Multi-Task MARL using shared distilled policies

Samuel Oliveira
Department of Computer Science
University College London
London, UK
samuel.oliveira.23@ucl.ac.uk

Max Jappert
Department of Computer Science
University College London
London, UK
max.jappert.23@ucl.ac.uk

Vishrut Malik
Department of Computer Science
University College London
London, UK
vishrut.malik.23@ucl.ac.uk

Abstract

Multi-task learning has been extensively applied to reinforcement learning problems to counteract its data inefficiency. One possible approach is based on transferring knowledge via policies: Due to the similarity between tasks, an agent's policy for one task is likely to share similarities with its policy for other tasks. This idea has been applied to single-agent RL to create the Distral Framework: An agent aims to learn a policy for each task, while constrained to find a policy that is similar to a shared policy. Based on this idea, we propose MultiDistral, an extension of Distral to the multi-agent setting. We show that MultiDistral outperforms a Q-Learning baseline given few games played. However, we observe that learning with MultiDistral in a semi-collaborative setting results in one player's performance worsening as more iterations are run. We hypothesise from behaviour simulations that this is due to competitiveness, as the better player learns to dominate the competitive resource, resulting in the disadvantaged player having to spend many time steps without the ability to access it and thus learn. We finish by studying the differences between two versions of this framework, as well as by analysing the impact of the different tasks' characteristics on the learning process.

1 Introduction

Reinforcement Learning (RL) has provided some of the most impressive advancements in artificial intelligence in the past few years, having been applied to various domains such as game playing [Silver et al., 2016, Mnih et al., 2015] and robotics [Levine et al., 2016], as well as to real-world settings such as power systems [a20, 2017]. However, RL methods are known to suffer from very large sample complexity [Kakade, 2003]. That is, typical RL algorithms, while potent, require particularly large amounts of data. One approach to mitigating this issue is to reuse knowledge, so agents require less data to learn. Under this realm, two typical approaches are Multi-Task Learning and Transfer Learning [Zhang and Yang, 2022, Taylor and Stone, 2009]. The former involves training an agent so it can perform well across a variety of concurrent tasks, whereas the latter focuses on using the knowledge gained in past tasks to attempt a new (previously unseen) task. These approaches, often intertwined, have been extensively applied to single agent RL problems [Parisotto et al., 2015, Zhang and Yang, 2022, Zhang et al., 2023].

While in theory the sharing of knowledge between related tasks seems intuitive, it's often not straightforward achieve good results with this approach in practice [Parisotto et al., 2015, Rusu et al.,

2016]. Using naive approaches such as sharing neural network parameters between different tasks, phenomena such as negative transfer tend to occur, where learning a task can actually make the agent worst at learning a related task.

Many practical applications of RL, such as multi-player games and multi-robot control, involve the interaction between multiple agents. Furthermore, as RL grows and more agents are deployed, the interaction between different agents will need to be controlled, in the same way that society requires a balance of aspects such as cooperation, communication and competition between different humans. This setting severely complicates the problem: Environment non-stationarity is often introduced, and the typical Bellman equations that dominate RL's theory no longer hold (at least not without some changes). This makes learning a much more challenging procedure, and thus the implementation of multi-task learning has yet to achieve as convincing results as in the single-agent setting, especially in sparsely-rewarded environments [Hu et al., 2015].

In previous work, knowledge has been transferred via a policy distillation [Rusu et al., 2016]. If tasks are similar, policies are likely to contain similar structure, and thus an agent aims to learn a policy for each task, while constrained to find a policy similar to central distilled policy that "summarises" common behaviours across tasks. This is the intuition behind Distral [Teh et al., 2017], a framework for distilling and transferring policies in multi-task single agent RL.

In this paper, we extend Distral to a multi-agent RL collaborative game-playing setting. We hypothesise two different possible ways to perform this extension, and analyse differences in performance between them. We find that MultiDistral achieves better sample efficiency than a Q-Learning baseline up to a certain point in training, but that excessive training iterations (of MultiDistral) decrease collaboration between agents.

2 Literature Review

2.1 Multi-Agent RL

Multi-agent RL extends RL methods to scenarios where multiple agents must interact in an environment. These interactions generate new challenges: The environment is often non-stationary (the policies of other agents, which an agent often does not have access to, impact the "model" of the environment), rewards are often hard to assign to specific agents, and the poor sample efficiency of RL methods is exacerbated as each particular state is even rarer.

There is a plethora of algorithms suited to this setting, and thus we present only a very limited overview. A good survey of such methods is present in [Matignon et al., 2012]. Decentralised Q-Learning (reference) suggests that each agent performs Q-Learning separately, similar to what is proposed in our decentralised method in Section 3.1 (except in the latter we perform Soft Q-Learning to promote exploration). Hysteretic Q-Learning [Matignon et al., 2007] aims to prevent mis-coordination between agents, and has been shown to provide solid empirical results. Finally, [Foerster et al., 2016] combines decentralised Q-Learning with Deep Recurrent Q-Networks.

2.2 Multi-Task and Transfer Learning

A good (albeit outdated) survey of multi-task and transfer learning is presented in [Taylor and Stone, 2009]. However, most methods for this type of learning are focused on single-agent RL. Recent works borrow heavily from the idea of policy distillation [Rusu et al., 2016], where knowledge gained across tasks is distilled (i.e. "condensed") into a policy, that captures the similarities in behaviours between the different tasks. [Parisotto et al., 2015] suggested an actor-mimic approach for transfer learning, utilising policy distillation of knowledge gained across different tasks. [Rusu et al., 2016] extended the idea of neural network distillation [Hinton et al., 2015] to Deep Q-Networks (DQNs), to learn an array of "teacher" DQNs that then were distilled into a single DQN capable of multi-tasking.

2.3 Multi-Task Multi-Agent RL

There is some work on Multi-Task Learning for multi-agent systems. A thorough survey is presented in [Silva and Costa, 2019]. However, the field is not yet as developed as for the single-agent setting. Focusing on a collaborative setting, transformers have been applied to transfer coordination knowledge for new tasks [Zhou et al., 2021], a method that can be applied in parallel with other

knowledge transfer methods. Other methods focus on networks that learn how to decompose a joint reward signal into incentives for each player [Sunehag et al., 2017].

More general approaches focus on transferring knowledge via value functions, such as the work done in [Liu et al., 2019]. However, these methods are prone to unstable learning, due to issues such as different reward schemes between tasks. It is common for a task to then dominate learning, and thus the agent does not learn to perform well across a variety of tasks.

A different approach could focus on transferring knowledge via policies: Due to the similarity between tasks, an agent's policy for one task is likely to share similarities with its policy for other tasks. Policy-focused methods can bypass the challenges arising from different reward schemes between tasks, as they simply focus on transferring patterns in behaviour. An interesting approach relies on policy distillation [Rusu et al., 2016]. This approach has been applied to single-agent RL to create the Distral Framework [Teh et al., 2017], which is thoroughly outlined in Section 3.1. This framework can be integrated with other methods described previously, and furthermore supports the integration of a wide array of RL algorithms to optimise its reward function. In this paper, we aim to extend this method to a multi-agent setting. The central idea of policy distillation has also been applied specifically to the multi-agent setting for partially observable environments Omidshafiei et al. [2017].

3 Method

3.1 Distral

Distral (short for "Distill and Transfer Learning") is a framework for multi-task RL, based on the idea of a shared policy resulting from the distillation [Rusu et al., 2016] of each task's specific policy. Thus this shared policy contains the behavioural patterns that are common in different tasks. This shared policy is then used to guide task-specific policies. The framework's name hints at this process: Task-specific policies are distilled into a shared policy, and then the knowledge in this shared policy is transferred into each task's specific policy. This idea can be seen in Figure 1.

Let us present a more formal introduction to this framework. Let us assume a multitask setting with n tasks, and typical RL notation with respect to states, actions and rewards. This section follows Section 2.1 of the original paper [Teh et al., 2017] closely, and thus borrows its notation. π_i is the strategy for each task, and π_0 is the shared policy. Mathematically, this framework is equivalent to maximizing the following reward function:

$$J(\pi_0, \{\pi_i\}_{i=1}^n) = \sum_i \mathbb{E}_{\pi_i}\left[\sum_{t\geq 0} \gamma^t R_i(a_t, s_t) - c_{KL}\gamma^t \log \frac{\pi_i(a_t|s_t)}{\pi_0(a_t|s_t)} - c_{Ent}\gamma^t \log \pi_i(a_t|s_t)\right] \quad (1)$$

$$= \sum_i \mathbb{E}_{\pi_i}\left[\sum_{t\geq 0} \gamma^t R_i(a_t, s_t) + \frac{\gamma^t \alpha}{\beta} \log \pi_0(a_t|s_t) - \frac{\gamma^t}{\beta} \log \pi_i(a_t|s_t)\right] \quad (2)$$

where $\alpha = \frac{c_{KL}}{c_{KL}+c_{Ent}}$, $\beta = \frac{1}{c_{KL}+c_{Ent}}$, and $c_{KL}, c_{Ent} \geq 0$.

Focusing on the first equality, our reward function now has three terms: the typical discounted return, a term with discounted KL divergences, that is maximised if $\pi_i = \pi_0$, and an extra term that aims to maximise the entropy of each task's policy. We direct any discussion regarding the relevance of the entropy term to Sections 2.1. and 2.2. of the original Distral paper [Teh et al., 2017]. The KL-divergence term is responsible for the regularisation of policies, encouraging each π_i to be close to π_0 (transfer), but also π_0 to be the centroid of each task's policy (distillation). Note that one can tune c_{KL} and c_{Ent} to tune the relative degrees of regularisation, achieving a trade-off between shared policy regularisation and entropy regularisation.

There are multiple RL algorithms that can be used to optimise this objective. We focus on a tabular RL setting, without the use of neural networks as function approximators. In this setting, the individual policies and the shared policy can be optimised in an alternating fashion that resembles EM (here the shared policy takes the role of the parameters in the EM algorithm). With π_0 fixed, optimising

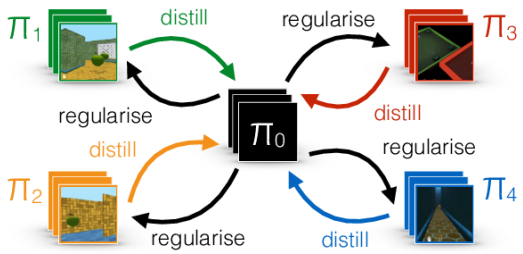

Figure 1: Illustration of the Distral framework. [Teh et al., 2017]

J can be done independently for each task, and it is equivalent to maximizing a regularised reward $R_i^{'}(a,s) = R_i(a,s) + \frac{\alpha}{\beta}\log\pi_0(a|s)$. This can be learnt using soft Q-Learning, yielding:

$$V_i(s_t) = \frac{1}{\beta}\log\sum_{a_t}\pi_0^{\alpha}(a_t|s_t)\exp\left[\beta Q_i(a_t,s_t)\right] \tag{3}$$

$$Q_i(a_t,s_t) = R_i(a_t,s_t) + \gamma\sum_{s_{t+1}} p_i(s_{t+1}|s_t,a_t)V_i(s_{t+1}) \tag{4}$$

with the optimal policy π_i being given by:

$$\pi_i(a_t|s_t) = \pi_0^{\alpha}(a_t|s_t)e^{\beta Q_i(a_t,s_t)-\beta V_i(s_t)} \tag{5}$$

In the second step of each iteration, these learnt π_i are fixed, and we aim to optimise π_0. This reduces to finding the MLE of:

$$\frac{\alpha}{\beta}\sum_i \mathbb{E}_{\pi_i}\left[\sum_{t\geq 0}\gamma^t\log\pi_0(a_t|s_t)\right] \tag{6}$$

which can be estimated by simply counting state visits under the different policies π_i.

3.2 MultiDistral

In this paper, we propose to extend Distral to a multi-agent setting in two possible ways. Both ways share a decentralised approach: agents will consider other agents simply as part of the environment, and thus this environment will be non-stationary. This severely hinders any guarantees of convergence of algorithms. Note, however, that this choice of a decentralised approach is mostly due to the scope of this project. MultiDistral could also be applied to centralised approaches, and its results studied to truly assess the potential of the use of distilled policies for multi-task multi-agent RL.

3.2.1 V1: One shared policy per agent

Firstly, we hypothesise that Distral can simply be extended by parallelizing its implementation so that each of the m agents has a shared policy across tasks. This policy encompasses the knowledge gained across different tasks for that specific agent. Each agent's shared policy is thus independent of other agents' shared policies. Under this setting (and assuming a decentralised approach), the global objective becomes:

$$J\left(\left\{\pi_0^{(j)}\right\}_{j=1}^{m},\left\{\pi_i^{(j)}\right\}_{i=1,j=1}^{n,m}\right) = \sum_i\sum_j \mathbb{E}_{\pi_i^{(j)}}\left[\sum_{t\geq 0}\gamma^t R_i^{(j)}(a_t,s_t) + \frac{\gamma^t\alpha}{\beta}\log\pi_0^{(j)}(a_t^{(j)}|s_t) - \frac{\gamma^t}{\beta}\log\pi_i^{(j)}(a_t^{(j)}|s_t)\right]$$

where we assume s_t and a_t to be the joint state and actions (across all agents). We aim to find the different π_i that maximise this reward using Soft Q-Learning, as in the original Distral. Note that the reward now depends on the actions and states of all agents, and thus if each agent will be learning using Soft Q-Learning, they will be learning in a non-stationary environment (as this algorithm will only consider the specific agent's state and action). Under this setting, both steps of our optimisation procedure reduce to the same expressions as in the original Distral (Equations 3 and 4), except we will have a different π_i and π_0 to optimise per agent. That is, each agent's objective can be maximised independently (altough with possible non-stationarity issues).

3.2.2 V2: One shared policy across all agents

Alternatively, we hypothesise that a shared policy, that is not only shared across tasks but also across agents, can yield better information transfer, especially in collaborative settings. Thus we only have one π_0. Thus, we aim to maximise:

$$J\left(\pi_0, \left\{\pi_i^{(j)}\right\}_{i=1,j=1}^{n,m}\right) = \sum_i \sum_j \mathbb{E}_{\pi_i^{(j)}}\left[\sum_{t\geq 0} \gamma^t R_i^{(j)}(a_t, s_t) + \frac{\gamma^t \alpha}{\beta} \log \pi_0(a_t^{(j)}|s_t) - \frac{\gamma^t}{\beta} \log \pi_i^{(j)}(a_t^{(j)}|s_t)\right] \tag{7}$$

We once again assume a decentralised approach, using Soft Q-Learning for the first optimisation step, which once again reduces to the expressions for Distral, as shown in Equations 3 and 4. That is, each agent's objective can be maximised independently (although with possible non-stationarity issues). For the π_0 optimisation step, we now find the MLE of:

$$\frac{\alpha}{\beta} \sum_i \sum_j \mathbb{E}_{\pi_i^{(j)}}\left[\sum_{t\geq 0} \gamma^t \log \pi_0(a_t|s_t)\right] \tag{8}$$

3.3 Grid World - MDP

We consider a two-player game in a simple grid environment, as seen in Figure 2, where two chambers are connected by a corridor [Fox et al., 2017, Teh et al., 2017]. The agents receive a reward penalty of -0.1 per step, and a single reward of 100 when they reach their respective goal state. Different initialisations of the game change the starting point of each agent, and its respective goal location. The agents learn across these different initialisations, each corresponding to a different task. Ideally, agents are expected to show signs of collaboration: whichever reaches the corridor first should keep on travelling, with the other agent letting the first agent through before attempting to cross the 1-player wide corridor. In each MDP, a single agent's state consists of a septuple (both agents' location coordinates, an agent's own previous action, the other agent's previous action, and a flag on whether the other player has reached the goal). This choice of agent state allows for a large degree of observability, thus reducing the impact of non-stationarity.

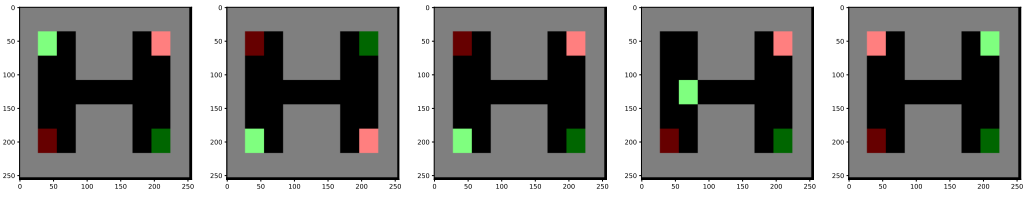

Figure 2: Different tasks that the agents learnt to solve using MultiDistral. Light green and light red squares represent Agents 1 and 2 respectively. Dark green and dark red squares represent each agent's respective goal. Note the symmetry in most tasks, as well as the asymmetry in the 4th task, where Agent 1 is closer to the goal.

3.4 MultiDistral

The E-Step of MultiDistral was implemented with Soft Q-Learning, with $\beta = 0.5$ and $\alpha = 0.7$. For each step of an iteration, 75 games were run. That is, each E-Step was approximated from 75 games, whereas each M-Step was also approximated using 75 games. Finally, this process was iterated 10 times. An informal study on the impact of the number of games per iteration on performance was performed, and we concluded a larger amount of games stabilised the learning curves.

3.5 Evaluation

We compare the return per episode of different learning algorithms for each agent. We use Q-Learning as a baseline to compare our results to. We compare the performance of Q-Learning across 5 tasks to both versions of MultiDistral implemented with Soft Q-Learning as the E-Step. We also compare the two implementations of MultiDistral based on the average return of each player. Finally, we analyse the impact on learning of changing the types of tasks the agents must solve.

4 Results

4.1 Q-Learning Baseline

To provide a baseline to compare MultiDistral to, the two agents were run each using Q-Learning on all five tasks in Figure 2. That is, agents learnt from scratch on each task. The results are shown in Figure 3. Learning was averaged across 20 independent runs to reduce noise in the learning curves.

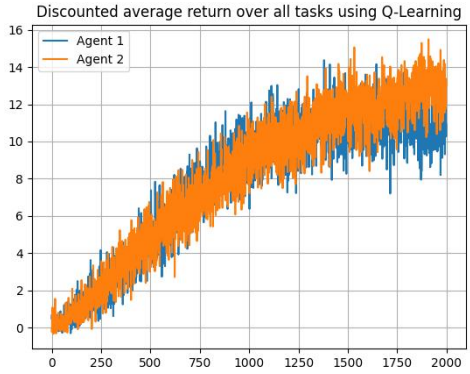

Figure 3: Average return over all tasks for both agents using Q-Learning, as a function of the number of episodes. Results averaged over 5 independent runs.

4.2 MultiDistral

The MultiDistral framework was run as specified in Section 3.4. We observed that a large number of games per iteration led to more stable learning, as opposed to more iterations with less games. This is likely due to the high variability in average returns of each of the agents. Furthermore, a larger number of games per iteration decreases the chance that sub-optimal policies get distilled and transferred.

4.2.1 Comparison between different versions

Figure 4 shows the average return (averaged over tasks) for each player, using both versions of the MultiDistral coursework (V1 on the left, V2 on the right). Note we observed that the learning curves present significant variability over different runs. However, the behaviour showcased in this report was by far the most frequently observed. In particular, it should be noted that in all runs Agent 2 outperformed Agent 1 after the first two iterations.

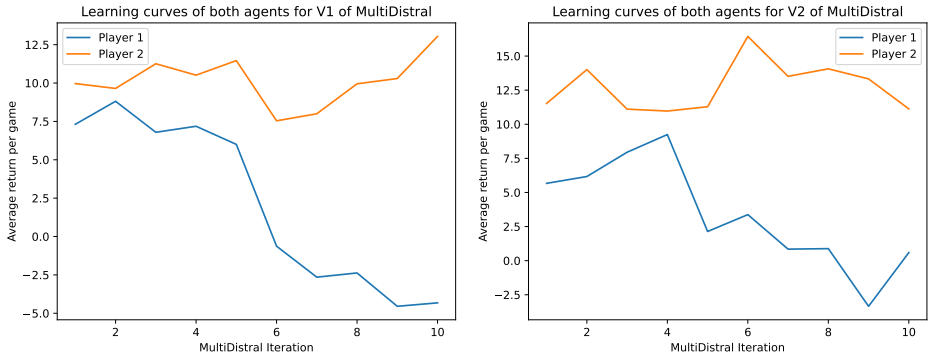

Figure 4: Average return for both agents in both versions of MultiDistral (V1 on the left, V2 on the right) for the tasks in Figure 2 (asymmetric tasks).

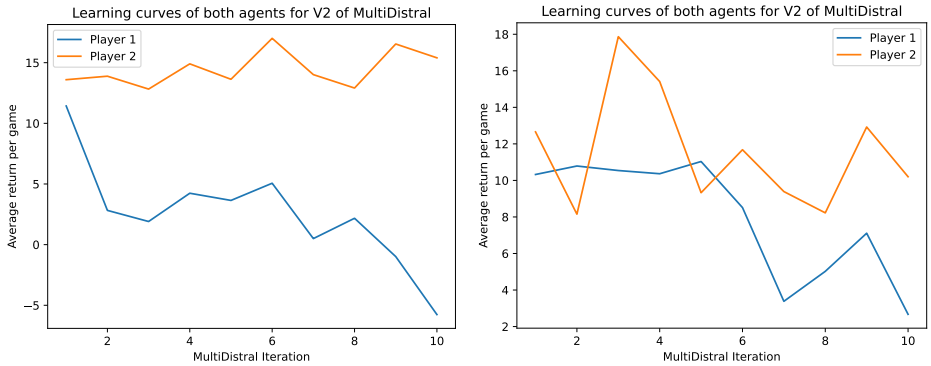

Figure 5: Average return for both agents in both versions of MultiDistral (V1 on the left, V2 on the right) for the new tasks (symmetric tasks).

4.2.2 Analysis of effect of task specification on MultiDistral performance

Due to the results presented in Figure 4 (and discussed in Section 5), the fourth task in Figure 2 was changed so it mimics Task 1, so that all tasks are symmetric (that is, the agents have equally hard goals to reach, in each of the tasks). Thus, neither agent has a particular advantage in any of the tasks. Previously this was the case for all tasks but task 4. Figure 5 shows the results of both versions of MultiDistral on the new symmetric tasks.

5 Discussion

As observed in Figure 2, the agents eventually learn using Q-Learning. However, their return, despite the large number of episodes, is still far away from the optimal return of 100 (although in reality, due to the discounting factor, the optimal return is around 91). Importantly, both agents' return stagnated after about 2000 episodes. Note that one of the agents seems to consistently (in different independent runs) outperform the other towards the later section of learning. In the case depicted in Figure 3, this implies that Agent 2 learns quicker, and once it learns to traverse the corridor, Agent 1's return is partly limited by Agent 2 usually already occupying the corridor. Analysis of sample episodes supports this hypothesis. Note however we do not observe the emergence of cooperative behaviour.

Each iteration of MultiDistral has 60 games (30 for E-step, 30 for M-step). Thus, 10 iterations correspond to 600 games per task. Note that for Q-Learning, the average return for this amount of learning is around 6 for both agents (Figure 3). Let us first focus on Figure 4. Firstly, note both players achieve better returns than Q-Learning for the first iteration (after 60 games), and one of the players consistently outperforms the Q-Learning implementation. Importantly, we see that for both versions Agent 1's performance drops as more games are played, whereas Agent 2's performance

increases or oscillates. By observing Figure 2, we can see that most tasks are symmetric, in the sense that they are equally hard for each agent. However, task 4 is an outlier, and in this task Player 1 is closer to its goal than Player 2 is. We hypothesise this difference in performance is due to the asymmetry of tasks, and thus perform the experiment described in 4.2.2. One of our hypotheses is that Player 1's asymmetric starting position in task 4 means this player struggles to learn in this task, and that poor performance ends up being distilled and transferred to other tasks. However, the differences in performance between versions would have to be more carefully assessed in a balanced setting before drawing stronger conclusions.

Regarding differences between versions, note that both players achieve larger rewards for V2, the version with one single shared policy. It is hypothesised that this shared policy does, to some extent, allow for some transfer of common knowledge between agents (note that, for example, Agent 1 in task 5 effectively solves the same task as Agent 2 does in task 1). However a more thorough investigation into agent behaviour (and in particular an analysis of the shared policies) would be required for proper conclusions to be drawn.

Finally, as described above, we changed Task 4 so all tasks would be symmetric (Figure 5). Note that for version 1, both players' performance initially improves (because their tasks are easier), but we still observe the same patterns in learning curves. However, for version 2, we see that while Player 2 still outperforms Player 1 in general, performance is much closer between players and noisier in general. It is unclear, even after observing player behaviour throughout learning, why this is the case throughout all experiments. Regardless, these results refute our earlier hypothesis that the task asymmetry was fully responsible for the differences in performance between agents. However, this difference in performance between versions (once symmetric tasks were introduced) suggests that in symmetric task settings, a single shared policy might actually lead to a phenomenon similar to negative transfer, where one agent's improvement ends up being "distilled away" by the other agent's sub-optimal policy. It would be interesting to further explore why this does not occur in asymmetric settings.

As discussed in Section 3.2, the extension of Distral to a multi-agent setting while not adapting the E-Step algorithm to a multi-agent algorithm can lead to convergence issues, due to the loss of stationarity. Thus, as we limit ourselves to single-agent Soft Q-Learning in this paper, it would be interesting to evaluate MultiDistral's potential using a more appropriate E-step, such as for example multi-agent Soft Q-Learning [Wei et al., 2019]. However, due to the need for deriving new formulas for the learned value functions and policies, this is beyond the scope of this project.

Note that the original Distral paper lacks reproducibility, and thus it is possible that some details were not accurately replicated. In particular, it is unclear whether Equation 3 in the Distral paper can be adapted to a model-free setting simply by sampling as we do in our code (instead of an expectation over model dynamics as presented in the paper). Furthermore, no details are given regarding suitable hyperparameters, and in particular those related specifically to MultiDistral (α and β). It is expected these can widely vary the performance of the algorithm, as one requires a good balance between each of the three loss function terms' strength (return, distillation related KL divergence, and the entropy term). Due to computational limitations, only very limited hyperparameter searches were performed in this project. Furthermore, the learning procedure of MultiDistral was limited to 10 iterations due to the same limitations.

6 Conclusion

We propose MultiDistral, a general framework for policy distillation and transfer for multi-task multi-agent reinforcement learning. We showed that both versions of MultiDistral outperform a Q-Learning baseline on an equal number of games. We showed, however, that learning with MultiDistral in a semi-collaborative setting has lead to one of the player's performance worsening as more iterations are run. We hypothesise from simulations that this is due to competitiveness, as the better player learns to overuse the shared resource, whereas the worse player has insufficient access to properly learn. Small differences in performance can thus grow larger, as one agent becomes better and thus worsens the other's chances to reach the goal.

Possible future work should focus on implementing a more theoretically sound E-Step algorithm. Furthermore, a thorough investigation into agent behaviour could be of interest, so as to better assess MultiDistral's potential for multi-task learning in more realistic multi-agent settings.

A Code

The source code for this project is available in the repository at `https://github.com/maxjappert/multi-agent_distral`.

B Contributions

This section (not counted in page limit) describes the contributions of each member of the team to the project (both to this paper but also to other related tasks):

- Samuel Oliveira
  - Initial research, project conception, and writing of the project proposal.
  - Derived the formulas needed to adapt to a multi-agent setting, and to implement the M-step of MultiDistral.
  - Drafted a presentation outline with its structure and content. Created the presentation slides. Presented the project to lecturer, TAs and peers.
  - Built parts of the grid environment, on top of work done by Max Jappert.
  - Created multiple baseline algorithms, including Q-Learning, Soft Q-Learning, and both of these algorithms also with rollout.
  - Created the code for MultiDistral (both versions), as well as for the E-Step algorithm (Soft Q-Learning adapted to MultiDistral's shared policies).
  - Wrote code to record gifs of episode simulations so as to better understand agent behaviour, as well as value function and policy plots.
  - Repository clean-up.
  - Created visualizations included in this paper.
  - Wrote this research paper.
- Max Jappert
  - Wrote and presented part of the presentation.
  - Implemented the environment and its visualisation.
  - Debugged and fixed numerical issues occurring in the baseline algorithms.
  - Created the plots of the baseline algorithm performance used in this report.
  - Implemented Monte Carlo tree search as a sanity check for solving the environment in a top-down coordinated approach.
  - Assisted Samuel Oliveira in conceptualising the project and its execution.
  - Wrote the README file and cleaned up the repository.
  - Wrote part of the report.
- Vishrut Malik
  - Attempted to implement MADDPG. Code doesn't run.
  - Pasted the output of the `tree` command of the repository file structure into the `README.md` file.

References

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search, 01 2016.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning, 02 2015.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17:1–40, 2016.

Reinforcement learning for electric power system decision and control: Past considerations and perspectives. *IFAC-PapersOnLine*, 50:6918–6927, 07 2017.

Sham Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, 03 2003.

Yu Zhang and Qiang Yang. A survey on multi-task learning, 2022.

Matthew Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

Emilio Parisotto, Jimmy Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *ICLR*, 11 2015.

Grace Zhang, Ayush Jain, Injune Hwang, Shao-Hua Sun, and Joseph Lim. Efficient multi-task reinforcement learning via selective behavior sharing, 2023.

Andrei Rusu, Sergio Colmenarejo, Ç Aglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation, 2016.

Yujing Hu, Yang Gao, and Bo An. Learning in multi-agent systems with sparse interactions by knowledge transfer and game abstraction, 2015.

Yee Teh, Victor Bapst, Wojciech Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning, 2017.

Laëtitia Matignon, Guillaume Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *Knowledge Engineering Review*, 27:1–31, 2012.

Laëtitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Hysteretic q-learning : an algorithm for decentralized reinforcement learning in cooperative multi-agent teams., 10 2007.

Jakob N Foerster, Yannis M Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate to solve riddles with deep distributed recurrent q-networks, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Felipe Leno Da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 03 2019.

Tianze Zhou, Fubiao Zhang, Kun Shao, Kai Li, Wenhan Huang, Jun Luo, Weixun Wang, Yaodong Yang, Hangyu Mao, Bin Wang, Dong Li, Wulong Liu, and Jianye Hao. Cooperative multi-agent transfer learning with level-adaptive credit assignment, 2021.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning. 06 2017.

Yong Liu, Yujing Hu, Yang Gao, Yingfeng Chen, and Changjie Fan. Value function transfer for deep multi-agent reinforcement learning based on n -step returns, 2019.

Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability, 2017.

Roy Fox, Michal Moshkovitz, and Naftali Tishby. Principled option learning in markov decision processes, 03 2017.

Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. Multiagent soft q-learning, 2019.