Temporal–Difference Methods and Markov Models

Etienne Barnard

Abstract— **The relation between temporal-difference training methods and Markov models, which was first noticed by Sutton, is explored. This relation is derived from a new perspective, and in this way the particular association between conventional temporal-difference methods and *first-order* Markov models is explained. We then derive a generalization of temporal-difference methods that is suitable for Markov models of higher order. Finally, several issues related to the performance of mismatched temporal-difference methods (i.e., the performance when the temporal-difference method is not specifically designed to match the order of the Markov model) are investigated numerically.**

I. Introduction

MANY pattern-recognition problems involve the gradual revelation of information through time. Speech recognition is an obvious example: as a phoneme or word or sentence is uttered, the audience is progressively informed of the speaker's intended message. However, many other problems that are generally viewed as stationary are also actually time-dependent in similar vein. Visual perception, for example, very rarely involves a single view that has to be recognized—again, the recognition system usually has access to a time development of the object in the field of view, and it is only for the sake of simplification that designers of pattern-recognition systems often resort to classification of a single visual image.

The realization that time development furnishes much useful information has prompted researchers to investigate various means of utilizing this information. One of the most successful proposals in this regard is the method of temporal differences (TD) [1]. This method, which is closely related to Holland's bucket brigade [2] and Samuel's checker player [3], attempts to apportion learning throughout the time development of such a dynamic system, rather than concentrating it at the final moment when the "correct classification" (or its equivalent) is revealed. TD and related methods (which are collectively referred to as "reinforcement learning methods") have proven their worth in various contexts—see e.g., [4], where TD is used to construct a competitive program for playing backgammon, and [5]–[7], which introduce the combination of reinforcement learning and world models for intelligent control systems.

Sutton [1] has discussed a one-parameter family of temporal-difference methods, which he calls TD(λ), and his analysis leads to new insights into TD(0). The more general case ($\lambda \neq 0$) has been studied by Dayan [8], who also shows the close relation between TD(λ) and the so-called Q-learning method of Watkins [9]. Q-learning explicitly considers the effects of control actions, and is thus more general than TD(λ) in this sense. In the current work we will be concerned with the TD(0) method exclusively, for which the name TD will be reserved.

The most important conclusion to come out of Sutton's analysis [1] is that TD methods lead to a more accurate estimation of certain important statistical parameters than conventional supervised methods. This result holds when the system to be estimated is a first-order Markov model, which is somewhat surprising in the light of the fact that TD methods were developed without any apparent reference to Markov models.

In the current paper the relation between Markov models and TD methods is made somewhat clearer by deriving the idea behind TD methods from certain facts about first-order Markov models. This derivation in Section II not only clarifies the above relation, but also removes a misconception that arose in [1], regarding gradient-based approaches and TD.

Another issue that arises naturally in the context of TD methods is their performance when the system to be estimated is not truly first-order Markov. In Section III it is first shown that the basic TD method can straightforwardly be extended to handle higher-order Markov models if their order is known. The performance of the conventional TD method is then compared to that of the extended method, both for Markov models of first order and of higher order (see Section IV).

The superiority of TD methods compared to supervised learning techniques derives from their more detailed modeling of the underlying probabilistic model. TD methods are consequently able to estimate more efficiently than conventional supervised techniques. However, there generally is a negative aspect to more detailed models: they involve the (effective) determination of more parameters from a given data set, which is generally detrimental to the efficiency of a statistical estimator. The trade-off between accuracy and complexity of a model and its effect on estimation efficiency is also studied in Section IV.

In Section V we summarize the paper, draw certain conclusions, and point out some extensions that are suggested by the current work.

II. Deriving TD from Markov Models

The problem we consider is the following: the state s_t of a system is observed at each of a number of consecutive time steps $t = 1, 2, \cdots$. After m_σ such steps the system arrives at some terminal state z_σ. The goal is to predict the value of z_σ from the observed history $s_1, s_2, \cdots$, and to this end various sequences $\sigma = 1, 2, \cdots$ (the "training set") are observed. (Note that m_σ can be a function of σ, i.e., the

Manuscript received October 27, 1991; revised May 2, 1992.

The author is with the Department of Electronics and Computer Engineering, University of Pretoria, Pretoria, 0002, South Africa.

IEEE Log Number 9205786.

lengths of the sequences can be variable.) In particular, we would like to know the probability of eventually arriving at a particular terminal state given the current (nonterminal) state s the system is in.

To simplify the exposition we assume that there are only two terminal states - generalizations are briefly considered in Section V. We then only need to predict the probability of the terminal state being a particular terminal state (which we label by $z = Z$), since the probability of the other state as well as any expectation values depending on the terminal states can directly be calculated from this information. Let us thus introduce a set of variables w_s, which estimate the probability of the various states s giving rise to the terminal state Z. Our purpose is thus to estimate w_s, which is the probability that a sequence will end up in state Z given that it currently is in state s.

We now investigate estimators for $\boldsymbol{w} = (w_1, w_2, \cdots)$. An obvious estimator is simply the observed probability that Z is entered after s had been observed; that is

$$w_s = n_{Zs}/n_s \tag{1}$$

where n_s is the total number of times that s is entered into, and n_{Zs} is the number of these cases that ended in terminal state Z. One intuitively expects such an estimator to be unbiased, and this expectation is borne out by calculation. It is, however, not clear that this is an efficient estimator, since it does not utilize all available information. In particular, one knows not only the final state into which a given state evolves, but also which other states occur *en route* to the final state, and in which order.

By assuming that the states are generated by a Markov model it is possible to derive an estimator that uses this information in addition. Let p_{rs} denote the conditional probability of nonterminal state r going over into nonterminal state s (i.e., the probability of entering s at time $t+1$ if the state at time t was r), and let h_s denote the probability of entering terminal state Z at time $t+1$ given state s at time t. Then the vector $\boldsymbol{w}$ can be computed as follows: state r can lead to terminal state Z in two ways - Z can either follow directly upon r, or some state s can follow after r, and this state can eventually lead to Z. Since these events are mutually exclusive, their probabilities add. Also, if the states are generated according to a first-order Markov chain, the events (r goes to s) and (s eventually terminates in Z) are independent. Consequently, we must have that

$$w_r = h_r + \sum_s p_{rs} w_s. \tag{2}$$

This is a linear equation that can be solved for $\boldsymbol{w}$ given the matrix $\boldsymbol{P} = \{p_{rs}\}$ and the vector $\boldsymbol{h} = \{h_r\}$. However, these quantities also need to be estimated during the training process, so we rather attempt to set up a recursion that estimates them and approximates $\boldsymbol{w}$ simultaneously.

Now $\boldsymbol{P}$ and $\boldsymbol{h}$ are probabilities, so their values can be estimated in a manner equivalent to (1), i.e.,

$$p_{rs} = m_{rs}/n_r \tag{3}$$

and

$$h_r = m_{rZ}/n_r, \tag{4}$$

where m_{rs} and m_{rZ} are the numbers of times that state r is followed by states s and Z, respectively. Substituting these expressions into (2) and multiplying through by n_r leads to

$$\sum_s m_{rs} w_s - n_r w_r + m_{rZ} = 0 \tag{5}$$

or

$$(\boldsymbol{M} - \boldsymbol{N})\boldsymbol{w} + \boldsymbol{m} = \mathbf{o} \tag{6}$$

where $\boldsymbol{M}$ is the matrix with entries $\{m_{11}, m_{12}, \cdots\}$, $\boldsymbol{N}$ is a diagonal matrix with diagonal entries $\{n_1, n_2, \ldots\}$, and $\boldsymbol{m}$ is the vector $\{n_{1Z}, n_{2Z}, \cdots\}$.

Now, the iterative solution of equations of the form $\boldsymbol{A}w + \boldsymbol{b} = \mathbf{o}$ is a common problem in neural-network theory (see e.g., [10]), and a popular solution to this problem is to iterate according to

$$\boldsymbol{w} \rightarrow \boldsymbol{w} + \alpha(\boldsymbol{A}w + \boldsymbol{b}). \tag{7}$$

Clearly this has its only fixed point when $\boldsymbol{w}$ solves the original equation, and one tries to choose the sign and magnitude of α such that this fixed point is converged upon. This solution should be distinguished from that which arises from gradient descent on the surface $E = |\boldsymbol{A}w + \boldsymbol{b}|^2$, which results in $\boldsymbol{w} \rightarrow \boldsymbol{w} + \alpha(\boldsymbol{A}^T\boldsymbol{A}w + \boldsymbol{A}^T\boldsymbol{b})$. (The superscript T is used to indicate the matrix transpose.) As long as $\boldsymbol{A}$ is positive definite, (7) can be proven to converge with appropriate α, and in this case convergence is faster than with gradient descent [11].

What makes the TD method new is that $\boldsymbol{A}$ and $\boldsymbol{b}$ are not known beforehand - they also have to be estimated during the solution process. Thus, at each time step $\boldsymbol{A}$ is replaced by the contribution to it from the current and previous states, and similarly for $\boldsymbol{b}$. For example, if the current state is s and the previous state is r,

- we have observed a transition from r to s, so that m_{rs} is contributed to;
- state r has been observed, so that n_r is contributed to; and
- no transition to Z has occurred, so $\boldsymbol{m}$ is unaffected.

(State s is also observed, and therefore n_s will be incremented at the *next* time step.)

In other words, if the current state is s and the previous state is r, $\boldsymbol{A} = \boldsymbol{M} - N$ is estimated by $A_{ij} = \delta_{ir}\delta_{js} - \delta_{ir}\delta_{jr}$ and $\boldsymbol{b} = \boldsymbol{m}$ is estimated by the zero vector (o). (The symbol δ_{ij} denotes the Kronecker delta operator, which is one if its subscripts are equal; otherwise it equals zero.) A current state of Z and a previous state of r leads to an estimate of $A_{ij} = -\delta_{ir}\delta_{jr}$ for $\boldsymbol{A}$ and $(0, 0, \cdots 1, 0, \cdots)$ (the "1" is in the rth position) for $\boldsymbol{b}$, and when the other terminal state is entered into after state r these values are $-\delta_{ir}\delta_{jr}$ and $\mathbf{o}$.

To implement these equations in the most straightforward fashion we associate a vector $\boldsymbol{x}_s$ with each nonterminal state s. This vector is chosen to be the unit vector with a '1' in position s and zeroes elsewhere. The estimated probability of entering terminal state Z given state s can therefore be expressed as $y_s = \boldsymbol{x}_s^T\boldsymbol{w}$. If the specific value of $\boldsymbol{x}_s$ at time t is $\boldsymbol{x}_t$ (with $\boldsymbol{x}_s$

defined to be $\mathbf{o}$ if s is a terminal state), all three cases of the update equation for time $t+1$ can thus be summarized by

$$\boldsymbol{w} \rightarrow \boldsymbol{w} + \alpha((\boldsymbol{x}_t\boldsymbol{x}_{t+1}^T - \boldsymbol{x}_t\boldsymbol{x}_t^T)\boldsymbol{w} + \boldsymbol{x}_t\delta_{s_{t+1},Z}) \tag{8}$$

which is the customary form of the TD equations.

In summary, then, the TD method can be understood as an example of the recursion (7) for the linear equations (6), which were derived from the Markov assumption. (In practice one often prefers to replace (8) with its batch-mode equivalent, where updating of $\boldsymbol{w}$ is performed only after the changes occurring on the right-hand side of (8) have been accumulated over all of a fixed training set. In that case one can see that the matrices that occur in (8) represent the estimates of the corresponding quantities over the training set. That is, if the term multiplying α is summed over the training set, it equals the left-hand side of (6). This is analogous to the fact that the updates of the Widrow-Hoff rule equal the training-set estimate of the gradient of the LMS criterion function when used in batch mode [12].) This derivation does not introduce the changes in $\boldsymbol{w}$ as being proportional to the gradient of some criterion function, and one can in fact easily show that the update term can in general not be equal to such a gradient (see Appendix A).

From this perspective the vector outer products $\boldsymbol{x}_t\boldsymbol{x}_{t+1}^T$ and $\boldsymbol{x}_t\boldsymbol{x}_t^T$ appear in the update equation (8) because they serve as single-sample estimates of the matrices $\boldsymbol{M}$ and $\boldsymbol{N}$, respectively, and $\boldsymbol{x}_t\delta_{s_{t+1},Z}$ similarly estimates the vector $\boldsymbol{m}$. The vector $\boldsymbol{w}$ is updated until it and the estimated quantities (approximately) satisfy (6).

It is therefore probably preferable to view TD as an instantiation of Markov-model estimation theory according to this derivation, rather than as a technique that tries to minimize some prediction error, as in [1]. On the one hand, the prediction-error point of view is not rigorously correct, and on the other hand the derivation from estimation theory clearly indicates the relation between the TD method and Markov models.

III. Extending The TD Method to Second-Order Markov Models

Given the relation between the TD method and first-order Markov models, it is natural to ask whether it is possible to extend the TD method so that it also predicts the outcomes of Markov models of higher order accurately. This issue is of some practical importance, since it is often not valid to assume that the dependence of a subsequent state on its past is determined by only the most recent previous state.

To show how this generalization can be made, we now investigate second-order Markov models. In that case, the transition probabilities depend not only on the current state r, but also on the previous state q. The transition probabilities are consequently written p_{qrs} and h_{qr} for transitions to states s and Z, respectively, and we now wish to estimate w_{qr}. The analog of (2) is then

$$w_{qr} = h_{qr} + \sum_s p_{qrs}w_{rs}. \tag{9}$$

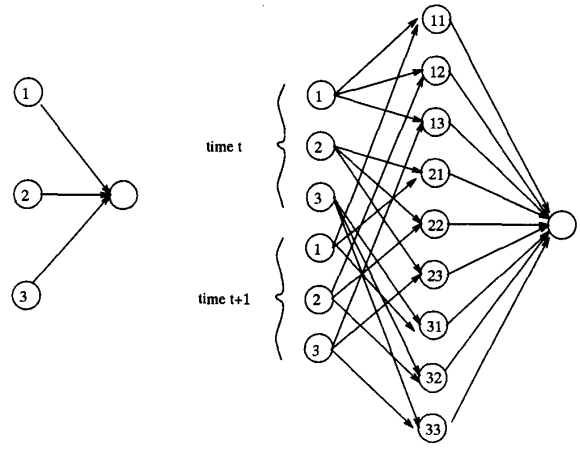


Fig. 1. Neural-network models for (a) standard TD and (b) TD extended to estimate second-order Markov models. In both cases, the system under consideration has three nonterminal states.

As in the first-order case, we can manipulate this equation to obtain a recursion in terms of the observed states, with the previous *two* time steps now contributing to the recursion:

$$w_{qr} \rightarrow w_{qr} + \alpha\left(\sum_s m_{qrs}w_{rs} - m_{qr}w_{qr} + m_{qrZ}\right). \tag{10}$$

The quantities m_{qrs}, m_{qr}, and m_{qrZ} are again estimated from the observed transitions, so that in (10) we employ

$$m_{qrs} = \delta_{s_{t-1},q}\delta_{s_t,r}\delta_{s_{t+1},s} \tag{11}$$

$$m_{qr} = \delta_{s_{t-1},q}\delta_{s_t,r} \tag{12}$$

and

$$m_{qrZ} = \delta_{s_{t-1},q}\delta_{s_t,r}\delta_{s_{t+1},Z}. \tag{13}$$

These quantities can again be written in terms of the state vectors $\boldsymbol{x}$, but now the resulting equation cannot simply be written in terms of conventional linear-algebra operations.

Since the estimated probability y_s of the first-order TD method is computed by the vector inner product $\boldsymbol{x}_s^T\boldsymbol{w}$, this computation can be performed by a simple neural network with one output neuron and n input neurons for n states (Fig. 1(a)) [13]. The extended TD method requires the output to depend on the two preceding states. This procedure is therefore best pictured in terms of a network with one hidden layer as in Fig. 1(b); each hidden node is activated at time t if and only if a particular pair of states occur at times t and $t+1$, and the output is then equal to the weight of this activated hidden neuron.

It is also clear that this technique can be used to extend the TD method to handle Markov models of arbitrary order by replacing (10) with the appropriate expression, and then estimating the quantities occurring in that equation as in (11)–(13).

There is a standard method of extending statistical methods for first-order Markov models to models of higher order [14].

For a model of order g containing n states, we form a new state space containing n^g states, this state space being the g-fold outer product of the original space. That is, each state in the new space is labeled by a g-tuple of the old states, where at any time these represent the g most recent g'th-order states that the model entered into. One can show that the model in this new space is first-order Markov. It is also not hard to see that the predictions of *this* extension are exactly the same as those of our (10). However, the formulation presented by (10) is considerably more efficient, since the standard method requires the manipulation of n^g x n^g matrices.

IV. The Performance of TD Methods of Mismatched Order

We have shown that it is possible to extend TD to estimate Markov models of arbitrary order. There are, however, at least three reasons to be interested in the performance of the TD method when it is applied to a Markov model that is of a different order than the order that it was designed for:

- In a practical application, the true order of the model to be estimated might not be known. A TD model might still be applied in such cases, and it is necessary to know how well it will fare.
- Even when the order of the model is known, this order may be so high that it is not practical to construct an appropriate TD method; for reasons of economy one would then like to use a method designed for a model of lower order.
- Since lower-order models effectively estimate fewer parameters, their efficiency for small sample sizes might actually be better than that of a model designed explicitly for the appropriate order under certain circumstances.

To investigate these issues, various experiments have been performed with TD models. In all cases, the system to be investigated was indeed a Markov model, but the order of the model was varied. A training set of S training sequences was obtained from the assumed Markov model, where a training sequence is defined as the sequence of states that the model enters into from some initial state until it reaches a terminal state. Using these S training sequences the TD equations were solved (i.e., the weights at which, e.g., (8) stabilizes were computed) for methods of first and second order, as well as the "0th-order method" of (1). (Two earlier states are considered in the computation of the second-order prediction, and one previous state for the first-order prediction. This justifies the name "0th order" for the computation (1), which employs no prior states. It is interesting to note that this method is the stable point of the equations that would be derived from the most straightforward supervised-learning approach—see Appendix II.) This training was performed using 500 different sets of S sequences each obtained from the same model, and the mean value of the squared error was computed. This error was defined as the difference between the exact probability of terminating in state Z given state s and the estimated probability, for each nonterminal state s, weighted by the probability p_s of that state occurring. The exact probability was computed analytically from the known Markov models used to generate the data. This process was repeated for various values of S.

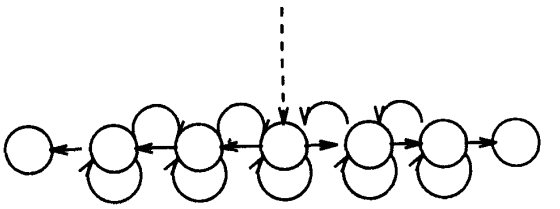


Fig. 2 Markov model employed for the first set of experiments: all indicated transitions had equal probability, and no other transitions occurred.

The mean error computed in this fashion is an indication of the difference between the probability of ending in state Z as predicted by each method, and the true value of that probability (as computed from the underlying model). By weighting the contribution from each state with the probability that the state will occur, one obtains the error that is to be expected without reference to the current state of the system. That is, the error thus computed equals the mean difference between the actual and predicted probabilities when the mean is computed over all states entered into by the model.

Note that the 0th-order and first-order methods give the required probabilities directly, whereas the second-order method gives the probability of terminating in state Z given a particular *transition*. Thus, to compare the various models, it is necessary to convert these probabilities to state-conditional probabilities. Not doing so would bias the procedure in favor of the second-order model, since it would then be able to use knowledge of the earlier transition to make a more accurate prediction of the expected outcome. The prediction of the second-order model given a state s is thus taken as the weighted average of the predictions for all transitions ending in that state, with the weighting factor for each transition being the relative likelihood that it occurs, that is

$$p(Z|s) = \sum_r p(Z|rs)p(rs)/p(s) \tag{14}$$

where $p(Z|s)$ $(= w_s)$ is the probability of state s terminating in Z, $p(Z|rs)$ $(= w_{rs})$ is the probability of transition rs ending in Z, and $p(s)$ and $p(rs)$ are the estimated probabilities of state s and transition rs, respectively. Maximum-likelihood estimators of these probabilities were again employed, e.g., $p(s) = n_s/m$ (m is the total number of states in the training set).

Our Markov models were similar in structure to those of Sutton [1]: they consisted of seven states, two of which were labeled the terminal states. These chains were always started from a fixed initial state, and the probability of reaching terminal state Z was estimated.

For the first set of experiments, the true Markov model was of first order, the states were linearly arranged, and if the system was in a given nonterminal state, it was equally likely to stay in that state, move to the left or move to the right at the next time step (see Fig. 2). The terminal states are the leftmost and rightmost states in the chain.

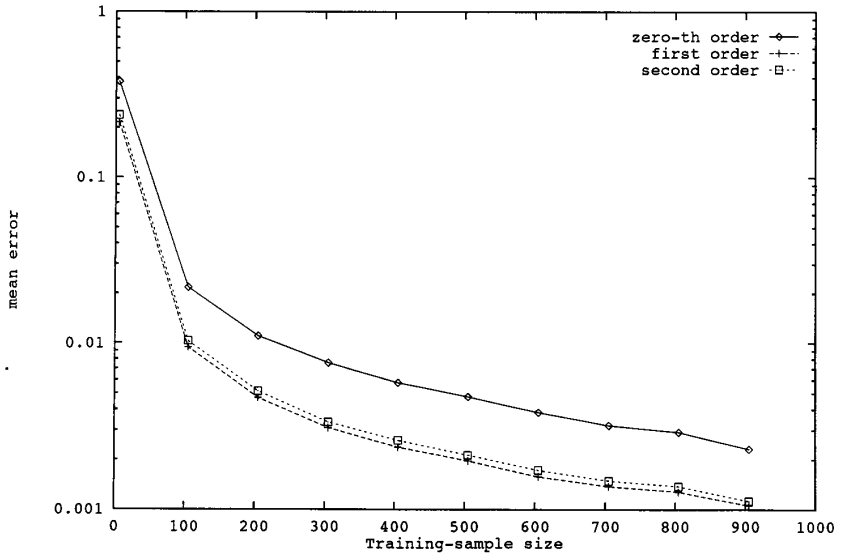


Fig. 3. Expected squared prediction error as a function of the number of training sequences, for a first-order Markov model

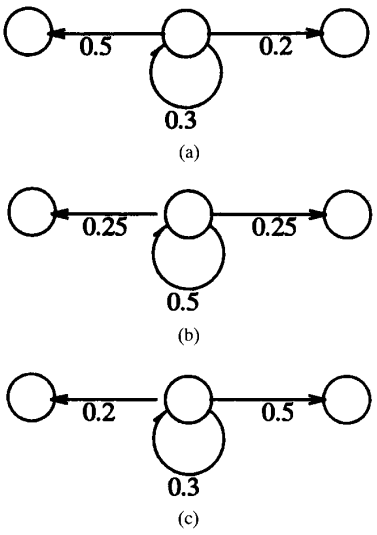


Fig. 4. For the first of the second-order Markov models employed in the second experimental set, the transition probabilities if the previous transition had been to the left is shown in (a), the probabilities if the previous transition had been to the same state is shown in the center, and the probabilities if the previous transition had been to the right is shown in (c). These probabilities are the same for all nonterminal states.

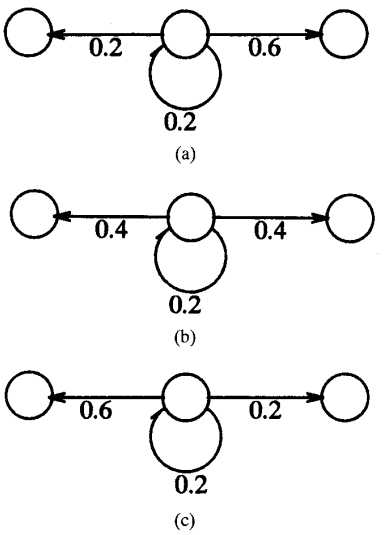
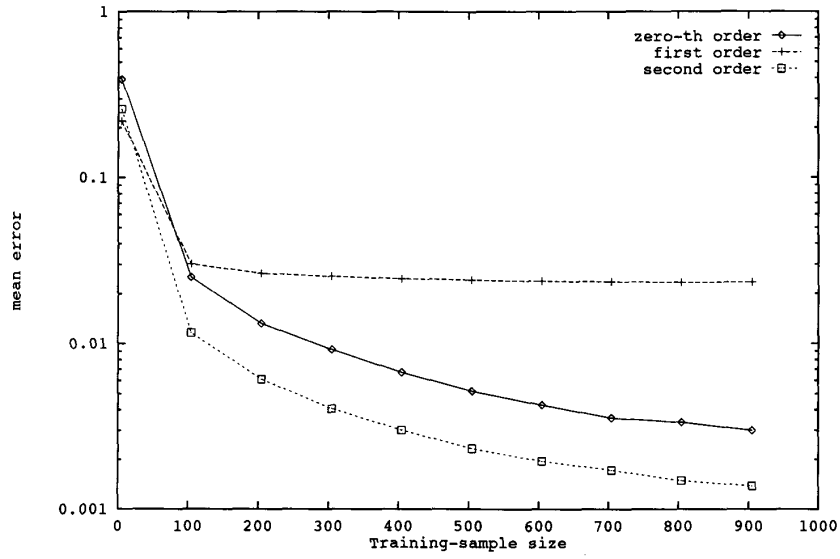


Fig. 5. As in Fig. 4, for the second model of the second set of experiments.

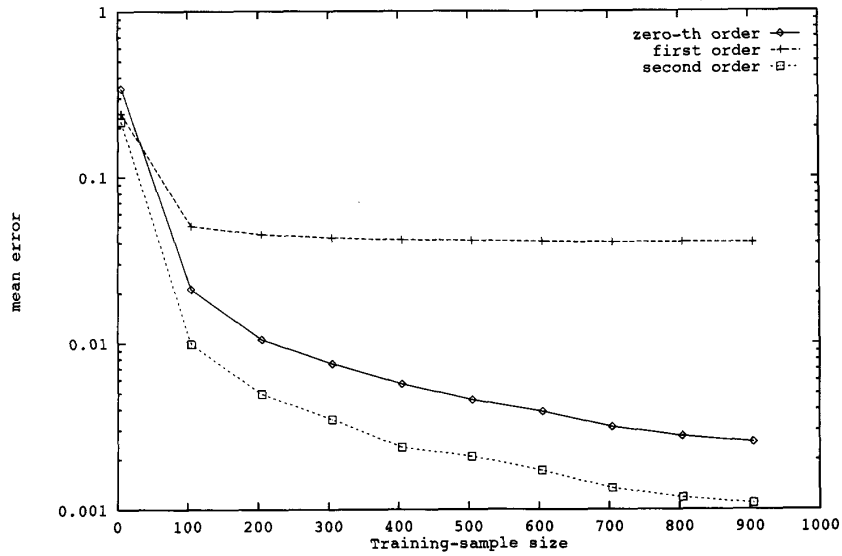
The results of this experiment are shown in Fig. 3. (In this figure, and similar figures below, the leftmost data point was obtained with 5 training samples.) It can be seen that the error for all methods decreases monotonically as the number of training sequences is increased, as would be expected. Also, since the model is really of first order, the 0th-order estimator is least accurate in this case. The first- and second-order methods perform similarly, with the first-order model being slightly more accurate than the second-order model.

The two models we investigated for our second set of experiments again used seven states in a linear arrangement, but now they were of second order. That is, if a model was in a particular nonterminal state, it could again stay in that state, move to the left, or move to the right. Now, however, the probability of taking each of these actions depends on the previous *transition*, as shown in Figs. 4 and 5.

The results of these experiments (see Fig. 6) indicate that the second-order method is considerably more successful than either of the other methods in estimating the terminal-state probabilities. For small training-sequence sizes the first-order model outperforms the 0th-order model, but as the number of samples is increased this trend is reversed. Since the second-order nature of the model is markedly different from any first-order model, the first-order estimate rapidly saturates —



(a)



(b)

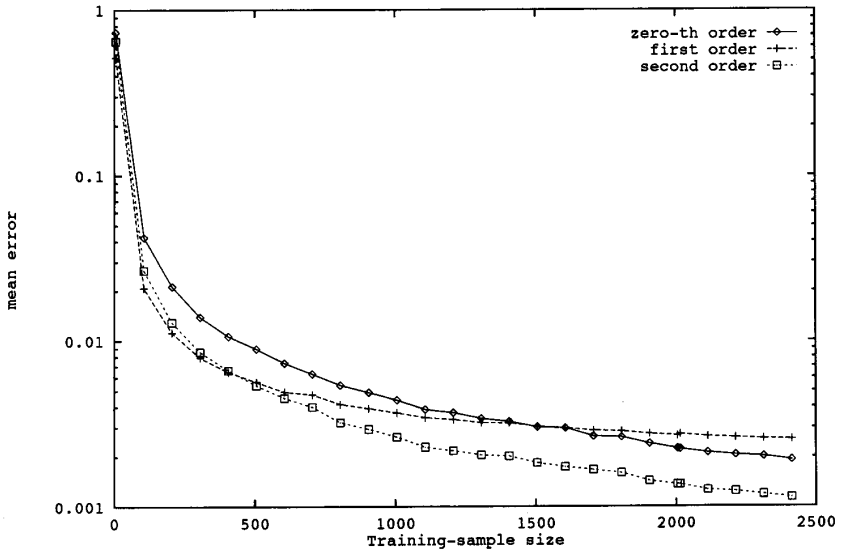
Fig. 6. Expected squared prediction error as a function of the number of training sequences, for the second-order Markov models of (a) Fig. 4 and (b) Fig. 5.

performance improves only very slowly when more than about 200 samples are presented to the first-order method. The methods for orders zero and two, on the other hand, continue improving substantially.

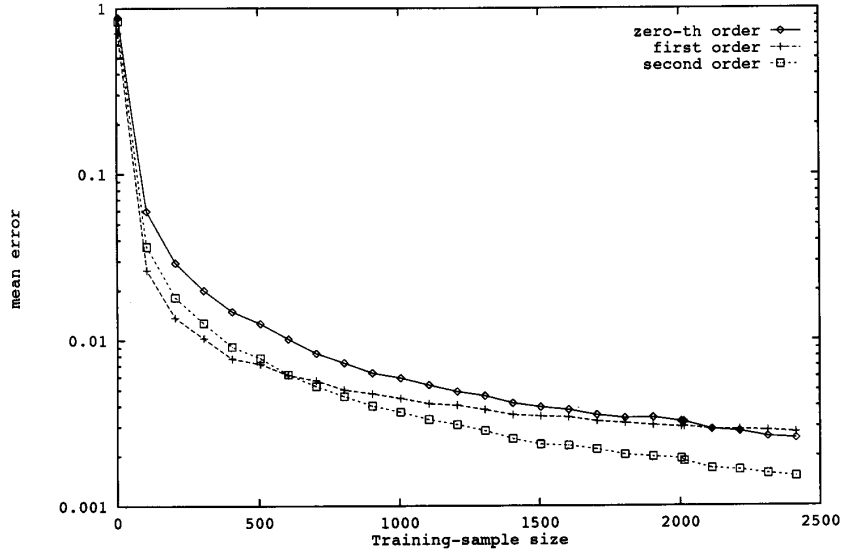
For our third experiment a second-order model was again used. The conditional probabilities in p_{qrs} were now generated randomly with the restriction that $\sum_s p_{qrs} = 1$ for all q and r. Two different models were generated in this fashion, and the above procedure was followed for each of these. As can be seen in the results of Fig. 7, the relative performance of the various methods in this case is initially similar to what was observed with the model that was really first-order (compare Fig. 3). As the number of training samples is increased, however, the second-order nature of the true model shows itself in that the second-order model eventually outperforms the other two models in both cases.

V. Conclusion

By considering various estimators of transition probabilities in Markov models, we have been able to show that TD methods naturally arise as efficient estimators. This point of view leads to a method of extending TD methods to estimate models of higher order as well.



(a)



(b)

Fig. 7. Expected squared prediction error as a function of the number of training sequence, for two random second-order Markov models.

Our numerical experiments have shown that the appropriateness of a model is generally more important for efficient estimation than the number of parameters effectively estimated. Thus, the first-order method is most accurate for the first set of experiments (the slight difference between the results for the first- and second-order methods indicating the effect of the superfluous parameters estimated by the second-order method), and the second-order method is most accurate for the second-order experiments. However, for the third set of experiments the second-order method is initially outperformed by the first-order method. This is probably due to the random nature of the transition probabilities for this experimental set - since there is no systematic trend of dependence of the current transition on the previous one, the cost of the additional parameters effectively estimated by the second order model is initially more important than the accuracy of the model. Only when the number of samples is increased sufficiently does the true second-order nature of the model become apparent, and then the second-order method does indeed perform better than the other methods.

The experiments also show that care must be exercised when a method of inappropriate order is used: for many training samples the 0th-order method actually outperforms the first-order method in the second and third experimental sets (Figs.

6 and 7.) It therefore appears that it might be preferable to use the 0th-order model when the true nature of the model to be estimated is unknown, and many training samples are available.

This work can be extended in various directions. One straightforward extension was alluded to in Section II: if more than two terminal states exist, it is necessary to estimate the probabilities of all terminal states. This is easily done by employing a separate output neuron for each terminal state, and only incrementing the $\boldsymbol{b}$-vector corresponding to that neuron (see (7)) when the particular state occurs. Similarly, other statistics than the terminal-state probabilities can be estimated.

A more substantial extension involves the application of "mixed" state vectors: we have been assuming that the current state of the system is always given by one of the unit basis vectors, and Sutton [1] has shown that similar results can be obtained if any linearly independent basis is used to describe the states. It might be interesting to study the extent to which uncertainty about the current state of the system (as in, e.g., hidden Markov models) can be handled by using linear superpositions of these vectors.

Appendix I
TD Is Not a Gradient-Descent Method

From (8), TD would be a gradient-descent method if

$$\Delta = (\boldsymbol{x}_t \boldsymbol{x}_{t+1}^T - \boldsymbol{x}_t \boldsymbol{x}_t^T)\boldsymbol{w} + \boldsymbol{x}_t \delta_{s_{t+1},Z} \tag{15}$$

(or possibly the sum of this quantity over all of the training set) could be written as the gradient of some function E. Thus,

$$\frac{\partial E}{\partial w_i} = x_{ti}((\boldsymbol{x}_{t+1}^T - \boldsymbol{x}_t^T)\boldsymbol{w} + \delta_{s_{t+1},Z}) \tag{16}$$

and

$$\frac{\partial E}{\partial w_j} = x_{tj}((\boldsymbol{x}_{t+1}^T - \boldsymbol{x}_t^T)\boldsymbol{w} + \delta_{s_{t+1},Z}). \tag{17}$$

Taking the derivative of (16) with respect to w_j and the derivative of (17) with respect to w_i give

$$\frac{\partial^2 E}{\partial w_i \partial w_j} = x_{ti}(x_{(t+1)j} - x_{tj}) \tag{18}$$

and

$$\frac{\partial^2 E}{\partial w_j \partial w_i} = x_{tj}(x_{(t+1)i} - x_{ti}), \tag{19}$$

respectively. Comparing these two expressions we see that they will generally not be equal for the single-sample case (since $x_{tj}x_{(t+1)i} \neq x_{ti}x_{(t+1)j}$ in general), and that they will also only be equal in expectation value if the transition probabilities of the model are symmetrical (i.e., $p_{ij} = p_{ji}$).

We therefore conclude that the assumption that Δ is a gradient is false - the TD model is generally not a gradient-descent technique.

Appendix II
Supervised Learning and the "0th Order" Model

The most straightforward way to apply supervised techniques to the problem at hand is to define a criterion function that is minimized when the prediction and the actual model output accord as well as possible (in the least-mean-squares sense). As in Section II, we introduce state vectors $\boldsymbol{x}_i$, and define the network's output (prediction of the probability of terminal state Z) when it is in state i by $y = \boldsymbol{x}_i^T \boldsymbol{w}$. The criterion function is thus

$$E = \frac{1}{2}\sum_{\sigma}\sum_{t=1}^{m_\sigma}(z_\sigma - \boldsymbol{x}_t^{\sigma T}\boldsymbol{w})^2 \tag{20}$$

where z_σ is defined to be 1 if the sequence σ terminated in state Z, and $z_\sigma = 0$ if it terminated in the other terminal state; $\boldsymbol{x}_t^\sigma$ corresponds to the state at time t of sequence σ.

Setting the derivative of this function equal to zero gives

$$\boldsymbol{w} = \boldsymbol{R}^{-1}\boldsymbol{q} \tag{21}$$

with $\boldsymbol{R} = \sum_{\sigma t} \boldsymbol{x}_t^\sigma \boldsymbol{x}_t^{\sigma T}$ and $\boldsymbol{q} = \sum_{\sigma t} z_\sigma \boldsymbol{x}_t^\sigma$. Writing this in component form, and using the definition of the $\boldsymbol{x}$-vectors, we see that $\boldsymbol{R}$ is a diagonal matrix with the number of times each state occurred as the corresponding diagonal entry (this matrix was called N above), and $\boldsymbol{q}$ is the vector that counts the number of times each state eventually led to terminal state Z. Thus, (21) is equivalent to (1).

References

[1] R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning*, vol. 3, pp. 9–44, 1988.

[2] J. H. Holland, "Escaping brittleness: the possibilities of general-purpose learning algorithms applied to parallel rule-based systems," in *Machine Learning: An Artificial Intelligence Approach*, R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, Eds. Los Altos, CA: Morgan Kaufmann, 1986.

[3] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. and Development*, vol. 3, pp. 210–229, 1959.

[4] G. Tesauro, "Practical issues in temporal difference learning," *Machine Learning*, to appear, 1992.

[5] S. D. Whitehead and D. H. Ballard, "A role for anticipation in reactive systemms that learn," in *Proc. 6th Int. Workshop Machine Learning*, Ithaca, NY, 1989, pp. 354–357.

[6] S. D. Whitehead, "Scaling reinforcement learning systems," Tech. Rep. 304, Dept. Comput. Sci., Univ. Rochester, 1989.

[7] R. S. Sutton, "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming," in *Proc. 6th Int. Conf. Machine Learning*, 1990, pp. 226–224.

[8] P. Dayan, "Temporal differences: TD(λ) for general λ," *Machine Learning*, to appear, 1992.

[9] C. J. C. H. Watkins and P. Dayan. Q-learning, *Machine Learning*, to appear, 1992.

[10] F. J. Pineda, "Dynamics and architecture for neural computation," *J. Complexity*, vol. 4, no. 3, pp. 216–245, Sept. 1988.

[11] G. Strang. *Introduction to Applied Mathematics*. Cambridge,MA: Wellesley-Cambridge, 1986.

[12] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[13] R. S. Sutton and A. G. Barto, "Toward a modern theory of adaptive networks: Expectation and prediction," *Psych. Rev.*, vol. 88, no. 2, 1981.

[14] P. Billingsley, "Statistical methods in markov chains," *Ann. Math. Statistics*, vol. 32, pp. 12–40, 1960.

Etienne Barnard was born in Pretoria, South Africa in 1963. He received the B.Eng degree in electronics from the University of Pretoria, Pretoria, South Africa, the B.Sc.(Hons) and M.Sc. degrees in physics from Witwatersrand University, Johannesburg, South Africa, and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, PA.

He is an Assistant Professor of Electronics and Computer Engineering at the University of Pretoria. His research interests include pattern recognition with application to speech and image understanding, neural networks, and computer-generated holography. He has acted as consultant to industry in all of these fields.