Policy Progress Score for Automatic Task Selection in Curriculum Learning

Golden Rockefeller
Oregon State University
rockefeg@oregonstate.edu

Scott Chow
Oregon State University
chows@oregonstate.edu

Yathartha Tuladhar
Oregon State University
tuladhay@oregonstate.edu

Kagan Tumer
Oregon State University
kagan.tumer@oregonstate.edu

ABSTRACT

This paper introduces policy progress scores as a principled means of selecting intermediate tasks that enable effective transfer learning. For some complex tasks, rewards can be sparse, delayed, noisy and/or uninformative, making it difficult to find good policies. For such tasks, a sequence of related tasks, i.e. curriculum, can be designed to enable and speed up learning by transferring knowledge between successive tasks in the curriculum. However, how to develop an effective curriculum, particularly without in depth domain knowledge, is a challenging problem.

In this paper, we present an automatic curriculum generation process that scores tasks on the anticipated change in policy parameters. This "policy progress" score is defined as the weighted sum of absolute expected change in a policy parameters. We apply the automated curriculum generation process using progress score to both single agent and multiagent Grid World domains. Our results on a complex multiagent Grid World domain show policy progress scores create similar curricula as the current state-of-the-art with reduced computation.

Keywords

Curriculum Learning, Learning Progress, Policy Progress, Transfer Learning, Task Selection

1. INTRODUCTION

Reinforcement learning has been used to train agents for a variety of applications including helicopter control [12], chess [8], and robotic arm control [5]. However, reinforcement learning is difficult for complex domains where rewards are sparse, rare, and/or have high variance [1, 6, 14]. These factors, often present in real-world problems, cause the reward to be non-informative.

To learn on complex domains, transfer learning methods have been proposed that allow agents trained on related tasks to transfer knowledge to a target task in order to improve the quality of learning on the target task [16]. This idea has been extended by Curriculum learning (CL) which transfers knowledge successively between a series of tasks, i.e. a curriculum, to speed up learning and/or increase converged performance for a target task [10]. Automatic curriculum generation methods aim to generate a curriculum online that is adaptive to the agent's current knowledge.

A key problem of automatic curriculum generation is task selection. For an automatic curriculum generation process (ACGP), task selection is the problem of choosing which intermediate tasks to train on in order to maximize the anticipated change in performance [9]. This anticipated change in performance is defined as the *performance progress*. In practice, performance progress is often estimated via an iterative update rule because it cannot be computed directly. However, the high variance and sparsity of rewards in these tasks makes it difficult to estimate performance progress. When performance progress estimation is inaccurate, ACGP is less effective at selecting good tasks for training.

Additionally, ACGP with performance progress scores is inefficient, since computing the performance progress score requires the program to run one training episode on the intermediate task and one evaluation episode on the target task. Performance progress' susceptibility to high variance and requirement of two episodes to compute makes it less feasible for real world problems [9].

The key contribution of this paper is to use policy progress as a proxy for performance progress to choose tasks for curriculum learning. Policy progress chooses intermediate tasks that maximize cumulative change in policy parameters. The benefits of policy progress is two-fold. First, the variance in policy progress comes from the intermediate task, not the main task. So even if the main task's reward has variance and sparse rewards, policy progress can be accurately computed as long as intermediate tasks have informative rewards. Secondly, policy progress can be computed with only a single training episode on the intermediate tasks, making it more computationally efficient than performance progress.

We organize the paper as follows. First we give background on transfer and curriculum learning as well as performance progress score in Section 2. Then we present a method for calculating policy progress in Section 3. Next, we describe the Grid World domain and the tasks used to test our approach in Sections 4 and 5. Finally, we present results and discuss their implications in Sections 6 and 7.

2. BACKGROUND

In this section, we describe transfer learning in the context of curriculum learning, automatic curriculum generation, and the performance progress score.

2.1 Transfer Learning

Transfer learning (TL) is the process of transferring knowledge from a source task to a separate target task to improve the quality of learning on the target task [16]. A parameterized labeling function is first trained on a source task. Knowledge is transfered when the labeling function after training on the source task is adapted for training on the target task. In context of model-free reinforcement learning, the agent's policy and/or evaluating Q function are transfered labeling functions.

2.2 Curriculum Learning

Curriculum Learning (CL) is a more specific form of Transfer Learning that aims to improve the quality of learning on a target task by creating a sequence of tasks (i.e. curriculum) on which the agent trains. Knowledge in the form of learned parameters is transfered between tasks. The application of CL involves progressively increasing the difficulty of the intermediate tasks, incrementally building scaffolding for an agent to accelerate the agent's training on the target task [10]. CL can be used to speed up learning on the target task by leveraging the information learned through these intermediate tasks.

2.3 Automatic Curriculum Generation

The automatic curriculum generation process creates a curriculum on-line that is adaptive to the agent's current knowledge through a principled selection of the task an agent should learn on for the next episode. A similar idea is present in active learning [2, 3] and self-paced learning [7] but sample data is selected for training instead of tasks. The automated curriculum generation process tackles the problem of selecting intermediate tasks in a way that maximizes a user-defined progress score. This problem is formulated as a *curriculum* Markov Decision Process (MDP), an extension to the MDP formulation [11]. The learning agent's policy space defines the MDP's state space. The task selection space defines the action space. The transition function describes how the policy might change given a task on which to train. The reward function is chosen so that the entire processes minimizes the time needed to reach a policy with an acceptable evaluation on the target task.

Practically, the anticipated progress for an intermediate task must be estimated from previous sampled measures of progress. In active and self-paced learning, progress can be defined as a reduction in the labeling function's variance [3], or reduction in a loss function [15, 7] which are both inexpensive to calculate relative to training time. However, with reinforcement learning, such definitions of progress may not apply, and other progress definitions like the increase in performance is costly to calculate exactly. Thus, an automated curriculum generation may maintain a set of progress estimates $\mathcal{P}_* = (P_{*,0}, ..., P_{*,m})$ for a finite set of m curriculum tasks $\mathfrak{T} = \{\mathcal{T}_0, ..., \mathcal{T}_m\}$ [9]. The progress estimates are kept for all tasks and updated iteratively as the agent learns. Other automated curriculum generation processes exist such as recursively breaking down a task into easy to learn subtasks and fully training on each task before moving on to harder tasks [11].

Maximizing cumulative progress requires a selection strategy that manages the exploration-exploitation trade-off. The algorithm must choose between the task estimated to have the highest progress score, and some other task to update the estimate of the selected task's actual progress score. The task that maximizes anticipated progress for the next step may change as there becomes less for an agent to learn from previously selected tasks. Thus, automatic curriculum generation is modeled as a non-stationary multi-armed bandit problem [4] where the tasks represents the arms and progress scores represent the payoff.

A formulation of basic on-line automatic curriculum generation with progress scores is presented in Algorithm 1. First, the algorithm calls a $SELECT$ function which describes a selection strategy for choosing intermediate task to train on for the next episode. ϵ-greedy is an example selection strategy that probabilistically selects either the task with the highest progress score or randomly chooses a task with some user-prescribed probability ϵ. After training, an *EVALUATE_PROGRESS* function is called to update progress scores for the selected task. Our contribution is a new method to compute these progress scores.

Algorithm 1 Automated Curriculum Generation Algorithm

```
Initialize P_*
while Training not completed do
  Task Index i ← SELECT(𝔗, P_*, *)
  π ← TRAIN(T_i, π, *)
  P_{*,i} ← EVALUATE_PROGRESS(*)
end while
```

2.4 Performance Progress

Performance progress scores have been introduced in previous works [2, 13] as a method to select tasks for a curriculum. Performance progress is defined by how much the expectation of the agent's performance changes after learning for one episode on a given task, that is:

$$P_R(\mathcal{T}) := \mathbb{E}[R' - R|\mathcal{T}, \pi] \tag{1}$$

where $P_R(\mathcal{T})$ is the anticipated performance progress for training on some intermediate task $\mathcal{T}$, R' is the performance measure on the target task to be received after the current training episode, R is the performance measure on the target task received before the current training episode, and π is the agent's policy before the start of the current training episode.

Practically, performance progress is approximated since computing the expectation of the agent's change in performance is not always viable. One estimation of performance progress is to iteratively update an estimate at the end of each learning episode using performance measures [9]

$$\hat{P}_R(\mathcal{T}) \leftarrow (1-\alpha)\hat{P}_R(\mathcal{T}) + \alpha(R' - R) \tag{2}$$

where α is the learning rate ($0 < \alpha \leq 1$). Ideally, the performance progress estimate converges to the actual performance progress.

There are a couple of drawbacks to using performance progress scores. First, it is susceptible to the high variance of the target task's reward. Additionally, updating the estimate of performance progress with Equation (2) requires

two episodes: one episode for training on the intermediate task and another episode for evaluation on the target task. In our paper, we propose policy progress scores as an alternative to performance progress scores in order to address these issues.

3. POLICY PROGRESS

High variance rewards can make performance progress estimates for automatic curriculum generation inaccurate and/or inefficient to calculate. Multiple episodes are required for each update of the performance progress estimate which takes time that could be used training directly on the target task. Thus, the automatic curriculum generation algorithm could counter-productively slow learning when the process of estimating performance progress increases computation costs beyond the learning speed up the curriculum can provide. Training can occur during an evaluation episode to make the automatic curriculum generation algorithm more efficient. However, this training introduces its own progress, making the performance progress estimate for just the evaluated intermediate task less accurate.

Policy progress scores are introduced as a principled means of selecting intermediate tasks to learn on so that transfer learning is sped up for a target task. A curriculum can be generated by choosing tasks in a particular sequence to maximize cumulative change in policy parameters, i.e. policy progress, as a progress score instead of performance progress. Ideally, policy progress scores should be used for task selection when policy progress is strongly correlated with performance progress and the former have less variance. It is reasonable to assume this correlation with tasks similar enough to the target task; learning algorithms manipulate the parameters in order to increase performance. Maximizing policy progress can address aforementioned difficulties with maximizing performance progress when there is adequately less uncertainty in the measure of policy progress compared to performance progress. Additionally, policy progress is a more efficient option as it can be measured without evaluation on the target task unlike performance progress.

The goal of learning algorithms is to make iterative change to an agent's policy to maximize performance; on average, each change should increase an agent's performance in the task being trained. More specifically, learning algorithms manipulate a parameter vector θ. Choosing an intermediate task with more change in policy parameters over time as the next part of the curriculum can lead to a faster rate of performance increase. Policy progress is thus defined as the weighted sum of absolute expected change in a policy's parameters. Policy progress must use a weighting factor for each parameter to take into account the relative variation in the impact and scaling of the parameter:

$$P_\pi(\mathcal{T}) := \sum_k^N w_k \Big| \mathbb{E}[(\theta'_k - \theta_k)|\mathcal{T}, \pi] \Big| \tag{3}$$

where $P_\pi(\mathcal{T})$ is anticipated policy progress for training on task $\mathcal{T}$, N is the number of parameters that determines the agent's policy π, w_k is the importance weighting for a given parameter, θ_k and θ'_k are the values for a given parameter before and after the current training episode, respectively.

An estimate of the expected change $\hat{\mathcal{D}}_k(\mathcal{T})$ is iteratively updated over time for each parameter θ_k :

$$\hat{\mathcal{D}}_k(\mathcal{T}) \leftarrow (1-\alpha)\hat{\mathcal{D}}_k(\mathcal{T}) + \alpha(\theta'_k - \theta_k) \tag{4}$$

where α is the learning rate. Like with the performance progress update (Equation (2)), the policy progress update is performed once per training episode. The policy progress estimate use $\hat{\mathcal{D}}_k(\mathcal{T})$ as the estimation of expected change in parameter term $\mathbb{E}[(\theta'_k - \theta_k)|\mathcal{T}, \pi]$ presented Equation (3):

$$\hat{P}_\pi(\mathcal{T}) = \sum_k^N w_k |\hat{\mathcal{D}}_k(\mathcal{T})| \tag{5}$$

Policy progress measures how much a chosen intermediate task pushes the parameters along regardless of whether doing so actually increases performance. Therefore, candidate tasks for selection should be carefully constructed such that, given the automatic curriculum generation algorithm, the average change for parameters over time will lead to an improvement in performance in the target task. Using policy progress for evaluating tasks is not recommended unless intermediate tasks often have positive performance progress.

Promising initializations [10] are potentially good candidates for intermediate tasks that place agents near goal areas to reduce the sparsity of the reward. Deviation to the optimal policy for the target task when training the optimal policy on an intermediate task is to be avoided as these deviations are counter-factually measured as positive policy progress according to formulation of policy progress. Promising initializations intermediate tasks avoid such deviations from the optimal policy for the target task that policy is also optimal for the promising initializations tasks.

4. GRID WORLD DOMAIN

The Grid World domain consists of an n by m grid with an agent in the grid world trying to reach a goal cell at some row g_r and some column g_c, with a reward of r_g rewarded to the agent for reaching the grid world and a timing reward (or cost) r_t rewarded to the agent for each time step the agent is in the grid world. The performance of the agent is the sum of all the rewards during the run. Each episode ends either after T time steps has passed or when the agent has reached the goal. Upon initialization, the agent is randomly placed in the grid world on any cell except the goal cell. Each step, the agent may move in the cardinal directions or stay in place, but they can not leave the boundaries of the grid world.

4.1 Multiagent Grid World

The multiagent extension to the Grid World domain makes for a good example of a complex domain with sparse rewards. In the multiagent case, we initialize N decentralized agents in the same worlds as described for the single agent Grid World Domain. Agents' actions are the same as in the single agent domain and multiple agents can occupy the same cell. A coupling requirement exist such that the goal reward is only received when a minimum number of agents are at the goal cell at the same time. A coupling requirement value c is the number of agents required to the receive the goal reward. This version of Grid World is difficult as agents do not know where are agents are and yet, they must coordinate the actions in order for the system to perform well. The episode ends either after T time steps have passed or when the coupling requirement is achieved.

Individualized rewards for agents may be shaped by the learning process for more effective training. The multiagent system's performance, or global reward, does not isolate

their contribution to the system, negatively impacting how well they learn due to poor credit assignment. Agents may instead receive the difference reward [17] which estimates the contribution of the agent as the difference in performance between the multiagent system current performance and the system's performance should that agent not exist or reach some counterfactual state-action. However, the multiagent system's performance as opposed to individual agent rewards must still be used in calculating progress.

The equation for the difference reward is

$$D_i(z) = G(z) - G(z_{-i}) \tag{6}$$

where D_i is the the difference reward for an evaluated agent i given the full environment state z (not the agent state), G is the global reward and z_{-i} the counterfactual full environment state, replacing the state-action of i with a counterfactual state-action.

As the decentralized multiagent system implies multiple policies, the policy progress score is calculated for each agent i as $P_{\pi,i}$ and the progress score P_{Π} is used for task selection as the sum of each agent's progress score:

$$\hat{P}_{\Pi} := \sum_{i}^{N} \hat{P}_{\pi,i} \tag{7}$$

5. EXPERIMENT SETUP

This paper compares averaged performance trajectories on the target task between using difference curriculum strategies. The curriculum strategies compared are 1) no curriculum, training on the target task, 2) no curriculum, training on the same non-target task for all learning episodes; two non-target tasks are compared 3) ACGP with random task selection 4) ACGP with an ϵ-greedy task selection process with performance progress scores and 5) ACGP with an ϵ-greedy task selection process with policy progress scores. All parameter weights are set to 1, when calculating policy progress scores. The ACGP selects from a set of five tasks, the target task and four intermediate tasks. The tasks for both the multiagent and single agent domains are defined by the goal cell and initialization cells locations illustrated by Figure 1 with the target task illustrated by (a) and the intermediate tasks illustrated (b-e).

5.1 Target Task

The target task defines a Grid World domain with the dimensions $n = m = 10$. The goal is located at $g_r = 5, g_c = 5$. The starting location of the agent is randomly determined with an equal probability for every cell except the goal cell. There is no timing reward $r_t = 0$ and the goal reward is set to $r_g = 1$. Each episode lasts $T = 10$ steps. In the single agent case, learning happens over 2000 episodes. In the multiagent case, the coupling requirement is $c = 3$ with $N = 6$ agents and learning happens over 20000 episodes.

5.2 Intermediate Tasks

Four intermediate tasks are generated from the target task by restricting where agents can be initialized. The four tasks are described by initialization rings that progressively get closer to the goal cell. These tasks are provided as intermediate tasks in the spirit of Promising Initializations [10]. The nearest Figure 1(c) and farthest Figure 1(b) rings are used as sample non-target tasks for the no-curriculum, non-target curriculum strategy. In the multiagent case, the same initialization restriction applies to all agents, i.e. agents are not initialized in different rings.

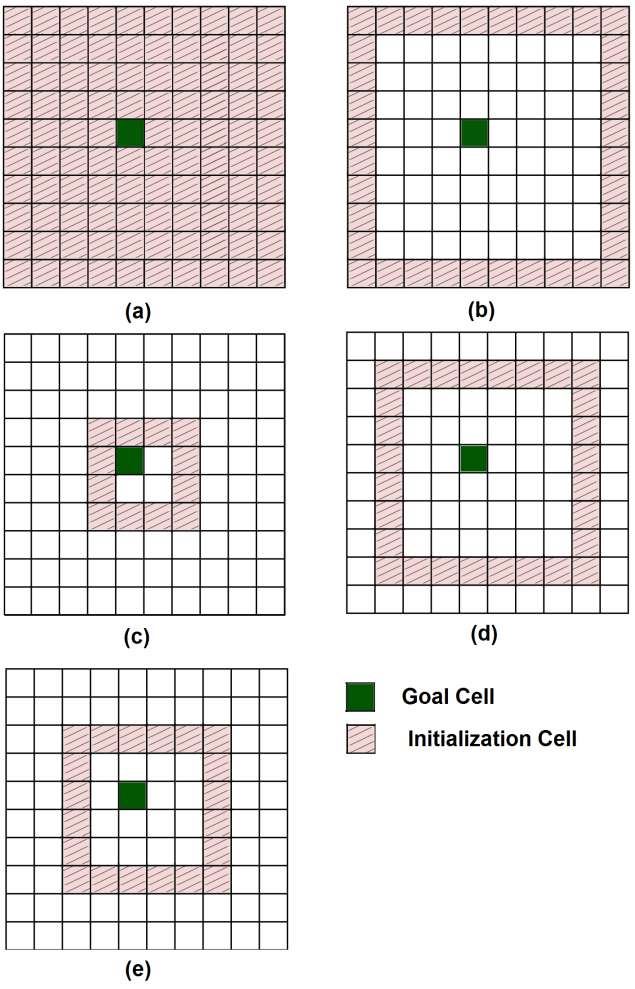


Figure 1: Illustration of five different Grid World tasks (a-e). At the start of a given task, agent(s) are initiated randomly with uniform probability in any of initialization cells (striped cells). Agent(s) must then navigate to the goal cell (dark cell) to receive a high reward and end the episode. As an example, the task illustrated by (c) initializes agents very close to the goal making the probability of the agents reaching the goal high compared to a task that initializes agents farther away like with the task illustrated by (b).

5.3 Learning Policies

Agents' policies are Q tables mapping state-action pairs to a reward for both the single and multiagent domains). The policies are update each time step using the tabular Q-learning algorithm:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma max_{a'}Q(s',a')) \tag{8}$$

for state s, next state s', action a, next action a' and reward r. The learning rate is set to $\alpha = 0.1$. The discount factor is set to $\gamma = 0.9$. The Q table is optimistically initialized to 1.0 to encourage exploration with small tie-breaking noise from a uniform distribution between 0 and 0.1. Agents chose the

action that yield the maximum Q value given a state but when learning, they follow ϵ-greedy strategy with $\epsilon = 0.1$.

In the multiagent agent case, evaluations for the compared curriculum strategies were generated both for when agents received the global reward, and when agents received the difference reward. In the target domain, evaluating the difference reward, agents receive a rewards of $r = 1$ when they reach the goal and the number of agents that reach the goal at the same time is exactly equal to the coupling requirement; otherwise, agents receive a rewards of $r = 0$.

5.4 ACGP Task Selection Strategy

The evaluated ACGP strategies apply ϵ-greedy with $\epsilon = 0.1$. The initial performance progress estimate is optimistically set to $\hat{P}_R(\mathcal{T}) = 0.1$ for all tasks to encourage exploration of which task to choose. The initial change in parameter estimate is optimistically set to $\hat{D}_k(\mathcal{T}) = 0.1$ for all parameters and for all tasks also to encourage exploration. All the data presented were averaged over a hundred statistical runs. Mean and confidence intervals are presented.

6. RESULTS

The learning curves for evaluated curriculum strategies for the single agent Grid World domain are presented in Figure 2. Training on only the nearest ring initialization task creates the fastest learning curve, converging before 500 training episodes. Training on only the farthest ring initialization task leads to the slowest learning curve. Training with ACGP using performance progress scores leads to the second fastest learning curve. All other curriculum strategies have similar learning curves. All curriculum strategies converge to a performance of 1.0 before episode 1500.

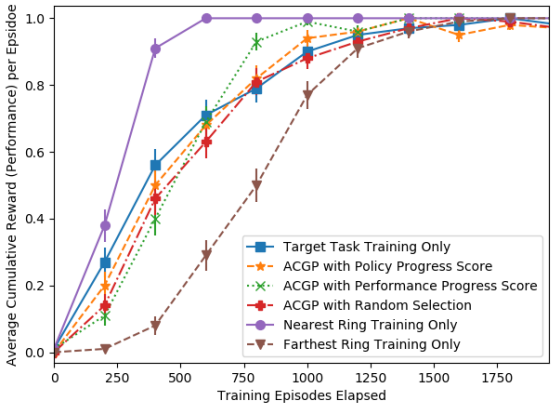


Figure 2: Average learning curves for different curriculum strategies in 10 × 10 **single agent** Grid World Domain. For a simple single agent domain, training using automatic curriculum generation processes (ACGPs) yields no benefits over training directly on the target task.

The learning curves for evaluated curriculum strategies for the multiagent Grid World domain with agents training on the difference reward are presented in Figure 3a. Training on only the farthest ring initialization task creates the fastest learning curve converging to a performance of 1.0 within 15000 training episodes. Training between only the nearest ring initialization task and only the target task leads to the slowest learning curve, neither converging within the alloted 20000 training episodes. All other curriculum strategies share similar learning curves, converging to a performance of 1.0 within 15000 training episodes.

All curriculum strategies do not converge with the alloted 20,000 training episodes. The learning curves for evaluated curriculum strategies for the multiagent Grid World domain with agents training on the global reward are presented in Figure 3b. Training between only the nearest ring initialization task and only the target task leads to the slowest learning curve. All other curriculum strategies have similar learning curves before 12500 training episodes, reaching a performance near 1.0 that decays afterwards with the learning curve of the ACGP using performance progress scores decaying faster at this point.

6.1 Task Selection Rate

Average task selection rate for the automated curriculum generation process using policy progress scores in the multiagent Grid World domain with agents training on the difference rewards are presented in Figure 4a. A strong preference for selecting nearer initialization tasks can be seen after 7500 training episodes. After this point, the strongest preferences for task selection is the nearest ring initialization task peaking at about 50 percent selection rate between 12500 and 15000 training episodes.

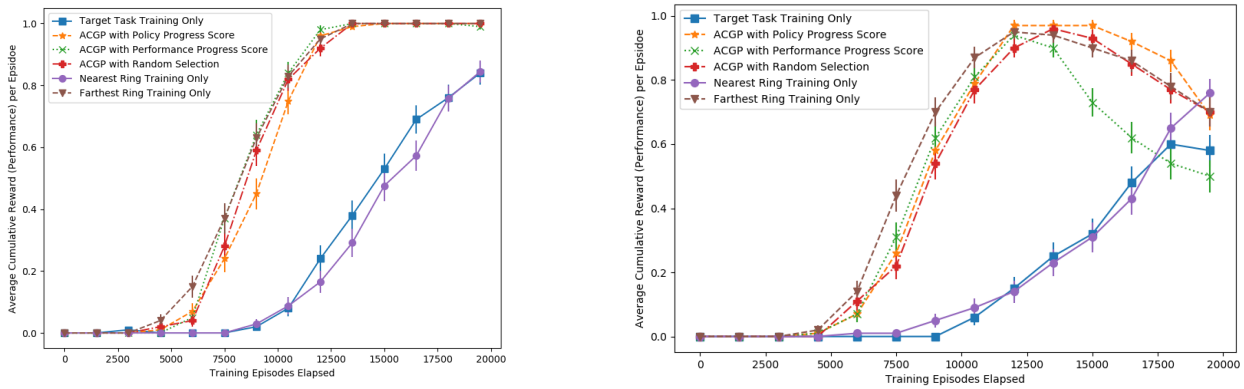
We compare this to the average task selection rate for the automated curriculum generation process using performance progress scores in the multiagent Grid World domain with agents training on the difference rewards are presented in Figure 4b. We, again, see a strong preference for selecting nearer initializations can be seen after 7500 training episodes. After this point, the strongest preference for task selection is also the nearest ring initialization task, which peaks at about 60 percent selection rate at 12500 episodes.

We see in Figure 4 that creating curricula using our policy progress and performance progress gives similar curricula for the multiagent Grid World task. Additionally, in all domains tested, the ACGP performs similarly when using policy progress scores as it performs when using performance progress scores (Figures 2 and 3). Also, recall that performance progress requires an extra episode of training on the target task to compute. Thus policy progress approximates performance progress with less computation.

7. CONCLUSION

Learning policies for tasks using reinforcement learning is difficult for domains where rewards are sparse, rare, and/or have high variance. For such domains, learning strategies may apply a curriculum to speed up learning on the target task. Automatic curriculum generation methods aim to generate a curriculum on-line that is adaptive to the agent's current knowledge. However, task selection is a key problem to address with automatic curriculum generation.

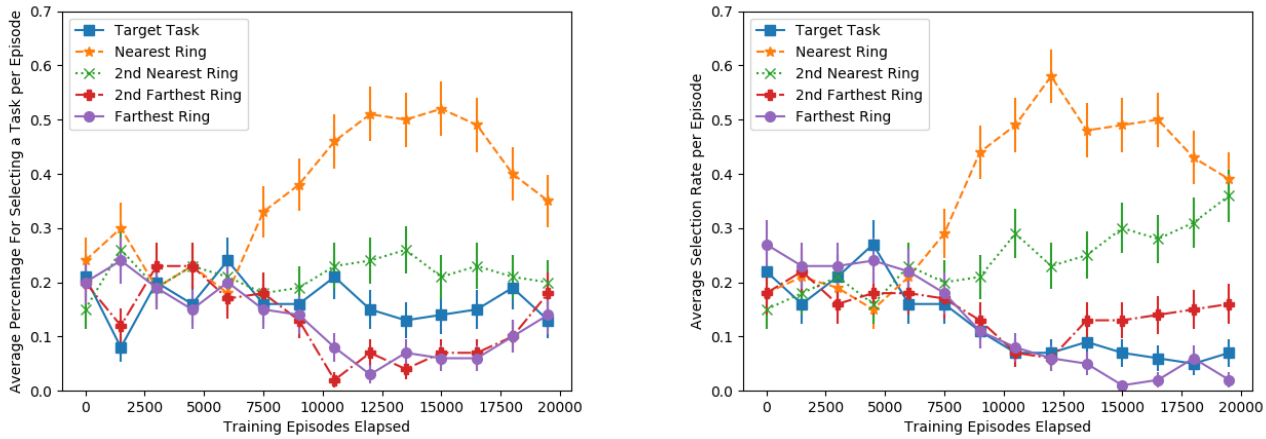
This paper uses policy progress scores as a principled means of selecting intermediate tasks to learn on so that curriculum learning is sped up for a target task. The variance of the policy progress score is dependent on the task updating the policy, the intermediate task, and not the target task. Thus, a good selection of the intermediate tasks will minimize policy progress variance, allowing for a clearer comparison between the tasks.



(a) Agents are trained on difference rewards.

(b) Agents are trained on global rewards.

Figure 3: Average learning curves for different curriculum strategies in 10 × 10 multiagent Grid World Domain with 6 agents and a coupling requirement of 3 agents simultaneously at the goal cell. The automatic curriculum generation process (ACGP) using policy progress scores performs on par with the ACGP that uses performance progress. Training agents on the global reward causes curriculum learning to fail due to the noise in the signal.



(a) Policy Progress: Average rate of selection for each intermediate task given the automatic curricula generated using policy progress scores.

(b) Performance Progress: Average rate of selection for each intermediate task given the automatic curricula generated using performance progress scores.

Figure 4: Average percentage for selecting a task using the automated curriculum generation process (ACGP) with either of the progress scores. For example, at about the 12500th training episode, ACGP using policy progress score selected the Nearest Ring task near 50 percent of the time; the ACGP using performance progress score selected the Nearest Ring task about 60 percent of the time. These results are evaluated in 10 × 10 **multiagent** Grid World Domain with 6 agents and a coupling requirement of 3 agents simultaneous at the goal cell. Our approach (policy progress) select tasks with the similar frequency as performance progress.

We compare ACGPs using performance progress versus policy progress on single agent and multiagent Grid World domains. For the single agent Grid World Domain, using either performance or policy progress scores does not alter the rate of learning over training directly. However, with the more complex multiagent Grid World domain, ACGPs using either progress score achieves a faster learning rate than learning on the target task alone. In general, the ACGPs using our policy progress approach performed comparably to ACGPs using performance progress scores; however, performance progress required an additional evaluation episode on the target task for estimating the change in performance. Thus our approach is more computational efficient. Future work will extend the evaluation on the merits of each progress score on additional domain, and address the disadvantages of each progress score.

REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1. ACM, 2004.

[2] A. Baranes and P.-Y. Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, 2013.

[3] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[4] M. Jaderberg, V. Mnih, W. M. Czarnecki, T. Schaul, J. Z. Leibo, D. Silver, and K. Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[5] P. Kormushev, S. Calinon, and D. G. Caldwell. Robot motor skill coordination with em-based reinforcement learning. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 3232–3237. IEEE, 2010.

[6] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[7] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*, pages 1189–1197, 2010.

[8] M. Lai. Giraffe: Using deep reinforcement learning to play chess. *arXiv preprint arXiv:1509.01549*, 2015.

[9] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman. Teacher-Student Curriculum Learning. *arXiv preprint arXiv:1707.00183*, 2017.

[10] S. Narvekar, J. Sinapov, M. Leonetti, and P. Stone. Source task creation for curriculum learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 566–574. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[11] S. Narvekar, J. Sinapov, and P. Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 147, page 149, 2017.

[12] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang. Autonomous inverted helicopter flight via reinforcement learning. In *Experimental Robotics IX*, pages 363–372. Springer, 2006.

[13] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[14] J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pages 463–471, 1998.

[15] B. Settles and M. Craven. An analysis of active learning strategies for sequence labeling tasks. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1070–1079. Association for Computational Linguistics, 2008.

[16] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[17] K. Tumer and A. Agogino. Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 591–598. ACM, 2005.