Adaptive Fuzzy Function Approximation for Multi-Agent Reinforcement Learning

Cheng Wu
Electrical and Computer Engineering
Northeastern University
Boston, United States
cwu@ece.neu.edu

Waleed Meleis
Electrical and Computer Engineering
Northeastern University
Boston, United States
meleis@ece.neu.edu

Abstract—**Reinforcement learning has difficulties in solving multi-agent problems because of the inefficiency of function approximation. Sparse distributed memories, which is implemented using Radial Basis Functions or Kanerva Coding, can be used to improve the efficiency. But this approach still often give poor performance when applied to large-scale multi-agent systems. In this paper, we attempt to solve a collection of instances in the predator-prey pursuit domain and argue that the poor performance that we observe 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 further show that prototype density varies widely across the state-action space and that this variation causes prototypes' receptive fields to be unevenly distributed. This distribution limits the ability of fuzzy Kanerva Coding to achieve better results. We demonstrate that another advantage of fuzzy Kanerva Coding is that it allows prototypes to tune their receptive fields for a target application. We conclude that fuzzy Kanerva Coding with prototype tuning and adaptation can significantly improve a reinforcement learner's ability to solve large-scale multi-agent problems.**

Keywords-**Reinforcement Learning, Function Approximation, Sparse Distributed Memory, Fuzzy Logic**

I. Introduction

Reinforcement learning [1] is a useful machine learning strategy for autonomous agents which interact with unknown environments with the objective of maximizing cumulative reward. Q-learning [2] has emerged as one of the most successful reinforcement learning strategies. The algorithm works by combining state space exploration and exploitation to learn the value of each state-action pair. Through repeated trials, the estimates of the values of each state-action pair can gradually converge to the true value, and these can be used to guide the agent to maximize its reward. Under certain limited conditions, Q-learning has been shown to always converge to an optimal policy.

However, problems with large state spaces, such as multi-agent problems, can be hard to solve. A key limitation on the effectiveness of Q-learning for solving such problems is the size of the table needed to store the state-action values. The requirement that an estimated value be stored for every state-action pair limits the size and complexity of the learning problems that can be solved. The Q-learning table is typically large because of the high dimensionality of the state-action space, or because the state or action space is continuous.

Function approximation [3], which stores an approximation of the entire table, is one way to solve this problem. Many function approximation techniques exist, including coarse coding [4] and tile coding [5] (also known as CMAC [2]), and there are guarantees on their effectiveness in some cases. Coarse coding, for example, uses a collection of overlapping regions within the state-action space, each of which corresponds to a binary feature. In tile coding, a special case of coarse coding, tilings partition the state-action space [6]. The receptive field of each binary feature corresponds to a tile. A limitation of these techniques is that they cannot handle continuous state-action spaces with high dimensionality [1].

Radial basis function networks (RBFNs) handle continuous state or action spaces by approximating a state-action value as a linear combination of basis functions. These basis functions represent features of the state-action space. Unlike binary features, RBFNs produce functions that vary smoothly and are differentiable. A Q-value in coarse coding and CMAC is constant across a feature's receptive field and falls sharply to 0 at the boundary, while the Q-value in an RBFN is largest at the center of a feature and then drops off gradually away from the center. This allows RBFNs to approximate a continuous state-action space more precisely.

However, two obstacles limit the usefulness of RBFNs. First, selecting the parameters for basis functions is difficult in general [1], [7], [8]. In an RBFN, the coefficients used to combine basis functions, and the parameters of the basis functions themselves such as the center and width, must be learned by training the solver using a large number of test instances. This process is complex, and manual tuning of the parameters before learning is often necessary. Second, 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. RBFNs have been found to be hard to apply to continuous problems with more than $10 - 12$ dimensions because of the incomplete information from the basic functions [9], [10].

Sparse Distributed Memories (SDM) [11] can also be used to reduce the amount of memory needed to store the state-action value table. This approach applied to reinforcement learning, also called Kanerva Coding [1], represents a function approximation technique that is particularly well-suited to problem domains with high dimensionality. 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.

Several researchers have investigated the use of Sparse Distributed Memories with reinforcement learning. Their results shows that 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 [12], [8]. Recent work shows that dynamically and adaptively selecting prototypes can refine the representation of state-action value functions and increase the efficiency of function approximation [8], [13]. However there have been no published studies that clearly explain the improved performance achieved using dynamic prototype allocation and adaptation. In addition, our experiments show that a traditional Kanerva-based reinforcement learner does not perform well as the state-action space increases, even using dynamic prototype allocation and adaptation.

In this paper, we show that prototype collisions are the key factor that reduces the efficiency of a Kanerva-based reinforcement learner, and that dynamic prototype allocation and adaptation improves performance by reducing these collisions. We then describe a more flexible approach by introducing prototypes with fuzzy receptive fields which can make prototype collisions less likely. In contrast to traditional RBFNs, our approach implements basis functions using prototypes from Kanerva Coding. Dynamic prototype allocation and adaptation is used to reassign the receptive fields of prototypes to the target domain of a specific application. We also employ the technique of Maximum Likelihood Estimation to tune the width of basis functions, allowing us to increase or decrease the resolutions of receptive fields within the target domain.

The paper is organized as follows. In Section II, we define prototype collisions and explain how they reduce the performance of a reinforcement learner with Kanerva-based function approximation. In Section III, we describe our reinforcement learning algorithm with fuzzy Kanerva Coding. In Section IV, we describe our prototype tuning algorithm. We discuss the relationship between RBFNs, Kanerva Coding and our Fuzzy approach in Section V and we conclude the paper in Section VI.

II. Prototype Collisions in Kanerva Coding

Kanerva Coding is an implementation of SDM in reinforcement learning. A collection of k *prototype state-action pairs*, (prototypes) is selected, each of which corresponds to a binary feature. 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. Normally, we set the threshold as 1 bits. In our previous work [14], we define the *membership grade* $\mu_i(s)$ of s with respect to p_i

$$\mu_i(s) = \begin{cases} 1 & \text{if } s \text{ is adjacent to } p_i, \\ 0 & \text{otherwise.} \end{cases}$$

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

$$\hat{Q}(s) = \sum_i \theta(i)\mu_i(s).$$

Therefore Kanerva Coding can greatly reduce the size of the value table that needs to be stored.

When two different state-action pairs, s_i and s_j, visited during reinforcement learning have the same membership vector, that is, the same membership grades over all prototypes, a prototype *collision* is said to have taken place between s_i and s_j. Kanerva Coding works best when each state-action pair is adjacent to a unique membership vector over all prototypes. 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, corresponding to identical membership vectors. Such prototype collisions reduce the quality of the results because the solver will not be able to distinguish distinct state-actions pairs, and the estimates of Q-values of such state-action pairs will be equal. We define the *collision rate* as the fraction of state-action pairs that are adjacent to no prototypes or adjacent to identical sets of prototypes.

Selecting a set of prototypes that distinguishes frequently-visit distinct state-actions pairs can improve the solver's ability to solve the problem. However, it is difficult to generate such a set of 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 during instance-based learning. In this way, prototypes can be adaptively adjusted to minimize prototype collisions for the specific problem domain.

A. Experimental Evaluation

We use Q-learning with Kanerva-Coding to solve instances of the predator-prey pursuit problem, a classic ex-

Table I
AVERAGE FRACTION OF TEST INSTANCES SOLVED BY TRADITIONAL AND ADAPTIVE KANERVA-BASED FUNCTION APPROXIMATION.

# of Prot.	Traditional (%)			Adaptive (%)		
	8x8	16x16	32x32	8x8	16x16	32x32
300	57.2	28.5	7.9	81.3	49.6	23.3
400	63.5	36.7	13.2	92.3	52.3	28.3
600	75.0	42.3	22.3	98.9	82.4	37.0
700	79.2	47.2	28.0	99.0	90.4	41.7
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
2000	93.1	75.4	40.6	99.5	95.9	90.5
2500	93.5	82.3	43.2	99.5	96.1	92.4

ample of machine learning problem [15]. This problem is challenging to solve because the size of its state-action space can be very large [16], [17], [18], [19].

In our experiments, pursuit takes place on a rectangular grid with open cells and closed blocks. The n closed blocks are distributed randomly. Each open cell in the grid represents a state that the agent may occupy. The agent is randomly placed in a starting cell. The problem is played in a sequence of time periods. In each time period, the agent can move to a neighboring open cell one horizontal or vertical step from its current location, or it can remain in its current cell. The agent receives a reward of 1 when it reaches the goal cell, and receives a reward of 0 in every other cell. The agent attempts to reach a fixed goal cell.

We use Q-learning with traditional and adaptive Kanerva-based function approximation. Traditional Kanerva-based function approximation follows Sutton [1]. Adaptive Kanerva-based function approximation is implemented using prototype deletion and prototype splitting. In prototype deletion, we delete prototypes with a probability equal to an exponential function of the number of visits. In prototype splitting, we create new prototypes near prototypes with the highest visit frequencies. A detailed description of prototype deletion and splitting can be found in [13].

In each epoch, we apply each learning algorithm to 40 random training instances followed by 40 random test instances. The exploration rate ϵ is set to 0.3, which we found experimentally to give the best results in our experiments. The initial learning rate α is set to 0.8, and it is decreased by a factor of 0.995 after each epoch. For every 40 epochs, we record the average fraction of test instances solved during those epochs within $2n$ moves. Each experiment is performed 3 times and we report the means and standard deviations of the recorded values. In our experiments, all runs were found to converge within 2000 epochs.

Table I shows the average fraction of test instances solved by traditional and adaptive Kanerva-based function approximation as the number of prototypes varies from 300 to 2500, and the size of the grid varies from 8 to 32. The values shown represent the final converged value of the solution rate.

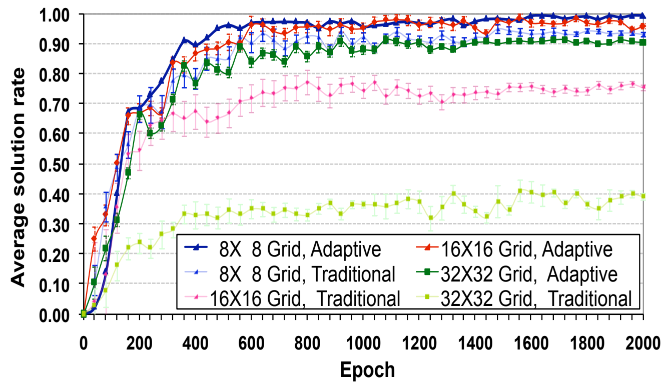

Figure 1. The fraction of test instances solved by traditional and adaptive Kanerva-based function approximation with 2000 prototypes.

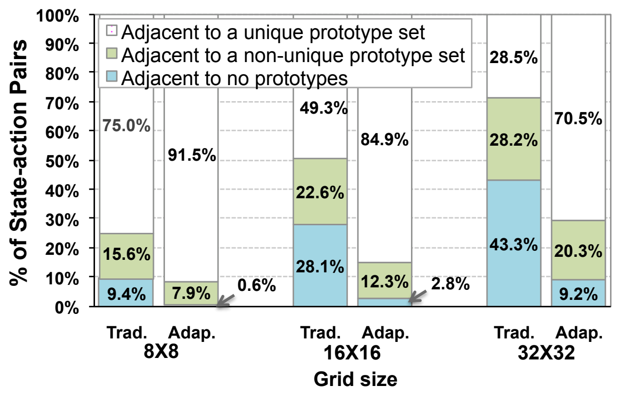

Figure 2. Prototype collisions using traditional and adaptive Kanerva-based function approximation with 2000 prototypes.

Figure 1 shows the average fraction of test instances solved by traditional and adaptive Kanerva-based function approximation with 2000 prototypes as the size of the grid varies from 8 to 32. The graph shows how the solvers converge as the number of epochs increases. Using traditional Kanerva-based function approximation with 2000 prototypes, the fraction of test instances solved decreases from 93.1% to 40.6% as the grid size increases. Using adaptive Kanerva-based function approximation with 2000 prototypes, the fraction of test instances solved decreases from 99.5% to 90.5% as the grid size increases.

These results show that as the size of the grid increases, the fraction of test instances solved decreases sharply using both traditional and adaptive Kanerva-based function approximation with different numbers of prototypes.

Figure 2 shows the fraction of state-action pairs that are adjacent to no prototypes, adjacent to identical sets of prototypes, and adjacent to a unique set of prototypes when traditional Kanerva and adaptive Kanerva with 2000 prototypes are applied to sample predator-prey instances of varying sizes.

These results show that as the collision rate increases, the performance of each algorithm decreases. For example, the average solution rate for the traditional algorithm decreases from 93.1% to 40.6% while the collision rate increases from 25.0% to 71.5% as the size of the grid increases. The average solution rate for the traditional algorithm decreases from 99.5% to 90.5% while the collision rate increases from 8.5% to 29.5% as the size of the grid increases.

The results also explain that the improved performance of the adaptive Kanerva algorithm over the traditional algorithm is due to the reduction of prototype collisions. For example, the adaptive Kanerva algorithm reduces the collision rate from 71.5% to 29.5% while the average solution rate for the adaptive algorithm increases from 40.6% to 90.5% for a grid size of 32x32.

However, the performance of the adaptive algorithm on large instances is still poor as the number of prototypes decreases, as shown in Table I. It is therefore necessary to consider a more effective approach for reducing the collision rate as the dimension of the state-action space increases.

III. Adaptive Fuzzy Kanerva Coding

A more flexible and powerful approach to function approximation is to allow a state-action pair to update θ values of all prototypes, instead of a subset of neighbor prototypes. Instead of being binary values, membership grades can vary continuously between 0 and 1 across all prototypes. Such fuzzy membership grades are larger for closer prototypes and smaller for more distant prototypes. Since prototype collisions occur only when two state-action pairs have the same real values in all elements of their membership vectors, collisions are less likely.

A. Fuzzy and Adaptive Mechanism

In the fuzzy approach to 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. Note that the membership grade of a prototype with respect to an identical state-action pair is 1, and the membership grade of a state-action pair and a completely different prototype approaches 0.

As with traditional Kanerva coding, 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 computed in the same way as before:

$$\hat{Q}(s) = \sum_i \theta(i)\mu_i(s).$$

The effect of an update $\Delta\theta$ to a prototype's θ-value is now a continuous function of the bit difference $||s-p_i||$ between the state-action pair s and the prototype p_i. The update can have a large effect on immediately adjacent prototypes, and a smaller effect on more distant prototypes.

In the adaptive Kanerva Coding algorithm described above, prototypes are updated based on their visit frequencies. In fuzzy Kanerva Coding the visit frequency of each prototype is identical, so we instead use membership grades which vary continuously from 0 to 1. 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.

Table II
Pseudo code of fuzzy Kanerva Coding

```
Main()
choose a set of prototypes $\vec{p}$ and initial their $\vec{\theta}$ value;
Repeat {for each episode}
    Generate initial state-action pair $s$ from initial state $\varsigma$
                                                   and action $a$
    Q-with-fuzzy-Kanerva($s$, $a$, $\vec{p}$, $\vec{\theta}$)
    Update-prototypes($\vec{p}$, $\vec{\theta}$)


Q-with-fuzzy-Kanerva($s$, $a$, $\vec{p}$, $\vec{\theta}$)
Repeat {for each step of episode}
    Take action $a$, observe reward $r$, and get next state $\varsigma$'
    $\vec{\mu}(s) = e^{\left(-\frac{||s-\vec{p}||^2}{2\sigma^2}\right)}$
    $\hat{Q}(s) = \sum \vec{\mu}(s) * \vec{\theta}$;
    for all actions $a*$ under new state $\varsigma$'
        Generate the state-action pair $s'$ from state $\varsigma$'
                                                   and action $a*$
        $\vec{\mu}(s') = e^{\left(-\frac{||s'-\vec{p}||^2}{2\sigma^2}\right)}$
        $\hat{Q}(s) = \sum \vec{\mu}(s') * \vec{\theta}$;
    $\delta = r + \gamma * maxQ(s') - Q(s)$
    $\Delta\vec{\theta} = \alpha * \delta * \vec{\mu}(s)$
    $\vec{\theta} = \vec{\theta} + \Delta\vec{\theta}$
    $m(s) = m(s) + \vec{\mu}(s)$
    with probability $1 - \varepsilon$
        for all actions $a*$ under current state $\varsigma$
            $\hat{Q}(s) = \sum \vec{\mu}(s) * \vec{\theta}$;
            $a = argmax_a Q(s)$
        else
        $a =$ random action
until s is terminal


Update-prototypes($\vec{p}$, $\vec{\theta}$)
$\vec{p} = \phi$
Repeat {all state-action pairs $s$}
    with probability $\lambda e^{-\lambda m(s)}$
        $\vec{p} = \vec{p} \bigcup \{s\}$
until $\vec{p}$ is full
```

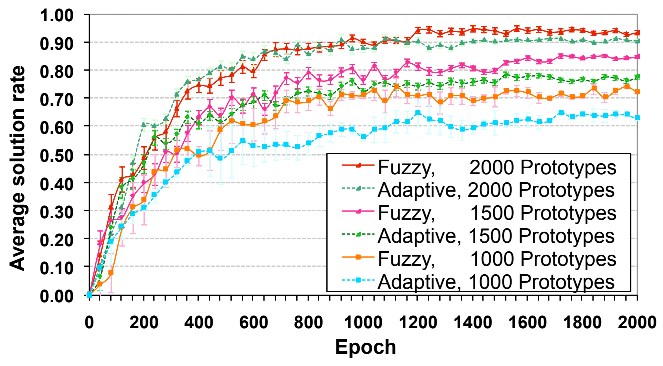

Figure 3. Average solution rate for adaptive fuzzy Kanerva Coding in 32x32 grid.

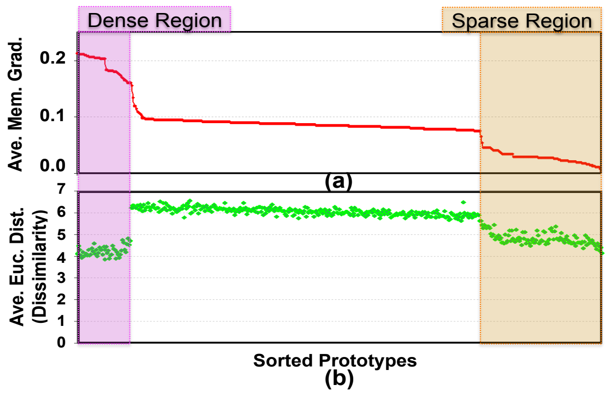

Figure 4. (a) Distribution of Average membership grades (Ave. Mem. Grad.) and (b) Distribution of Average Euclidean distance (Ave. Euc. Dist.) or prototype similarity across sorted prototypes.

B. *Adaptive Fuzzy Kanerva Coding Algorithm*

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

C. *Experimental Evaluation*

We evaluate the performance of adaptive fuzzy Kanerva Coding by applying Q-learning with adaptive Kanerva Coding and adaptive fuzzy Kanerva Coding with different number of prototypes to pursuit instances on grids of size 32x32.

Figure 3 shows the average fraction of test instances solved when adaptive Kanerva and adaptive fuzzy Kanerva-based function approximation are applied to our instances as the number of prototypes varies. The results show that the fuzzy algorithm increases the fraction of test instances solved over the adaptive Kanerva algorithm. For example, with 2000 prototypes, using the fuzzy algorithm increases the fraction of test instances solved over the adaptive algorithm from 90.5% to 93.5%. With 1000 prototypes, using the fuzzy algorithm increases the fraction of the test instances solved over the adaptive algorithm from 62.8% to 72.3%.

These results also demonstrate that the adaptive fuzzy Kanerva approach can give a larger improvement in the quality of the results when fewer prototypes are used. But the improvement is small and the solution rate is still low.

IV. Prototype Tuning

While fuzzy Kanerva Coding can give good results for our instances, the quality of the results is still poor as we decrease the number of prototypes. An explanation for these results can be found by considering the similarity of membership vectors across state-action pairs.

Figure 4(a) shows the average membership grade of each prototype with respect to all other prototypes. The prototypes are ordered by decreasing average membership grade. The results show that prototypes fall into three general regions. On the left, the prototypes have a higher average membership grade, corresponding to prototypes that are closer on average to other prototypes. On the right, prototypes have a lower average membership grade, corresponding to prototypes that are on average farther from other prototypes. The prototypes on the left are in a region of the state-action space where the distribution of prototypes is more dense, and prototypes on the right are in a region where the distribution of prototypes is more sparse. This variation in the distribution of the prototypes causes the receptive fields to be unevenly distributed across the state-action space.

State-action pairs in the dense region of the space are near to more prototypes and therefore have large membership grades that are near the top of the Gaussian response function. Similarly, state-action pairs in the sparse region of the space are far from more prototypes and therefore have small membership grades that are near the tail of the Gaussian response function. A state-action pair's membership grade is less sensitive to variations when the membership grade is near 1 or 0, as shown in Figure 5(a). Two state-action pairs in the dense region are therefore more likely to have membership vectors that are similar, and the same is true for two state-action pairs in the sparse region. This similarity between the membership vectors of state-action pairs is equivalent to the prototype collisions observed with traditional Kanerva Coding, and may have similar negative effect on the quality of the results.

Figure 4(b) shows how the similarity between prototypes varies across the state-action space. The graph shows the average Euclidean distance between each prototype and every other prototype. As expected, prototypes in the dense and sparse regions have a smaller average Euclidean distance,

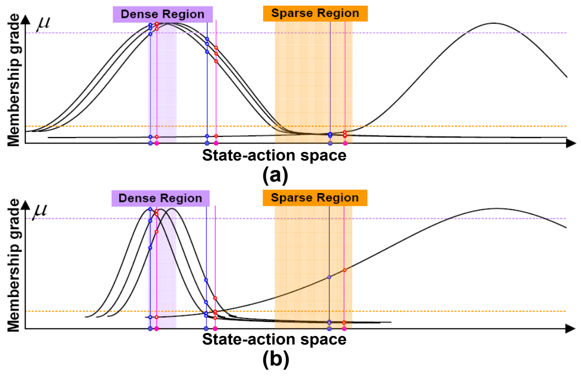

Figure 5. Illustration of the similarity of membership vectors across sparse and dense prototype regions, before (a) and after (b) prototype tuning.

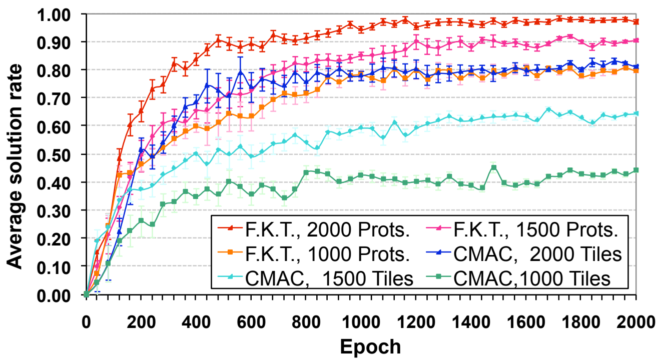

Figure 6. Average solution rate for adaptive fuzzy Kanerva Coding with Tuning (F.K.T.) and CMAC in the random-block gridworld of size 32x32.

indicating that they are more similar to one another.

We reduce the effect of similar membership vectors by adjusting the variance of the Gaussian response function used to compute membership grades. The variance is decreased in the dense region which narrows the Gaussian response function, and the variance is increased in the sparse region which broadens the Gaussian response function. This *prototype tuning* increases the sensitivity of state-action pairs' membership vectors to variations in the state-action space in these regions, as shown in Figure 5(b). We use Maximum Likelihood Estimation to compute an estimate $\hat{\sigma_i^2}$ of the variance of a prototype's membership function. Given a prototype i, we let d_{ij} be the bit difference between prototype p_i and all other prototypes p_j, where $j \neq i$, and $\bar{d}_i$ the sample mean of d_{ij}. The estimate of $\hat{\sigma_i^2}$ is

$$\hat{\sigma_i^2} = \sum_{j=1}^{n} (d_{ij} - \bar{d}_i)^2/n,$$

where n is the number of prototypes.

A. Experimental Evaluation

We evaluate our implementation of adaptive fuzzy Kanerva Coding with prototype tuning by using it to solve pursuit instances on a grid of size 32x32. As a comparison, a typical CMAC is also implemented to solve the same instances. We implement Tile Coding by representing each state-action pair as a binary vector. Each tiling corresponds to a 3-tuple of bit positions. To obtain a fair baseline for performance, we compare approximation algorithms where the number of tiles used by CMAC is similar to the number of prototypes used by adaptive fuzzy Kanerva Coding.

Figure 6 shows the average fraction of test instances solved by adaptive fuzzy Kanerva Coding with prototype tuning. In comparison to Figure 3, we can see that using prototype tuning increases the fraction of the test instances solved over the fuzzy algorithm. For example, with 2000 prototypes, using prototype tuning increases the fraction of the test instances solved over the fuzzy algorithm from 93.5% to 97.1%. With 1000 prototypes, using prototype tuning increases the fraction of the test instances solved over the fuzzy algorithm from 72.3% to 79.6%. These results demonstrate that using prototype tuning can greatly improve the efficiency of adaptive fuzzy Kanerva Coding.

In addition, Figure 6 compares the average fraction of test instances solved by CMAC and adaptive fuzzy Kanerva Coding with prototype tuning. The results show that using adaptive fuzzy Kanerva Coding with prototype tuning increases the fraction of the test instances solved over CMAC. For example, using adaptive fuzzy Kanerva Coding with prototype tuning with 2000 prototypes increases the fraction of the test instances solved over CMAC with 2000 tiles from 81.1% to 97.1%. Using adaptive fuzzy Kanerva Coding with prototype tuning with 1000 prototypes increases the fraction of the test instances solved over CMAC with 1000 tiles from 44.3% to 79.6%. These results demonstrate that using adaptive fuzzy Kanerva Coding with prototype tuning can greatly improve the quality of the results obtained.

We further evaluate our adaptive fuzzy Kanerva Coding algorithm with prototype tuning by applying it to solve the four-room problem employed by Sutton, Precup and Singh [20] and Stone and Veloso [21]. To increase the size of the state space, we extend the grid to size 32x32, shown in Figure 7. Pursuit takes place on a rectangular grid with 4 rooms. The agent can move to a neighboring open cell one horizontal or vertical step from its current location, or it can remain in its current cell. To go to another room, the agent must pass through a door. The agent is randomly placed in a starting cell, and the agent attempts to reach a fixed goal cell. The agent receives a reward of 1 when it reaches the goal cell, and receives a reward of 0 in every other cell.

Figure 8 compares the average fraction of test instances solved by CMAC and adaptive fuzzy Kanerva Coding with prototype tuning to solve the instances of the four-room problem. The results show that using adaptive fuzzy Kanerva Coding with prototype tuning increases the fraction of the

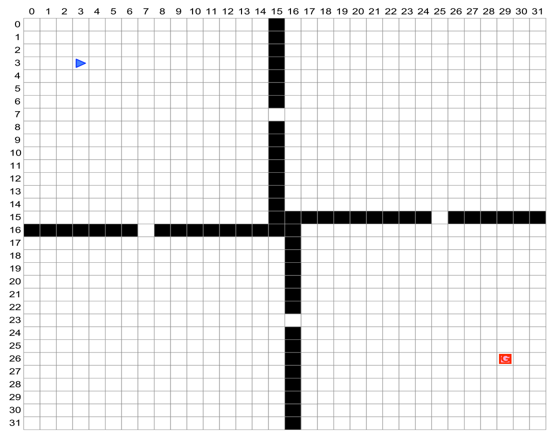

Figure 7. The four-room gridworld

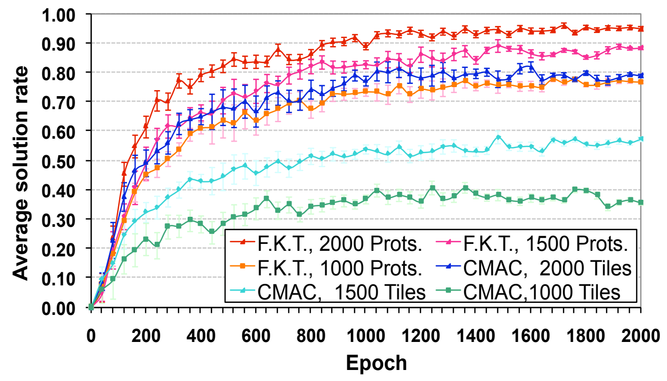

Figure 8. Average solution rate for adaptive Fuzzy Kanerva Coding with Tuning (F.K.T.) and CMAC in the four-room gridworld of size 32x32.

test instances solved over CMAC. For example, using adaptive fuzzy Kanerva Coding with prototype tuning with 2000 prototypes increases the fraction of the test instances solved over CMAC with 2000 tiles from 78.9% to 94.9%. Using adaptive fuzzy Kanerva Coding with prototype tuning with 1000 prototypes increases the fraction of the test instances solved over CMAC with 1000 tiles from 35.6% to 76.8%. These results again demonstrate that using adaptive fuzzy Kanerva Coding with prototype tuning can greatly improve the quality of the results obtained.

V. Related Work

RBFNs are a powerful technique for implementing function approximation in continuous state or action spaces. A radial basis function (RBF) is actually a real-valued function whose value depends only on the distance from its center. It also can be considered a fuzzy logic membership function, and in this sense RBFNs represent a fuzzy function approximation technique. This allows RBFNs to approximate a continuous state-action space more smoothly and precisely.

But RBFNs are the natural generalization of coarse coding with binary features to continuous features. A typical RBF feature unavoidably represents information about some, but not all, dimensions of the state-action space. RBFNs also can be interpreted as a simple single-layer artificial neural network, in which Radial Basis Functions are used as the activation functions of the network. In this case, the coefficients of each activation function, and the parameters of the activation functions themselves must be learned by training a neural network on a large number of test instances. This limits RBFNs from approximating large-scale, high-dimension state-action spaces efficiently.

Kanerva Coding provide a useful framework for investigating and modeling large-scale state-action spaces. The prototypes, which are randomly selected from the entire state-action space, represent information about all dimensions of the state-action space. When used in approximation, prototypes are represent binary features. Then the complexity of the functions that can be learned depends entirely on the number of features, and is not necessarily related to the dimensionality of the task.

However the target function approximated by binary features is coarser than one approximated continuous features. Hence our Fuzzy Kanerva-based function approximation uses prototypes with fuzzy receptive fields, which are described by real-valued membership grades. In fact, our approach can be seen as a variation of RBFNs built upon the architecture of Kanerva coding. However, in our work we clearly explain how fuzzy basis functions can improve the efficiency of function approximation, and describe a more convenient and efficient way to set and tune parameters adaptively under the architecture of Kanerva coding.

Fuzzy reinforcement learning is very different approach to handling complex state-action spaces, while avoiding direct function approximation. In our approach, we apply fuzzy logic to function approximation, but not to the learning algorithm itself. In fuzzy reinforcement learning, fuzzy rules map continuous state-action values to fuzzy actions, which themselves produce crisp actions. Examples of this approach include fuzzy Q-learning [22], [23], [24], [25] and fuzzy Sarsa [26]. Most fuzzy approaches require complex fuzzy rules, which can limit their usefulness for modeling complex systems. Furthermore, fuzzy inference on value functions themselves cannot reduce the size of the table needed to store the state-action values, and cannot handle large-scale state-action spaces with high dimensionality.

VI. Conclusion

Traditional function approximation techniques can give poor performance when applied to the problems with large state-action spaces. In this paper, we evaluated a collection of pursuit instances of the predator-prey problem and argued that this poor performance is caused by frequent prototype collisions. We also showed that dynamic prototype allocation and adaptation can reduce these collisions and give better results. However the collision rate remained quite high and the performance was still poor for large-scale instances. It was therefore necessary to consider a more effective approach for reducing the collision rate as the dimension of the state-action space increases.

Our new fuzzy approach to Kanerva-based function approximation 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 the collision rate. Our adaptive fuzzy Kanerva approach gives better performance than the pure adaptive Kanerva algorithm. We then showed that prototype density varies widely across the state-action space, causing prototypes' receptive fields to be unevenly distributed across the state-action space. State-action pairs in dense or sparse regions of the space are more likely to have similar membership vectors which limits the performance of a reinforcement learner based on Kanerva Coding. Our fuzzy framework for Kanerva-based function approximation allows us to tune the prototype receptive fields to balance the effects of prototype density variations, further increasing the fraction of test instances solved using this approach. We conclude that adaptive fuzzy Kanerva Coding with prototype tuning can significantly improve a reinforcement learner's ability to solve large-scale high dimension problems.

References

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

[2] C. Watkins, "Learning from delayed rewards," *Ph.D thesis, Cambridge Univeristy, Cambridge, England*, 1989.

[3] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Proc. of the 12th Intl. Conf. on Machine Learning*. Morgan Kaufmann, 1995.

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

[5] J. Albus, *Brains, Behaviour, and Robotics*. McGraw-Hill, 1981.

[6] R. Sutton, "Generalization in reinforcement learning: Successful examples using sparse coarse coding." in *Proc. of Conf. on Neural Information Processing Systems*, 1995.

[7] G. J. Gordon, "Stable function approximation in dynamic programming," in *Proc. of Intl. Conf. on Machine Learning*, 1995.

[8] 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.

[9] R. Munos and A. Moore, "Variable resolution discretization in optimal control." *Machine Learning*, 2002.

[10] 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.

[11] P. Kanerva, *Sparse Distributed Memory*. MIT Press, 1988.

[12] K. Kostiadis and H. Hu, "Kabage-rl: kanerva-based generalisation and reinforcement learning for possession football," in *Proc. of Intl. Conf. on Intelligent Robots and Systems*, 2001.

[13] C. Wu and W. Meleis, "Adaptive kanerva-based function approximation for multi-agent systems." in *Proc. of 7th Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 2008.

[14] ——, "Fuzzy kanerva-based function approximation for reinforcement learning." in *Proc. Of 8th Intl. Conf. on Autonomous Agents and Multiagent Systens (AAMAS)*, 2009.

[15] M. Benda, V. Jagannathan, and R. Rodhiawalla, "On optimal cooperation of knowledge sources," *Technical Report, Boeing Computer Services*, 1985.

[16] T. Haynes and S. Sen, "The evolution of multiagent coordination strategies," *Adaptive Behavior*, 1997.

[17] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative learning," in *Readings in Agents*. Morgan Kaufmann, 1997, pp. 487–494.

[18] M. Adler, H. Racke, N. Sivadasan, C. Sohler, and B. Vocking, "Randomized pursuit-evasion in graphs," in *Proc. of the Intl. Colloq. on Automata, Languages and Programming*, 2002.

[19] V. Isler, S. Kannan, and S. Khanna, "Randomized pursuit-evasion with local visibility," *SIAM Journal on Discrete Mathematics*, vol. 20, no. 1, pp. 26–41, 2006.

[20] R. Sutton, D. Precup, and S. Singh, "Between mdps and semi-mdps: A framework for temporal abstration in reinforcement learning." in *Artificial Intelligence, 112(1-2):181-211*, 1999.

[21] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[22] P. Glorennec, "Fuzzy q-learning and dynamical fuzzy q-learning," in *IEEE Intl. Conf. on Fuzzy Systems*, 1994.

[23] T. Iloriuchi, A. Fujino, Katai, and T. Sawaragi, "Based q-learning with continous state and actions," in *IEEE International Conference on Fuzzy Systems*, 1996.

[24] M. Appl and W. Brauer, "Fuzzy model-based reinforcement learning," in *Advances in Computational Intelligence and Learning: Methods and Applications*, 2002.

[25] T. Nakashima, M.Udo, and H. Ishibuchi, "Implementation of fuzzy q-learning for a soccer agent," in *IEEE International Conference on Fuzzy Systems*, 2003.

[26] L. Tokarchuk, J. Bigham, and L. Cuthbert, "Fuzzy sarsa: An approach to fuzzifying sarsa learning." in *Intl. Conf. on Computational Intelligence for Modelling, Control and Automation*, 2004.