Fuzzy Kanerva-based Function Approximation for Reinforcement Learning

(Extended Abstract)

Cheng Wu
Northeastern University
Boston, MA, U.S.A.
cwu@ece.neu.edu

Waleed Meleis
Northeastern University
Boston, MA, U.S.A.
meleis@ece.neu.edu

ABSTRACT

Radial Basis Functions and Kanerva Coding can give poor performance when applied to large-scale multi-agent systems. In this paper, we attempt to solve a collection of predator-prey pursuit instances and argue that the poor performance is caused by frequent prototype collisions. We show that dynamic prototype allocation and adaptation can give better results by reducing these collisions. We then describe our novel approach, fuzzy Kanerva-based function approximation, that uses a fine-grained fuzzy membership grade to describe a state-action pair's adjacency with respect to each prototype. This approach completely eliminates prototype collisions. We conclude that adaptive fuzzy Kanerva Coding can significantly improve a reinforcement learner's ability to solve large-scale multi-agent problems.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search

General Terms

Algorithms, Experimentation

Keywords

Reinforcement Learning, Function Approximation, Fuzzy Logic

1. INTRODUCTION

Q-learning [5] has emerged as one of the most successful reinforcement learning strategies. However, problems with large state spaces, such as multi-agent problems, can be still be difficult to solve. A key limitation is the size of the table needed to store the state-action values, which is typically large because of high dimensionality of the state-action space, or because the state or action space is continuous.

Function approximation, which stores an approximation of the entire table, is one way to solve this problem. Many function approximation techniques exist, including coarse coding [2] and tile coding (also known as CMAC) [1], and Radial basis function networks (RBFNs) [4]. An RBFN can handle continuous state or action spaces by approximating a state-action value as a linear combination of basis functions. However, selecting the parameters for basis functions is difficult in general [5, 4]. In addition, a typical RBF feature represents information about some, but not all, dimensions of the state-action space because the computational complexity increases exponentially with the number of dimensions. Therefore RBFNs have been found to be hard to apply to continuous problems with more than $10-12$ dimensions [3].

Kanerva Coding [5] can also reduce the memory needed to store the state-action value table. In Kanerva Coding, a collection of prototype state-action pairs is selected to represent binary features. An advantage of this approach is that each feature contains information about all dimensions of a state-action space. The performance of a reinforcement learner with Kanerva Coding depends largely on the number of prototype state-action pairs and the size of the target state-action space [4]. Recent work shows that dynamically selecting prototypes can improve the performance of function approximation [4, 6]. However there have been no published studies that explain the improved performance. In addition, our experiments show that traditional Kanerva Coding still does not perform well as the state-action space increases, even using dynamic prototype allocation.

2. PROTOTYPE COLLISIONS

In Kanerva Coding, a collection of k *prototype state-action pairs*, (prototypes) is selected. A state-action pair s and a prototype p_i are said to be *adjacent* if their bit-wise representations differ by no more than a threshold number of bits. We define the *membership grade* $\mu_i(s)$ of s with respect to p_i to be equal 1 if s is adjacent to p_i, and 0 otherwise. A state-action pair's *membership vector* consists of its membership grades with respect to all prototypes. A value $\theta(i)$ is maintained for the ith feature, and $\hat{Q}(s)$, an approximation of the value of a state-action pair s is then the sum of the θ values of the adjacent prototypes, that is $\hat{Q}(s) = \sum_i \theta(i)\mu_i(s)$.

When two state-action pairs, s_i and s_j have the same membership vector, that is, the same membership grades over all prototypes, a prototype *collision* is said to have taken place. Kanerva Coding works best when each state-action pair is adjacent to a unique membership vector. If prototypes are not well distributed across the state-action space, many state-action pairs will either not be adjacent to any prototypes, or adjacent to identical sets of prototypes. Such prototype collisions reduce the quality of the results because the solver can not distinguish distinct state-actions

Cite as: Fuzzy Kanerva-based Function Approximation for Reinforcement Learning, (Extended Abstract), Cheng Wu, Cheng Wu, *Proc. of 8th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2009),* Decker, Sichman, Sierra and Castelfranchi (eds.), May, 10–15, 2009, Budapest, Hungary, pp. 1257–1258

Table 1: The average fraction of test instances solved by traditional and adaptive Kanerva Coding.

# of	Traditional (%)			Adaptive (%)		
Prot.	8x8	16x16	32x32	8x8	16x16	32x32
300	57.2	28.5	7.9	81.3	49.6	23.3
600	75.0	42.3	22.3	98.9	82.4	37.0
1000	90.9	50.3	32.1	99.2	94.5	62.8
1500	91.4	59.1	36.6	99.3	95.7	77.6
2500	93.5	82.3	43.2	99.5	96.1	92.4

pairs, and the estimates of their Q-values will be equal.

However, it is difficult to generate such a set of well-distributed prototypes for several reasons: the space of possible subsets is very large, and the state-action pairs encountered by the solver depend on the specific problem instance being solved. Dynamic prototype allocation and adaptation removes unnecessary prototypes and adds new prototypes that cover parts of the state-action space that are frequently visited . In this way, prototypes can be adaptively adjusted to minimize prototype collisions for the specific problem.

We evaluate the effect of prototypes collision by applying Q-learning with traditional and adaptive Kanerva-Coding to solve the predator-prey pursuit instances. The details of the experiment design can been found in our pervious work [6]. Table 1 shows that as the size of the grid increases from 8 to 32, the fraction of test instances solved decreases sharply using both traditional and adaptive Kanerva-based function approximation with different number of prototypes.

3. ADAPTIVE FUZZY KANERVA CODING

A more flexible and powerful approach is to allow a state-action pair to update θ values of all prototypes, not only neighboring prototypes. Instead of being binary values, membership grades vary continuously between 0 and 1. Such fuzzy membership grades are larger for closer prototypes and smaller for more distant ones. Since prototype collisions occur only when two state-action pairs have same real values in all elements of their membership vectors, collisions are less likely.

In the fuzzy Kanerva Coding, the membership grade is defined as follows. Given a state-action pair s, the ith prototype p_i, and a constant variance σ^2, the membership grade of s with respect to p_i is $\mu_i = e^{-\frac{||s-p_i||^2}{2\sigma^2}}$, where $||s-p_i||$ represents the bit difference between s and p_i. A value $\theta(i)$ is maintained for the ith prototype and an approximation $\hat{Q}(s)$ of the value of a state-action pair is $\hat{Q}(s) = \sum_i \theta(i)\mu_i(s)$. The effect of an update $\Delta\theta$ to a prototype's θ-value is now a continuous function of the bit difference $||s - p_i||$.

In adaptive Kanerva Coding, prototypes are updated using their visit frequencies. In fuzzy Kanerva Coding, the visit frequency of each prototype is identical, so we instead use membership grades.The probability $p_{update}(s)$ that a state-action pair s with cumulated membership grade $m(s)$ is chosen as a prototype is $p_{update}(s) = \lambda e^{-\lambda m(s)}$, where λ is a parameter that can vary from 0 to 1. In this mechanism, prototypes that are weakly adjacent to frequently-visited state-action pairs tend to be probabilistically replaced by prootypes that are strongly adjacent.

Our adaptive fuzzy Kanerva Coding algorithm begins by initializing parameters and repeatedly executes Q-learning with fuzzy Kanerva Coding. The algorithm computes fuzzy membership grades for all state-action pairs with respect to all prototypes. Current prototypes are then periodically replaced probabilistically with state-action pairs with the highest accumulated membership grades.

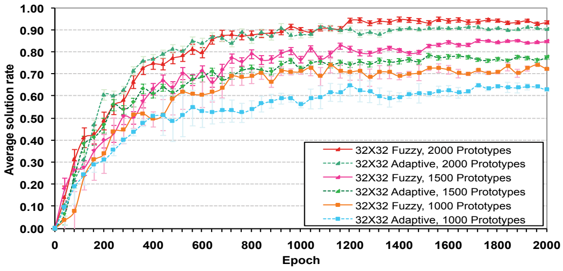

Figure 1: Average solution rate for adaptive fuzzy Kanerva Coding in 32x32 grid.

We evaluate the performance of adaptive fuzzy Kanerva Coding by applying Q-learning with adaptive Kanerva Coding and adaptive fuzzy Kanerva Coding with different numbers of prototypes to pursuit instances on grids of size 32x32. Figure 1 shows the average fraction of test instances solved by these two algorithms. The results show that the fuzzy algorithm greatly increases the fraction of the test instances solved over the adaptive algorithm.

4. CONCLUSION

Traditional function approximation techniques give poor performance when applied to problems with large state-action spaces. We evaluated a collection of pursuit instances and argued that poor performance is caused by frequent prototype collisions. Our fuzzy approach uses a fine-grained fuzzy membership grade to describe a state-action pair's adjacency with respect to each prototype. This approach, coupled with adaptive prototype allocation, allows the solver to distinguish membership vectors and reduce collision rate. We showed that adaptive fuzzy Kanerva Coding can significantly improve a reinforcement learner's ability to solve large-scale high dimension problems.

5. REFERENCES

[1] J. Albus. *Brains, Behavior and Robotics.* McGraw-Hill, 1981.

[2] G. Hinton. Distributed representations. *Technical Report, Department of Computer Science, Carnegie-Mellon University, Pittsburgh*, 1984.

[3] P. W. Keller, S. Mannor, and D. Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. In *Proc. of International Conference on Machine Learning*, 2006.

[4] B. Ratitch and D. Precup. Sparse distributed memories for on-line value-based reinforcement learning. In *Proc. of the European Conf. on Machine Learning*, 2004.

[5] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction.* Bradford Books, 1998.

[6] C. Wu and W. Meleis. Adaptive kanerva-based function approximation for multi-agent systems. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2008.