Q-Learning: A Tutorial and Extensions[1]

George Cybenko
Robert Gray
Katsuhiro Moizumi

Thayer School of Engineering
8000 Cummings Hall
Dartmouth College
Hanover, NH 03755 USA

Abstract

In the past decade, research in neurocomputing has been divided into two relatively well-defined tracks: one track dealing with cognition and the other with behavior. Cognition deals with organizing, classifying and recognizing sensory stimuli. Behavior is more dynamic, involving sequences of actions and changing interactions with an external environment. The mathematical techniques that apply to these areas, at least from the point of neurocomputing, appear to have been quite separate as well. The purpose of this paper is to give an overview of some recent powerful mathematical results in behavioral neurocomputing, specifically the concept of Q-learning due to C. Watkins, and some new extensions. Finally, we propose ways in which the mathematics of cognition and the mathematics of behavior can move closer to build more unified systems of information processing and action.

1 Introduction

The study of artificial neural networks has burgeoned in the past decade. Two distinct lines of research have emerged: the cognitive and the behavioral. Cognitive research deals with the biological phenomenon of recognition, the mathematics of pattern analysis and statistics, and applications in automatic pattern recognition. Behavioral research deals with the biological phenomena of planning and action, the mathematics of time dependent processes and applications in control and decision-making.

To be mathematically precise, let us discuss simple formulations of each problem type. A cognitive problem typically involves so-called *feature* vectors, $x \in \mathbf{R}^n$. These feature vectors are sensory stimuli or measurements and are presented to us in some random way – that is, we cannot predict with certainty which stimuli will occur next. These observations must be classified and the classification is accomplished by a function $f : \mathbf{R}^n \to \mathbf{R}^m$. The problem is to build an estimate of the true function, f, based on a sample set of the form $S = \{(x_i, y_i), i = 1, ..., N\}$ which are drawn according to a joint probability distribution on $(x, y) \in \mathbf{R}^{n+m}$, say $\mu(x, y)$. Call the estimate $g(x)$. Typically, $f(x)$ is the conditional expected value of y: $f(x) = \int_y y d\mu(x, y) / \int_y d\mu(x, y)$. A number of distinct situations in cognitive neurocomputing arise depending on the type of information available for estimating f. (See one of the numerous excellent textbooks on neural computing and machine learning for more details.)

- Supervised Learning – The sample data, S, is as above so that correct classifications, apart from noise, are part of the data. Moreover, an error criterion is typically provided or constructed on the classifications: $|g(x) - f(x)|$ is some error metric. This situation is closest to traditional statistical regression and uses many classical statistical methods.

[1]Supported in part by the Air Force Office of Scientific Research research grant F49620-93-1-0266. Presented at Mathematics of Artificial Neural Networks, Oxford University, England, July 1995.

- Unsupervised Learning – No corrrect or approximate classification values, y, are given. The problem is to organize the data into equivalence classes or clusters and then to label the classes. These labels serve to define f from above and the performance criterion is related to how accurately the equivalence classes are formed. Unsupervised learning is closely related to clustering and uses techniques from that area.

- Reinforcement Learning – As in unsupervised learning, no "correct" response is given but a *reinforcement* is available. A reinforcement can be thought of as an error when $g(x)$ is the estimated response for feature vector x but the correct response f is not available. Reinforcement learning is thus between supervised and unsupervised learning: some performance feedback is provided but not in the form of the correct answer and the deviation of the response from it. The difference between supervised and reinforcement learning has been characterized as the difference between learning from a teacher and learning from a critic. The teacher provides the correct response but the critic only states how bad the system's response was. Reinforcements are often referred to as "rewards" and "punishments" depending on their sizes and the context.

Behavioral neurocomputing derives its problems from planning, decision-making and other applications in which actions over time are of fundamental interest. The mathematical framework, which will be formalized below, involves a system with *states*. Actions are available to move the system into another state, perhaps stochastically. There is an immediate cost associated with taking an action. An action at any given time may be optimal in the short run but may not be the best over the long run when future costs and actions are taken into account. Behavioral applications are therefore most naturally cast as reinforcement learning problems. The goal of behavioral learning is to have a system infer from empirical data, the state-action pairs which give the smallest average cost.

There are a number of ways that such a behavioral system could be reduced to a cognitive problem. One involves building the following type of training set. Given some initial state, generate a sequence of actions randomly, stopping at some future time. Record the initial state, the action sequence and the ultimate cost of taking that action sequence from that initial state. From this data, build a classifier whose input feature vector is a state and whose output is the action sequence which has minimal cost over all sequences tried. This estimate of the minimal cost to solve the problem from an initial state is an estimate of the *cost-to-go* function.

A related way to solve the problem involves estimating these *cost-to-go* values for each state-action pair. Notice that in a stochastic setting, the action which empirically resulted in the least cost solution may not be the action which leads to the least cost expected value. So averaging is more appropriate than merely keeping track of the minimum cost solution for a given action sequence.

	Cognitive Learning	Behavioral Learning
Goal	Classifying inputs	Optimizing actions
Related Areas	Statistics Pattern recognition	Control theory Operations research
Relevant Mathematics	Approximation theory Probability	Dynamic programming Stochastic Systems
Status	Sound Theoretical Foundations	Emerging Analytic Theory

Table 1: Cognitive vs. Behavioral Learning

The artificial intelligence community has had considerable difficulties with behavioral learning because of difficulties with temporal delays and combinatorial explosions resulting from brute force approaches. Only recently has the mathematics of controlled Markov chains been explored along with solution techniques such as dynamic programming. A major breakthrough in learning optimal actions for such processes has been Watkins' Q-learning [3, 2, 4, 1].

In this paper, we give a quick overview of controlled Markov processes in Section 2. Section 3 presents Watkins' basic results and Section 4 extends those results to describe a system that learns and simultaneously converges to the optimal policy solution. Section 5 discusses possible areas of intersection between the cognitive and behavioral problems.

2 Markov Decision Processes

The paradigm for *behavioral* neurocomputing can typically be cast as a controlled Markov process which is described below. An environment has a number of states, $X = \{x_i\}$. For each state, there is a set of actions, $A_i = \{a_{ij}\}$, each of which transforms the current state to another state stochastically. This is quantified as a collection of transition probabilities: $p(x_i, a_{ij}, x_k)$ being the probability that applying action a_{ij} in state x_i we land in state x_k at the next time step. These probabilities are independent of time and previous states, hence the stationary Markov character.

Each action engenders a *cost*, $k(x_i, a_{ij})$, which depends only on the state and the action. The goal of a behavioral problem is to minimize this cost over time. Such a temporal cost depends on a *policy* which specifies the actions to be taken at various times in various states. Thus for each time, t, in a policy, we have a vector of actions a_t which tells us which action to apply for any of the possible states the system could be in at time t. Specifically, $a_t(x_i)$ is one of the allowable actions a_{ij} for state x_i. There are a number of ways in which to quantify this temporal cost minimization:

- Free-time minimization – A designated state, x_0, terminates the process. We are concerned with the cost of the behavior up to the time this state is reached. Starting in some state x_i and following a policy, $\pi = (a_0, a_1, a_2, ...)$, suppose we follow the trajectory $x_i, x_{i_1}, x_{i_2}, ..., x_{i_{\tau-1}}, x_0$ the cost will then be
$$\sum_{j=0}^{\tau-1} k(x_{i_j}, a_j(x_{i_j})).$$
However, since the transitions are stochastic, we must take an expectation over all possible trajectories, starting with x_i and terminating in x_0 with probabilities determined by the policy, π, and the corresponding transitions.
$$V(\pi, i) = E\{\sum_{j=0}^{\tau-1} k(x_{i_j}, a_j(x_{i_j}))\}.$$

- Fixed-time minimization – As above but the cost is computed up to a fixed time as opposed to when a terminal state is reached.

- Average-cost minimization – The average future cost is minimized (here $0 < \gamma \leq 1$ is the discount factor):
$$V(\pi, i) = \lim_{L \to \infty} \frac{1}{L+1} E\{\sum_{j=0}^{L} \gamma^j k(x_{i_j}, a_j(x_{i_j}))\}.$$

- Discounted cost minimization – Here the cost of future actions are discounted by a constant rate, $0 < \gamma < 1$, so that
$$V(\pi, i) = E[\sum_{j=0}^{\infty} \gamma^j k(x_{i_j}, a_j(x_{i_j}))].$$

We consider only discounted cost minimization problems here. A large body of literature in optimal control theory has been devoted to solving such problems and we review the key elements, leaving out the occasional technical requirements for the results to hold. We focus instead on the general flow of ideas.

First of all, we need only consider policies which are independent of time if we are interested in the optimal policy. The reason is that the process is Markovian and once we reach a state at a point in time, only the future costs can be affected. Since the costs are additive (in a discounted way), an optimal choice of action depends only on the current state. This simplification allows us to consider only policies that are of the form $\pi = (a, a, a, ...)$ where a is a mapping from states to actions.

Secondly, the cost-to-go functions, which we now write as $V(a, i)$ because we are restricting ourselves to stationary policies, satisfy a recurrence relation of the form:

$$V(a, i) = k(x_i, a(x_i)) + \gamma \sum_j p(x_i, a(x_i), x_j) V(a, j).$$

This identity has the following interpretation: using action $a(x_i)$ we go to state x_j with probability $p(x_i, a(x_i), x_j)$; once in state x_j we have the cost-to-go value $V(a, j)$. Weighing these cost-to-go values with the corresponding transition probabilities and adding gives the identity. Stacking these values to form a vector $V(a)$, the above equation becomes a vector-matrix equation of the form:

$$V(a) = k(a) + \gamma P(a) V(a).$$

Thus, every stationary policy satifies an equation of the above form which can be solved by rewriting it as:

$$(I - \gamma P(a)) V(a) = k(a)$$

where it is assumed that $P(a)$ and $k(a)$ are known for a given policy a. Thus this is a simple linear system of equations which can be solved using standard matrix iterative techniques.

What this shows is that for any policy, there is a computational procedure for computing the cost-to-go function for a fixed action policy. What remains is to find an action policy which will lead to the overall optimal strategy – one that minimizes the cost-to-go for each state.

Specifically, the optimal policy a^* has the property that

$$V(a^*) = V^* = min_a \{k(a) + \gamma P(a) V^*\}.$$

This policy can be computed using the iteration, starting with V_0 arbitrary,

$$V_{n+1} = min_a \{k(a) + \gamma P(a) V_n\}.$$

Then $V_n \to V^*$ and the optimal policy is the policy which realizes the minimization above.

3 Watkins' Q-Learning

The behavioral learning problem is to learn the optimal (minimal cost) policy using samples of the following form:

$$s_i = (\eta_i, \zeta_i, \alpha_i, k_i), i = 1, 2, 3, ...$$

These samples are four-tuples with the interpretation that s_i involves a trial use of action α_i on state η_i which resulted in a transition to state ζ_i at a cost of k_i. Given many samples with the same initial state and action, we can estimate the transition probabilities as well as the expected cost of taking that action from that state. With such estimates, a standard solution procedure for computing optimal policies for a Markov decision process can be carried out. This is an off-line, batch approach.

Is there an on-line approach which will learn the optimal strategy adaptively, with guaranteed convergence? The on-line strategy does not compute transition probabilities or costs explicitly. Watkins has given an elegant solution to this problem which he named *Q-learning*. Q-learning uses samples of the above form to update a simple table whose entries converge and whose limit values can easily be used to infer an optimal policy.

Q-learning is motivated by so-called Q-values which are defined as follows:

$$Q^a(x_i, a_{ij}) = K(x_i, a_{ij}) + \gamma P(a)V(a).$$

$Q^a(x_i, a_{ij})$ is the expected cost when starting in state x_i, performing action a_{ij} and then following policy a thereafter. Let Q^* be the Q-values for the optimal policy a^* meaning we take some initial action and then follow it with optimal actions thereafter. For these Q-values, we have

$$V(a^*)_i = \min_{a_{ij}}\{Q^*(x_i, a_{ij})\}$$

and the optimal action from state x_i is precisely the action a_{ij} which achieves the minimum.

The beauty of Watkins' Q-learning is that we can adaptively, estimate the Q-values for the optimal policy using samples of the above type. The Q-learning algorithm begins with a tableau, $Q_0(x_i, a_{ij})$, initialized arbitrarily. Using samples of the form

$$s_i = (\eta_i, \zeta_i, \alpha_i, k_i), i = 1, 2, 3, \ldots$$

we perform the following update on the Q-value tableau:

$$Q_i(\eta_i, \alpha_i) = (1 - \beta_i)Q_{i-1}(\eta_i, \alpha_i) + \beta_i(k_i + \gamma V(\zeta_i))$$

where $V(\zeta_i) = \min_\alpha\{Q_{i-1}(\zeta_i, \alpha)\}$. All other elements of Q_i are merely copied from Q_{i-1} without change. The parameters $\beta_i \to 0$ as $i \to \infty$.

Theorem (Watkins [6, 7, 5]) – Let $\{i(x, \alpha)\}$ be the set of indices for which the (x, α) entry of the Q-tableau is updated. If

$$\sum_{i(x,\alpha)} \beta_{i(x,\alpha)} = \infty \quad \text{and} \quad \sum_{i(x,\alpha)} \beta^2_{i(x,\alpha)} < \infty$$

then $Q_i(x, \alpha) \to Q^*(x, \alpha)$ as $i \to \infty$. Accordingly, $\alpha_i(x) = \mathrm{argmax}_\alpha Q_i(x, \alpha)$ converges to the optimal action for state x.
Proof – See [7, 5].

This result is remarkable in that it demonstrates that a simple update rule on the Q-tableau results in a learning system which computes the optimal policy. In the next section we show how this method can be embedded into an online system which simultaneously uses the current Q-values to generate policies *and* uses the results to update Q-values in such a way as to satisfy Watkin's theorem. Consequently, the online system learns the optimal policy to which it ultimately converges.

4 Universal On-line Q-Learning

Online optimal learning and asymptotic optimal performance must be a compromise between performing what the learning system estimates to be the optimal actions and persistently exciting all possible state-action possibilities often enough to satisfy the frequencies stipulated in Watkins' Q-learning theorem.

To begin, let us introduce a notion of asymptotically optimal performance. Suppose we have an infinite sequence of state-action pairs that result from an actual realization of the Markov decision process, say $\{(y_i, \alpha_i), i = 0, 1, \ldots\}$ with (y_i, α_i) meaning that at time i we are in state y_i and execute action α_i resulting in a transition to state y_{i+1} at the next time. For such a sequence, we have a corresponding sequence of costs v_i which represent the actual costs computed from the realized sequence, v_i being the cost of following the sequence from time i onwards.

We now define *asyptotic average optimality.* Suppose that V_i is the optimal expected cost-to-go for state i. Let $v_i(n_{ij})$ be the observed cost-to-go values for state i when the system is in state i at time n_{ij}. Let $N_M(i)$ be the number of times the process has visited state i up to time M. Then we say that the policy is *asyptotically optimal on average for state i* if

$$\frac{1}{N_M(i)} \sum_{n_{ij}<M} v_i(n_{ij}) \to V_i$$

as $M \to \infty$.

In order to establish this result, we need to construct an auxiliary Markov process with states (x_i, a_{ij}) and transition probabilities, $p((x_i, a_{ij}), (x_m, a_{mn})) = p(x_i, a_{ij}, x_m)/J_m$ where J_m is the number of allowable actions in state x_m. This auxiliary process has the following interpretation: the transition probabilities between states are determined by the transition probabilities of the original controlled process with the added ingredient that the actions corresponding to the resulting state are uniformly randomized.

Theorem – Suppose that there is at least one stationary policy under which the states of resulting Markov process constitute a single recurrent class (that is, the process is ergodic). Then, with probability 1, there is a mixed nonstationary strategy for controlling the Markov decision process with the following two properties:

1. The optimal policy is estimated by Q-learning asymptotically;
2. The control is asymptotically optimal on average for all states that have a nonzero stationary occupancy probability under the process' optimal policy.

Proof – The proof consists of three parts. We begin by showing that under the hypothesis that the original process has a stationary policy under which the Markov process consists of a single recurrent class, the corresponding auxiliary process also has a single recurrent class. Let $a(x)$ be the action for a state x which makes the original process have a single recurrent set. Denote the corresponding transition probabilities by $p_*(x, x') = p(x, a(x), x')$. Let D be the maximal number of actions per state: $D \geq |\{a_{ij}\}|$ for all i.

Consider the auxiliary process' canonical partitioning into subclasses of states in which each set of the partition consists of states which are either a) transient or b) recurrent and communicate with each other. Now the $(x, a(x))$ form a subset of the states of the auxilliary process.

Step 1. The Auxilliary Process is Recurrent – We will show that any given state of the auxilliary process communicates with every other state. Let (x, a) and (x', a') be any two states of the auxilliary process. We must show that

$$p^{(n)}((x, a), (x', a')) > 0$$

for some n. Pick any state y for which $p(x, a, y) > 0$. By assumption, there is an m for which

$$p_*^{(m)}(y, x') > 0$$

under the special choice of actions, $a(\cdot)$. This means that there is a sequence of states $y_1 = y, y_2, ..., y_m = x'$ for which

$$p(y_i, a(y_i), y_{i+1}) > 0$$

and so

$$p^{(m+1)}((x, a), (x', a')) \geq \frac{p(x, a, y)}{D} \Pi_{i=1}^{m-1} [p(y_i, a(y_i), y_{i+1})/D] > 0.$$

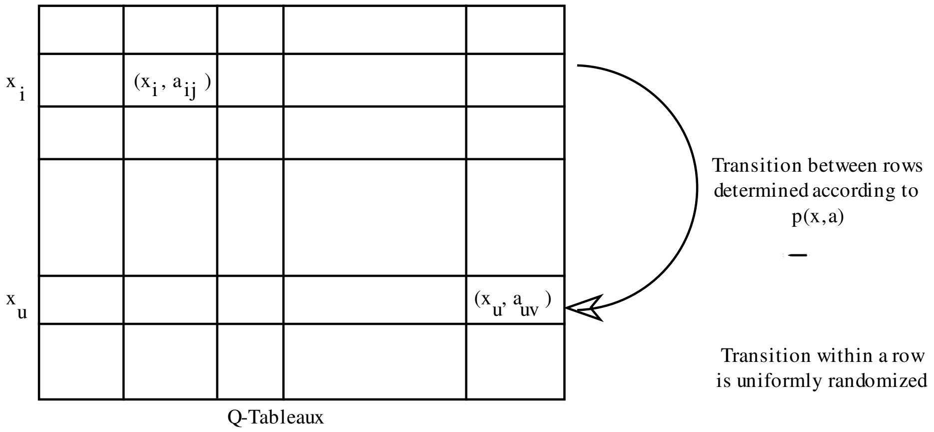

Figure 1: Auxilliary Process Schematic

Since all states of the auxilliary process communicate with each other, all states must be recurrent. Thus the auxilliary process consists of a single recurrent set of states.

Because the auxilliary process consists of recurrent states, the expected time to visit all states of the auxilliary process is finite and all states will be visited in a finite time with probability 1.

Step 2. A Mixed Policy – We next define a mixed time-dependent strategy for switching between the auxilliary process and a stationary policy based on an estimate of the optimal policy. The mixed strategy begins by chosing actions according to the auxilliary process. At times n_k we switch from the auxilliary process to the process controlled by the policy determined by the Q-table according to

$$a_j(n_k) = \text{argmin}_\alpha Q_{n_k}(x_j, \alpha).$$

At time m_k we switch back to the auxilliary process. For $k = 1, 2, 3, ...$ we have $1 = m_0 < n_k < m_k < n_{k+1} < m_{k+1} ... < \infty$.

Let N' the largest expected time to visit all states of the auxilliary process starting from any state and let $N = 2 * N'$. Consider the following experiment which is relevant to the discussion below. Start in any state of the auxilliary process and follow it for N steps at which time we jump to an arbitrary state and run for another N steps, repeating this process indefinitely. Call each such run of N steps a *frame*, numbering them F_i for $i = 1, 2, 3, ...$ We want to concatenate the frames to produce a realization of the auxilliary process. To do this, consider the final state of frame F_i. Since N was chosen to be twice as large as the expected time to visit all states starting from any state, with probability 1 some future frame, say F_{i+j} includes a visit to the final state of F_i in the first $N/2 = N'$ states of that frame. Concatenate to the end of F_i the history in F_{i+j} following the visit to the final state of F_i. Each step of this concatenation adds at least N' consecutive steps of the auxilliary process so there are an infinite number of visits to all states in the concatenation with probability 1. In the following construction, we implicitly use this property of a concatenation of frames.

The definition of n_k is as follows. At time m_{k-1}, we began following the auxilliary process which is recurrent. We store state transitions of the form

$$(\eta, \zeta, \alpha, \kappa),$$

where η is a state x_i, α is an action taken by the auxiliiary process, ζ is the state of the original process to which the process transitioned and κ is the observed cost of that transition, in a list. Proceed in this manner until either N steps have been taken in this way or until the list contains a sample for each

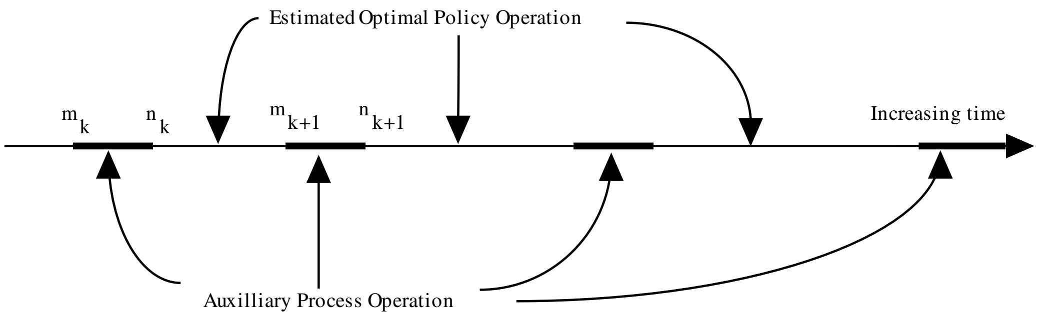

Figure 2: The Mixed Policy Transitions

element in the Q-table. If the list to update the Q-table is completed first, we update the Q-table and compute the current optimal actions. In this case, n_k is defined as the time after which the Q-table was updated. Otherwise, define $n_k = m_{k-1} + N$ and continue using the previous estimated optimal actions as a policy. In either case, $m_k = n_k + k * N$ so we operate the process using the estimated optimal policy for increasingly longer periods of time $k * N$. We then use this list to update the Q-table with the samples in the list.

Assume that the Q-table has M entries. Then for updating the ith element for the jth time in the Q-table from the sample list, use

$$\beta_{j*M+i} = \frac{1}{j*M+i}.$$

According to this scheme, element i is updated for $j = 1, 2, 3, ...$ and the corresponding subset of β's forms an arithmetic subsequence of the harmonic sequence, hence it satifies Watkins' criteria for divergence and squared-convergence.

By the discussion above about frames, this list is filled in a finite amount of time infinitely often with probability 1 but we impose a deterministic limit on the time spent in this mode generating a frame. By construction and Watkins' Theorem, the Q-table values converge to the optimal Q-values and hence the optimal action policy is eventually found.

Step 3. Asymptotic Optimal Convergence – To prove the asymptotic optimal convergence, note that the Q-table updates are performed according to Watkins' criteria so that in the limit the Q-table values determine an optimal stationary policy. Moreover, the time spent operating in the auxilliary mode becomes an asymptotically small fraction of the time spent operating in the estimated optimal policy. Hence, the asymptotic convergence to optimality.

To make this formal, note that asymptotic average optimality depends only on what happens in the limit. Specifically, if we can show that the average of the observed cost-to-go values after some fixed time converges to the optimal value then we are done.

To see this, run the process long enough so that the estimated optimal policy determined by the Q-table is as close as desired to the optimal cost-to-go for the process. This will happen at some finite time with probability 1 although the time is not known a priori. Since we can also wait until k is arbitrarily large, the fraction of time spent in the appoximately optimal mode can be made as close to 1 as we like. Now for a state that has nonzero stationary occupancy probability under the optimal policy, the fraction of time spent operating under the estimated optimal policy approaches 1 and so the empirical cost-to-go also approaches the optimal. For states that have zero occupancy probability under the optimal policy, the empirical cost-to-go will be dominated by the empirical values determined during the auxilliary process operation which will not be optimal in general. Thus the average within the mixed mode of operation is guaranteed to converge to the estimated cost-to-go for the optimal policy but only for states that have nonzero stationary occupancy probabilities under the optimal policy.

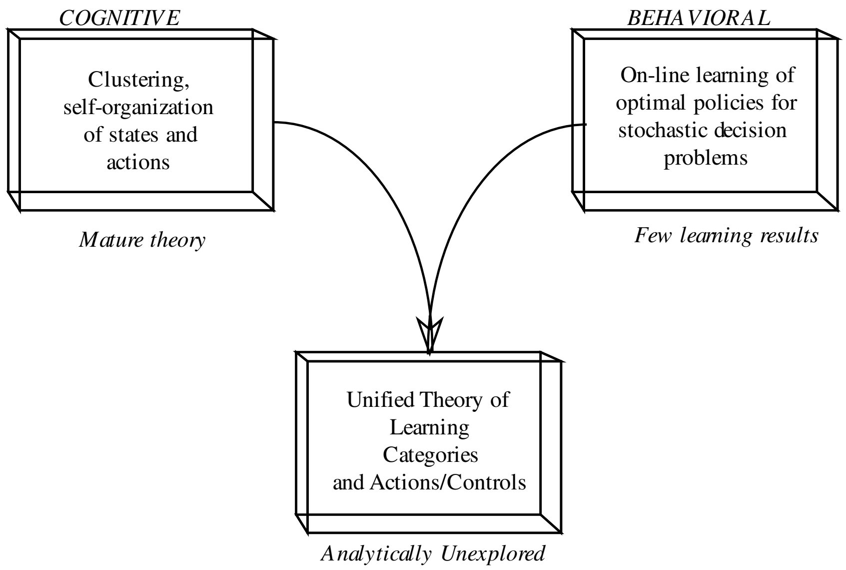

Figure 3: A Unified Learning Theory

5 Discussion

We have shown that there exists a strategy for operating a Markov Decision Process (under simple conditions) in such a way that the optimal strategy is both learned and the asymptotic operation of the system approaches optimality for states that have nonzero occupancy probabilities under the computed optimal policy. This is only one of many possible strategies for performing both of these simultaneously. The question of which operating procedures are best for fastest convergence to the optimal cost-to-go values is beyond the scope of the techniques that we use. It is an important question to pursue in the future.

One of the weaknesses of the Q-Learning framework is that states and actions for the Q-Tableaux must be known a priori. This is restrictive in most dynamic learning situations – it may not be appropriate or possible to select the actual states or actions of a system without significant experimentation first. In general, we would like to simultaneously learn the states, actions and corresponding optimal policies at one time. This subject has been looked into by various researchers but with few analytic results yet. There has been growing interest in dealing with systems that have an infinite (contiuum) of possible states and actions, requiring discretization for Q-Learning. How these states can be clustered or otherwise organized to achieve optimal operation is a challenging question that requires serious future research.

References

[1] D. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Belmont, MA, 1995.

[2] R. Howard. *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, MA, 1960.

[3] W.T. Miller III, R.S. Sutton, and P.J. Werbos. *Neural Networks for Control.* MIT Press, Cambridge, MA, 1990.

[4] R.S. Sutton, A.G. Barto, and R.J. Williams. Reinforcement learning is direct adaptive control. *IEEE Control Systems Magazine*, pages 19–22, April 1992.

[5] J. Tsitsiklis. Asynchronous stochastic approximation and Q-Learning. *Machine Learning*, 16:185–202, 1994.

[6] C.I.C.H. Watkins. Learning from delayed rewards. Ph.D. Dissertation, University of Cambridge, 1989.

[7] C.I.C.H. Watkins and P. Dayan. Q-Learning. *Machine Learning*, 8:279–292, 1989.