Macro-Action Discovery Based on Change Point Detection and Boosting

Leonidas Lefakis
Idiap Research Institute
Martigny, Switzerland
leonidas.lefakis@idiap.ch

François Fleuret
Idiap Research Institute
Martigny, Switzerland
francois.fleuret@idiap.ch

***Abstract*—We present a novel approach to automatic macro-action discovery and its application to a complex goal-planning task. The problem of macro-action discovery is framed as one of multiple change point detection and is addressed with the help of the Dynamic Programming Boosting algorithm. The procedure is then employed to solve a complex goal-planning problem which entails an avatar navigating a 3D environment. By using DPBoost to decompose the problem into a number of simpler ones, we are able to successfully address both the complexity and partial observability of the environment.**

***Keywords*-Goal-Planning, Imitation Learning, Macro-Action Discovery.**

I. Introduction

Our main contribution is a novel approach to automatically decompose a teacher policy into a mixture of policies referred to as *macro-actions*. Given a training set demonstrating the teacher's policy, our procedure encompasses two separate steps: The first uses DPBoost algorithm [1] which comprises an alternating procedure that goes back and forth between learning macro-actions and estimating at any time step in the exemplar sequences which macro-action is being executed. The second learns a switch function which indicates when to switch from one macro-action to the other.

In our main experiments, we wish to learn a policy to pilot an avatar in a virtual 3D environment (see Figure 1). The task presented to the avatar is one of reaching two flags in the correct order. More specifically, a blue flag and a red flag are dispatched in the environment at random locations. The goal is then to come in contact with (reach) the blue flag initially and to subsequently come in contact with the red flag. Touching the red flag without having first touched the blue one results in a failed run.

This environment is challenging from a goal-planning perspective due to its complexity, the size of the state-space is considerably larger than that of typical maze problems often found in the literature. The signal passed to the goal-planner consists of images, thus making the problem one of visual goal-planning; as a consequence the goal-planner must also address the task from a computer vision point of view. The task is further complicated by the fact that given solely the avatar's view, the environment is only partially observable; without further information the avatar does not know when confronted with the red flag whether it must move towards it or ignore and move to find the blue flag as there is no information in the signal regarding whether the avatar has already reached the blue flag.

II. Related Work

Standard reinforcement learning [2] settings typically rely on simple actions through which a goal-planning algorithm interacts with an environment, typically modeled as a Markov Decision Process, by performing actions and receiving rewards at various time steps. Tasks tackled by reinforcement learning algorithms are typically of limited complexity (e.g. 2D maze problems). Even in the case of more complex tasks which have been addressed successfully (e.g. Backgammon [3]), typically a large number of interactions with the environment are required before a near-optimal policy is learned. For many tasks however this demand is not feasible thus limiting the applicability of RL approaches.

A. Macro-Actions

The introduction of hierarchical reinforcement learning [4] and more specifically the options framework [5] is a promising route towards scaling up RL. In this setting, the agent has at its disposal, beyond the simple primitive actions a number of options (variably known as skills or macro-actions) which consist of a policy, a termination condition, and an initiation set.

One of the core issues for the options framework is that of automatically discovering these macro-actions. In [6] the authors use novelty detection to develop options which lead to *surprising* events. The authors in [7] use a graph-cut approach to discover useful sub-goals of the task at hand, they then use the options framework to learn macro-actions that lead to these sub-goals. Konidiaris *et al.* [8] learn what they name a skill chain. Their algorithm effectively learns a series of options by backtracking from the goal in such a way that the termination state of each option is in the initiation set of the next. In [9] the authors segment demonstration trajectories to create skill trees which are constructed from macro-actions. Like the work presented here the authors cast the problem of macro-action discovery as one of change point detection.

B. Imitation Learning

In the setting of goal-planning another approach that has proven fruitful in solving complex tasks is that of imitation learning [10] also referred to as learning from demonstration or mimicking. In this framework, the agent is presented with a number of trajectories which are typically considered to be generated by an optimal or near-optimal policy. In practice these trajectories are created by having an expert, i.e. the teacher, navigate the environment.

Several different approaches have been proposed towards exploiting the information present in these trajectories. Inverse reinforcement learning [11] aims to infer the reward function $R(s_t)$ given the demonstration trajectories. Maximum margin planning [12] casts imitation learning as a structured prediction problem, solving it by a sub-gradient approach. In their seminal paper, Atkeson and Schaal used demonstration trajectories to learn both a model and a reward function and subsequently used planning to calculate a good policy [13].

Perhaps most closely related to the work presented here is the approach where learning is cast as a classification problem. In this approach, typically one or more classifiers are used to learn a direct state-to-action mapping of the optimal policy. A well-known example of such an approach is the ALVINN autonomous land vehicle [14]. More recently Lecun *et al.* [15] used a convolutional neural network structure to effectively learn a state-to-action mapping for off-road driving.

C. Change point detection

As noted the present work casts the problem of discovering macro-actions in demonstration trajectories as a change point detection problem [16]. In change point problems one is confronted with a set of samples collected over time and which are assumed to have been generated disjointly by different models.

Given this setting, the problem of macro-actions discovery becomes one of segmenting the teacher trajectories so that each segment is assigned to a different predictor. This casting of macro-action discovery as a change point detection problem is similar to that proposed in [9] where the problem is solved using a Viterbi algorithm to estimate the models of maximum *a posteriori* probability.

Contrary to these approaches we aim to leverage more powerful discriminative learning algorithms, namely Adaboost, and have thus employed the DPBoost algorithm [1] which can exploit these discriminative learners while avoiding the very tricky issue of normalizing properly a probabilistic transition model and a discriminative emission model, by considering a regularization that upper-bounds strictly the number of transitions.

III. Dynamic Programming Boosting

As summarized in the introduction, the first step in our approach is to automatically decompose the expert trajectories into a mixture of several simple macro-actions.

Precisely, we get a sequence of training samples, each composed of an available signal (an image) and an action, and we know that each of these samples is associated to a macro-action, which is unknown. The only prior knowledge is the regularity of these macro-actions, which do not change often, and that given these macro-actions, the action can be predicted from the available signal. Thus we consider the case of training a family of Q multi-class classifiers from a sequence of N training samples that we know is composed of Q underlying macro-classes.

Using an alternating procedure, DPBoost [1] builds Q multi-class classifiers $f_q = \sum_1^T a_q^t h_q^t$, by re-estimating at each time step t the optimal macro-labeling of the N samples and adding weak learners h_q^t in the strong classifiers f_q associated to each macro-label (macro-action), with a standard Boosting criterion.The corresponding macro-labels are unknown during training but given these macro-labels, the samples are considered i.i.d..

In the case of macro-actions discovery, the macro-label stands for the macro-action currently in progress, and the label for the action executed. So classifiers and classes can be assigned respectively to macro-action policy and actions, and macro-classes correspond to macro-action labels q.

IV. Macro-Action Switching

During the training phase of the algorithm, change point detection is handled by the dynamic programming step of the algorithm. During the testing phase however, the algorithm must rely on switching functions

$$H_q^{q'} : \mathbb{R}^D \rightarrow \{-1, 1\}, \quad \forall q \neq q'$$

which will signal the shift from macro-class q to macro-class q'. Such functions can be trained from the $q_1, \ldots, q_N$ estimated by $DPBoost$. In the context of goal-planning these functions would act as indicators for moving from one macro-action to the next.

In order for these switch functions to be effective in practice, it is imperative that they do not signal a transition too soon. Thus we would like that $\forall n \leq m, H_q^{q'}(x_n) < 0$, where m signifies the moment of transition. On the other hand, the switching function need only respond positively at the moment of transition, thus its response is indifferent $\forall n > m$.

Based on this, the training set for each switching function $H_q^{q'}$ is built as follow: For each trajectory where the transition $q \rightarrow q'$ occurs, at moment m, we gather the samples x_n for which $n < m$ and couple them with a negative label $y_n^{q \rightarrow q'} = -1$, where $y_n^{q \rightarrow q'}$ signifies the label of sample x_n for training $H_q^{q'}$ as opposed to its true label y_n. For the positive samples we simply need the samples x_m at the moment the transitions from $q \rightarrow q'$ occur in the various trajectories. In practice however in order to make our switching functions

more robust we add J positive samples $x_m, ..., x_{m+J-1}$ from each trajectory.

Using the constructed training sets $\{x_n, y_n^{q \to q'}\}$, the functions $H_q^{q'}$ are learned using AdaBoost with classification stumps. Of course any other discriminative approach can be employed at this step.

V. Experimental results

The motivation for the use of the DPBoost algorithm has been to address a complex goal-planning task. The task entails an avatar navigating a 3D simulated environment generated with the OGRE 3D graphics rendering engine. This engine produces a realistic rendering of the scene as seen by the avatar. The engine is capable of depicting complex 3D objects, as well as the effects of differing lightning sources. As can be seen in Figure 1, the rendered result is of high quality and reminiscent of modern 3D video games. The simulator is furthermore equipped with a physics engine that allows for the realistic simulation of the physical interactions between the avatar and its environment.

The possible actions that the avatar can take in this application, are "move forward", which moves the avatar forward by 3cm, and "turn left" or "turn right" which alter the avatar's orientation by $\pi/300$. The size of the room is generated at random to be between 5 and 20 meters, while the textures displayed on the walls and ground are also picked at random. The lights are at fixed locations to avoid overly dark areas.

In order to test our approach experimentally we train two goal-planners, one with two and another with four macro-actions, We compare DPBoost against a baseline which consists of a goal-planner based on a single AdaBoost classifier which performs state-to-action mapping; in this case there is no Dynamic Programming component to the learning phase nor any switching function. The baseline is denoted with a dash in Table I.

The data used for training consist of twenty trajectories generated using a hard-coded teacher. The teacher has knowledge of whether the blue flag has already been reached or not, information which is not passed on to the goal-planner, instead the goal-planner must infer the correct action solely from visual information. To be more specific, the data x_n available to DPBoost consist of the RGB values of a scaled down version (from 320×240 to 64×48 pixels) of the avatar's current view, as well as the difference between the RGB values of the down scaled version of the avatar's current view and the RGB values of the down scaled version of the avatar's previous view i.e. before the last action. As mentioned earlier the avatar has no memory of past events, in particular it has no information regarding whether it has already reached the blue flag. The information passed to the goal-planner can be seen in Figure 2

In our original experiments the positions of the two flags are generated at random and then remain fixed throughout the training and test runs. The results obtained can be seen in Table I marked Fixed Flag Locations, the percentages here refer to the percentage of successful runs.

Table I
Results

Fixed Flag Locations			
Nb. of macro-actions	-	2	4
Success Rate	92%	92%	68%
Randomized Flag Locations			
Nb. of macro-actions	-	2	4
Success Rate	28%	56%	40%

This results may at a first glance seem surprising, the single classifier baseline goal-planner succeeds in solving the task despite the partial observability of the environment. In order to accomplish this the single classifier planner must have effectively learned when to ignore the red flag and when not. Though as stated this may seem surprising it can be explained by the fact that for fixed flag locations, in an successful trajectory the avatar reaches the blue flag and then approaches the red flag always from more or less the same avenue i.e. the trajectory from the blue flag to the red flag. Thus a single classifier can side-step the issue of partial observability by memorizing, so to speak, this avenue from blue flag to red flag ignoring the red flag whenever its view of the flag is not along this avenue, effectively over-fitting to the specific flag configuration.

To test this hypotheses we present the avatar with a more complex problem. Namely the positions of the flags are reinitialized to random positions after each run both in training and in test. As can be seen by the results in Table I marked Randomized Flag Locations, this setting proves too challenging for the baseline planner and its performance degrades to a greater extent than that of the multiple macro-action planners. It should be noted here that the teacher providing the optimal trajectories was hard-coded and was not itself capable of solving the problem for all flag configurations; such issues may arise for example when one flag hides the other flag from view. Testing the teacher on the same Randomized Flag Locations, we find that the expert the planner is trying to emulate is only capable of solving the task 72% percent of the time.

VI. Conclusion

We presented a novel approach to macro-action discovery. The strength of the proposed method was shown empirically in a complex partially observable goal-planning environment.

In the context of goal-planning, DPBoost can be extended to function in conjunction with other methods, as for example the DAGGER algorithm [17]. We aim to incorporate DPBoost into such a framework in order to solve more complex tasks.

Finally, we also aim to extend our work on switch-function training in order to embed it into the DPBoost training procedure.

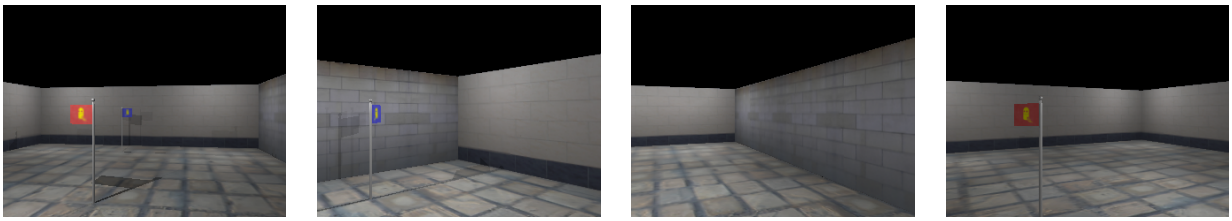
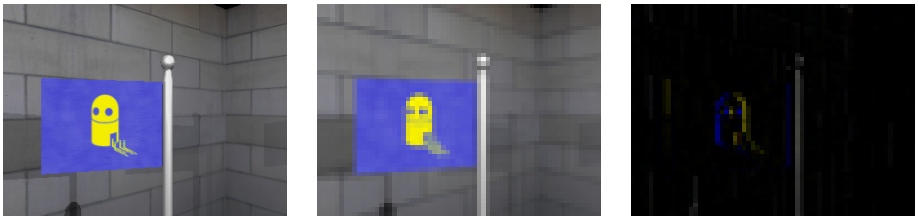


Figure 1. Rendering of the avatar's view.



(A) (B) (C)

Figure 2. (A) shows the original image I_t rendered by the simulator and the images passed to DPBoost. (B) is a downscaled version I_t^s of the image while (C) shows the difference between the images I_t^s and I_{t-1}^s

ACKNOWLEDGMENT

This work was supported by the European Community's Seventh Framework Programme FP7 - Challenge 2 - Cognitive Systems, Interaction, Robotics - under grant agreement No 247022 - MASH.

REFERENCES

[1] L. Lefakis and F. Fleuret, "Dynamic programming boosting," *Idiap Research Institute Technical Report*, October 2012.

[2] R. S. Sutton and A. G. Barto, *Reinforcement learning : an introduction*, ser. A Bradford Book. The MIT Press, 1998.

[3] G. Tesauro, "Temporal difference learning and td-gammon," *Commun. ACM*, vol. 38, no. 3, pp. 58–68, Mar. 1995.

[4] A. G. Barto and S. Mahadevan, "Recent advances in hierarchical reinforcement learning," *Discrete Event Dynamic Systems*, vol. 13, no. 1-2, pp. 41–77, Jan. 2003.

[5] R. S. Sutton, D. Precup, and S. P. Singh, "Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, no. 1-2, pp. 181–211, 1999.

[6] S. P. Singh, A. G. Barto, and N. Chentanez, "Intrinsically motivated reinforcement learning," in *NIPS*, 2004.

[7] O. Şimşek, A. P. Wolfe, and A. G. Barto, "Identifying useful subgoals in reinforcement learning by local graph partitioning," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 816–823.

[8] G. Konidaris and A. G. Barto, "Skill discovery in continuous reinforcement learning domains using skill chaining," in *NIPS*, 2009, pp. 1015–1023.

[9] G. Konidaris, S. Kuindersma, A. G. Barto, and R. A. Grupen, "Constructing skill trees for reinforcement learning agents from demonstration trajectories," in *NIPS*, 2010, pp. 1162–1170.

[10] B. Argall, S. Chernova, M. M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[11] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *In Proceedings of the Twenty-first International Conference on Machine Learning*. ACM Press, 2004.

[12] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, "Maximum margin planning," in *In Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[13] C. G. Atkeson and S. Schaal, "Aprobot learning from demonstration," in *In Proceedings of the Fourteenth International Conference on Machine Learning*. Morgan Kaufmann, 1997.

[14] D. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *NIPS*, 1988, pp. 305–313.

[15] R. Hadsell, P. Sermanet, J. Ben, A. Erkan, M. Scoffier, K. Kavukcuoglu, U. Muller, and Y. LeCun, "Learning long-range vision for autonomous off-road driving," *J. Field Robot.*, vol. 26, no. 2, pp. 120–144, Feb. 2009.

[16] P. Fearnhead and Z. Liu, "Online inference for multiple changepoint problems," *Journal of the Royal Statistical Society: Series B*, vol. 69, no. 4, pp. 589–605, 2007.

[17] G. G. Stephane Ross and J. A. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011.