Lateral Inhibition Overcomes Limits of Temporal Difference Learning

Jacob Rafati & David C. Noelle
(jrafatiheravi@ucmerced.edu, dnoelle@ucmerced.edu)
Computational Cognitive Neuroscience Laboratory
University of California, Merced
5200 North Lake Road
Merced, CA 95343 USA

Abstract

There is growing support for Temporal Difference (TD) Learning as a formal account of the role of the midbrain dopamine system and the basal ganglia in learning from reinforcement. This account is challenged, however, by the fact that realistic implementations of TD Learning have been shown to fail on some fairly simple learning tasks — tasks well within the capabilities of humans and non-human animals. We hypothesize that such failures do not arise from natural learning systems because of the ubiquitous appearance of lateral inhibition in the cortex, producing sparse conjunctive internal representations that support the learning of predictions of future reward. We provide support for this conjecture through computational simulations that compare TD Learning systems with and without lateral inhibition, demonstrating the benefits of sparse conjunctive codes for reinforcement learning.

Keywords: reinforcement learning; lateral inhibition; sparse conjunctive codes; computational cognitive neuroscience

Introduction

Humans and non-human animals are capable of learning highly complex skills by reinforcing appropriate behaviors with reward. The midbrain dopamine system has long been implicated in reward-based learning (Schultz, Apicella, & Ljungberg, 1993), and the information processing role of dopamine in learning has been well described by a class of reinforcement learning algorithms called *Temporal Difference (TD) Learning* (Montague, Dayan, & Sejnowski, 1996; Schultz, Dayan, & Montague, 1997). While TD Learning, by itself, certainly does not explain all observed reinforcement learning phenomena, increasing evidence suggests that it is key to the brain's adaptive nature (Dayan & Niv, 2008).

Beyond empirical support for the TD Learning account of biological reinforcement learning, the power of this learning method suggests that it may be capable of explaining the neural basis of the successful learning of even fairly complex tasks (Sutton & Barto, 1998). This algorithm can learn elaborate decision making skills, such as playing the game of Backgammon at the Grand Master level (Tesauro, 1995). There are even proofs that TD Learning will converge to optimal performance, given enough experience (Dayan, 1992).

Despite these strengths, a mystery remains. There are some relatively simple reinforcement learning problems for which TD Learning has been shown to fail (Boyan & Moore, 1995). These problems arise when the space of possible sensory states of the learning agent is so large that it is intractable to store the agent's learned assessment of the *value* or *quality* of each state (i.e., its expectation of future reward, given that it is in that state) in a large look-up table. In these cases, it is necessary to encode the agent's learned *value function*, mapping from sensory state features to an expectation of future reward, using some form of function approximator. Formally, this function approximator is a parameterized equation that maps from state to value, where the parameters can be constructively optimized based on the experiences of the agent. One common function approximator is an artificial neural network, with the parameters being the connection weights in the network. Such a network, adapted using the *backpropagation of error* learning method (Rumelhart, Hinton, & Williams, 1986), was used in the previously mentioned Backgammon playing program (Tesauro, 1995). As illustrated by this program, TD Learning with an artificial neural network approximating the value function, can solve apparently complex tasks. Using a function approximator to learn the value function has the added benefit of potentially supporting generalization by including a bias toward mapping similar sensory states to similar predictions of future reward.

Surprisingly, some tasks that superficially appear very simple cannot be perfectly mastered using this method. For example, learning to navigate to a goal location in a simple two-dimensional space in which there are obstacles has been shown to pose a substantial challenge to TD Learning using a backpropagation neural network (Boyan & Moore, 1995). Note that the proofs of convergence to optimal performance depend on the agent maintaining a potentially highly discontinuous value function in the form of a large look-up table, so the use of a function approximator for the value function violates the assumptions of those formal analyses. Still, it seems unusual that this approach to learning can succeed at some difficult tasks but fail at some fairly easy tasks.

The power of TD Learning to explain biological reinforcement learning is greatly reduced by this observation. If TD Learning fails at simple tasks that are well within the reach of humans and non-human animals, then it cannot be used to explain how the dopamine system supports such learning.

In this paper, we demonstrate how incorporating a ubiquitous feature of biological neural networks into the artificial neural networks used to approximate the value function can allow TD Learning to succeed at simple tasks that have previously challenged it. Specifically, we show that the incorporation of *lateral inhibition*, producing competition between neurons so as to produce *sparse conjunctive representations*, can produce success in learning to approximate the value function using an artificial neural network, where only failure had been previously found. Thus, through computational simulation, we provide preliminary evidence that lateral inhibition in the brain may help compensate for a weakness of TD Learning, further buttressing the TD Learning account of dopamine-based reinforcement learning.

In the remainder of this paper, we initially provide some background concerning the reported failure of TD Learning on fairly simple problems. We then provide details concerning our computational simulations of TD Learning with lateral inhibition included. The results of these simulations, comparing the performance of our approach to previously examined methods, are then described. We close with a discussion of these results and ideas for future work.

Background

Consider a very simple two-dimensional "grid world" environment that remains static over time. A reinforcement learning agent in this environment may be faced with the choice, at each time step, to move a fixed distance either North, South, East, or West. On each time step, the agent receives mild negative reinforcement, until it moves to a goal location in the Northeast corner of the space, at which point the agent is relieved of negative reinforcement. Further, imagine that this environment contains "puddles" through which the agent can move, but moving into puddle locations produces extremely strong negative reinforcement. Finally, consider the case in which the agent can perfectly sense its location in the grid environment (i.e., its Cartesian coordinates), but it otherwise has no senses. This situation is illustrated in Figure 1. Equipped with TD Learning, using a function approximator to learn the value function, could such an agent learn to avoid the puddles and move rapidly to the goal location, after substantial experience in the environment?

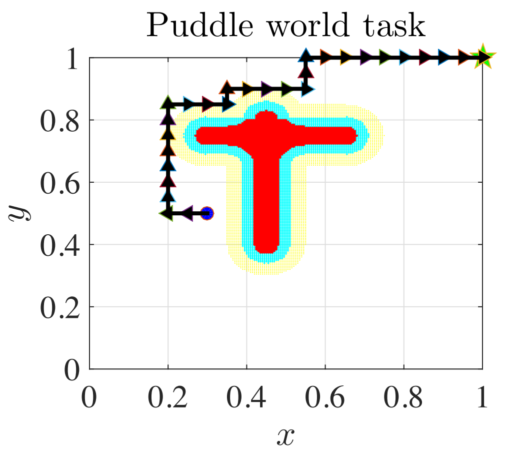

Figure 1: The agent receives -1 reward on each time step until it reaches the goal in the Northeast corner. The agent moves a distance of 0.05 either North, South, East, or West on each time step. Entering a puddle produces a reward of $(-400 \times d)$, where d is the shortest distance to a puddle edge.

Boyan and Moore (1995) provided simulation evidence suggesting that such learning is impossible. They tried a variety of different value function approximators, including a backpropagation network with a single hidden layer, but none of them converged on a good solution to the problem. Indeed, as the agent continued to explore the environment, the estimate of future reward for locations kept changing, failing to settle down to a fixed value function.

This observation suggests that the difficulty in learning this problem arises from a specific feature of reinforcement learning. In particular, the value function that the function approximator is trying to learn is, in a way, a moving target. Early in training, when the agent is unlikely to make it to the goal location, the expected future reward for a given location might be quite low. Later, if the agent has had some successes, the value for the same location might be higher. If the value function is stored in a large look-up table, then adjusting the value of one location has no influence on the values associated with other locations, allowing for small incremental changes in the value function. When using a function approximator, however, adjusting parameters (e.g., backpropagation network connection weights) for one location will likely change the value assigned to many other locations, potentially causing the un-learning of appropriate values for those other locations. This is a reasonable hypothesis for the observed lack of convergence.

In the following year, Sutton (1996) showed that this task could be learned by a TD Learning agent by hard-wiring the hidden layer units of the backpropagation network used to learn the value function to implement a fixed sparse conjunctive (coarse) code of the agent's location. The specific encoding used was one that had been previously proposed in the CMAC model of the cerebellum (Albus, 1975). Each hidden unit would become active only when the agent was in a location within a particular range of x values *and* within a particular range of y values. The conjoining of conditions on both coordinates is what made this code "conjunctive" in nature. Also, for any given location, only a small fraction of the hidden units displayed non-zero activity. This is what it means for the hidden representation to be a "sparse" code. Locations that were close to each other in the environment produced more overlap in the hidden units that were active than locations that were separated by a large distance. By ensuring that most hidden units had zero activity when connection weights were changed, this approach kept changes to the value function in one location from having a broad impact on the expected future reward at distant locations. (In the backpropagation of error learning algorithm, a connection weight is changed in proportion to the activity on the sending side of that connection, so there is no change if there is no activity being sent.) By engineering the hidden layer representation, this reinforcement learning problem was solved.

This is not a general solution, however. If the same approach was taken for another reinforcement learning problem, it is quite possible that the CMAC representation would not be appropriate. Thus, the method proposed by Sutton (1996) does not help us understand how TD Learning might flexibly learn a variety of reinforcement learning tasks. This approach requires prior knowledge of the kinds of internal representations of sensory state that are easily associated with expected future reward, and there are simple learning problems for which such prior knowledge is unavailable.

We hypothesize that the key feature of the Sutton (1996) approach is that it produces a sparse conjunctive code of the sensory state. Representations of this kind need not be fixed, however, but might be learned at the hidden layers of neural networks. Computational cognitive neuroscience models have shown that a combination of feedforward and feedback inhibition naturally produces sparse conjunctive codes over a collection of excitatory neurons (O'Reilly & Munakata, 2001). Such patterns of lateral inhibition are ubiquitous in the mammalian cortex (Kandel, Schwartz, Jessell, Siegelbaum, & Hudspeth, 2012). Importantly, networks containing such lateral inhibition can still learn to represent input information in different ways for different tasks (O'Reilly & Munakata, 2001), retaining flexibility while producing the kind of sparse conjunctive codes that may support reinforcement learning.

Simulation Methods

In order to assess our hypothesis that biasing a neural network toward learning sparse conjunctive codes for sensory state inputs will improve TD Learning when using a value function approximator, we constructed a variant of a backpropagation network with a single hidden layer in which the number of hidden units that could be highly active at any one time was restricted. Like some computational cognitive neuroscience models of lateral inhibition (O'Reilly & Munakata, 2001), we implemented this through a k-Winners-Take-All (kWTA) mechanism akin to pooled lateral inhibition. After calculating the *net input* values of hidden units based on the network inputs (i.e., the weighted sum of the inputs), we identified a single scalar amount of inhibition (i.e., negatively weighted input) that, when added to all of the hidden unit net input values, would result in the k hidden units with the highest net input values to have adjusted net input values that were positive, while all hidden units with lower net input values would be adjusted to become negative. These adjusted net input values were transformed into unit activation values using a logistic sigmoid activation function (gain of 1, offset of -1), resulting in hidden unit activation values in the range between 0.0 and 1.0, with the top k units having activations above 0.27 (due to the -1 offset) and the "losing" hidden units having activations below that value. The k parameter controlled the degree of sparseness of the hidden layer activation patterns, with low values producing more sparsity (i.e., fewer hidden units with high activations). In the simulations reported here, we set k to be 10% of the total number of hidden units.

In addition to encouraging sparse representations, this kWTA mechanism has two properties that are worthy of note. First, introducing this mechanism violates some of the assumptions needed to formally relate the backpropagation of error procedure to gradient descent in error. Thus, the connection weight changes recommended by the backpropagation procedure may slightly deviate from those which would lead to local error minimization in this network. We opted to ignore this discrepancy, however, trusting that a sufficiently small learning rate would keep these deviations small. Second, it is worth noting that this particular kWTA mechanism allows for a distributed pattern of activity over the hidden units. This provides the learning algorithm with some flexibility, allowing for a graded range of activation levels when doing so reduces network error. As connection weights from the inputs to the hidden units grow in magnitude, however, this mechanism will drive the activation of the top k hidden units closer to 1 and the others closer to 0. Indeed, an examination of the hidden layer activation patterns in the kWTA-equipped networks used in this study revealed that the k winning units consistently had activity levels close to the maximum possible value, once the learning process was complete.

Our reinforcement learning agent used this neural network, with kWTA, as an adaptive value function approximator. We used a version of TD Learning called SARSA (Sutton & Barto, 1998), which calculates a separate prediction of expected future reward for each action that might be taken from the current state. Thus, the value function can be formalized as $Q(s,a)$, where s is the current sensory state (i.e., the location of the agent expressed as an $< x, y >$ coordinate pair), and a is a considered next action (i.e., one of North, South, East, and West). The value of $Q(s,a)$ is then the expected future reward at state s when a will be the next action, and future actions will be those, at each future state, that maximize the expected future reward at that state (i.e., $argmax_a\ Q(s,a)$). The expected future reward value is discounted, so that rewards that are received soon are weighted more heavily than rewards that are received in the distant future. The amount of discounting is controlled by a discounting parameter, $\gamma \in (0,1]$, such that the expected future reward at time t is:

$$\sum_{k=0}^{m} \gamma^k \ r(t+k)$$

Here, $r(t)$ is the instantaneous reward received at time t (see below for specific values), and m is either the number of time steps remaining until the goal is reached or, if the goal is not reached, a time step maximum (such that $t+m = 80$; about twice the time needed to get from any point in the environment to any other). For these simulations, temporally distal rewards were fairly highly weighted by using a value of $\gamma = 0.99$. The backpropagation network was expected to learn an approximation of the $Q(s,a)$ function, $\hat{Q}(s,a)$, where the state, s, is provided as input to the network, and there is one output for each action, a, specifying $\hat{Q}(s,a)$ for the pair.

For the puddle world task, the x-coordinate and the y-coordinate of the current state, s, were presented to the neural network over two separate pools of input units. Note that these coordinate values were in the range $[0,1]$, as shown in Figure 1. Each pool of input units consisted of 21 units, with each unit corresponding to a coordinate location between 0 and 1, inclusive, in increments of 0.05. To encode a coordinate value for input to the network, a Gaussian distribution with a peak value of 1, a standard deviation of 0.05, and a mean equal to the given continuous coordinate value was used to calculate the activity of each of the 21 input units. For example, for a coordinate value of 0.15, the input unit corresponding to this value was set to a maximum activation of 1 and input units corresponding to coordinate values above and below 0.15 were set to activity levels that decreased with distance from 0.15 according to the Gaussian function.

The network had four output units, each corresponding to one of the four directions of motion. Between the 42 input units and the 4 output units was a layer of 220 hidden units. The hidden layer was subject to the previously described kWTA mechanism, parameterized so as to allow 10%, or 22, of the hidden units to be highly active. The hidden units used a logistic sigmoid activation function on net input values that were adjusted to allow only 22 units to be highly active at any one time. The output units used a linear activation function (i.e., their activation was equal to their net input). There was complete connectivity between the input units and the hidden units and between the hidden units and the output units, with all connection weights initialized to uniformly sampled random values in the range $[-0.05, 0.05]$.

Following the SARSA version of TD Learning, the reinforcement agent was controlled in the following way. The current location of the agent, s, was provided as input to the neural network, producing four output activation values. With a small exploration probability, ϵ, these values were ignored, and an action was selected uniformly at random from the four cardinal directions. (The value of ϵ is discussed further, below.) Otherwise, the output unit with the highest activation level determined the action, a, to be taken. The agent then moved a distance of 0.05 in the direction specified by the selected action, placing it in a new state, s'.

At this point, the agent received a reward signal, r, based on its current location, s'. This value was -1 for most of the environment, but it had a higher value, 0, at the goal location in the Northeast corner. If the agent was currently located in a puddle, the reward signal was calculated as $(-400 \times d)$, where d is the shortest distance from the current location to the edge of the puddle. Finally, the agent received a reward signal of -2 if it had just attempted to leave the square environment. Thus, the agent was "punished" for being anywhere except the goal location, and it was severely "punished" for entering a puddle. This pattern of reinforcement was selected to parallel that used in Sutton (1996).

The action selection process was then repeated at location s', determining a subsequent action, a'. Before this action was taken, however, the neural network value function approximator had its connection weights updated according to the SARSA Temporal Difference (TD) Error:

$$\delta = \left(r + \gamma \hat{Q}(s', a')\right) - \hat{Q}(s, a)$$

The TD Error, δ, was used to construct an error signal for the backpropagation network implementing the value function. The network was given the input corresponding to s, and activation was propagated through the network. Each output unit then received an error value. This error was set to zero for all output units except for the unit corresponding to the action that was taken, a. The selected action unit received an error signal equal to the TD Error, δ. These error values were then backpropagated through the network, using the standard backpropagation of error algorithm (Rumelhart et al., 1986), and connection weights were updated (with a low learning rate, α, of 0.005). This process was then begun again, starting at location s' and taking action a'.

The agent explored the puddle world environment in *episodes*. Each episode began with the agent being placed at a location within the environment sampled uniformly at random. Actions were then taken, and connection weights updated, as described above. The episode ended when the agent reached the goal location or after the maximum of 80 actions had been taken. At the beginning of a simulation, the exploration probability, ϵ, was set to a relatively high value of 0.1, and it remained at this value for much of the learning process. Once the average magnitude of δ over an episode fell below 0.2, the value of ϵ was reduced by 0.1% each time the goal location was reached. Thus, as the agent became increasingly successful at reaching the goal location, the exploration probability, ϵ, approached zero. (Annealing the exploration probability is commonly done in systems using TD Learning.) The agent continued to explore the environment, one episode after another, until the average absolute value of δ was below 0.01 and the goal location was consistently reached, or a maximum of 44,100 episodes had been completed. (This value was heuristically selected as a function of the size of the environment: $(21 \times 21) \times 100 = 44,100$.)

When this reinforcement learning process was complete, we examined both the behavior of the agent and the degree to which its value function approximations, $\hat{Q}(s, a)$, matched the correct values, $Q(s, a)$, where the correct values were determined by running SARSA to convergence while using a large look-up table to capture the value function.

Simulation Results

We compared the performance of our kWTA neural network with that produced by using a standard backpropagation network with identical parameters. We also examined the performance of a "linear" network, which had no hidden units but only complete connections from all input units directly to the four output units. Twenty simulations were conducted for each of these three neural network architectures.

Figure 3 shows the value function (plotted as $max_a\ \hat{Q}(s, a)$ for each location, s) for representative networks of each kind. Also displayed are selected actions at a grid of locations. Finally, we show learning curves displaying the episode average value of the TD Error, δ, over episodes.

In general, the linear network did not consistently learn to solve this problem, sometimes failing to reach the goal or choosing paths through puddles. The backpropagation network performed much better, but its value function approximation still contained sufficient error to produce a few poor action choices. The kWTA network, in comparison, consistently converged on good approximations of the value function, almost always resulting in optimal paths to the goal.

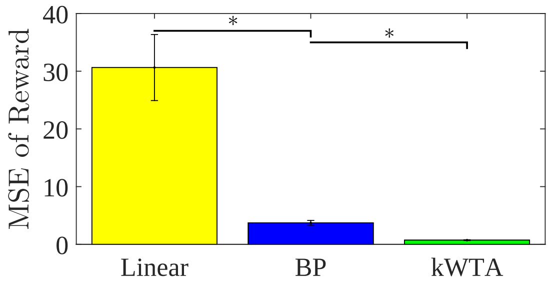

Figure 2: Averaged over 20 simulations of each network type, these columns display the mean squared deviation of accumulated reward from that of optimal performance. Error bars show one standard error of the mean.

For each network, we assessed the quality of the paths produced. For each simulation, we froze the connection weights, and we produced an episode for each possible starting location, except for locations inside of the puddles. The reward accumulated over each episode was recorded, and, for each episode that ended at the goal location, we calculated the sum squared deviation of these accumulated reward values from that produced by an optimal agent (constructed using SARSA with a look-up table for the value function). The mean of this error measure, over all successful episodes, was recorded for each of the 20 simulations run for each of the 3 network types. Figure 2 shows the means of these values, over 20 simulations. The backpropagation network had significantly less error than the linear network ($t(38) = 4.692$; $p < 0.001$), and the kWTA network had significantly less error than the standard backpropagation network ($t(38) = 7.314$; $p < 0.001$). On average, the kWTA network deviated from optimal performance by less than one reward point.

We also recorded the fraction of episodes which succeeded at reaching the goal for each simulation run. The mean rate of goal attainment, across all non-puddle starting locations, for the linear network, the backpropagation network, and the kWTA network were 93.3%, 99.0%, and 99.9%, respectively. Despite these consistently high success rates, the linear network exhibited significantly more failures than the backpropagation network ($t(38) = 2.306$; $p < 0.05$), and the backpropagation network exhibited significantly more failures than the kWTA network ($t(38) = 2.138$; $p < 0.05$).

Conclusions, Discussion, & Future Work

These simulation results demonstrate that a mechanism for learning sparse conjunctive codes for the agent's sensory state can help overcome learning problems observed when using TD Learning with a value function approximator. Artificial neural networks can be biased toward producing such sparse codes over their hidden units by including a process akin to the sort of pooled lateral inhibition that is ubiquitous in the cerebral cortex.[1] In this way, these simulation results lend preliminary support to the hypothesis that the midbrain dopamine system does, indeed, implement a form of TD Learning, and the observed problems with TD Learning do not arise in the brain due to the encoding of sensory state information in circuits that make use of lateral inhibition.

This work was prompted by the failures of TD Learning on some simple machine learning tasks, reported by Boyan and Moore (1995). It is interesting to note that our standard backpropagation network performance results are much better than the analogous results reported by those researchers. It is not clear if this discrepancy involves subtle differences in learning parameters, arises from our scheme for encoding coordinate values as inputs to the neural networks, or was caused by some other unknown factor. While our simulations demonstrate clear benefits from using the kWTA mechanism, the performance of our backpropagation networks was not as bad as what was previously reported in the literature.

We are currently extending this work in two primary directions. First, we are applying our kWTA value function approximator to other reinforcement learning problems that have posed difficulties for TD Learning. The first of these is the "mountain car" control problem, which involves a value function with difficult non-linearities (Sutton & Barto, 1998). Second, while the work reported here focuses on improving machine learning methods, we are also interested in approaches that more closely match computational cognitive neuroscience models of feedforward and feedback inhibition in the brain, as well as more biologically plausible learning procedures (O'Reilly & Munakata, 2001).

Acknowledgments

The authors thank the Computational Cognitive Neuroscience Laboratory (CCNL) at the University of California, Merced (UCM), as well as five anonymous reviewers, for their feedback. We are grateful for travel support from the UCM School of Engineering, awarded to the first author.

References

Albus, J. S. (1975). A new approach to manipulator control: The cerebellar model articulation controller CMAC. *Journal of Dynamic Systems, Meaasurement, and Control*, *97*(3), 220–227.

Boyan, J. A., & Moore, A. W. (1995). Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. S. Touretzky, & T. K. Leen (Eds.), *Advances in neural information processing systems 7* (pp. 369–376). Cambridge, MA: MIT Press.

Dayan, P. (1992). The convergence of TD(λ) for general λ. *Machine Learning*, *8*, 341–362.

[1]Of course, sparse codes can be produced in other ways, as well, such as introducing a "sparseness constraint" regularization term to the learning process (French, 1991; Hoyer, 2004).

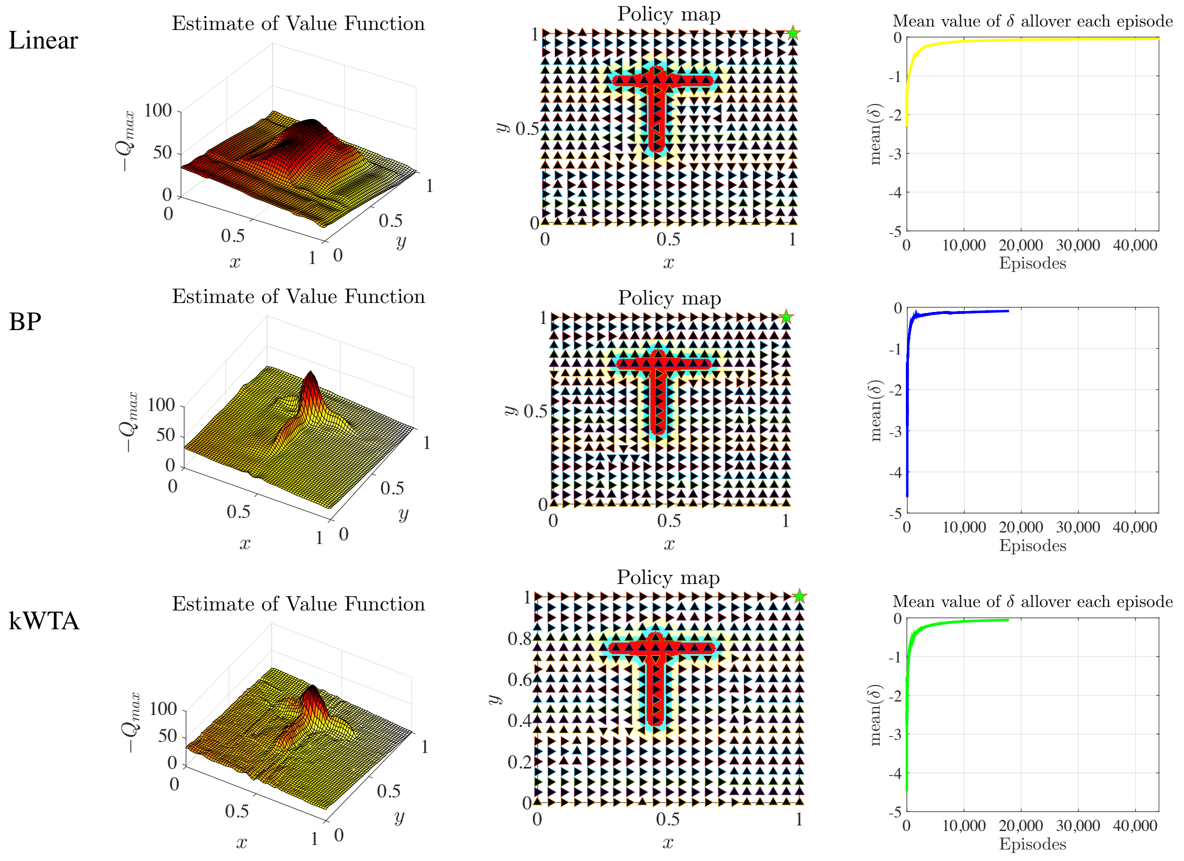

Figure 3: The performance of various learned value function approximators may be compared in terms of their success at learning the true value function, the resulting action selection policy, and the amount of experience in the environment needed to learn. The approximated value functions, expressed as $max_a\ \hat{Q}(s,a)$ for each location, s, appears on the left. The actions selected at a grid of locations is shown in the middle column. The learning curve, showing the TD Error over learning episodes, is shown on the right. The rows display results for representative neural networks of the three kinds explored: linear, backpropagation, and kWTA, respectively, from top to bottom.

Dayan, P., & Niv, Y. (2008). Reinforcement learning: The good, the bad and the ugly. *Current Opinion in Neurobiology*, *18*, 185–196.

French, R. M. (1991). Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks. In *Proceedings of the 13th annual cognitive science society conference* (pp. 173–178). Hillsdale, NJ: Lawrence Erlbaum.

Hoyer, P. O. (2004). Non-negative matrix factorization with sparseness constraints. *CoRR*, *cs.LG/0408058*. Retrieved from `http://arxiv.org/abs/cs.LG/0408058`

Kandel, E., Schwartz, J., Jessell, T., Siegelbaum, S., & Hudspeth, A. J. (2012). *Principles of neural science* (Fifth Edition ed.). New York: McGraw-Hill.

Montague, P. R., Dayan, P., & Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of Neuroscience*, *16*, 1936-1947.

O'Reilly, R. C., & Munakata, Y. (2001). *Computational explorations in cognitive neuroscience*. Cambridge, Massachusetts: MIT Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*, 533–536.

Schultz, W., Apicella, P., & Ljungberg, T. (1993). Responses of monkey dopamine neurons to reward and conditioned stimuli during successive steps of learning a delayed response task. *Journal of Neuroscience*, *13*, 900–913.

Schultz, W., Dayan, P., & Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, *275*(5306), 1593–1599.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In D. S. Touretzky, M. C. Mozer, & M. E. Hasselmo (Eds.), *Advances in neural information processing systems 8* (pp. 1038–1044). Cambridge, MA: MIT Press.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press.

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*(3).