Learning, then Compacting Visual Policies (Extended Abstract)

Sébastien Jodogne S.Jodogne@ULg.ac.be
Justus H. Piater Justus.Piater@ULg.ac.be
Montefiore Institute (B28), University of Liège, B-4000 Liège, Belgium

Abstract

We propose a way to fight overfitting in the recently-published *Reinforcement Learning of Visual Classes* (RLVC) algorithm. RLVC interactively learns mappings from images to actions. In RLVC, the visual space is incrementally partitioned into a smaller set of visual classes, by testing the presence of highly informative visual features. Since the splitting rule is local, useless features are often selected, which reduces the performance of the algorithm. We introduce a method to aggregate visual classes that are similar with respect to the theory of Markov Decision Processes using *Binary Decision Diagrams*. We illustrate the practical interest of this approach on a complex visual navigation task.

1. Introduction

Constructing an optimal control policy through Reinforcement Learning (RL) becomes exponentially harder as the size of the perceptual (input) space grows. This is known as the *curse of dimensionality*. There exist only little work on perceptual spaces consisting of images, which are extremely high-dimensional and noisy. Likewise, in the Computer Vision community, there has been so far only very little focus on what could be called the *vision-for-action* paradigm, which consists in computing image-to-action mappings suitable for a given visual task. However, in many robotic control tasks, the learning agent is equipped with cameras. We therefore argue that RL algorithms able to directly tackle visual spaces should be developed.

The issue of dealing with large, discrete perceptual spaces in RL was previously addressed by the G Algorithm (Chapman & Kaelbling, 1991) and by the so-called "selective attention" mechanism of U Tree (McCallum, 1996). These algorithms incrementally refine a decision tree, in a sequence of attempts to remove perceptual aliasing. The decision tree partitions the input space into several perceptual classes by testing the presence of domain-specific features that are highly relevant for solving the task. This idea was later extended to continuous input spaces, like in Continuous U Tree (Uther & Veloso, 1998) and Variable Resolution Grids (Munos & Moore, 2002). However, these papers do not answer the question of which type of features could be used in the case of visual tasks.

We have recently proposed to take advantage of the visual features that are used in *local-appearance methods* in Computer Vision (Jodogne & Piater, 2005). Such methods have had much success in applications such as image matching, image retrieval and object recognition. They are indeed powerful and flexible, as they are robust to partial occlusions, and do not require segmentation or 3D models of objects. They rely on the detection of discontinuities in the visual signal thanks to *interest point detectors*. The image patch around each interest point is then summarized as a vector of real numbers, which is called a *visual feature* (Mikolajczyk & Schmid, 2003). Once a suitable metric (e.g., Mahalanobis' or Euclidean distance) is defined upon the visual features, two images can be matched if they share a sufficient number of matching visual features.

In our framework, which we called *Reinforcement Learning of Visual Classes* (RLVC), we build a decision tree that tests, at each of its internal nodes, the existence of one visual feature in the visual stimulus. The leaves of the decision tree define a set of visual classes, which is hopefully exponentially smaller than the original visual space, and upon which it is possible to apply directly any usual RL algorithm. The construction of the decision tree is carried on incrementally, like in the G Algorithm and in U Tree, but using a different splitting rule that targets zero Bellman residuals and that relies on an information-theoretic measure. We have shown that this algorithm is of practical interest, by solving a visual navigation task.

Unfortunately, because of its greedy nature, RLVC is highly subject to overfitting. Splitting *one* visual class has an impact on the Bellman residuals of *all* the visual classes. Therefore, the splitting strategy can get stuck in local minima: Once a split is made that subsequently proves useless, it cannot be undone in the

original description of RLVC. In this work, we propose to provide RLVC with the possibility of aggregating visual classes that share similar properties. Doing so has at least three potential benefits: (i) Useless features are discarded, which enhances generalization capabilities; (ii) RLVC can escape from local minima by re-initializing the search for good features; and (iii) the number of samples that the embedded RL algorithm has at its disposal for each visual class is increased, which results in better visual control policies.

2. Compacting Visual Policies

We cannot include here a thorough discussion of RLVC for space reasons. The reader is kindly invited to consult our previous papers for more details (Jodogne & Piater, 2005). For the purpose of this work, it is sufficient to know that RLVC builds a sequence $(\mathcal{C}_0, \ldots, \mathcal{C}_k, \ldots)$ of decision trees until no further perceptual aliasing is detected. Each $\mathcal{C}_k$ induces a *mapped* Markov Decision Process (MDP), the states of which correspond to the leaves of $\mathcal{C}_k$. Furthermore, for each k in the sequence, a traditional RL algorithm is applied, so that properties like the optimal value function $V_k^*(\cdot)$, the optimal state-action value function $Q_k^*(\cdot,\cdot)$ and the optimal control policy $\pi_k^*(\cdot)$ are known for the MDP that is mapped through $\mathcal{C}_k$. Using those properties, it is easy to define a whole range of equivalence relations between the visual classes. For instance, given a threshold $\varepsilon \in \mathbb{R}^+$, we list hereunder three possible equivalence relations for a pair of visual classes (c_k, c'_k):

Optimal Value Equivalence:
$|V_k^*(c_k) - V_k^*(c'_k)| \leq \varepsilon$.

Optimal Policy Equivalence:
$|V_k^*(c_k) - Q_k^*(c'_k, \pi_k^*(c_k))| \leq \varepsilon \wedge$
$|V_k^*(c'_k) - Q_k^*(c_k, \pi_k^*(c'_k))| \leq \varepsilon$.

Optimal State-Action Value Equivalence:
$(\forall a \in A)|Q_k^*(c_k, a) - Q_k^*(c'_k, a)| \leq \varepsilon$.

We therefore propose to modify RLVC so that, periodically, visual classes that are equivalent with respect to one of those criteria are merged together. We have experimentally observed that the conjunction of the first two criteria seems to lead to the best performance. This way, RLVC alternatively splits and merges visual classes. Therefore, a visual class cannot be represented as a single branch of a decision tree anymore. Rather, a visual class is characterized by a general Boolean function, the variables of which correspond to the tested visual features, so that we need a more expressive tool than decision trees to represent the visual classes.

One such suitable tool is the *Binary Decision Diagram* (BDD) (Bryant, 1992). BDD is a tree-based symbolic representation for encoding arbitrary Boolean functions, and has had much success in the field of computer-aided verification. A BDD is unique when the ordering of its variables is fixed, but different variable orderings can lead to different sizes of the BDD, since some variables can be discarded by the reordering process. Although the problem of finding the optimal variable ordering is NP-complete, heuristics can in practice find orderings that are closed to optimality. This is interesting in our case, since reducing the size of the BDD amounts to discarding irrelevant variables, which correspond to useless visual features.

To summarize, the modified RLVC does not use decision trees anymore, but assigns one BDD to each visual class. The process of refining, with a visual feature f, a visual class v that is labeled by the BDD $\mathcal{B}(v)$, consists in replacing v by two visual classes v_1 and v_2 such that $\mathcal{B}(v_1) = \mathcal{B}(v) \wedge f$ and $\mathcal{B}(v_2) = \mathcal{B}(v) \wedge \neg f$. Merging a pair of visual classes (v_1, v_2) amounts to deleting v_1 and v_2, and adding a new visual class v such that $\mathcal{B}(v) = \mathcal{B}(v_1) \vee \mathcal{B}(v_2)$. Every time a merging operation takes place, variable reordering is carried on.

3. Experimental Results

We have applied the modified version of RLVC to a simulated navigation task. In this task, the agent moves between 11 spots of our campus (Figure 1). Every time the agent is at one of the 11 locations, its body can aim at four possible orientations (north, south, west, east). The state space is therefore of size $11 \times 4 = 44$. The agent has three possible actions (turn left, turn right, go forward). Now, the agent does not have direct access to its position and its orientation. Rather, it only perceives a picture of the area that is in front of it (Figure 2). So, the agent has to connect images to the appropriate reactions without knowing the underlying physical structure of the task.

We have used the SIFT keypoints as visual features (Lowe, 2004). There was a total of 2,591 distinct features. The original version of RLVC has identified 644 visual classes by selecting 336 SIFT features. The error on the computed visual policy was 2% on the learning set and 22% on the test set[1]. The modified RLVC was then applied, with one compacting every 5 steps. The results are shown in Figure 3 and are clearly superior: There is an improvement in the generalization abilities, as well as a reduction of the number of visual classes and selected features.

As a conclusion, compacting visual policies is probably a required step to deal with realistic visual tasks, if an

[1] With respect to the optimal policy when the agent has direct access to its position and viewing direction.

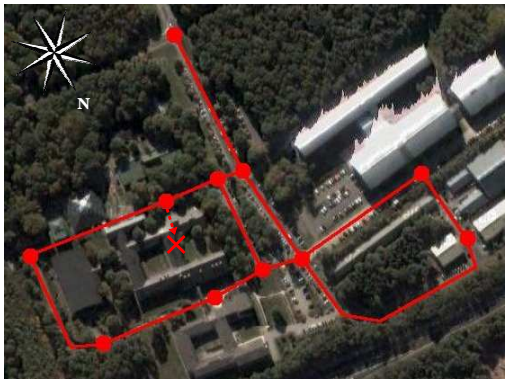

Figure 1. The Montefiore campus at Liège. Red spots corresponds to the places between which the agent moves. The agent can only follow the links between the different spots. Its goal is to enter the Montefiore Institute, that is labeled by a red cross, and where it gets a reward of 100. Turning left or right induces a penalty of -5, and moving forward, a penalty of -10. The discount factor γ is set to 0.8.

iterative splitting process is applied. Future work will focus on a theoretical justification of the used equivalence relations. This implies bridging the gap with the theory of MDP minimization (Givan et al., 2003).

References

Bryant, R. (1992). Symbolic boolean manipulation with ordered binary decision diagrams. *ACM Computing Surveys*, *24*, 293–318.

Chapman, D., & Kaelbling, L. (1991). Input generalization in delayed reinforcement learning: An algorithm and performance comparisons. *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence* (pp. 726–731).

Givan, R., Dean, T., & Greig, M. (2003). Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, *147*, 163–223.

Jodogne, S., & Piater, Journal (2005). Interactive learning of mappings from visual percepts to actions. *Proc. of the 22nd Int. Conf. on Machine Learning* (pp. 393–400).

Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision*, *60*, 91–110.

McCallum, R. (1996). *Reinforcement learning with selective perception and hidden state*. Doctoral dissertation, University of Rochester, New York.

Mikolajczyk, K., & Schmid, C. (2003). A performance evaluation of local descriptors. *IEEE Conf. on Computer Vision and Pattern Recognition* (pp. 257–263).

Munos, R., & Moore, A. (2002). Variable resolution discretization in optimal control. *Machine Learning*, *49*, 291–323.

Uther, W. T. B., & Veloso, M. M. (1998). Tree based discretization for continuous state space reinforcement learning. *Proc. of the 15th National Conf. on Artificial Intelligence (AAAI)* (pp. 769–774).

Figure 2. The percepts of the agent. For each possible location and each possible viewing direction, a database of 24 images of size 1024×768 with viewpoint changes has been collected. Those 44 databases have been divided into a learning set of 18 images and a test set of 6 images. 4 different percepts are shown, that correspond to the location and viewing direction marked in yellow on the top image.

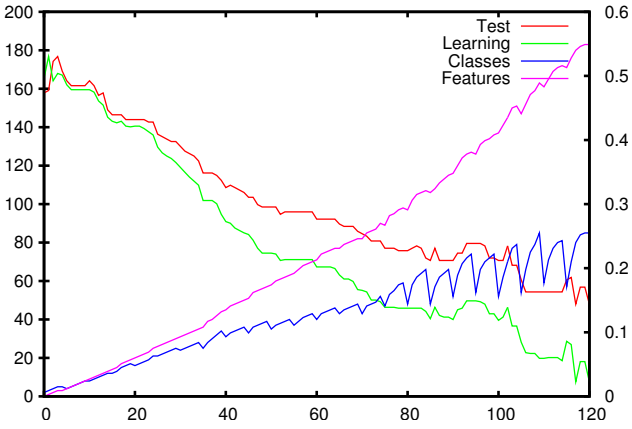

Figure 3. Statistics about the modified RLVC as a function of the step counter k. The green (resp. red) plot corresponds to the error of the computed policy on the learning (resp. test) set. At the end, the learning error is 2% and the test error is 14%, which is superior to the scores of the original RLVC. The number of visual classes (resp. selected features) is plotted in blue (resp. purple). The decreases in these curves indicate the compacting phases. Only 183 visual features were selected. Interestingly, 57 visual classes were identified, which is close to 44, the number of states.