Pairwise Relative Offset Features for Atari 2600 Games

Erik Talvitie
Department of Mathematics and Computer Science
Franklin & Marshall College
Lancaster, PA, 17552 USA

Michael Bowling
Department of Computing Science
University of Alberta
Edmonton, AB, T6G 2E8 Canada

Abstract

We introduce a novel feature set for reinforcement learning in visual domains (e.g. video games) designed to capture pairwise, position-invariant, spatial relationships between objects on the screen. The feature set is simple to implement and computationally practical, but nevertheless allows for substantial improvement over existing baselines in a wide variety of Atari 2600 games. In the most dramatic results the features allow multiple orders of magnitude improvement in performance.

1 Introduction

Selecting a good representation of state has long been understood to be a key concern for reinforcement learning with function approximation. While the representation is often tuned by an agent designer specifically for individual problems, when the agent must succeed in many problems the challenge of finding a good representation is even greater. This paper introduces a new set of features for use in domains where the agent's observations are images (e.g. video games). The features encode the fact that the relative positions of objects to each other is often more important than their absolute positions. The utility of these features for reinforcement learning with linear value function approximation is demonstrated in over 50 Atari 2600 games using the Arcade Learning Environment (ALE) (Bellemare et al. 2013), where they exhibit robustly good performance and outperform existing baselines in a wide variety of examples.

2 Linear Value Function Approximation

We focus our attention on Markov Decision Processes (MDP). At each discrete time step t the environment is in some state $s_t \in \mathcal{S}$. The agent chooses an action $a_t \in \mathcal{A}$, which causes the environment to enter state $s_{t+1} \sim T(\cdot \mid s_t, a_t)$ sampled from a transition probability distribution conditioned on the current state and action. The agent also receives a reward $r_{t+1} = R(s_t, a_t, s_{t+1})$. The agent's goal is to obtain a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected discounted return $V^\pi(s) = \mathrm{E}\left[\sum_{k=1}^{\infty} \gamma^{k-1} r_{t+k} \mid s_t = s\right]$ for every state s, where the expectation is over the stochasticity in the environment and $\gamma \in (0, 1]$ is the discount factor. The function V^π is called the *state value function for policy* π. The associated *state-action value function* $Q^\pi(s,a) = \mathrm{E}_{s'}\left[R(s,a,s') + \gamma V^\pi(s')\right]$ represents the expected return from taking action a in state s and following π forever after.

Many reinforcement learning algorithms learn a policy from experience by continually updating an estimate of the state-action value function of the current policy, and then updating the policy to behave greedily (or nearly greedily) with respect to that value function. When the state space is large, it is common to approximate the value function, most commonly as a linear function. In that case, the state is represented by some vector of k features generated by the function $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^k$ and the value function is parameterized by a real vector θ. Then $Q(s,a) \approx \theta^\top \phi(s,a)$. For a detailed introduction to reinforcement learning methods, the reader is referred to Sutton and Barto (1998); the main issue concerning this paper is that the choice of ϕ can have a dramatic impact on learning performance in terms of both the number of samples needed to achieve good performance and the asymptotic performance of the learned policy.

3 Features for Atari 2600 Games

Multiple choices for ϕ have been explored in the ALE environment. The feature set most relevant to this work is the *Basic* feature set (Bellemare et al. 2013). Basic features are generated by dividing the 160×210 pixel Atari 2600 screen into 10×15 pixel blocks. Then for every block (c, r) with $c \in \{0, 15\}$ and $r \in \{0, 13\}$ and NTSC color $k \in \{0, ..., 127\}$ there is a binary feature $\phi_{(c,r),k}$ that is 1 if any pixel contained in block (c, r) has color k, and 0 otherwise. Thus there are $16 \times 14 \times 128 = 28,672$ Basic features (most of which are 0 in a given frame). Basic features try to exploit the fact that many Atari 2600 games use color to distinguish object types and that most objects in Atari 2600 games are larger than a single pixel to give a crude idea of the location and type of each object in the game. Bellemare et al. (2013) made the Basic feature vector more sparse by extracting a background image for each game offline and subtracting it from each frame before encoding.

Closely related to Basic is the *BASS* feature set (Bellemare et al. 2013). The BASS feature set includes the Basic features, and also every pairwise product of the Basic features. BASS features try to capture the fact that not only is the position of each individual object important, but so are the spatial relationships between objects. Note that if the full NTSC color palette were used there would be

$28,672^2 = 822,083,584$ pairwise features. Even though the feature vector is quite sparse, it can be impractical to work with nearly a billion features. So the smaller SECAM palette, which has only 8 colors was used, yielding 1,792 Basic features and 3,211,264 pairwise features.

Other more sophisticated techniques have been proposed as well. Bellemare et al. (2013) also presented results using Local Sensitivity Hashing (hashing a bit vector representing the image down to a smaller number of features) and the DISCO features which used basic computer vision to detect moving "blobs" of color, categorize the blobs into object types, and finally perform tile-coding over blob positions, pairwise relative positions, and velocities. Bellemare, Veness, and Bowling (2012a) used a learned predictive model in conjunction with a Bayesian filter to determine the portion(s) of the screen directly under control of the player (called *contingent regions*). The location of the contingent regions were discretized and the product of the resulting binary features and the Basic feature set was used for learning. Bellemare, Veness, and Bowling (2012b) later extended this idea to consider patches of color of various size within Basic blocks, rather than only single pixels. This allows the features to capture features of shape as well as color, but creates a prohibitively large feature vector, which was hashed down to a more manageable feature set.

More recently Mnih et al. (2013) took a very different approach, approximating the value function using a deep convolutional neural network. They call their algorithm Deep Q-Networks (DQN). The raw input to the network was a sequence of 4 downsampled frames, converted to greyscale. The hidden layers of the network learned a more abstract representation over this input. DQN was not evaluated on the entire suite of Atari 2600 games, but on the 7 games reported it substantially outperformed all previous approaches.

4 Basic Pairwise Relative Offset Features

Note that several of the above discussed representations share some common themes. Many acknowledge the importance of pairwise spatial relationships between objects. Some also attempt to capture a sense of spatial invariance – often the relative positions of objects to each other are more important than their absolute positions (e.g. it is more important to know that an enemy is near the player's avatar than the position of the enemy on the screen). That said, several of the existing attempts to capitalize on these observations are complicated to implement and computationally demanding. Basic Pairwise Relative Offset (B-PRO) features, the subject of this paper, aim to encode these observations in a simple, computationally light-weight feature set in the spirit of Basic or BASS features.

The B-PRO feature set includes the Basic features. It also includes a binary feature that indicates whether any pair of Basic features are on at some particular offset from each other. More specifically, for every offset (i, j) with $i \in \{-15, \ldots, 15\}$ and $j \in \{-13, \ldots, 13\}$ and every pair of NTSC colors $k_1, k_2 \in \{0, \ldots, 127\}$ there is a B-PRO feature $\phi_{(i,j),k_1,k_2}$ which is 1 if there exists any block (c, r) such that a pixel of color k_1 is contained within the block and a pixel of color k_2 is contained within the block $(c+i, r+j)$.

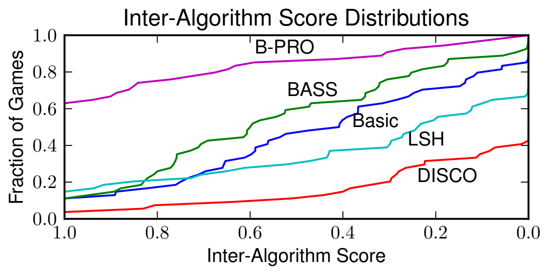

Figure 1: Inter-algorithm score distributions: the y-axis shows the fraction of games in which each method achieved the corresponding inter-algorithm score or better.

Note that the B-PRO features can be viewed as a specific form of max-pooling over the BASS features (pooling over absolute position). As such, the complexity of computing the B-PRO feature set is similar to that of BASS. However, there are ultimately many fewer B-PRO features than BASS features over the same palette. This makes it practical to use the full NTSC palette with B-PRO features, resulting in $31 \times 27 \times 128^2 = 13,713,408$ offset features.

B-PRO features are simple and computationally manageable. The experiments in the next section will demonstrate that they also capture state information that is important to success in many Atari 2600 games.

5 Experiments

B-PRO features were combined with the SARSA(λ) agent tested by Bellemare et al. (2013), using the same experimental setup for comparison's sake. Specifically, for each game, in 30 independent trials the agent was trained for 5000 episodes, after which the learned policy was evaluated in 500 test episodes. The average score over all test episodes and trials is reported. As in those experiments, $\gamma = 0.999$ and the agent used an ϵ-greedy policy with $\epsilon = 0.05$ (for both training and test episodes). The learning rate α and the eligibility trace decay rate λ were selected via a parameter sweep using 5 "training games" (ASTERIX, BEAM RIDER, FREEWAY, SEAQUEST, and SPACE INVADERS) [1].

SARSA with B-PRO features was compared to the SARSA results reported by Bellemare et al. (2013) for Basic, BASS, DISCO, and LSH features. For each game, the inter-algorithm scores (Bellemare et al. 2013) were computed: if s_i is the score for method i, the inter-algorithm score $a_i = \frac{s_i - \min_j s_j}{\max_j s_j - \min_j s_j}$. Figure 1 compares the inter-algorithm score distributions of the 5 feature sets. Note that B-PRO obtained the maximum score in the majority of games. Also note that B-PRO's curve remains higher than those of the other methods indicating that it tends to perform comparably to the best performer and rarely did the worst.

Table 1 reports the raw scores obtained by SARSA with B-PRO features as well as the results reported by for Basic, BASS, DISCO, and LSH features by Bellemare et al. (2013). B-PRO often substantially outperforms the other baselines.

[1] Parameter values $\alpha = 0.5$ and $\lambda = 0.9$ performed best from ranges $\alpha \in \{0.001, 0.01, 0.1, 0.5\}$ and $\lambda \in \{0, 0.1, 0.5, 0.9\}$.

Game	Basic	BASS	DISCO	LSH	B-PRO (std. dev.)		CAF	DQN
Asterix	862.3	859.8	754.6	987.3	**2641.0**	(568.9)	1332.0	
Beam Rider	929.4	872.7	563.0	793.6	**1501.0**	(391.1)	1742.7	4092
Freeway	11.3	16.4	12.8	15.4	**18.0**	(1.4)	19.97	
Seaquest	579.0	664.8	421.9	508.5	**715.0**	(98.8)	722.89	1705
Space Invaders	203.6	250.1	239.1	222.2	**284.0**	(48.9)	267.93	581
Alien	**939.2**	893.4	623.6	510.2	788.0	(304.7)		
Amidar	64.9	103.4	67.9	45.1	**125.0**	(33.1)		
Assault	465.8	378.4	371.7	**628.0**	478.0	(151.5)		
Asteroids	829.7	800.3	744.5	590.7	**991.0**	(106.6)		
Atlantis	**62687.0**	25375.0	20857.3	17593.9	31367.0	(4959.7)		
Bank Heist	98.8	71.1	51.4	64.6	**116.0**	(45.4)		
Battle Zone	**15534.3**	12750.8	0.0	14548.1	13221.0	(4819.8)		
Berzerk	329.2	491.3	329.0	441.0	**543.0**	(105.2)		
Bowling	28.5	**43.9**	35.2	26.1	39.0	(15.4)		
Boxing	-2.8	**15.5**	12.4	10.5	3.0	(5.1)		
Breakout	3.3	5.2	3.9	2.5	**6.0**	(1.4)		168
Carnival	2323.9	1574.2	1646.3	1147.2	**2838.0**	(1053.5)		
Centipede	7725.5	8803.8	6210.6	6161.6	**9466.0**	(1918.4)		
Chopper Command	1191.4	1581.5	1349.0	943.0	**2760.0**	(1118.2)		
Crazy Climber	6303.1	7455.6	4552.9	**20453.7**	14612.0	(5195.4)		
Demon Attack	**520.5**	318.5	208.8	355.8	420.0	(282.4)		
Double Dunk	-15.8	-13.1	-23.2	-21.6	**-10.0**	(8.1)		
Elevator Action	3025.2	2377.6	4.6	3220.6	**7402.0**	(2959.1)		
Enduro	111.8	129.1	0.0	95.8	**147.0**	(32.2)		470
Fishing Derby	-92.6	-92.1	**-89.5**	-93.2	-96.0	(5.3)		
Frostbite	161.0	161.1	176.6	**216.9**	176.0	(26.0)		
Gopher	545.8	1288.3	295.7	941.8	**2412.0**	(889.9)		
Gravitar	185.3	251.1	197.4	105.9	**413.0**	(68.8)		
H.E.R.O.	6053.1	**6458.8**	2719.8	3835.8	5868.0	(2012.1)		
Ice Hockey	-13.9	-14.8	-18.9	-15.1	**-3.0**	(2.0)		
James Bond	197.3	202.8	17.3	77.1	**323.0**	(96.5)		
Journey Escape	-8441.0	-14730.7	-9392.2	-13898.9	**-6329.0**	(2696.4)		
Kangaroo	962.4	1622.1	457.9	256.4	**2175.0**	(1643.3)		
Krull	2823.3	3371.5	2350.9	2798.1	**4012.0**	(546.6)		
Kung-Fu Master	16416.2	19544.0	3207.0	8715.6	**23423.0**	(5838.5)		
Montezumas Revenge	10.7	0.1	0.0	0.1	**185.0**	(167.6)		
Ms. Pac-Man	1537.2	1691.8	999.6	1070.8	**1913.0**	(380.7)		
Name This Game	1818.9	2386.8	1951.0	2029.8	**3397.0**	(484.9)		
Pong	-19.2	**-19.0**	-19.6	-19.9	-23.0	(0.0)		20
Pooyan	800.3	1018.9	402.7	**1225.3**	1098.0	(110.9)		
Private Eye	81.9	100.7	-23.0	**684.3**	105.0	(21.7)		
Q*Bert	**613.5**	497.2	326.3	529.1	585.0	(139.6)		1952
River Raid	1708.9	1438.0	0.0	1904.3	**2079.0**	(576.3)		
Road Runner	67.7	65.2	21.4	42.0	**10992.0**	(3804.5)		
Robotank	**12.8**	10.1	9.3	10.8	12.0	(2.4)		
Star Gunner	850.2	**1069.5**	1002.2	722.9	928.0	(125.5)		
Tennis	-0.2	**-0.1**	**-0.1**	**-0.1**	-1.0	(0.0)		
Time Pilot	1728.2	2299.5	0.0	**2429.2**	2159.0	(901.0)		
Tutankham	40.7	52.6	0.0	85.2	**100.0**	(30.8)		
Up and Down	3532.7	3351.0	2473.4	2475.1	**4087.0**	(1071.5)		
Venture	0.0	**66.0**	0.0	0.0	62.0	(235.0)		
Video Pinball	15046.8	12574.2	10779.5	9813.9	**18366.0**	(3072.4)		
Wizard of Wor	1768.8	1981.3	935.6	945.5	**2417.0**	(799.6)		
Zaxxon	1392.0	2069.1	69.8	**3365.1**	2909.0	(1172.9)		
Times Best	6	6.33	1.33	7.33	**33**			

Table 1: Control performance of in Atari 2600 games. Bold indicates the best mean score in the first five columns.

The most dramatic individual comparison is in the game ROAD RUNNER in which B-PRO features yield an average 10,992 points and the second best result is 67.7 points. A notable failure of B-PRO is in PONG where it actually achieves the *worst* performance, despite the fact that the most important feature on the screen is the offset between the agent's paddle and the ball. In this game the score display matches the colors of the two paddles. Since B-PRO ignores absolute position, it may be that color confusion between the score and the paddle is enough to prevent effective learning.

Finally, the reported results for Contingency Aware Features (CAF) (Bellemare, Veness, and Bowling 2012a) and DQN (Mnih et al. 2013) are included for reference. Note that results for CAF and DQN were not reported for all games and were generated under different experimental protocols and are therefore not directly comparable to the B-PRO results. Nevertheless, it can be noted that, even though CAF was given more training data per trial (10,000 episodes as opposed to 5,000), SARSA with the B-PRO features performs comparably or better in all 5 training games.

The B-PRO results do not reach the levels of performance obtained using DQN. However, note that B-PRO features are far simpler to implement and far less computationally expensive than the deep network. Also, because DQN learns its own representation, it is not clear what representational insights were key to its success. Though clearly DQN is capturing other important state information, it may be that features similar to B-PRO account for part of its success.

5.1 Unpromising Extensions

Given the positive results observed using B-PRO features, we investigated several variations on the features. Each of the following feature sets was progressively added to the feature vector and evaluated in the 5 training games using the same experimental set up as above.

- Row offsets: max-pooling over column offsets. For each pair of colors and each row offset, there is a feature that indicates if any B-PRO feature with that pair of colors and that row offset is on, regardless of its column offset.
- Column offsets: the same as above but with columns.
- Sign offsets: max-pooling over offset magnitudes. For each pair of colors there are four binary features, one for each setting of the signs for the row and column offsets.
- Direction offsets: like sign offsets, but signs are treated independently. For each color pair there are four binary features indicating "above", "below", "left" and "right."

Additionally, all of these variations were evaluated with a higher resolution Basic grid (each block was 5×10 instead of 10×15). None of these variations were found to significantly improve performance in the training games.

6 Conclusions and Discussion

We introduced the B-PRO feature set for Atari 2600 games that captures pairwise, position-invariant, spatial relationships between objects on the screen. Empirically, these features allow better performance than existing baselines in the literature in a wide variety of games while retaining their simplicity and computational practicality.

One of the main lessons from these results is that there are still simple, game-independent representation ideas that can make a substantial impact on performance in Atari. B-PRO features performed better or at least comparably to other representations that use far more complicated methods to extract relative positions of objects (e.g. DISCO and CAF). This feature set is easy to implement and computationally cheap to use – we believe it would make sense for B-PRO to supplant Basic/BASS as the go-to "simple baseline" feature set for comparison or augmentation.

The impressive results obtained using DQN may tempt one to conclude that hand-coded features are no longer necessary. Certainly adaptive representations like DQN are likely to play an important role in achieving general competence in video games or other domains. However, we argue that it is still valuable to study simple, interpretable representation schemes, as doing so may give insight into which issues are key to success. Furthermore, using such insights it may be eventually be possible to achieve DQN-like results, or better, using a more light-weight architecture, or to seed something like DQN with a better base representation.

It is clear that the B-PRO features do have weaknesses and they did behave poorly in some games where they were expected to perform well (e.g. PONG). The preliminary findings regarding variations on B-PRO features suggest that additional max-pooling over B-PRO features or refining the resolution of the Basic features upon which they are built will not likely yield improvements on these results. Other possible directions for augmenting these features include generating features from more than one recent frame (e.g. to represent velocities), using shape in addition to color to detect important objects on the screen, or representing spatial relationships between 3 or more objects.

Acknowledgments

This research was supported by Alberta Innovates Technology Futures through the Alberta Innovates Centre for Machine Learning. We would also like to thank Ujjwal Das Gupta, Marlos Machado, and Srinivasan Sriram for helpful discussions and insights.

References

Bellemare, M. G.; Naddaf, Y.; Veness, J.; and Bowling, M. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research* 47:253–279.

Bellemare, M. G.; Veness, J.; and Bowling, M. 2012a. Investigating contingency awareness using atari 2600 games. In *Proceedings of the 26th Conference on Artificial Intelligence (AAAI)*, 861–871.

Bellemare, M. G.; Veness, J.; and Bowling, M. 2012b. Sketch-based linear value function approximation. In *Advances in Neural Information Processing Systems 26*, 2213–2221.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Reidmiller, M. 2013. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Sutton, R. S., and Barto, A. G. 1998. *Introduction to reinforcement learning*. MIT Press.