CS234 Final Project Report: Robot Navigation in Crowds via Meta-learning

Bilan Jackie Yang [1] **Michelle Mengxuan Zhang** [1]

Abstract

In this work, we propose a robot navigation policy network that is compatible with the meta-learning framework. The robot is expected to efficiently navigate in a pedestrian-rich environment with randomly assigned goals. It needs to adapt to different pedestrian speeds and traveling directions while minimizing collision rates. The proposed policy network enables the robot to be aware of the crowd with attention-based human-behavior-capture mechanism. The meta-learning framework tackles the uncertainties of the environment, and maintains the awareness of unobservable human control policies implicitly. Six models with different settings are trained and analyzed with the effort to find the policy network with the best setting. As a result, the robot has a higher efficiency in navigating towards the assigned goal while avoiding the pedestrians with the proposed policy network as compared to an implementation of the meta-learning method for general reinforcement learning applications.

1. Introduction

With increasing interests and expanding needs in autonomous mobility, a moving-robot is envisioned to navigate safely and efficiently to reach its goal. However, part of the environment may be unobservable to the robot, the world may change rapidly, or environmental agents may interact with the robot. All of those factors could affect the quality of robot navigation, which in turn leads to task failures. One of the most complex scenarios is when the environment contains human-agents. Humans may not only follow different policies, but also sometimes be irrational when making decisions or have an inconsistent set of preferences (Ariely, 2008). In the context of human-robot interactions (HRI), the robot communicates and reasons about human maneuvers in order to facilitate its own decision making (Goodrich & Schultz, 2008).

Traditional approaches utilize rule-based methods to model human behaviors. In particular, we encode human-domain knowledge to help robot build an initial understanding of the world, or (implicitly) extract features to construct robot control models. The rule-based approach assumes perfect modeling of the world, with its performance relying greatly on the relevance of extracted features and the human-understanding of the world dynamics (Elbanhawi & Simic, 2014). However, as the environment becomes complex, especially when robot needs to interact with obstacles in the environment, it is usually hard to generate a rule to encode or find proper weights to model the dynamics of the environment.

Model-free approaches become popular with the help of growing machine intelligence. For example, in the reinforcement learning field, robots are trained by exploring the world according to itinerary, which may end up with a superior performance as compared to what can be achieved by humans. Recent popular approaches for robot motion planning involve value-based learning (Yan et al., 2018) (Chen et al., 2017) (Chen et al., 2018), and policy-gradient based learning (Peters & Schaal, 2006). However, enabling robot exploration of the world requires a large number of trials in the training process, especially in reinforcement learning based robot motion planning problem. Further, the resultant network is restricted to be applied to a single task that is identical to the training environment.

In this project, we investigate robot navigation control policies that can be applied to complex and rapidly changing environments. In particular, we focus on pedestrian-rich environments, where robot moves towards a goal position, while avoiding collision with human pedestrians. This involves efficient navigation to the goal while capturing human-moving behaviors to evaluate humans' hidden intents. Also, robot should quickly adapt its pre-trained control policy to the new environment. The resultant policy is assumed to be responsive to changes of the environment. We have investigated the performance of robot navigation with uncertainties from goal positions and moving velocities and direction of surrounding pedestrians.

[1] School of Engineering, Stanford University, Stanford, California, United States. Correspondence to: Bilan Jackie Yang <bilany@stanford.edu>, Michelle Mengxuan Zhang <zhangmx@stanford.edu>.

The paper is structured as such: in Section 2, we provide references to some relevant work, including the meta-learning framework in reinforcement learning and social attention network. In the following section, we present our approach to tackle the problem. In Section 4, we describe our models and implementation details. In the final section, we present the training and evaluation results, as well as explanations and discussions correspondingly.

2. Related Work

We tackle the robotic navigation problem using the meta-learning technique through a social attention policy network. Our project is an integration and extension of model-agnostic meta-learning (MAML) (Finn et al., 2017) and social-attention reinforcement learning (SARL) (Chen et al., 2018). We take advantage of a meta-learning based policy to improve the robustness of the robot to various possible changes in the environment, which enables quick adaption to a similar environment. Meanwhile, the social attention network captures human behavior models in dense crowds and thus contributes to effective learning of pedestrian avoidance strategy.

2.1. Model-agnostic meta-learning

MAML is a generalized optimization method that can be applied to any model that learns through gradient decent (Finn et al., 2017). Instead of training a task-specific policy, the robot learns a policy that can be quickly adapted to any random task, as demonstrated in Fig. 1. The optimized parameters minimize the surrogate loss among all tasks, in stead of converging to a task-specific optimal policy. The parameters of the model are trained such that only small number of gradient steps are required given any new task.

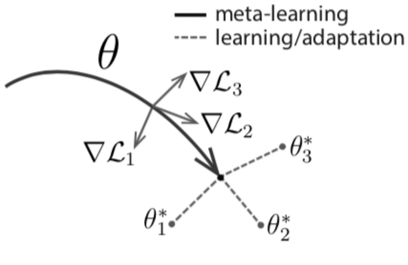

Figure 1. MAML diagram, where the resultant policy θ is close to any optimal task-specific policy θ_1^*, θ_2^*, and θ_3^*. (Finn et al., 2017)

We adapt MAML in particular to the robot navigation application, where each task represents one particular combination of pedestrian behaviors and goal positions. The reinforcement learning algorithm with meta-learning optimization is shown in Alg. 1. The learning is achieved by 1) updating parameters for each task-specific policy (Line 7), and 2) meta-optimizing the surrogate loss of all tasks (Line 10).

Algorithm 1 MAML for Reinforcement Learning

```
Require: $p(\mathcal{T})$: distribution over tasks
Require: $\alpha, \beta$: step size hyperparameters
 1: randomly initialize $\theta$
 2: while not done do
 3:   Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
 4:   for all $\mathcal{T}_i$ do
 5:     Sample $K$ trajectories $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_\theta$ in $\mathcal{T}_i$
 6:     Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ using $\mathcal{D}$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
 7:     Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
 8:     Sample trajectories $\mathcal{D}_i' = \{(\mathbf{x}_1, \mathbf{a}_1, ...\mathbf{x}_H)\}$ using $f_{\theta_i'}$ in $\mathcal{T}_i$
 9:   end for
10:   Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$ using each $\mathcal{D}_i'$ and $\mathcal{L}_{\mathcal{T}_i}$ in Equation 4
11: end while
```

2.2. Crowd-Robot Interaction

Understanding crowd-robot interactions involves modeling human-robot interactions and human-human interactions. SARL is a value-based network that captures interactions among agents (pedestrians) occurring in dense crowds, which indirectly affects the robot's anticipation capability (Chen et al., 2018). An interactive module is proposed for each pairwise human-robot interactions, and an attentive pooling module is incorporated to assist, in terms of learning the collective importance of neighboring humans with respect to their future states. These two modules aggregate together in the robot planning module which gives an approximation of the optimal value function. The whole module is described in Fig. 2

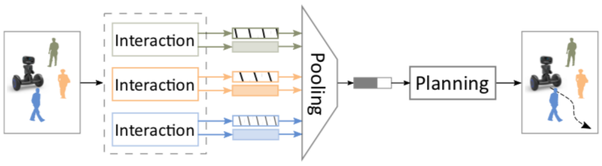

Figure 2. SARL method for socially attentive navigation made of 3 modules: Interaction, Pooling, and Planning. (Chen et al., 2018)

3. Approach

In our work, we refer to the structure of SARL to build the policy network with a meta-learning framework. We take advantages of meta-learning's capability of fast adaption to new tasks, as well as SARL's mechanism to effectively learn to avoid pedestrians while navigating to a goal. We made two major changes based on the original implementation of MAML for reinforcement learning: 1) a new policy network capable of extracting features regarding robot-human pairwise interactions; 2) a new environment that simulates a dense crowd and returns the environmental response given an action.

3.1. Robot Navigation Policy Network

A similar structure to the value approximation network in SARL is applied to approximate the policy in our model. The original implementation of SARL uses a value network and it uses the robot state transition model when extracting a policy. For sake of robustness and simplicity considerations, we decide to follow the MAML paper to parameterize a stochastic policy and directly optimize it via gradient descend. Instead of the simple neural network with 2 hidden linear layers and ReLU nonlinearities in the original MAML implementation, a more sophisticated network is constructed in order to capture robot-human interactions (Fig. 3).

The input to the policy network are robot-human state pairs stacking vertically. The robot state includes its positions (p_x, p_y), velocities (v_x, v_y), and goal positions (g_x, g_y) in both x and y directions. The pedestrian state for human i is composed of its positions $(p_{x,i}, p_{y,i})$ and velocities $(v_{x,i}, v_{y,i})$. The robot and pedestrian states are expressed as follows in a 2D continuous state space as:

$$\begin{aligned} s_{\mathcal{R}} &= [p_x, p_y, v_x, v_y, g_x, g_y], \\ s_{\mathcal{P},i} &= [p_{x,i}, p_{y,i}, v_{x,i}, v_{y,i}],\ i = 1, ..., n. \end{aligned} \tag{1}$$

The joint state input has a dimension as the integration of the robot and pedestrian state dimensions. It is linearly embedded into a fixed-size vector through a multi-layer perceptron (MLP), which consists of 3 linear layers and ReLU activation following each of the first two layers. The output from the first MLP goes through ReLu activation and is fed into MLP2 as well as the Attention MLP. The Attention MLP shrinks the input dimension to one and thus assigns exactly one attention score to each pedestrian. This score is used to measure how important of the behavior of one particular pedestrian is to the robot's control policy. For example, when a pedestrian is closer to the robot, a higher attention score will be expected. This score then goes through a masked softmax operation into a weight-parameter so that to transform the output from MLP2 into weighted features. This weighted features are expected to convey the information of robot human interactions. They are concatenated with the robot state to form the input to MLP3. The output of MLP3 serves as the mean of the normal distribution that represents the executed action, which is the robot velocity in 2D continuous space in our application.

3.2. Human-behavior Modeling and Simulator

A key part of the environment is behaviors of human pedestrians, where they should be governed by some rational policies, but are unobservable to the robot. In this work, we construct human-pedestrians as reciprocal velocity obstacles (RVO) (van den Berg et al., 2008), where each human acts fully independently and follows a hidden intended preferred velocity. Humans using the optimal reciprocal collision avoidance (ORCA) policy can observe each other's position and safety radius, as well as optimize their own velocity without much interference of the preferred velocity (van den Berg et al., 2011).

The simulation environment for human agents is constructed based on the official implementation of RVO algorithm – "RVO2", which is an open source library for interactive navigation and planning of large numbers of agents in two-dimensional (crowded) environments. With a fixed number of agents in the environment, we specify preferred velocities (in 2D) and collision avoidance regions at each step. The actual executed velocities are determined by RVO, which are assumed to be optimal with respect to their surroundings. Further documentation can be found at http://gamma.cs.unc.edu/RVO2/.

3.3. Environment

In our problem, the robot starts at a fixed position, and learn to navigate to a randomly assigned goal position while avoiding n moving pedestrians. The joint state s is composed of six robot states $s_{\mathcal{R}}$ and four human states $s_{\mathcal{P},i}$ for each pedestrian i. Actions of the robot include velocities in both directions v_x, v_y, which are clipped into $[-0.1, 0.1]$. The state space and action space are given as

$$\begin{aligned} s &= [s_{\mathcal{R}}, s_{\mathcal{P},1}, ..., s_{\mathcal{P},n}] \in \mathcal{R}^{6+4n} \\ a &= [v_x, v_y] \in \mathcal{R}^2 \end{aligned}$$

The reward function $r(s)$ in our formulation is state-dependent, which is the weighted sum of robot distance to the goal position r_d and robot-pedestrian collision indicator r_c. In particular

$$\begin{aligned} r(s) &= r_d(s) + r_c(s), \\ r_d(s) &= -w_d \cdot \sqrt{(g_x - p_x)^2 + (g_y - p_y)^2}, \\ r_c(s) &= \sum_{i=1}^{n} r_{c,i}(d_i). \end{aligned}$$

where w_d is the weight for distance reward. We use $w_d = 0.2$ in the following training and evaluation results. $r_{c,i}$

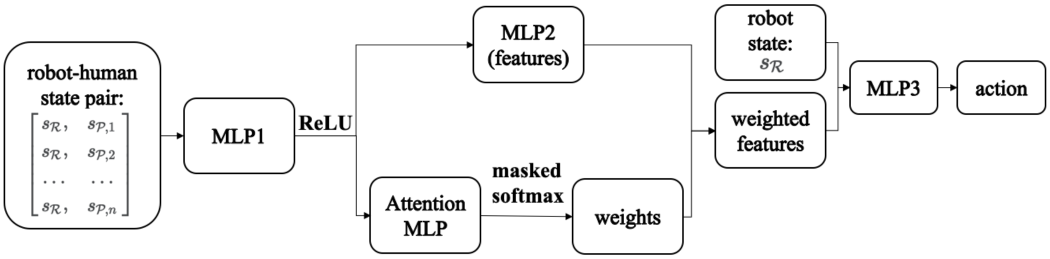
robot-human state pair:
$s_{\mathcal{R}}, \; s_{\mathcal{P},1}$
$s_{\mathcal{R}}, \; s_{\mathcal{P},2}$
$\cdots \; \cdots$
$s_{\mathcal{R}}, \; s_{\mathcal{P},n}$
MLP1
ReLU
MLP2 (features)
Attention MLP
masked softmax
weights
robot state: $s_{\mathcal{R}}$
weighted features
MLP3
action

Figure 3. Robot navigation network that replicates SARL features but using policy-based network.

is the collision-checker rewards for each pedestrian as a function of robot-pedestrian distance d_i, such that

$$r_{c,i}(d_i) = \begin{cases} -1.5 & \text{if } d_i \leq r_{\text{crit}}, \\ -0.2(r_{\text{safe}} - d_i) & \text{if } r_{\text{crit}} < d_i \leq r_{\text{safe}}, \\ 0 & \text{if } d_i > r_{\text{safe}}. \end{cases}$$

where r_{crit} is the critical collision distance, which is represented by pedestrian's radius; r_{safe} is the minimum safe separation distance between the robot and a pedestrian. The robot starts to be aware of the presence of a pedestrian once it falls in the r_{safe} zone centralized at the robot.

To facilitate MAML training, the environment samples tasks in order to create various training scenarios as well as to represent the stochasticity of the pedestrian-rich environment. A task captures a specific setting of goal position and pedestrian intentions. The pedestrian intentions are unobservable by the robot. In particular, the goal position is uniformly sampled within a square of $[-1, 1]$. Pedestrians are initialized at corners of the box with different moving directions and preferred velocities (ranges from 0.03 to 0.15). Once a pedestrian moves out of the boundary, it gets recycled and sent back to its initial position so as to maintain the same complexity level of the environment. Note that we only specify the preferred velocity of a pedestrian, but the actual executed travelling velocity is controlled by ORCA policy. Also, we prohibit pedestrians from avoiding the robot, so that the robot policy would not rely on the performance of ORCA.

4. Experiments

4.1. Training Models

A total of six models have been trained and evaluated, as shown in Table. 1. Models that are constructed under our proposed policy network (Section 3.1) are denoted as "MAML-SA", while models with the two-layer policy network from the original MAML implementation are denoted as "MAML". All models have same environment settings as detailed in Section 3.3. We vary 1) the number of pedestrians, 2) the number of robot-human interaction features, 3) the observable states, and 4) the coordinate system to evaluate their effect on the policy network performance.

As the goal of this project is to efficiently navigate through pedestrian crowds, we are interested in how the density of the crowd would make an impact on the robot's behavior. Also, different amounts of features are allocated to target at understanding the robot-human interactions. For MAML models, the cases when the policy network is trained with and without pedestrian states are tested. Moreover, a robot-centric coordinate system is implemented. The parameterization of the systems joint state (Eqn. (1)) may appear redundant from a geometric perspective. To reduce computational load and improve learning efficiency, a robot-centric frame is defined such that the origin is at the current robots position, and the x-axis is pointing toward the goal position:

$$\begin{aligned} s'_{\mathcal{R}} &= [d_g, v_x, v_y], \\ s'_{\mathcal{P},i} &= [p_{x,i}, p_{y,i}, v_{x,i}, v_{y,i}, d_i], \; i = 1, ..., n. \end{aligned}$$

where $d_g = \|\boldsymbol{p_g} - \boldsymbol{p}\|$ is the distance between robot and goal; $d_i = \|\boldsymbol{p_i} - \boldsymbol{p}\|$ is the distance between robot and a pedestrian. The last three columns describe the model performance in terms of rewards, which will be elaborated in Section 5.

4.2. Implementation Setup

As mentioned in the Section 3.1, the robot state has a dimension of 6 and the pedestrian state has a dimension of 4, which makes an input dimension of 10 for the policy network. The input and output dimensions for MLP1, MLP2, Attention MLP are $(10, 100)$, $(100, 4)$, $(100, 1)$. The concatenation of the robot state and weighted features gives a dimension of 14, which makes the input and output dimension of MLP3 to be $(14, 2)$.

We implemented our training model[1] with the proposed policy network model and the crowd environment in PyTorch (Paszke et al., 2017). The rest of the implementation

[1] Our model: https://github.com/biy001/pytorch-maml-navigation-in-crowds

Model	Pedestrian number	Interaction feature number	Observable states	Coordinate system	Distance rewards	Collision rewards	Total rewards
MAML-SA 1	4	4	robot + pedestrians	global	-21.3	-1.30	-22.6
MAML-SA 2	8	4	robot + pedestrians	global	-21.4	-2.65	-24.0
MAML-SA 3	8	50	robot + pedestrians	global	-23.9	-2.81	-26.7
MAML-SA 4	8	50	robot + pedestrians	robot-centric	-7.41	-0.55	-7.95
MAML 1	8	N/A	robot	global	-53.9	-1.87	-55.3
MAML 2	8	N/A	robot + pedestrians	global	-22.6	-2.44	-25.1

Table 1. Training model descriptions and their rewards.

is based on the meta-learning framework[2] available to the public. A total of 17 trajectories are sampled for each of the 22 tasks. Each trajectory has 100 time steps. The robot and pedestrians would reset to default positions after the robot reaches its goal, and receives zero-reward afterwards. Each task differs each other by the goal position, pedestrian speeds and their moving directions. A learning rate of 0.1 is used for the one-step fast adaption for each task. A discount factor of 0.9 is used to calculate returns. A linear baseline based on handcrafted features (Duan et al., 2016) are used to calculate the advantages estimated with Generalized Advantage Estimation (Schulman et al., 2015b). The meta-optimization step is achieved with Trust Region Policy Optimization (Schulman et al., 2015a). Parameters related to optimization includes maximum Kullback-Leibler (KL) divergence = 1e-2, Conjugate gradient (CG) iterations = 10, CG damping = 1e-5, line search maximum step = 15, line search backtrack ratio = 0.5.

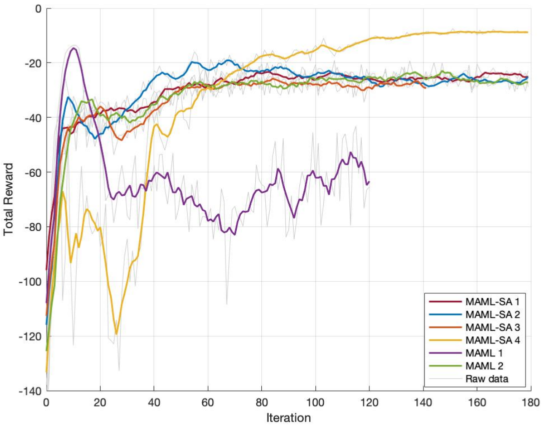

Figure 4. Smoothed training histories of total rewards.

5. Results and Discussions

5.1. Training Results

The rewards after convergence associated with the six models are summarized in the last three columns in Table 1. Models in the first four rows ("MAML-SA") use our proposed policy network which explicitly takes the robot-human interaction into consideration. The last two models are from the implementation of the original MAML paper with the crowd environment that we designed for our application (Section 3.3). Fig. 4, 5 present the reward progress during training. The oscillations in early iterations are subject to randomness, and we will mainly consider results in late training stage when they are close to convergence.

From Table 1, "MAML-SA 4" has the highest distance reward while having very low variance when close to convergence. "MAML 1" has the worst distance reward and high variance. "MAML-SA 1" has the highest collision reward, followed by "MAML-SA 4". All other models have similar performance.

Discussions: The distance reward has a much higher proportion in the total reward, which makes curves that corresponds to distance reward and total reward are very similar. Intuitively, it seems to attempt to lower the weight of distance and raise that of collision in rewards to better help the model to learn to avoid moving pedestrians. However, since the goal position is randomly generated for each task, it makes the model very confusing and cannot even direct the robot towards the goal in the first place. Since the pedestrian crowd is only modeled in the neighborhood of the robot's way to the goal, having a heavy weight in collision reward may cause the robot moving farther and farther away from the goal and the crowd so as to boost the collision reward. Thus, navigation towards the goal is of a higher priority, on the base of which the probability of collision is minimized. The discussion below elaborates on the impact of each factor on the training performance.

[2] Implementation of Model-Agnostic Meta-Learning (MAML) applied on Reinforcement Learning problems in Pytorch: https://github.com/tristandeleu/pytorch-maml-rl

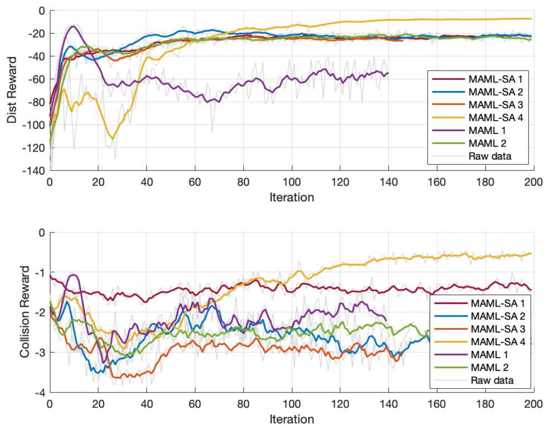

Figure 5. Smoothed training histories of distance and collision rewards correspondingly.

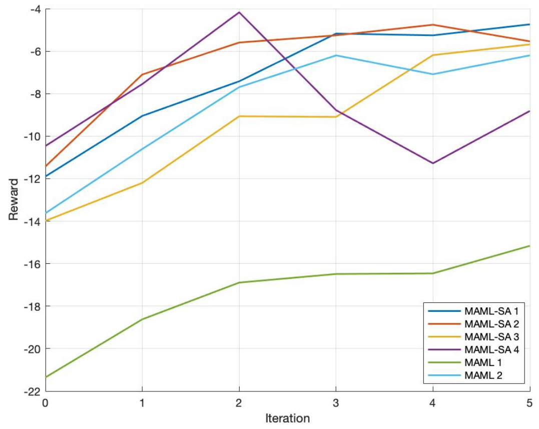

Figure 6. Evaluation results of task-specific gradient updates with 0.01 learning rate.

1) Number of pedestrians: "MAML-SA 1" has 4 pedestrians in the crowd simulation while all other models all have 8. "MAML-SA 1" has the highest collision reward, which is expected since by nature agents in a less dense crowd are less likely to collide with each other. A lower number of pedestrians does not necessarily impact the robot's trajectory due to the relatively low weight in collision reward.

2) Number of robot-human interaction features: "MAML-SA 2" and "MAML-SA 3" give a pure comparison for the case of varying the number of features describing robot-human interactions. It seems that simply increasing the amount of features will not boost the overall performance but could even slightly worsen it due to redundancy to a certain extent.

3) Observable states: The two MAML models differ by whether or not the pedestrian states are included in the states of the policy network. Observing pedestrians' states appear to be important in understanding where the collision reward is coming from. "MAML 1" does not observe pedestrians, and its strategy is to move away from the crowd as well as the goal so as to avoid the source of collisions.

4) Coordinate system: Among all 6 models listed in Table 1. "MAML-SA 4" is the most successful model with the highest rewards in both distance and collision. It is featured by its usage of the robot-centric frame in policy network. Even though the robot-centric frame contributes to high performance of the MAML-SA model, it actually causes the MAML model to diverge significantly to negative hundreds in total rewards (not shown in plots). A possible reason might be that the operation of using the robot-centric frame does not only simply change the coordinate system but also include using distance variables in the state representation to replace position variables. Explicitly including distance variables in the human-robot state pairs could boost the efficiency of learning the joint behaviors in robot-human interactions. However, in the MAML model, all robot and pedestrian state variables are concatenated in the input. This structure helps with learning how to reach higher rewards when the states are expressed in a 2D format globally in contrast to distance scalars.

5.2. Evaluation Results

In evaluation, we focus on task-specific policy performances, which include policy converging rate, robot trajectories and robot-human interactions examples.

Firstly, we compare model adaptation ability to a new task up to 5 gradient updates. The results in Fig. 6 show the adaptation performances of models that are initialized with MAML-optimal (pre-update) policy, and are updated with gradient descent on a particular task. Each curve is averaged from 5 tasks and 40 trajectories each. We see that a policy usually converges to task-specific optimal value within 3 steps, which indicates that the MAML training framework is able to quickly adapted to a new environment. The "MAML-SA 4" model is particularly sensitive and oscillate as it updates its policy gradient after two updating iterations. It indicates that the current model is close to the optimum; further updating (fine-tuning) requires smaller learning-rate.

The trajectory simulation results are shown in Fig. 7 and 8. We use one-step task-specific gradient update to demonstrate the quick-learning feature of our model. In both plots, we assign a randomly placed goal and pedestrian maneuvers to the environment. The one-step update has a great improvement over the original policy, in the sense that the

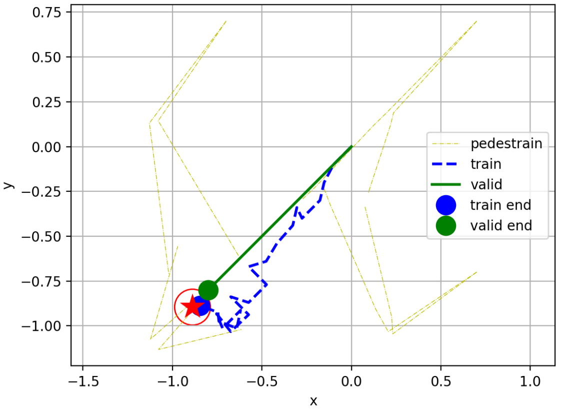

Figure 7. Trajectory simulation using model "MAML-SA 4", where the goal(marked by the red star) is located at $[-0.89, -0.90]$. The training trajectory (dashed-blue line) is generated by the pre-update policy, which takes 20 steps to reach the goal; whereas the validation trajectory (green line) is the one-step task-specific policy from the pre-update MAML policy, and takes 9 steps to reach the goal.

validation-policy takes less time (time steps) to navigate the robot to reach its goal while avoiding pedestrians. If no pedestrian is interfering with the robot (Fig. 7), the robot can figure out the shortest path and executes the maximum velocity. When pedestrians are close to the robot (Fig. 8), it avoids human with small steps, so that not to deviates too far from its shortest path. Further, we simulate the robot-pedestrian interactions, as shown in Fig. 9. We can see from pictures that the robot follows a zig-zag trajectory to approach the goal in order to avoid collision with pedestrians.

6. Conclusions and Future Work

In this work, we introduce MAML to the robot navigation application in a highly stochastic environment. A structured social attention model with features targeting at capturing human-robot interactions is used to represent robot policy network. A task-specific policy usually reaches its optimum within three steps. The simulation results show that the robot is able to safely navigate to its goal in a rapidly-changing pedestrian-rich environment. The source code and further demonstrations can be found at: https://github.com/biy001/pytorch-maml-navigation-in-crowds.git. In the future, we are interested in making our policy network more robust by only considering neighboring pedestrians when evaluating pedestrians' attention levels in training.

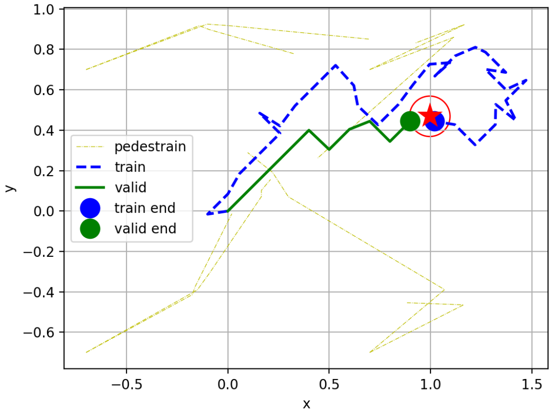

Figure 8. Trajectory simulation using model "MAML-SA 4", where goal position is $[0.990.47]$. Pre-update policy takes 33 steps to reach the goal, while one-step update policy takes 10 steps.

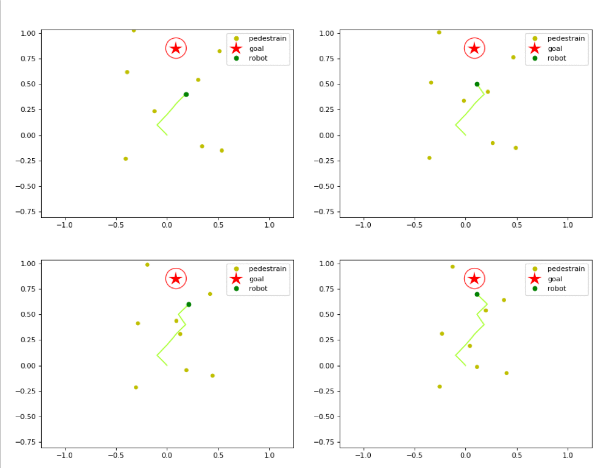

Figure 9. Trajectory simulation with pedestrian interactions.

References

Ariely, D. *Predictably Irrational: The Hidden Forces That Shape Our Decisions*. New York: Harper, 2008. URL http://radio.shabanali.com/predictable.pdf.

Chen, C., Liu, Y., Kreiss, S., and Alahi, A. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. *arXiv preprint arXiv:1809.08835*, 2018.

Chen, Y. F., Everett, M., Liu, M., and How, J. P. Socially aware motion planning with deep reinforcement learning.

2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Sep 2017. doi: 10.1109/iros.2017.8202312. URL http://dx.doi.org/10.1109/IROS.2017.8202312.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016.

Elbanhawi, M. and Simic, M. Sampling-based robot motion planning: A review. *IEEE Access*, 2:56–77, 2014. ISSN 2169-3536. doi: 10.1109/ACCESS.2014.2302442.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Goodrich, M. A. and Schultz, A. C. Human-robot interaction: A survey. *Foundations and Trends in Human-Computer Interaction*, 1(3):203–275, 2008. ISSN 1551-3955. doi: 10.1561/1100000005. URL http://dx.doi.org/10.1561/1100000005.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Peters, J. and Schaal, S. Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2219–2225, Oct 2006. doi: 10.1109/IROS.2006.282564.

Schulman, J., Levine, S., Abbeel, P., Jordan, M. I., and Moritz, P. Trust region policy optimization. In *Icml*, volume 37, pp. 1889–1897, 2015a.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

van den Berg, J., Lin, M., and Manocha, D. Reciprocal velocity obstacles for real-time multi-agent navigation. pp. 1928–1935, 05 2008. doi: 10.1109/ROBOT.2008.4543489.

van den Berg, J., Guy, S. J., Lin, M., and Manocha, D. Reciprocal n-body collision avoidance. In Pradalier, C., Siegwart, R., and Hirzinger, G. (eds.), *Robotics Research*, pp. 3–19, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-19457-3.

Yan, T., Zhang, Y., and Wang, B. Path planning for mobile robot's continuous action space based on deep reinforcement learning. pp. 42–46, 06 2018. doi: 10.1109/BDAI.2018.8546675.