# Comparison of CMACs and Radial Basis Functions for Local Function Approximators in Reinforcement Learning

R. Matthew Kretchmar
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
kretchma@cs.colostate.edu

Charles W. Anderson
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
anderson@cs.colostate.edu

## Abstract

CMACs and Radial Basis Functions are often used in reinforcement learning to learn value function approximations having local generalization properties. We examine the similarities and differences between CMACs, RBFs and normalized RBFs and compare the performance of Q-learning with each representation applied to the mountain car problem. We discuss ongoing research efforts to exploit the flexibility of adaptive units to better represent the local characteristics of the state space.

## 1 INTRODUCTION

Despite the recent rise of interest in reinforcement learning applied to control problems, successful implementation is still more of an art than a science. This is especially true when selecting and optimizing a function approximator. Sutton (1996) demonstrated the effectiveness of the CMAC function approximator on several control problems.

In this paper, we reconsider the CMAC and two other local function approximators – the radial basis function and the normalized radial basis function. We compare their relative strengths and weaknesses in Section 2. We address the task of choosing an appropriate local function approximator and then optimizing that choice for the control problem at hand. In Section 3, we apply each of the three local function approximators to the mountain car problem and find that the continuous function approximators using the radial basis functions provide better performance. We conclude, in Section 4, with a discussion of ongoing research efforts with the continuous function approximators to distribute their resources according to the local characteristics of the function to be learned.

## 2 Local Function Approximators

The cerebellar model articulation controller (CMAC) is essentially a complex table lookup mechanism (Miller, et al., 1990). The state space is partitioned into a grid, or tiling. Each tiling divides the space into $b$ boxes along each dimension. Commonly, there are many of these tilings overlaying the state space, each one shifted slightly from the others. With $t$ tilings, the total number of boxes (also called resources) is $tb^d$ where $d$ is the state space dimensionality. For Q-learning, each resource stores Q-values to be used in computing an estimate of the Q-function over the state space. In the general CMAC design, the tiles would be stored in a hash table with each table entry associated with one set of Q-values that would apply to all tiles in that entry. Thus, in this general form, a CMAC is not necessarily a local representation.

Given a fixed number of resources, there is a trade-off between the number of tilings and the number of boxes per tiling. Notice that decreasing the number of boxes per tiling (having fewer boxes and hence fewer partitions per dimension) implies that each box must be bigger to span the entire state space. We refer to this as an increase in *generalization* because each box generalizes across more of the state space.

The radial basis function (RBF) is an alternative local function approximator. Whereas the CMAC representation scheme discretized the state space, RBFs cover the state space in a continuous manner: the CMAC tile's Q-value is constant across its entire domain and falls sharply to 0 at the tile boundary, while the RBF Q-value is maximal at the center of the RBF and then drops off gradually away from the center. The RBF's gaussian shape is typically computed by:

$$RBF(x) = w\, e^{-\frac{||c-x||^2}{\sigma^2}},$$

where $c$ is the center of the RBF, $w$ is the weight, or height, that determines the maximum value of the RBF, and $\sigma$ effectively determines the width of the RBF.

The peculiarities of the RBF become evident as we combine several of them and alter their parameters. In Figure 1a, we see 4 RBFs with increasing heights. The width, $\sigma$, for these RBFs is 1 after normalizing by dividing by the separation between adjacent centers. Figure 1b shows the sum of these 4 RBFs – this is the function which is represented by these RBFS. We see a smooth ramp-like function in the middle, but the curve drops off dramatically near the edges. Another quirk of the RBF is its behavior as the width decreases. In Figure 1c and d, we see the sum of the same 4 RBFs with 3/8 of the previous width. Not only do we still have the drop-off at the edges, but the interior of the function now has several humps which may be an unwanted "feature" of the RBF if we wish to approximate a smooth Q-function.

The normalized radial basis function (NRBF) is an adaptation of the basic RBF unit that has not seen much attention in reinforcement learning literature. The NRBF differs from the RBF by normalizing by the sum of all RBF activations:

$$NRBF_j(x) = w_j \, \frac{e^{-\frac{||c_j - x||^2}{\sigma_j^2}}}{\sum_i e^{-\frac{||c_i - x||^2}{\sigma_i^2}}}$$

With widths of 1 or more, the NRBF behaves similarly to its non-normalized cousin. Shown in figure 1e are the 4 NRBFs with their activations normalized by the sum of the activations. Added together they produce a curve similar to that of the RBF except for the support at the edges. As we approach an edge, the activation for the NRBF at the edge is much larger than the activation for the other NRBFs. This closest NRBF dominates the term in the denominator which causes its weight (height) to be distributed to the state-space boundary.

The behavior of "dominating an area" is further emphasized as we decrease the widths. When we decrease the widths, each unit dominates its local area and spreads its weight value across its region. This behavior causes a step-like function as shown in Figure 1g and h. Thus the NRBF eliminates two undesirable side-effects of the RBF. At the edges of the state space, the NRBF avoids the sharp drop-off of the RBF by propagating the nearest NRBF weight to the boundary. Also, as the widths narrow, the NRBF avoids the large variations of the RBF. The amount of generalization of the NRBF is obviously not related to their widths in the same way as it is for the RBF. For NRBF, the width has less of an effect on the size of an NRBF function's area of support.

# 3 Experimental Results for the Mountain Car Problem

For the experiments in this paper we use the mountain car problem in Sutton(1996). A car is located in a deep valley. The car must drive out of the valley to the right (the goal state) but does not have enough power to drive straight up the hill; it must rock back and forth to gain enough momentum before it can drive up the steep hill to the right.

The two dimensional state space is defined by the car's position, $p = [-1.2, 0.5]$, and the car's velocity, $v = [-0.07, 0.07]$. The topology of the hill is governed by $\sin(3p)$. The state transition is dictated by the following equations:

$$\begin{aligned} v_{t+1} &= \text{bound}[v_t + 0.001 a_t - g \cos(3p_t)] \\ p_{t+1} &= \text{bound}[p_t + v_{t+1}] \end{aligned}$$

where $a_t$ is the action ($-1$= push left, 0 = coast, +1 = push right) and $g$ is the gravitational constant ( $-0.0025$ ). There is a $-1$ reward provided by the environment at each time step except at the goal where the reward is 0. To maximize the reward, the controller will learn the quickest way to drive the car out of the valley from any given starting state.

We first determine the best width parameter for each different local representation scheme. Having optimized each scheme, we then compare the local function approximators to determine which provides the best control for the mountain car task. We fix the number of computational units ( boxes, RBFs, or NRBFs ) to be 100.

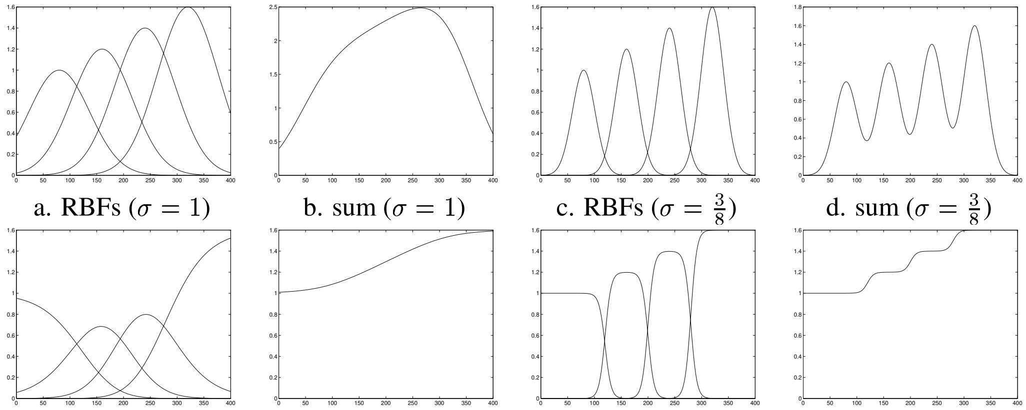
At the start of each experiment, we initialized all Q-values to 0. We ran the mountain car task on 3,000 different trials starting from random points in the state space. This entire process was repeated 10 times and the steps-to-goal was averaged over the 10 runs.

For the CMAC we test four different width configurations (tilings): 2 7x7 tilings, 4 5x5 tilings, 11 3x3 tilings, and 25 2x2 tilings. In Figure 2 we see that the CMACs with more generalization (25 2x2 tilings and 11 3x3 tilings) learned the fastest and provided the best final performance of 68 steps to goal. For the RBF, we performed the same set of experiments using widths of 5, 2, 1, and 0.5. Again, as shown in Figure 2, the RBF with the most generalization performs best; the RBFs of width 5 learned the quickest and achieved a steady state control performance of 56 steps-to-goal. However, in the NRBF case, the units with the smallest width (and least generalization) performed the best. While widths of 1/2, 1, and 2 all learned equally quickly, the width of 1/2 resulted in the best steps-to-goal measurement of 56.

These results demonstrate two important trends. First, there is a general tendency with CMAC and RBF for increased generalization to provide better performance. The NRBF function approximator violates this trend, illustrating that the relationship between the width and the generalization properties does differ significantly between RBF and NRBF approximators. Secondly, we notice that RBF and NRBF outperformed CMAC in steady-state control performance. Their continuous nature enabled them to learn the Q-function more accurately.

# 4 ADAPTIVE UNITS

In the analysis and experiments of the previous sections, we did not utilize the adaptive capabilities of the RBF and NRBF representations. Here we review ongoing research in which



a. RBFs ($\sigma = 1$) b. sum ($\sigma = 1$) c. RBFs ($\sigma = \frac{3}{8}$) d. sum ($\sigma = \frac{3}{8}$)

e. NRBFs ($\sigma = 1$) f. sum ($\sigma = 1$) g. NRBFs ($\sigma = \frac{3}{8}$) h. sum ($\sigma = \frac{3}{8}$)

Figure 1: Examples of RBFs (top row) and NRBFs (bottom row) and the functions they represent.

we adapt the centers and widths of the NRBFs to better suit the local characteristics of the Q-function.

One obvious method is to backpropagate the temporal-difference error (as was done to update the weights in the previous experiments). This yields the following update equations for the NRBF case with a single input dimension:

$$
\begin{aligned}
\epsilon &= r + \max_{a'} Q(x', a') - Q(x, a) \\
e_i &= e^{-\dfrac{||c_i - x||^2}{\sigma_i^2}} \\
\Delta w_i &= \eta_h \epsilon \frac{e_i}{\Sigma e_k} \\
\Delta \sigma_i &= \eta_\sigma \epsilon \frac{e_i}{\Sigma e_k}(w_i - Q)\frac{(c_i - x)^2}{\sigma_i^2} \\
\Delta c_i &= \eta_c \epsilon \frac{e_i}{\Sigma e_k}(w_i - Q)\frac{c_i - x}{\sigma_i}
\end{aligned}
$$

where Q is the Q-value computed at state $x$, $\epsilon$ is the temporal-difference error, and $\eta$ are the learning rates. For each unit $i$, $e_i$, $w_i$, $\sigma_i$ and $c_i$ are the activation, weight, width and center.

Our research indicates that this is not particularly effective as an adaptive method. *Minimizing the temporal-difference error* is a different goal than *providing a good control policy*. In many of our experiments, we find that the NRBFs move toward locations in the state space for which the current value function produces small temporal-difference errors. This does succeed in finding a local minimum in the overall temporal-difference error, but at the expense of rendering large regions of the state space void of resources. These empty regions are not able to develop a finely tuned policy and hence suffer in control performance.

We are currently developing a number of alternatives to backpropagation of the temporal-difference error. One alternative is to augment an unsupervised approach for estimating the probability density of experienced states with a factor related to the temporal-difference error. Initial results on some simple problems show that this approach can produce much better results than static RBFs (Rittwager, 1996). In other work, we are extending the algorithms developed by Anderson (1993) which prunes the least useful radial basis functions and adds functions centered on problematic states. This involves extensions of Platt's (1991) resource allocation algorithm to reinforcement learning. We plan to investigate recent developments by Molina and Niranjan (1996) for setting the parameters of the added functions.

# 5 SUMMARY

In this paper we have examined three local function approximators as they apply to reinforcement learning control tasks. The CMAC representation partitions the state space by overlaying several shifted tilings on the state space. We discovered that having more tilings each with a fewer number of boxes increases the generalization capability. A "continuous-version" of CMACs is realized with the radial basis function. RBFs are better able to represent gradual-continuous transitions in the function to be approximated; however, RBFs do have drawbacks; they tend to drop-off dramatically at the edge of the state space and introduce unnatural waviness as their widths decrease. The normalized radial basis function is similar to the RBF except that the NRBF provides constant support at the state space edge and also demonstrates better narrow-width characteristics. When applied to the mountain-car control task, the CMAC and RBF demonstrated that increased generalization resulted in faster learning and ultimately better control. The continuous nature of the RBF and NRBF outperformed the CMAC in steady state steps-to-goal control performance. Finally, we considered using adaptive methods to update the widths and centers of the NRBF units.

## Acknowledgements

This work was supported by the National Science Foundation through grants CMS-9401249 and CISE-9422007.

## References

Anderson, C. (1993) Q-Learning with Hidden-Unit Restarting. *Advances in Neural Information Processing Systems*, volume 5, S. J. Hanson, J. D. Cowan, and C. L. Giles, eds., Morgan Kaufmann Publishers, San Mateo, CA, pp. 81–88.

Miller, W. T., F. H. Glanz, and L. G. Kraft. (1990). CMAC: An associative neural network alternative to backpropagation. *Proceedings of IEEE*, 78, pp. 1561–1567.

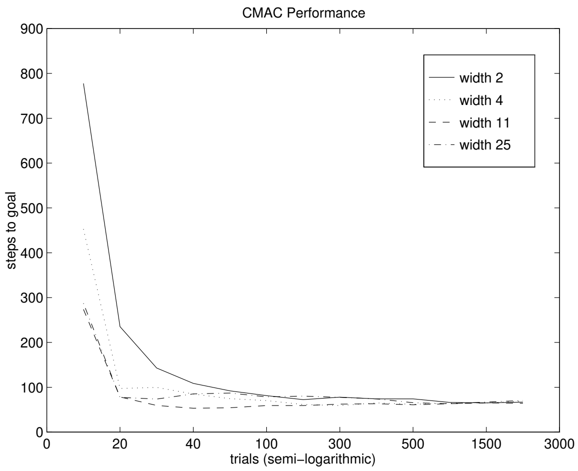
Molina, C. and Niranjan, M. (1996) Pruning with replacement on limited resource allocating networks by f-projections. *Neural Computation*, vol. 8., no. 4, pp. 855–868.

Platt, J.C. (1991) A resource-allocating network for function interpolation. *Neural Computation*, 3:213–225.

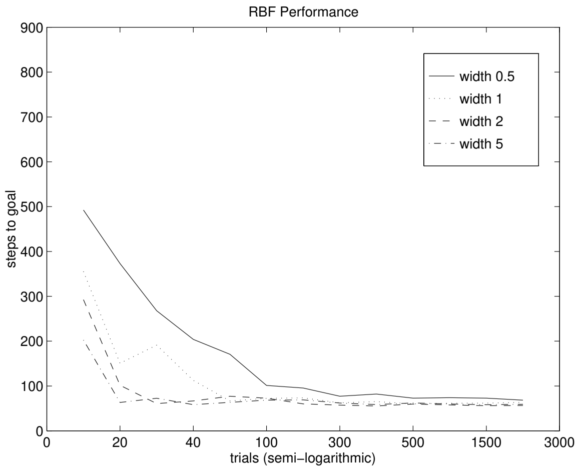
Rittwager, F. (1996) Temporal-difference based self-organization for reinforcement learning with radial basis functions. Masters Thesis, Dept. of Computer Science, Colorado State University.

Sutton, R.S. (1996) Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, Cambridge, MA: MIT Press.

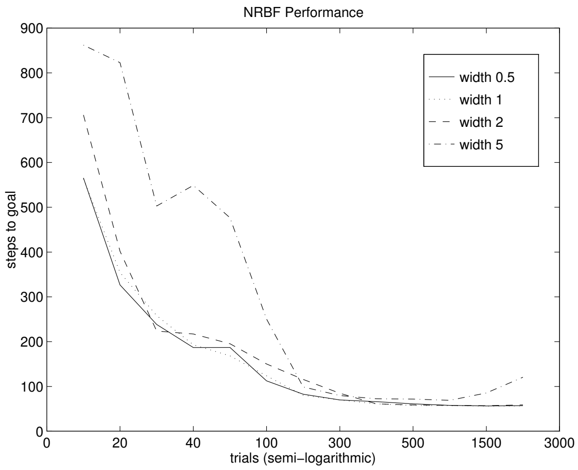
Watkins, C. J. (1989) Learning from delayed rewards. Ph.D. Thesis, Cambridge University.




a. CMAC




b. RBF




c. NRBF

Figure 2: CMAC/RBF/NRBF Mountain Car Performance