested which combines the two policies is

$$w_{\text{avg}} : (s,a) \mapsto \frac{\pi_1(a|s) + \pi_2(a|s)}{2}$$

which is a simple average of the two policies. The reason for introducing this was to explore the scenario where two states have a high probability of being in the same cluster when both policies strongly prefer movement between the two states. However, this weighting loses information about the difference between the two policies. This is why the second weighting that combines two policies was

$$w_{\text{diff}} : (s,a) \mapsto |\pi_1(a|s) - \pi_2(a|s)|$$

which would increase the probability of two states being connected in a cluster whenever the disagreement between the two policies' preferences of movement between the two states increased. To further explore different weightings and since both w_{avg} and w_{diff} take values between 0 and 1 and can thus be interpreted as probabilities, the softmax and complement variations of these weightings were introduced, whereby the softmax was defined with an inverse temperature β for a weighting w by

$$\sigma_\beta(w) : (s,a) \mapsto \frac{\exp(\beta w(s,a))}{\sum_{a' \in \mathcal{A}} \exp(\beta w(s,a'))}$$

and the complement was simply defined for a weighting w as $w^c : (s,a) \mapsto 1 - w(s,a)$.

6.2 Combining policies using the clustered state space

Given a clustering of the state space $\mathbf{P}_\mathcal{S}$, the next step was to find a subset of clusters $P \subseteq \mathbf{P}_\mathcal{S}$ such that $C(\bigcup P, \pi_1, \pi_2)$ would be an improvement to both π_1 and π_2. In this project, the greedy subset selection approach was used, which, one-by-one adds to the the subset of clusters if a new cluster can be found that improves the subset. The description of the algorithm is as follows. First, let $P \leftarrow \{\emptyset\}$. If a cluster $K \in \mathbf{P}_\mathcal{S} \setminus P$ can be chosen such that by the policy ordering oracle $O_\mathcal{R}$ we have

$$C(K \cup \bigcup P, \pi_1, \pi_2) > C(\bigcup P, \pi_1, \pi_2)$$

and

$$C(K \cup \bigcup P, \pi_1, \pi_2) > C(K' \cup \bigcup P, \pi_1, \pi_2)$$

for all $K' \in \mathbf{P}_\mathcal{S} \setminus P, K' \neq K$, then set $P \leftarrow P \cup \{K\}$. This process is repeated until a suitable K cannot be chosen. In the interest of not preferring a single policy, this

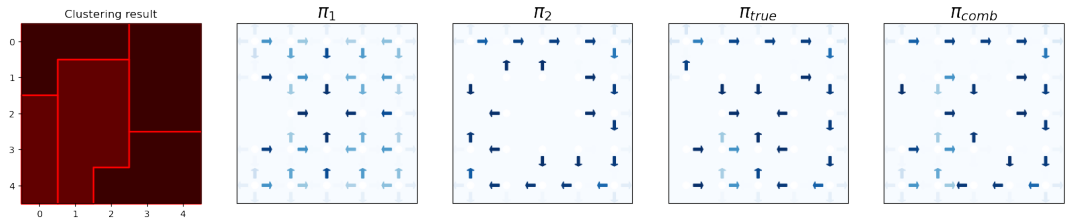

Figure 6.1: Spectral clustering result for w^c_{diff} and the test environment no. 6 with number of clusters $n = 5$. π_1 and π_2 are shown alongside the true policy π_{true} for the environment in the second, third, and fourth subfigure from the left, respectively. On the right is the combined policy $\pi_{\text{comb}} = C(\bigcup P, \pi_1, \pi_2)$. On the left, the clusters that were selected from π_1 are in a a lighter red, i.e. they make up P.

whole procedure is repeated to find a suitable $P' \subseteq \mathbf{P}_{\mathcal{S}}$, such that $C(\bigcup P', \pi_2, \pi_1)$ is an improvement to π_1 and π_2. Then, either $C(\bigcup P, \pi_1, \pi_2)$ or $C(\bigcup P', \pi_2, \pi_1)$ is chosen depending on which choice was better using the oracle $O_{\mathcal{R}}$. Figure 6.1 contains an example result from the clustering approach with weighting w^c_{diff}. In the figure, the similarity of π_{comb} to π_{true} is apparent. This approach has a time-complexity of $O(n^2)$ where n is the number of clusters. For the purposes of having $n << |\mathcal{S}|$, such as $2 \leq n \leq 8$, this algorithm is suitable. More examples can be seen in Appendix C.

6.3 Evaluating Expertise Combination

Several experiments were ran to deem whether using spectral clustering in order to combine policies is a viable approach to expertise combination. The goal was to understand if it is possible to use spectral clustering to create combined policies that better suit the task at hand than the policies being combined, and, whether using the original policies π_1 and π_2 in the weightings w in the creation of the adjacency matrix A_w can yield more optimal results than using just w_{rand}.

6.3.1 Evaluation with combined policy average return

The different weightings and their combinations across multiple different tasks were compared through measuring the average discounted return of the resulting combined policies. Several values for the number of clusters n with $n \in N$ where $N = \{2,3,4,5,6,7\}$ and several environments were tried.

Two metrics were used to compare the weighting matrices that were input to the expertise combination algorithm. The first, most obvious one was comparing the performance of the resulting, combined policy. Since being successful using a small number of clusters is more useful and shows the success of the choice of w, a weighted scoring $\mathbf{s}$ was defined by

$$\mathbf{s}(w) = \frac{\sum_{n \in N} \alpha_n s_n(w)}{\sum_{n \in N} \alpha_n}$$

for each choice of weighting w, where $s_n(w)$ is the average discounted return (normalised to between 0 and 1 across all used n and w) in the environment given the

resulting combined policy using the clustering defined by A_w, and where n is the number of clusters and α_n is the respective weight given the number of clusters n. The choice of weight α_n was $\alpha_n = e^{\frac{1}{n}}$ which means that for numbers of clusters $n, m \in N, n < m$ implies that $\alpha_n > \alpha_m$.

The second metric that was used was the monotonicity of the scores as cluster sizes increased. An adjacency matrix A_w would have a high positive monotonicity score if the resulting combined policy's score increased each time the number of clusters was increased. The reason for this metric is to analyse the reliability of the clustering as the number of clusters are increased given some adjacency matrix. That is, it is desirable that if the cluster number is increased by one, there is a high probability that performance will not be worse than before. The monotonicity score $\mathbf{m}$ was defined, for a choice of weighting w, by

$$\mathbf{m}(w) = o(w)\frac{\max_i(s_i(w)) - \min_i(s_i(w))}{\sum_{j\in N\setminus\{7\}} |s_{j+1}(w) - s_j(w)|}$$

where $o(w) = \text{sgn}(\arg\max_i(s_i(w)) - \arg\min_i(s_i(w)))$ and where sgn is the sign function. A monotonicity score $\mathbf{m}$ that is close to 1 indicates high monotonicity in the sequence $(s_n(w))$ where $n \in N$ increases. On the other hand, a monotonicity score that is negative indicates that the respective sequence tends to decrease more than increase as the number of clusters n increases.

Several experiments in multiple 5×5 and 9×9 gridworld environments were run, all of which have been included in Appendix A. Figure 6.2 contains the $s_n(w)$ values for the several w and $n \in N$ that were tested in the 5×5 gridworlds. As can be seen in the figure, the performance of the combined policy was often better than the original policies π_1 and π_2, even with a low number of clusters. However, there is no clear indication of a statistical difference between the different weightings. Also, it was often the case that average discounted returns were yielded which were better than the expected discounted return of the agent that directly solves the true problem, specified by $\mathcal{R}$. This is because the policy calculation from the value function that was learnt used the softmax function to convert q-values into probabilities, instead of a greedy policy.

The scoring results for the weightings are included in Appendix B, with mean values and standard deviations included. The correlation between the two gridworld types, the 5×5 and 9×9 gridworlds, was also measured. There was a 0.75 Pearson's correlation coefficient between the mean values for $\mathbf{s}(w)$ in the 5×5 and 9×9 gridworlds (see the mean values in Tables B.1 and B.3) with a low p-value of 0.0031. This could imply that there is a high probability that the scoring function $\mathbf{s}$ retains some consistency across environment size and hence it could be the case that certain weightings have better performance than others in general, for example, the values $\mathbf{s}(\sigma_{10}(w_{\text{avg}}))$ in both gridworld sizes were one of the best. For the monotonicity scores $\mathbf{c}(w)$ for weightings w, the respective Pearson's correlation coefficient between the mean values was a low 0.07. This means that there was not a strong consistency of $\mathbf{c}$ between the 5×5 and 9×9 gridworlds.

In general, with these results it has been shown that the proposed approach of expertise

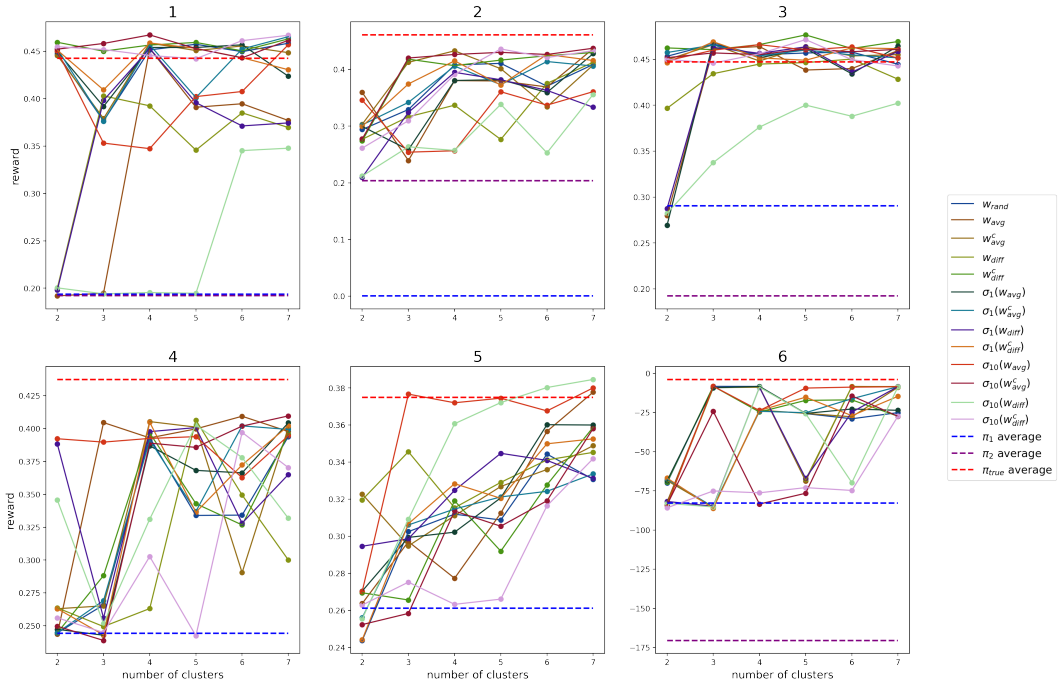

Figure 6.2: Spectral clustering results (average discounted return for the combined policies) for the 6 different test 5×5 gridworld environments from Figure A.1. Each clustering result is recorded for each weighting w and for four different numbers of clusters. For the fourth and fifth environments, the π_2 average discounted return values were omitted from the figure (they were -0.748 and -0.7 respectively).

combination works well and that it has consistency in producing the desired results. These results also demonstrate that certain weightings seem to have an edge. However, since the standard deviations of results are relatively high, then it is hard to single out one weighting as the best and it is also not possible to conclude that, for example using w_{rand}, the only weighting that did not incorporate information from π_1 and π_2, was somehow worse that using the other weightings. Even though policy combination with spectral clustering is clearly successful, since large differences between the results for the different weightings were not seen, it was hypothesised that these differences will come out in larger gridworlds. The experiment designed for larger gridworlds is described next.

6.3.2 Evaluation using policy mean euclidean distance

For larger gridworlds, it was decided to compare clusterings directly instead of through greedy subset selection and the approximation of the resulting combined policies' expected discounted returns. The latter can be an expensive calculation in large environments when a large number of simulations is required for the average value would be statistically meaningful. For the direct approach, for each cluster in a clustering, either π_1 or π_2 was chosen depending on which policy the target policy π_{true} was closest to. The distance metric that was used was the root mean squared error $\text{RMSE} : \Pi \times \Pi \times \mathcal{P}(\mathcal{S}) \to \mathbb{R}$ which is expressed as:

$$\text{RMSE}(\pi, \pi', C) = \sqrt{\frac{1}{|C||\mathcal{A}|} \sum_{s \in C} \sum_{a \in \mathcal{A}} (\pi(s,a) - \pi'(s,a))^2}$$

where π, π' are arbitrary policies and C is a cluster within the state space. Then as the new policy π_{comb} was defined this way, the resulting mean euclidean distance was recorded $\text{RMSE}(\pi_{\text{comb}}, \pi_{\text{true}}, \mathcal{S})$.

For the experiment, one of the 9×9 gridworlds was superimposed (see Appendix A, Figure A.2, environment no. 1) into the bottom right corner of a simple 50×50 gridworld. Then, the same weightings as for the previous experiment were used for creating the clusters and above procedure was performed for a longer sequence of number of clusters n. The results are shown in Figure 6.3. Clearly, there is a large difference between the weightings that were used. The scores for w_{rand} did not seem to be getting better as the number of clusters was increased, but for some others, like w_{avg}, it was definitely the case. From this the conclusion was made that incorporating π_1 and π_2 definitely has an impact towards getting better policy combination results and some weightings have clear advantages over others.

6.4 Expertise Combination with Inverse Reinforcement Learning

The final goal of this project was to validate spectral clustering as an approach to guide expertise clustering within an inverse reinforcement learning cycle. Within multiple gridworld environments, the learner (IRL) agents were trained to see how

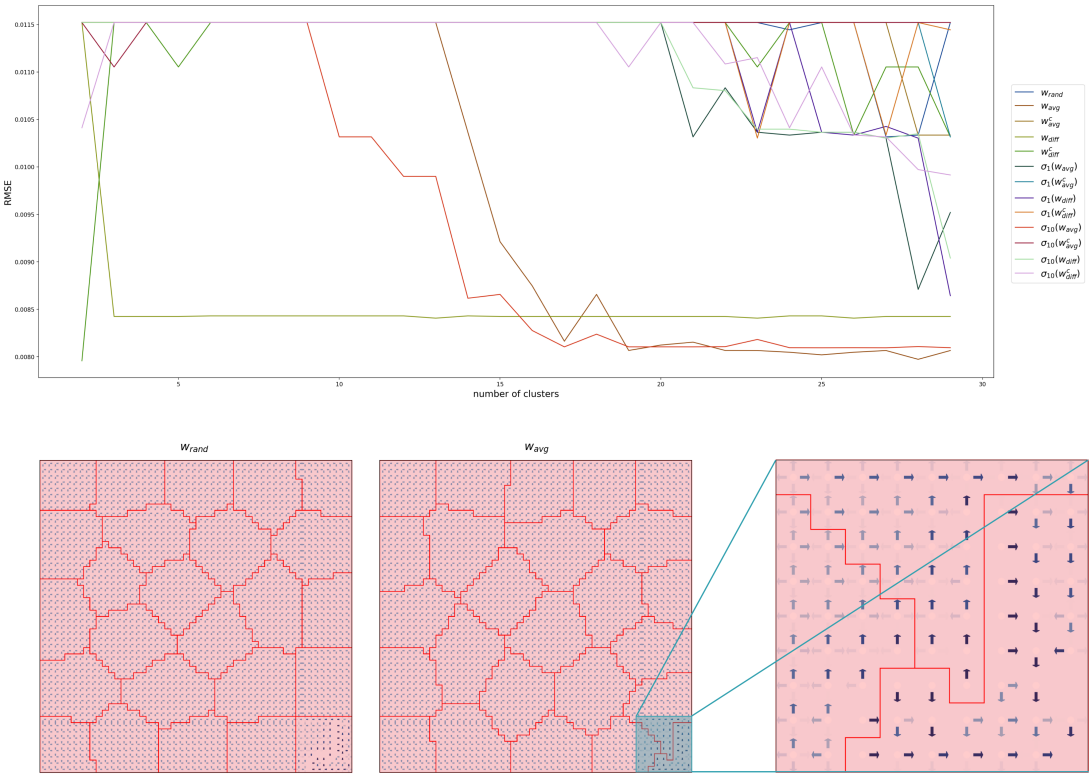

Figure 6.3: Clustering results for several different weightings in a 50×50 gridworld whereby the first 9×9 gridworld from A.2 has been superimposed in the bottom right corner. Top: root mean squared error between the true policy and the best clustering depending on the number of clusters for different weightings. Bottom: clusterings displayed on top of π_1 in the gridworld environment. Bottom left: clustering of the gridworld into 25 clusters using the w_{rand} weighting. Bottom middle: clustering of the gridworld into 20 clusters using the w_{avg} weighting. Bottom right: zoomed in view of the relevant 9×9 area of the clustering on its left.

useful the Kendall rank correlation coefficient is as a threshold for performing expertise combination between the learner's policy π_L and the estimated suboptimal expert's policy π_E. Thus, multiple thresholds were tested alongside multiple target number of clusters and the results compared.

For this experiment the complement of the average policy π^c_{avg} was used to weigh the transition matrix to create the affinity matrix required for spectral clustering. The result was also compared to clustering results based on the uniform policy π_{rand}. Both weightings resulted in combined policies that were better than the demonstrator and learner originally. As can be seen in Figure 6.4, π^c_{avg} displayed better monotonicity (as cluster sizes increased, returns tended to increase) than π_{rand}. This correlates with the fact that in general, π_{rand} had lower monotonicity scores (as seen in Appendix 2, tables B.2 and B.4). Returns were quite varying up until the high threshold levels when in most cases the mean return increased. Also, in a lot of cases, variability decreased at higher threshold levels.

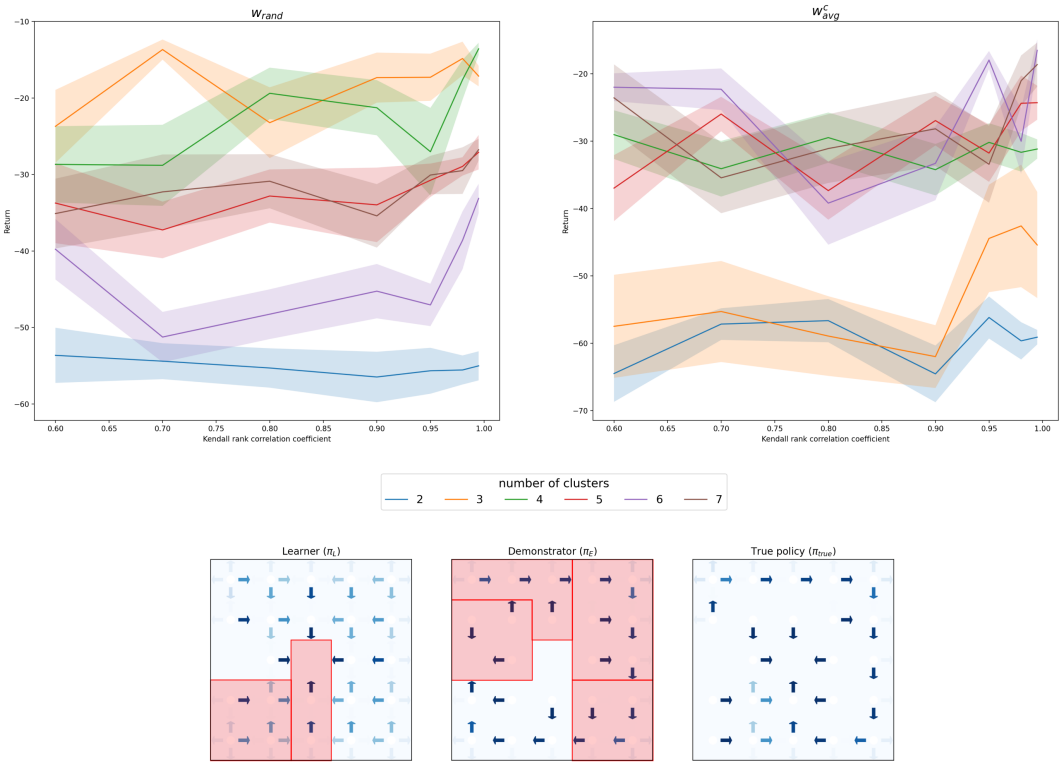

Figure 6.4: Top: spectral clustering results in the sixth 5x5 gridworld (seen in Figure A.1). Discounted return given the Kendall rank correlation correlation as a threshold to performing expertise combination. Each line is the average discounted return given a number of clusters over 10 runs. Shaded areas are 30% standard deviations. Top left: results with clustering using the adjacency matrix computed using a uniformly random policy. Top right: results with clustering using the adjacency matrix computed using the complement of the average policy between the learner's policy and the estimated expert's policy. For this simulation, the learner's own expected discounted return was -82.7, while for the demonstrator and the optimal expert the values were -170.5 and -3.91 respectively. Bottom: the learner's, demonstrator's and the true policy for the environment. Contains an example clustering made by the weighting w^c_{avg}. Selected areas for policy combination from the learner and demonstrator are highlighted on the learner's and demonstrator's policies, respectively.

Chapter 7

Discussion

Being able to combine the expertise of a demonstrator with one's own in order to complete a new task is a highly beneficial skill. In this project, we showed that spectral clustering as the method driving expertise combination works well in reducing search space and most importantly allows producing policies that are better than the original input policies. This means that there are concrete implications to the per-cluster policy combination we have presented. This list includes a positive effect on performance, that is, there is less learning from scratch. Also, expertise combination allows for less exploratory interaction with the original environment, which, in a real-life setting, for example in robotics, could be highly beneficial in terms of safety.

Expertise Combination

For smaller environments we did not observe stark differences between using the different weightings for creating the adjacency matrix which was the input for spectral clustering. There was however some correlation in the scoring $\mathbf{s}$ between the two environment sizes and thus there is an indication that this correlation may scale into even bigger environment sizes. For example, the $\sigma_{10}(w_{\text{avg}})$ weighting was one of the best in both the 5×5 and 9×9 gridworld environments. The same weighting also performed well in our 50×50 gridworld experiment. The large-scale environment experiment also showed that the best performing results came when weightings that consisted of a combination of π_1 and π_2 were used.

For future research, we believe that a larger set of environments and weightings should be explored. Due to the connection between the weight and the probability of two states being connected in a cluster, there could be further mathematical analyses done to derive bounds in the space of possible weighting functions. Also, an interesting avenue would be to explore the search of weightings by training a neural network to estimate an optimal per-state weighting w_{net} with an input of two probability distributions over actions and an action. For this function the loss could be derived from the fact that for some $s \in \mathcal{S}$ and $a \in \mathcal{A}$, the loss of $w(s,a)$ is linked to whether the states $S \subset \mathcal{S}$ that are taken to by action a from state s should belong to the same cluster as s and hence should have the same source policy in the resulting combined policy. Thus, the resulting weighting over π_1 and π_2 could be defined as $w : (s,a) \mapsto w_{\text{net}}(\pi_1(\mathcal{A}|s), \pi_2(\mathcal{A}|s), a)$.

Another aspect to note is that there is a loss of information when combining two policies within a single weighting. For example, w_{avg} loses information about whether preferences were aligned or not, i.e. we lose information on the difference of the policies. To minimise this loss when performing clustering, we believe that another avenue to explore would be to perform clustering separately such that two adjacency matrices A_{π_1} and A_{π_2} are used whereby the two resulting clusterings are combined instead. Additionally, while we explored weightings which combined two policies symmetrically in order to have no bias, introducing bias when performing weighting *could* be beneficial, for example by using the policy ordering oracle $O_{\mathcal{R}}$ to give a larger weight to the policy which is deemed better.

As to the use of spectral clustering, we hypothesise that the fact that spectral clustering loses information about the directionality of the policies means that the resulting clusterings are not as optimal. This loss of information could be quite large, since, given a Markov chain where the adjacency matrix has been post-processed to be symmetric, we can infer about original trajectories, but not the direction of trajectories. Moving forward, to address the issue of the requirement of symmetry, we believe that exploring weighted directed graph clustering methods would yield benefits. In this project, we tested the DEDICOM model, which, compared to spectral clustering, failed to consistently produce clusters that were connected. However, there are multiple other models which could be used, such as directed graph clustering using weighted cuts (see Meilă and Pentney, 2007).

The assumption of the existence of a policy ordering oracle $O_{\mathcal{R}}$ was important as it allowed us to choose the relevant subset $P \subseteq \mathbf{P}_{\mathcal{S}}$. However, we believe there are additional, relatively simple optimisations which could decrease the search space and increase performance even further and thus reduce the reliance on the oracle within the best-subset search algorithm. Firstly, clusters where both policies are highly similar (e.g. have a low RMSE between the two policies in the cluster) could be ignored by the best subset selection algorithm. Secondly, certain inefficient policy combinations could be excluded by performing simple trajectory analysis to keep the number of cluster boundary crossings to a minimum. That is, if for a trajectory, the number of cluster boundary crossings is close to the length of the trajectory, then it is highly probable that the trajectory is not exhibiting behavior similar to either source policy.

The use of the greedy subset selection should also be noted. This algorithm does not guarantee that it will find the optimal subset from $\mathbf{P}_{\mathcal{S}}$. Finding the optimal subset from a set is an NP-hard problem (Natarajan, 1995 and Thompson, 2022), and employing existing, better solutions in this domain than the greedy approach are left for subsequent research.

Expertise Combination in Inverse Reinforcement Learning

We put expertise combination in an inverse reinforcement learning scenario and showed that it is possible to get desirable results even when one of the policies being combined is derived from an estimation of a reward function. Hence it was shown that even if the demonstrating agent is highly suboptimal, we are able to use expertise combination to produce near-optimal policies.

On average there is a decrease in the variability of the average returns as the expertise combination threshold, the Kendall rank correlation coefficient, was increased. This confirms the fact that a higher expertise combination threshold is desirable when looking for consistent results. Also, more importantly, increasing the threshold of when expertise combination is used from around 0.95 onwards, on average showed a slight increase in average returns. Both of these results we believe are due to the fact that at these high thresholds, the estimated demonstrator's policy is very close to the true one and hence policy combination benefits from this increased accuracy.

For future research, we would like to explore the scenario when multiple demonstrators are available to the learner. This method of expertise combination could be scaled to combine the expertise of multiple agents. For example, one learner agent could track multiple demonstrators and perform expertise combination with the demonstrator with whom the combined policy is the best. This could be defined as an iterative process, whereby the underlying environment and agents could be changed over time. Also, while we used maximum entropy inverse reinforcement learning, there is a large collection of IRL models that could also be used with the expertise combination method proposed in this project.

Chapter 8

Conclusion

This project investigated policy combination with spectral clustering in a multi-task inverse reinforcement learning setting. Inspired by scenarios from real life and robotics, the prospect of being able to join two policies to better complete a new task than either the two policies can is desirable. This is because, by being able to efficiently combine expertise, there is a potential to interact less with the environment directly, thus reducing learning time and possibly expensive interaction with the environment. Also, this opens up the potential to learn from other agents, where their behavior may not be optimising for performing the target task directly, thereby allowing the relaxation of the assumption of expert optimality in inverse reinforcement learning.

With the availability of a policy ordering oracle which can compare two policies in the context of the true reward function, expertise combination was introduced whereby one policy is combined with another such that at every state the new policy either follows the first or the second policy. Since the search space for a good expertise combination combinatorically explodes as state space increases, we looked at clustering the state space so as to reduce search space and thus combine policies not on a per-state basis, but a per-cluster basis.

We showed that in the gridworld environment, using spectral clustering to combine policies is a useful technique and weighing the adjacency matrix by the two policies that are being combined yield clustering results whereby less clusters are required to yield more optimal results. We also demonstrated that the Kendall rank correlation coefficient is a useful metric to estimate the convergence of the inverse reinforcement learning model, so as to choose an optimal time to combine the policy of the learner agent and the demonstrating agent. Even at low correlation thresholds combining policies yielded results that outperform the original learner and demonstrator.

In future research, we would like to explore the use of expertise combination in continuous environments using techniques such as discretisation with basis functions. We would like to look into the benefits of using clustering methods that allow for asymmetric adjacency matrices. Furthermore, we believe that limiting the search space for the best-subset algorithm by excluding clusters which yield redundant combined policies is an avenue that could further advance the approach presented in this thesis.

Bibliography

Babes, Monica et al. (2011). "Apprenticeship learning about multiple intentions". In: *ICML*.

Barreto, André et al. (2019). "The option keyboard: Combining skills in reinforcement learning". In: *Advances in Neural Information Processing Systems* 32.

Bauckhage, Christian et al. (2014). "Beyond heatmaps: Spatio-temporal clustering using behavior-based partitioning of game levels". In: *2014 IEEE Conference on Computational Intelligence and Games*, pp. 1–8. DOI: 10.1109/CIG.2014.6932865.

Brown, Daniel S., Wonjoon Goo, Prabhat Nagarajan, et al. (2019). *Extrapolating Beyond Suboptimal Demonstrations via Inverse Reinforcement Learning from Observations*. arXiv: 1904.06387 [cs.LG].

Brown, Daniel S., Wonjoon Goo, and Scott Niekum (2019). *Better-than-Demonstrator Imitation Learning via Automatically-Ranked Demonstrations*. arXiv: 1907.03976 [cs.LG].

Choi, Jaedeug and Kee-Eung Kim (2012). "Nonparametric Bayesian inverse reinforcement learning for multiple reward functions". In: *Advances in Neural Information Processing Systems* 25.

Fernández, Fernando and Manuela Veloso (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.

Harshman, Richard A (1978). "Models for analysis of asymmetrical relationships among N objects or stimuli". In: *First Joint Meeting of the Psychometric Society and the Society of Mathematical Psychology, Hamilton, Ontario, 1978*.

Hussein, Ahmed et al. (2017). "Imitation learning: A survey of learning methods". In: *ACM Computing Surveys (CSUR)* 50.2, pp. 1–35.

Jeong, Rae et al. (2021). *Learning Dexterous Manipulation from Suboptimal Experts*. arXiv: 2010.08587 [cs.RO].

Kakade, Sham M. (2003). "On the sample complexity of reinforcement learning." In:

Meilă, Marina and William Pentney (2007). "Clustering by weighted cuts in directed graphs". In: *Proceedings of the 2007 SIAM international conference on data mining*. SIAM, pp. 135–144.

Mnih, Volodymyr et al. (2015). "Human-level control through deep reinforcement learning". In: *Nature* 518, pp. 529–533.

Natarajan, Balas Kausik (1995). "Sparse approximate solutions to linear systems". In: *SIAM journal on computing* 24.2, pp. 227–234.

Pateria, Shubham et al. (2021). "Hierarchical reinforcement learning: A comprehensive survey". In: *ACM Computing Surveys (CSUR)* 54.5, pp. 1–35.

Piot, Bilal, Matthieu Geist, and Olivier Pietquin (2017). "Bridging the Gap Between Imitation Learning and Inverse Reinforcement Learning". In: *IEEE Transactions on Neural Networks and Learning Systems* 28.8, pp. 1814–1826. DOI: 10.1109/TNNLS.2016.2543000.

Rajasekaran, Siddharthan, Jinwei Zhang, and Jie Fu (2017). "Inverse reinforce learning with nonparametric behavior clustering". In: *arXiv preprint arXiv:1712.05514*.

Russell, Stuart (1998). "Learning Agents for Uncertain Environments (Extended Abstract)". In: *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*. COLT' 98. Madison, Wisconsin, USA: Association for Computing Machinery, pp. 101–103. ISBN: 1581130570. DOI: 10.1145/279943.279964. URL: https://doi.org/10.1145/279943.279964.

Sutton, R.S. (Aug. 1988). "Learning to Predict by the Methods of Temporal Differences". In: *Mach. Learn.* 3.1, pp. 9–44. ISSN: 0885-6125. DOI: 10.1023/A:1022633531479. URL: https://doi.org/10.1023/A:1022633531479.

Sutton, R.S. and A.G. Barto (2018). *Reinforcement Learning, second edition: An Introduction*. Adaptive Computation and Machine Learning series. MIT Press. ISBN: 9780262352703. URL: https://books.google.co.uk/books?id=uWV0DwAAQBAJ.

Taylor, Matthew E., Halit Bener Suay, and Sonia Chernova (2011). "Integrating Reinforcement Learning with Human Demonstrations of Varying Ability". In: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '11. Taipei, Taiwan: International Foundation for Autonomous Agents and Multiagent Systems, pp. 617–624. ISBN: 0982657161.

Thompson, Ryan (2022). "Robust subset selection". In: *Computational Statistics & Data Analysis*, p. 107415.

Tomasello, Michael (2009). *Why we cooperate*. MIT press.

Von Luxburg, Ulrike (2007). "A tutorial on spectral clustering". In: *Statistics and computing* 17.4, pp. 395–416.

Watkins, Christopher (Jan. 1989). "Learning From Delayed Rewards". In:

Yang, Mengjiao, Sergey Levine, and Ofir Nachum (2022). "TRAIL: Near-Optimal Imitation Learning with Suboptimal Data". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=6q_2b6u0BnJ.

Ziebart, Brian D et al. (2008). "Maximum entropy inverse reinforcement learning." In: *Aaai*. Vol. 8. Chicago, IL, USA, pp. 1433–1438.

Appendix A

Environments

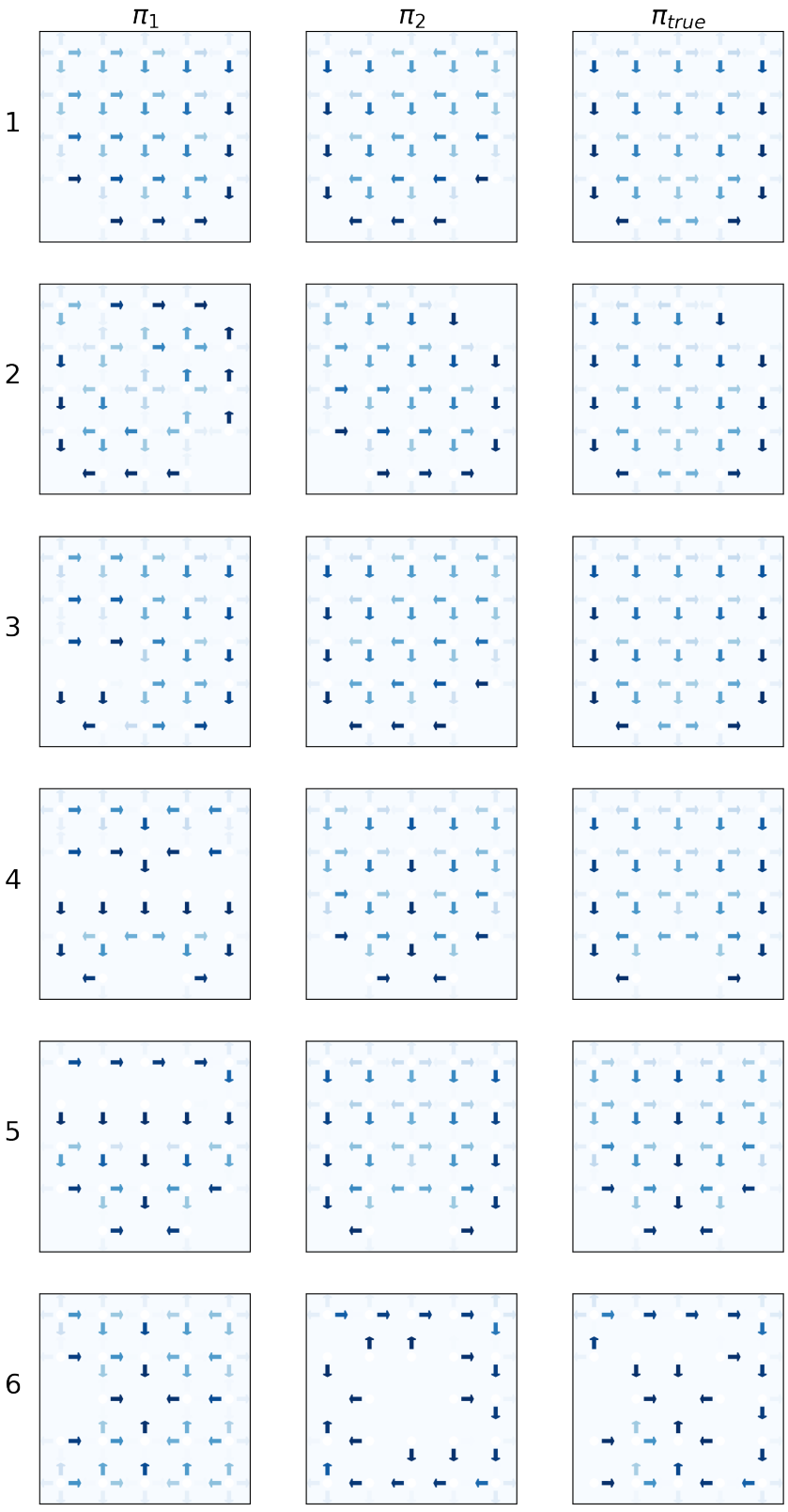

Figure A.1: 6 different 5×5 gridworlds (per row) with π_1 (left), π_2 (centre) and the true policy π_{true} (right). The darker the arrow, the larger the preference in respective action. A state with no arrows is a terminal state.

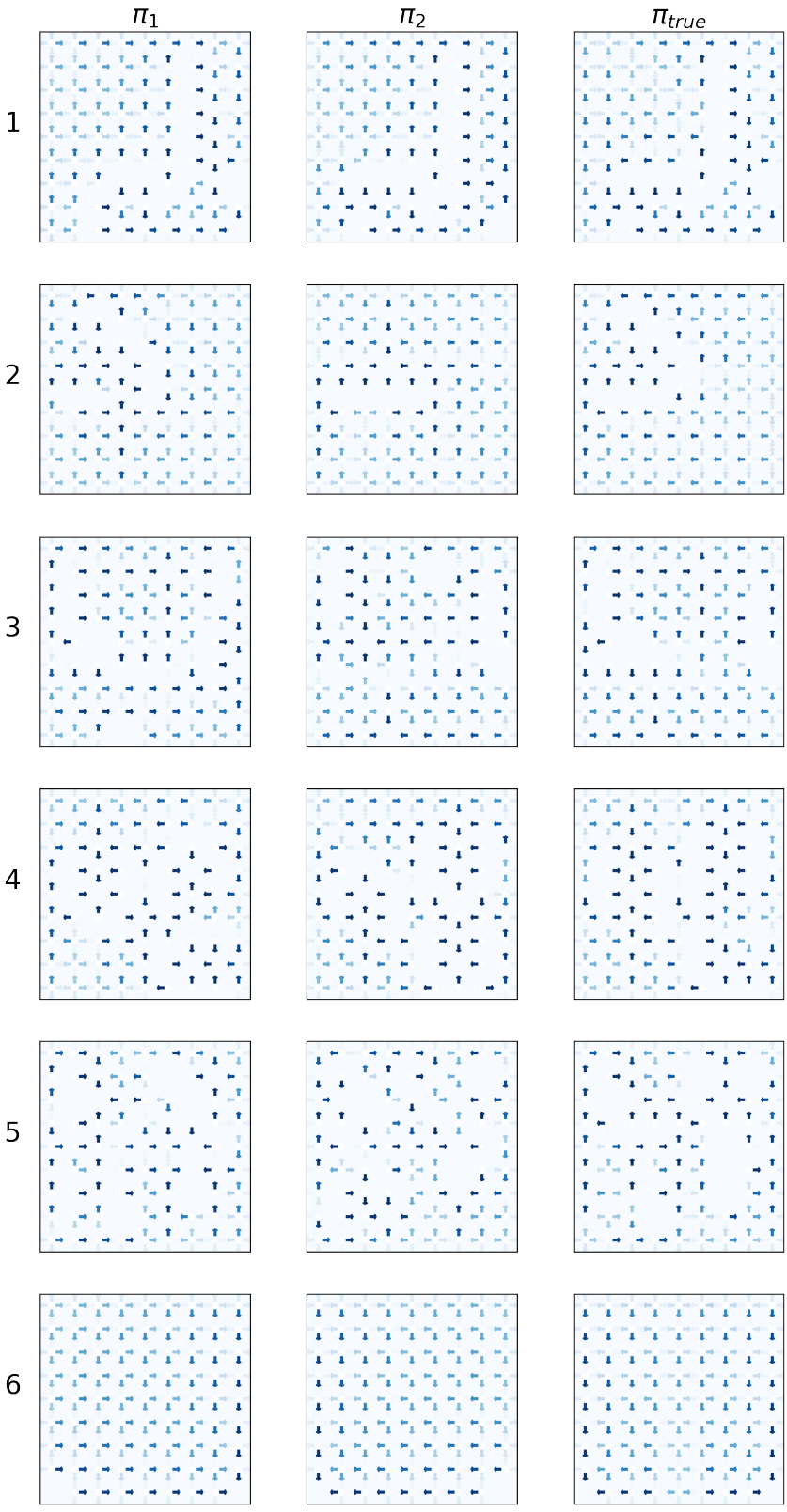

Figure A.2: 6 different 9×9 gridworlds (per row) with π_1 (left), π_2 (centre) and the true policy π_{true} (right). The darker the arrow, the larger the preference in respective action. A state with no arrows is a terminal state.

Appendix B

Weigthing scores

w	1	2	3	4	5	6	mean	std
w_{rand}	0.90	0.68	0.90	0.48	0.42	0.73	0.69	0.18
w_{avg}	0.48	0.65	0.71	0.76	0.47	0.68	0.62	0.11
w^c_{avg}	0.90	0.74	0.90	0.55	0.56	0.56	0.70	0.16
w_{diff}	0.55	0.52	0.78	0.37	0.63	0.68	0.59	0.13
w^c_{diff}	**0.96**	0.79	**0.95**	0.51	0.41	0.73	0.73	0.20
$\sigma_1(w_{\text{avg}})$	0.89	0.60	0.71	0.53	0.51	0.74	0.66	0.14
$\sigma_1(w^c_{\text{avg}})$	0.88	0.70	0.91	0.55	0.44	0.58	0.68	0.17
$\sigma_1(w_{\text{diff}})$	0.60	0.52	0.73	0.69	0.54	0.48	0.59	0.09
$\sigma_1(w^c_{\text{diff}})$	0.90	0.74	0.90	0.54	0.49	0.57	0.69	0.17
$\sigma_{10}(w_{\text{avg}})$	0.77	0.48	0.91	**0.87**	**0.77**	**0.76**	**0.76**	0.14
$\sigma_{10}(w^c_{\text{avg}})$	0.96	**0.82**	0.91	0.58	0.38	0.42	0.68	0.23
$\sigma_{10}(w_{\text{diff}})$	0.18	0.29	0.43	0.58	0.67	0.47	0.44	0.17
$\sigma_{10}(w^c_{\text{diff}})$	0.95	0.69	0.88	0.34	0.29	0.21	0.56	0.29

Table B.1: Weighted scores $\mathbf{s}(w)$ for different weightings w across the six 5×5 gridworlds in Figure A.1. Includes means and standard deviations.

w	1	2	3	4	5	6	mean	std
w_{rand}	0.49	0.60	-0.54	0.56	0.84	0.71	0.44	0.45
w_{avg}	0.76	0.58	0.80	0.83	0.74	0.38	0.68	0.16
w^c_{avg}	0.48	0.43	-0.29	0.39	0.66	0.75	0.40	0.33
w_{diff}	0.64	0.53	0.71	0.57	-0.35	0.39	0.41	0.36
w^c_{diff}	0.32	0.87	0.40	0.52	0.62	0.64	0.56	0.18
$\sigma_1(w_{\text{avg}})$	0.42	0.66	0.70	0.78	1.00	0.74	**0.72**	0.17
$\sigma_1(w^c_{\text{avg}})$	0.32	0.60	-0.40	0.59	**1.00**	0.80	0.48	0.45
$\sigma_1(w_{\text{diff}})$	0.75	0.75	0.74	0.37	0.79	0.39	0.63	0.18
$\sigma_1(w^c_{\text{diff}})$	0.41	0.57	0.40	0.51	0.87	0.62	0.57	0.16
$\sigma_{10}(w_{\text{avg}})$	0.52	0.43	0.45	0.46	0.83	0.71	0.57	0.15
$\sigma_{10}(w^c_{\text{avg}})$	-0.42	**0.96**	0.50	**0.91**	0.87	0.35	0.53	0.48
$\sigma_{10}(w_{\text{diff}})$	**0.95**	0.44	**0.83**	0.48	1.00	0.38	0.68	0.25
$\sigma_{10}(w^c_{\text{diff}})$	0.66	0.87	-0.49	0.50	0.77	**0.91**	0.53	0.48

Table B.2: Monotonicity scores $\mathbf{c}(w)$ for different weightings w across the six 5×5 gridworlds in Figure A.1. Includes means and standard deviations.

w	1	2	3	4	5	6	mean	std
w_{rand}	0.65	0.59	0.50	0.54	0.58	0.90	0.63	0.13
w_{avg}	0.67	**0.77**	0.57	0.60	0.60	0.74	**0.66**	0.07
w^c_{avg}	0.69	0.71	0.50	0.54	0.71	0.72	0.64	0.09
w_{diff}	0.56	0.40	**0.79**	0.41	0.74	0.41	0.55	0.16
w^c_{diff}	0.69	0.69	0.47	0.48	0.64	0.89	0.64	0.14
$\sigma_1(w_{\text{avg}})$	0.62	0.52	0.52	0.54	0.63	0.72	0.59	0.07
$\sigma_1(w^c_{\text{avg}})$	0.53	0.68	0.50	0.55	0.65	0.88	0.63	0.13
$\sigma_1(w_{\text{diff}})$	0.49	0.57	0.67	0.59	0.69	0.55	0.59	0.07
$\sigma_1(w^c_{\text{diff}})$	0.65	0.53	0.43	0.60	0.71	0.89	0.63	0.14
$\sigma_{10}(w_{\text{avg}})$	0.62	0.77	0.59	**0.68**	0.78	0.50	0.66	0.10
$\sigma_{10}(w^c_{\text{avg}})$	**0.77**	0.76	0.34	0.52	0.61	0.89	0.65	0.18
$\sigma_{10}(w_{\text{diff}})$	0.51	0.49	0.48	0.50	0.79	0.21	0.50	0.17
$\sigma_{10}(w^c_{\text{diff}})$	0.75	0.73	0.37	0.33	**0.84**	**0.91**	0.65	0.22

Table B.3: Weighted scores $\mathbf{s}(w)$ for different weightings w across the six 9×9 gridworlds in Figure A.2. Includes means and standard deviations.

w	1	2	3	4	5	6	mean	std
w_{rand}	0.49	-0.61	0.35	-0.48	0.32	-0.32	-0.04	0.44
w_{avg}	-0.51	-0.50	0.41	**0.66**	-0.46	0.46	0.01	0.51
w^c_{avg}	0.36	-0.98	0.67	-0.45	-0.46	**0.84**	-0.00	0.67
w_{diff}	0.38	-0.38	-0.59	0.60	-0.37	0.66	0.05	0.51
w^c_{diff}	-0.55	0.49	0.62	-0.44	-0.56	-0.61	-0.17	0.52
$\sigma_1(w_{\text{avg}})$	0.55	-0.54	**0.74**	0.60	-0.44	0.45	0.23	0.52
$\sigma_1(w^c_{\text{avg}})$	-0.49	-0.93	0.65	-0.38	-0.36	-0.42	-0.32	0.48
$\sigma_1(w_{\text{diff}})$	-0.35	-0.83	0.43	-0.45	0.41	0.80	0.00	0.58
$\sigma_1(w^c_{\text{diff}})$	-0.45	-0.52	-0.39	0.34	-0.40	-0.68	-0.35	0.32
$\sigma_{10}(w_{\text{avg}})$	**0.56**	0.43	0.44	0.48	**0.75**	-0.52	0.36	0.41
$\sigma_{10}(w^c_{\text{avg}})$	0.52	0.87	0.31	0.52	0.72	-0.50	**0.41**	0.44
$\sigma_{10}(w_{\text{diff}})$	-0.40	-0.74	0.53	-0.46	-0.44	0.66	-0.14	0.53
$\sigma_{10}(w^c_{\text{diff}})$	0.46	**0.95**	0.44	0.46	-0.40	0.33	0.37	0.40

Table B.4: Monotonicity scores $\mathbf{c}(w)$ for different weightings w across the six 9×9 gridworlds in Figure A.2. Includes means and standard deviations.

Appendix C

Examples of Expertise Combination

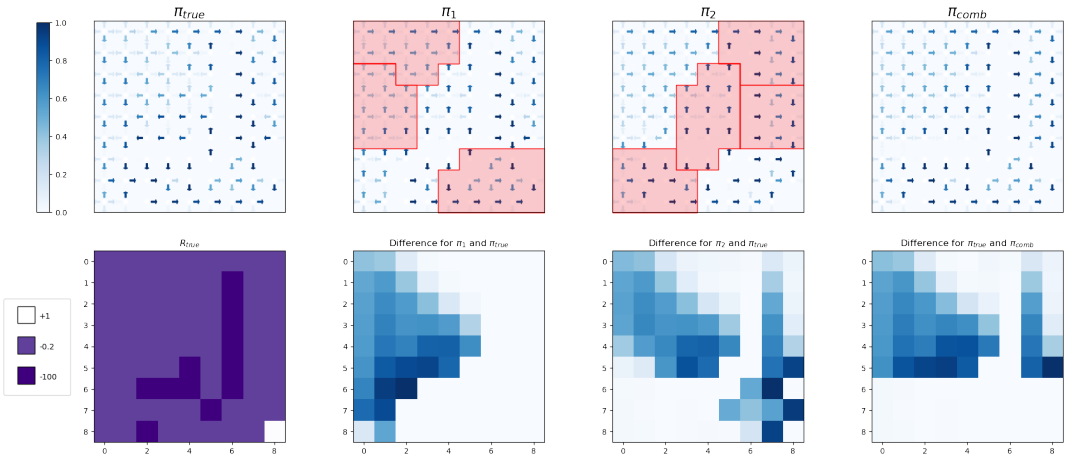

Figure C.1: Policy combination in the first 9×9 gridworld from Figure A.2 using weighting w_{rand} and $n = 7$. Top: true policy (left), combined policy (right), policies that were combined (middle two). Selected clusters from π_1 and π_2 are highlighted. Bottom: true reward (left) and per-state euclidean distances between π_{true} and the respective policy above (others).

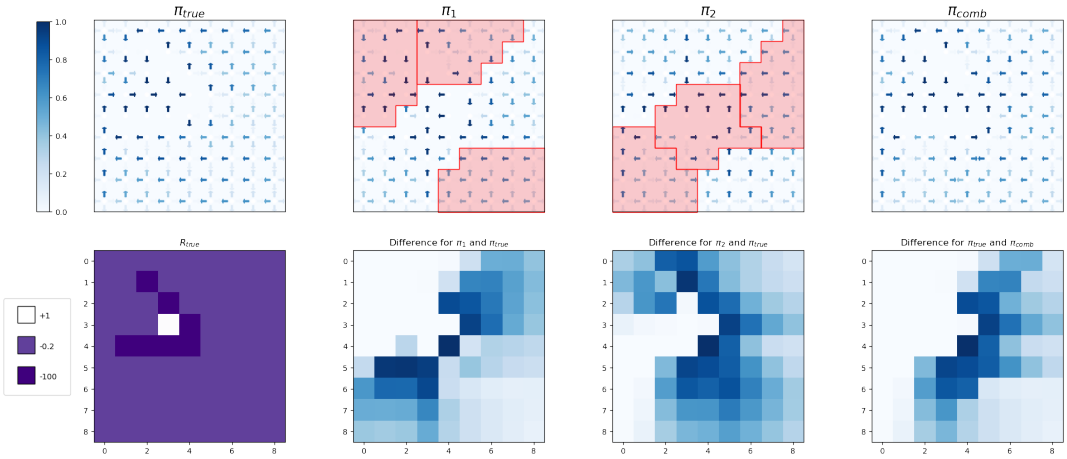

Figure C.2: Policy combination in the second 9×9 gridworld from Figure A.2 using weighting $\sigma_q(w_{\text{avg}})$ and $n = 6$. Top: true policy (left), combined policy (right), policies that were combined (middle two). Selected clusters from π_1 and π_2 are highlighted. Bottom: true reward (left) and per-state euclidean distances between π_{true} and the respective policy above (others).

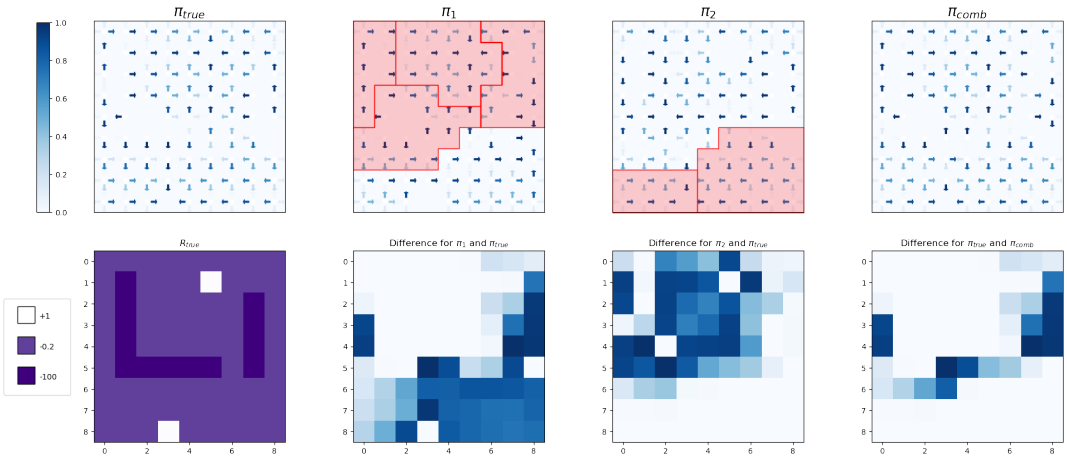

Figure C.3: Policy combination in the third 9×9 gridworld from Figure A.2 using weighting w^c_{diff} and $n = 6$. Top: true policy (left), combined policy (right), policies that were combined (middle two). Selected clusters from π_1 and π_2 are highlighted. Bottom: true reward (left) and per-state euclidean distances between π_{true} and the respective policy above (others).

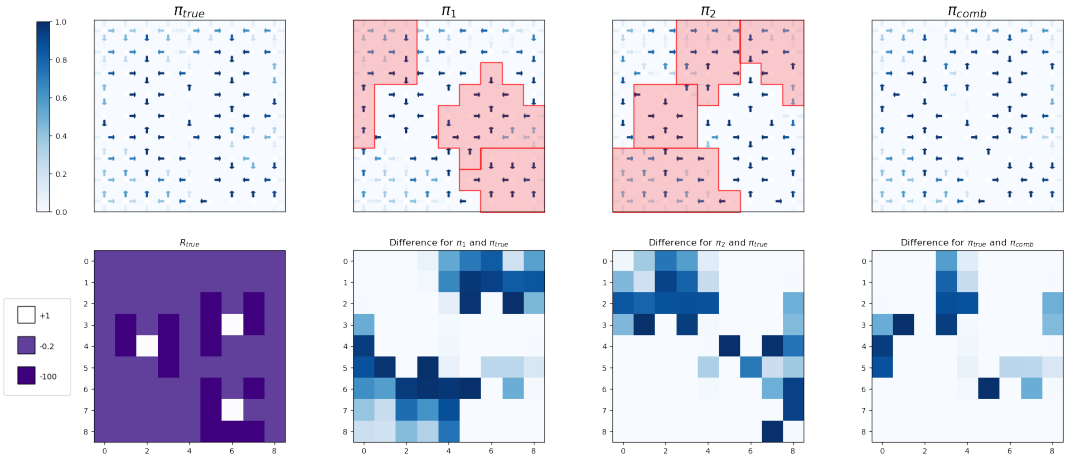

Figure C.4: Policy combination in the fourth 9×9 gridworld from Figure A.2 using weighting w_{avg} and $n = 7$. Top: true policy (left), combined policy (right), policies that were combined (middle two). Selected clusters from π_1 and π_2 are highlighted. Bottom: true reward (left) and per-state euclidean distances between π_{true} and the respective policy above (others).