Temporal difference learning

Andrew G. Barto (2007), Scholarpedia, 2(11):1604. doi:10.4249/scholarpedia.1604 revision #186927 [link to/cite this article]

- **Dr. Andrew G. Barto**, Dept. of Computer Science, University of Massachuestts - Amherst

Temporal difference (TD) learning is an approach to learning how to predict a quantity that depends on future values of a given signal. The name TD derives from its use of changes, or differences, in predictions over successive time steps to drive the learning process. The prediction at any given time step is updated to bring it closer to the prediction of the same quantity at the next time step. It is a supervised learning process in which the training signal for a prediction is a future prediction. TD algorithms are often used in reinforcement learning to predict a measure of the total amount of reward expected over the future, but they can be used to predict other quantities as well. Continuous-time TD algorithms have also been developed.

Contents

The Problem

Suppose a system receives as input a time sequence of vectors (x_t, y_t), $t = 0, 1, 2, \ldots$, where each x_t is an arbitrary signal and y_t is a real number. TD learning applies to the problem of producing at each discrete time step t an estimate, or prediction, p_t, of the following quantity:

$$Y_t = y_{t+1} + \gamma y_{t+2} + \gamma^2 y_{t+3} + \cdots = \sum_{i=1}^{\infty} \gamma^{i-1} y_{t+i},$$

where γ is a *discount factor*, with $0 \leq \gamma < 1$. Each estimate is a prediction because it involves future values of y. The signal x_t is the information available to the system at time t to enable it to make the prediction p_t. In other words, p_t is a function of x_t, and we can write $p_t = P(x_t)$, where P is a *prediction function*. The discount factor determines how strongly future values of y influence current predictions. When $\gamma = 0$, p_t predicts just the next value of y, that is, y_{t+1}. As γ increases toward one, values of y in the more distant future become more significant.

The problem, then, is to select a function P so that $p_t = P(x_t) \approx Y_t$ as closely as possible for $t = 0, 1, 2, \ldots$. This is the *infinite-horizon discounted* prediction problem. TD algorithms apply to other prediction problems as described below.

Of course, it is not always possible to find a prediction function that does a good job of solving this problem. If there are no regularities in the relationship between signals x_t and future values of y, then the best predictions will be no better than random guessing. However, if there are regularities in this relationship, then predictions may be possible that are more accurate than chance. The usual way to model possible regularities is to assume that the signals x_t are derived from the states of a Markov chain, on which the numbers y also functionally depend. Depending on how the x_t represent the Markov chain's states (e.g., do these signals unambiguously identify states, or are some states *unobservable*), a prediction function may exist that accurately gives the expected value of the quantity Y_t for each t.

The Simplest TD Algorithm

A good way to explain TD learning is to start with a simple case and then extend it to the full algorithm. Consider first the problem of making a one-step-ahead prediction, i.e., the above problem with $\gamma = 0$, meaning that one wants $p_t = y_{t+1}$ for each t. Incremental error-correction supervised learning can be used to update the prediction function as inputs arrive. Letting P_t denote the prediction function at step t, the algorithm updates P_t to a new prediction function P_{t+1} at each step. To do this, it uses the error between the current prediction, p_t, and the prediction target (the quantity being predicted), y_{t+1}. This error can be obtained by computing p_t by applying the current prediction function, P_t, to x_t, *waiting one time step* while remembering p_t, then observing y_{t+1} to obtain the information required to compute the error $y_{t+1} - p_t = y_{t+1} - P_t(x_t)$.

The simplest example of a prediction function is one implemented as a lookup table. Suppose x_t can take only a finite number of values and that there is an entry in a lookup table to store a prediction for each of these values. At step t, the table entry for input x_t is $p_t = P_t(x_t)$. When y_{t+1} is observed, the table entry for x_t is changed from its current value of p_t to $p_t + \alpha(y_{t+1} - p_t) = (1 - \alpha)p_t + \alpha y_{t+1}$, where α is a positive fraction that controls how quickly new data is incorporated into the table and old data forgotten. Only the table entry corresponding to x_t is changed from step t to step $t + 1$, to produce the a table, P_{t+1}. The fraction α is called the *learning rate* or *step-size* parameter. Note that if $\alpha = 1$, the table entry is simply set to y_{t+1}, which is appropriate if the predictive relationship is deterministic. Using $\alpha < 1$, on the other hand, causes the table entries to approach the *expected* prediction targets when the predictive relationship is not deterministic.

To extend this approach to the full prediction problem with $\gamma \neq 0$, the prediction target would be $Y_t = y_{t+1} + \gamma y_{t+2} + \gamma^2 y_{t+3} + \cdots$ instead of just y_{t+1}. The algorithm above would have to update the table entry for x_t by changing its value from p_t to $p_t + \alpha(Y_t - p_t) = (1 - \alpha)p_t + \alpha Y_t$. But to do this would require *waiting for the arrival of all the future values of* y instead of waiting for just the next value. This would prevent the approach from forming a useful learning rule. TD algorithms use the following observation to get around this problem. Notice that

$$Y_t = y_{t+1} + \gamma y_{t+2} + \gamma^2 y_{t+3} + \cdots :$$

$$= y_{t+1} + \gamma[y_{t+2} + \gamma y_{t+3} + \gamma^2 y_{t+4} + \cdots] :$$

$$= y_{t+1} + \gamma Y_{t+1}, \qquad (1)$$

for $t = 0, 1, 2, \ldots$. Therefore, since $P_t(x_{t+1})$ is the estimate of Y_{t+1} available at step t (i.e., using the step-t table), one can estimate Y_t by the quantity $y_{t+1} + \gamma P_t(x_{t+1})$. That is, the current prediction function, P_t, *applied to the next input*, x_{t+1}, multiplied by γ, gives an estimate of the part of the prediction target that would otherwise require waiting forever (in this case) to obtain. The result in this lookup-table case is an algorithm that, upon receiving input (x_{t+1}, y_{t+1}), updates the table entry for x_t by changing its value from p_t to $p_t + \alpha[y_{t+1} + \gamma P_t(x_{t+1}) - p_t] = (1 - \alpha)p_t + \alpha[y_{t+1} + \gamma P_t(x_{t+1})]$.

More concisely, define the following *temporal difference error* (or TD error):

$$\delta_{t+1} = y_{t+1} + \gamma P_t(x_{t+1}) - P_t(x_t).$$

Then the simplest TD algorithm updates a lookup-table as follows: for each $t = 0, 1, 2, \ldots$, upon observing (x_{t+1}, y_{t+1}):

$$P_{t+1}(x) = \begin{cases} P_t(x) + \alpha\, \delta_{t+1} & \text{if } x = x_t \\ P_t(x) & \text{otherwise,} \end{cases} \qquad (2)$$

where x denotes any possible input signal. [Note that some authors would label the TD error as defined here δ_t instead of δ_{t+1}, giving the impression that TD learning is somehow not *causal* in that its updates depend on unavailable future values of certain variables. If it is understood that the update defined here occurs on step $t + 1$, it is completely causal. The time subscript for δ is unimportant.]

Although the detailed behavior of this algorithm is complicated, an intuitive understanding is possible. The algorithm uses a prediction of a later quantity, $P_t(x_{t+1})$, to update a prediction of an earlier quantity, $P_t(x_t)$, where each prediction is computed via the same prediction function, P_t. This may seem like a futile process since neither prediction need be accurate, but in the course of the algorithm's operation, later predictions tend to become accurate sooner than earlier ones, so there tends to be an overall error reduction as learning proceeds. This depends on the learning system receiving an input sequence with sufficient regularity to make predicting possible. In formal treatments, the inputs x_t represent states of a Markov chain and the y values are given by a function of these states. This makes it possible to form accurate predictions of the expected values of the discounted sums Y_t.

Another view of the TD algorithm is that it operates to maintain a consistency condition that must be satisfied by correct predictions. Since $Y_t = y_{t+1} + \gamma Y_{t+1}$, for $t = 0, 1, \ldots$ (Equation (1), the following relationship should hold between successive predictions:

$$P(x_t) = y_{t+1} + \gamma P(x_{t+1})$$

for $t = 0, 1, 2, \ldots$. One can show, in fact, that within an appropriate mathematical framework any function that satisfies this condition for all t must actually give the correct predictions. TD algorithms adjust the prediction function with the goal of making its values always satisfy this condition. The TD error indicates how far the current prediction function deviates from this condition for the current input, and the algorithm acts to reduce this error. This view of TD learning is connected to the theory of Markov decision processes, where the prediction function corresponds to a value function and the update process is closely related to dynamic programming methods. Most of the theoretical results about TD learning derive from these connections.

TD with Function Approximation

Beyond lookup tables, TD learning can update prediction functions represented in a variety of ways. Consider, for example, the case in which each prediction is a linear function of the input signals, where each input signal x_t is now a vector of real numbers

$$x_t = (x_t^1, x_t^2, \ldots, x_t^n) .$$

Then the prediction function of step t applied to the signal at step t' , where t' can be different from t , is defined by

$$P_t(x_{t'}) = \sum_{i=1}^{n} v_t^i x_{t'}^i,$$

where the v_t^i , $i = 1, \ldots, n$, are the coefficients, or weights, of the linear prediction function of step t , and the $x_{t'}^i$ are components of the vector $x_{t'}$. Then TD learning adjusts the weights according to the following rule: for each $t = 0, 1, 2, \ldots$, upon observing (x_{t+1}, y_{t+1}) :

$$v_{t+1}^i = v_t^i + \alpha[y_{t+1} + \gamma P_t(x_{t+1}) - P_t(x_t)]x_t^i :$$

$$= v_t^i + \alpha\delta_{t+1}x_t^i. \quad (3)$$

for $i = 1, \ldots, n$, where $\alpha > 0$ is a step-size parameter.

This learning rule adjusts each weight in a direction that tends to reduce the TD error. It is similar to the conventional Least Mean Square (LMS) or delta rule for supervised learning with the difference being the presence of $\gamma P_t(x_{t+1})$ in the error term. TD learning with linear function approximation is the best understood extension of the lookup-table version. It is widely used with input vectors consisting of the outputs of a possibly large number of sophisticated feature detectors, which is equivalent to employing representations involving sophisticated basis vectors. Note that lookup-tables correspond to linear function approximation using standard unit basis vectors (in which case Equation (3) reduces to Equation (2). TD learning with nonlinear function approximation is also possible. In general, any incremental function approximation, or regression, method can be adapted for use with TD learning. In all of these cases, it is important to recognize that TD learning does not entail specific assumptions about how stimuli are represented over time. There is a wide latitude in the alternatives that can be employed, and each has implications for the behavior of the algorithm.

Eligibility Traces

TD learning can often be accelerated by the addition of eligibility traces. When the lookup-table TD algorithm described above receives input (y_{t+1}, x_{t+1}) , it updates the table entry only for the immediately preceding signal x_t . That is, it modifies only the immediately preceding prediction. But since y_{t+1} provides useful information for learning earlier predictions as well, one can extend TD learning so it updates a collection of many earlier predictions at each step. Eligibility traces do this by providing a short-term memory of many previous input signals so that each new observation can update the parameters related to these signals. Eligibility traces are usually implemented by an exponentially-decaying memory trace, with decay parameter λ . This generates a family of TD algorithms TD(λ), $0 \leq \lambda \leq 1$, with TD(0) corresponding to updating only the immediately preceding prediction as described above, and TD(1) corresponding to equally updating all the preceding predictions. This also applies to non lookup-table

versions of TD learning, where traces of the components of the input vectors are maintained. Eligibility traces do not have to be exponentially-decaying traces, but these are usually used since they are relatively easy to implement and to understand theoretically.

Action-Conditional Prediction

Consider a setting in which the future values of inputs (x_t, y_t) are influenced by an agent's actions. The prediction at step t is often denoted $Q(x_t, a_t)$, where x_t is the input signal and a_t is the agent action at step t. The objective is to find a function that accurately predicts Y_t, the discounted sum of future values of y, on the basis of x_t and a_t. In the usual reinforcement learning setting of a Markov decision process, Y_t also depends on all the agent's actions taken after step t. Two different cases are considered for the actions taken after step t: (1) assume that after step t, the agent selects actions that generate the largest possible value of Y_t, or (2) assume that the agent follows a fixed rule, or policy, for selecting actions as a function of future inputs. In either case, the desired predictions are well-defined.

A TD learning process for case (1), known as Q-Learning, works as follows for a lookup-table representation. For each $t = 0, 1, 2, \ldots$, upon generating action a_t and observing (x_{t+1}, y_{t+1}), the prediction function Q_t is updated to Q_{t+1} defined as follows:

$$Q_{t+1}(x,a) = \begin{cases} Q_t(x,a) + \alpha[y_{t+1} + \gamma \max_a Q_t(x_{t+1}, a) - Q_t(x_t, a_t)] & \text{if } x = x_t \text{ and } a = a_t \\ Q_t(x,a) & \text{otherwise,} \end{cases}$$

where x denotes any possible input signal and a denotes any of a finite number of possible actions. For case (2), the update is the same except that $\max_a Q_t(x_{t+1}, a)$ is replaced by $Q_t(x_{t+1}, a_{t+1})$, producing a TD learning rule called Sarsa (for state-action-reward-state-action). These learning rules can also be extended to allow function approximation.

Q-Learnng and Sarsa are useful in reinforcement learning where y_t is a reward signal. An agent selecting actions that maximize $Q(x, a)$ for each current x is fully exploiting the knowledge contained in Q in its attempt to maximize the measure of long-term reward, Y_t. Under appropriate conditions, both Q-Learning and Sarsa converge to prediction functions that allow an agent to make optimal action choices.

Other Prediction Problems and Update Rules

TD learning is not restricted to the infinite-horizon discounted problem. If the input sequence is divided up into finite-length episodes, and the predictions are not expected to extend beyond single episodes, then the discount factor γ can be set to one, resulting in the *undiscounted* case. TD algorithms also exist for problems requiring the prediction of the average value per-unit-time of the target variable. In reinforcement learning where the target variable is reward, this is called the *average reward* case (Mahadevan, 1996). TD learning has also been generalized to TD Networks (Sutton and Tanner, 2005). Conventional TD learning is based on a consistency condition relating the prediction of a quantity to the prediction of the same quantity at a later time. TD Networks generalize this to conditions relating predictions of one quantity to a set of predictions of other quantities at a later time. More sophisticated update methods have been studied that use the basic TD idea but that often have better performance. Examples include *least squares TD*, an adaptation of a conventional recursive least squares regression method (Bradtke and Barto, 1996), and *Gaussian Process TD* (Engle, et al., 1993), an adaptation of Gaussian Process methods.

Psychology and Neuroscience

Theories of animal learning as studied by psychologists were a major influence on the development of TD learning. In particular, TD learning can be viewed as an extension of the Rescorla-Wagner model of Pavlovian conditioning in which the timing of stimuli within learning trials plays a significant role in how associative strengths change. A slightly modified version of the TD algorithm given above accounts for a range of phenomena observed in Pavlovian conditioning experiments. Of particular significance is the ability of the algorithm to produce an analog of *secondary reinforcement* when used as a component of a reinforcement learning system. Predictions of future reward formed by TD learning can be treated as rewarding themselves, thereby acting as secondary reinforcers. This is most clearly demonstrated in reinforcement learning systems that use the TD error δ as a reward signal. TD learning is also related to the neuroscience of reward learning through the similarity of the behavior of δ in analogs of conditioning experiments and the behavior of midbrain dopamine neurons in the brain. This observation has stimulated both neuroscientific and computational research based on the computational theory of TD learning in efforts to improve understanding of the brain's dopamine system and its role in reward-related behavior and drug addiction.

History

TD learning is most closely associated with R. S. Sutton, whose 1984 Ph.D. dissertation addressed TD learning and whose 1988 paper, in which the term Temporal Difference was first used, has become the definitive reference. TD learning is the basis of the *adaptive critic* component of the actor-critic architecture for reinforcement learning (Barto et al., 1983), as well as the basis of models of Pavlovian conditioning (Sutton and Barto, 1990). This line of research work began with the exploration of Klopf's 1972 idea of *generalized reinforcement* which emphasized the importance of sequentiality in a neuronal model of learning (see Klopf, 1982). Mention should also be made of the work of S. E. Hampson, whose 1983 Ph.D. dissertation (see Hampson, 1990) presented a similar algorithm. Earlier precursors are the method used by A. L. Samuel (1959) in his learning program for the game of checkers, I. H. Witten's (1977) algorithm presented in the context of Markov decision processes, P. J. Werbos' 1974 mention of similar ideas in the context of robust prediction and his later introduction of *Heuristic Dynamic Programming* (see Werbos, 1994). The similarity between the behavior of the TD error and the activity of dopamine neurons, based on the findings of W. Schultz (reviewed in Schultz, 1998), was first noted by Houk et al. (1995) and Schultz et al. (1997). Q-Learning was introduced in C. J. C. H. Watkins' 1989 Ph.D. dissertation, though it also has precursors in L. Tesfatsion's introduction in 1976 of *Criterion Filtering* (see also Tesfatsion, 1982), and S. Bozinovski's 1982 *Crossbar Adaptive Array* (see Bozinovski, 1995). Sarsa, proposed by Rummery and Niranjan (1994) and named by Sutton (1996), is similar to J. H. Holland's *Bucket Brigade* algorithm (Holland, 1986).

References

- Barto, A.G., Sutton, R.S., Anderson, C.W., 1983, Neuronlike Elements That Can Solve Difficult Learning Control Problems, *IEEE Transactions on Systems, Man and Cybernetics*, 13, pp. 835-846, Reprinted in J. A. Anderson and E. Rosenfeld, eds., 1988, *Neurocomputing: Foundations of Research*, Cambridge, MA: MIT Press.
- Bozinovski, S., 1995, *Consequence Driven Systems*, Bitola Macedonia: GOCMAR Publishers.
- Bradtke, S.J, Barto, A.G., 1996, Linear Least-squares Algorithms for Temporal Difference Learning", *Machine Learning*, 22, pp. 33-57.
- Engel, Y., Mannor, S., Meir, R., 2003, Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning, in *Proceedings of the International Conference on Machine Learning*.
- Hampson, S.E., 1990, *Connectionist Problem Solving*, Boston, Basal, Berlin: Birkhauser.

- Holland, J.H., 1986, Escaping Brittleness: The Possibility of General-Purpose Learning Algorithms Applied to Rule-Based Systems, in *Machine Learning: An Artificial Intelligence Approach, Volume II*, Michalski, R.S., Carbonell, J.G., Mitchell, T.M., eds., pp. 593-623, San Mateo, CA: Morgan Kaufmann.
- Houk, J.C., Adams, J.L., Barto, A.G., 1995, A Model of how the Basal Ganglia Generates and uses Neural Signals that Predict Reinforcement, in *Models of Information Processing in the Basal Ganglia*, J.C. Houk, J.L. Davis, D.G. Beiser, eds., pp. 249-270, Cambridge, MA: MIT Press.
- Klopf, A.H., 1982, *The Hedonistic Neuron: A Theory of Memory, Learning, and Intelligence*, Washington, D.C.: Hemisphere.
- Mahadevan, S., 1996, Average Reward Reinforcement Learning: Foundations, Algorithms, and Empirical Results, *Machine Learning*, 22, pp. 159-196.
- Rummery,G.A., Niranjan, M., 1994, On-Line Q-Learning using Connectionist Systems, Cambridge University Engineering Department Technical Report CUED/F-INFENG/TR 166.
- Samuel, A.L., 1959, Some Studies in Machine Learning Using the Game of Checkers, IBM Journal on Research and Development, pp. 210-229. Reprinted in E.A. Feigenbaum and J. Feldman, eds., 1963, *Computers and Thought*, New York: McGraw-Hill.
- Schultz, W., 1998, Predictive Reward Signal of Dopamine Neurons, *Journal of Neurophysiology*, 80, pp. 1-27.
- Schultz, W., Dayan, P., Montague, P.R., 1997, A Neural Substrate of Prediction and Reward, *Science*, 275, pp. 1593-1598.
- Sutton, R.S., 1996, Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding, in *Advances in Neural Information Processing Systems: Proceedings of the 1995 Conference* (NIPS 8), pp. 1038-1044, D. S. Touretzky, M. C. Mozer, M. E. Hasselmo, eds., Cambridge, MA: MIT Press.
- Sutton, R.S., 1988, Learning to Predict by the Method of Temporal Differences, *Machine Learning*, 3, pp. 9-44.
- Sutton, R.S., 1984, Temporal Credit Assignment in Reinforcement Learning, Ph.D. Dissertation, University of Massachusetts, Amherst, MA.
- Sutton, R.S., Barto, A.G., 1990, Time-Derivative Models of Pavlovian Reinforcement, in *Learning and Computational Neuroscience: Foundations of Adaptive Networks*, M. Gabriel, J. Moore, eds., pp. 497 -537, Cambridge MA: The MIT Press.
- Sutton, R. S., Tanner, B., 2005, Temporal-Difference Networks. *Advances in Neural Information Processing Systems 17*, pp. 1377-1384.
- Tesfatsion, L., 1976, Bayes' Theorem for Utility, Discussion Paper 76-65, Center for Economic Research, University of Minnesota.
- Tesfatsion L., 1982, A Dual Approach to Bayesian Inference and Adaptive Control, *Theory and Decision*, 14, pp. 177-194.
- Watkins, C.J.C.H., 1989, Learning from Delayed Rewards, Ph.D. Dissertation, Cambridge University, Cambridge, England.
- Werbos, P.J., 1994, *The Roots of Backpropagation: From Ordered Derivatives to Neural Networks and Political Forecasting*, New York: John Wiley & Sons, Inc.
- Witten, I. H., 1977, An Adaptive Optimal Controller for Discrete-time Markov Environments, *Information and Control*, 34, pp. 286-295.

Internal references

- Valentino Braitenberg (2007) Brain. Scholarpedia, 2(11):2918.
- Wolfram Schultz (2007) Reward. Scholarpedia, 2(3):1652.

- Wolfram Schultz (2007) Reward signals. Scholarpedia, 2(6):2184.

Recommended Readings

- Sutton, R.S., Barto, A.G., 1998, *Reinforcement Learning: An Introduction*, Cambridge, MA: MIT Press.
- Bertsekas, D.P., Tsitsiklis, J.N., 1996, *Neuro-Dynamic Programming*, Belmont, MA: Athena Scientific.
- Powell, W. B., 2007, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, Hoboken, NJ: John Wiley & Sons, Inc.

External links

- Author's web page (http://www-anw.cs.umass.edu/~barto/)
- R. S. Sutton's web page (http://www.cs.ualberta.ca/~sutton/index.html)

See Also

Adaptive Critic, Conditioning, Dopamine, Dynamic Programming, Markov Decision Processes, Neuroeconomics, Optimal Control, Q-Learning, Reinforcement Learning, Rescorla-Wagner Model, Reward, Reward Signals

Sponsored by: Eugene M. Izhikevich, Editor-in-Chief of Scholarpedia, the peer-reviewed open-access encyclopedia
Reviewed by (http://scholarpedia.org/w/index.php?title=Temporal_difference_learning&oldid=25755) : Anonymous
Accepted on: 2007-11-19 01:09:50 GMT (http://scholarpedia.org/w/index.php?title=Temporal_difference_learning&oldid=25755)

Categories: Machine Learning | Reinforcement Learning