SEQUENTIAL DETECTION USING LEAST SQUARES TEMPORAL DIFFERENCE METHODS

Anthony Kuh

University of Hawaii
Dept. of Electrical Engineering
Honolulu, HI 96822
kuh@spectra.eng.hawaii.edu

Danilo Mandic

Imperial College
Electrical and Electronic Engineering
London SW7 2AZ
d.mandic@imperial.ac.uk

ABSTRACT

This paper considers sequential detection problems where we learn from sets of training sequences. The sufficient statistics can be learned quickly using a least squares temporal difference (TD) learning algorithm. This algorithm converges much quicker than previously applied TD learning algorithms. The algorithm can easily be implemented in an on-line manner and can also be applied to more complicated decentralized detection problems.

1. INTRODUCTION

This paper extends earlier research on sequential detection where we learn from sets of training sequences [6]. In the earlier research we used a popular reinforcement learning algorithm, Temporal Difference (TD) learning [12]. A learning algorithm was designed to learn the sufficient statistics of the sequential detection problem given training sequences. The training sequences were of varying length and were drawn from the Sequential Probability Ratio Test (SPRT) designed by Wald [15]. The TD learning algorithm learned the sufficient statistics and given enough training sequences, the algorithm approximated the SPRT closely. The major problem with the TD learning algorithm is that learning was slow and was sensitive to parameter adjustments such as the step size μ and the TD λ value. More powerful reinforcement learning algorithms have been developed [4, 5, 8] that are based on least squares methods combined with TD learning algorithms. In this paper we apply the Least Squares TD (LSTD) λ algorithm developed by [4] to learn sufficient statistics given SPRT training sequences. We demonstrate through simulations that the algorithm requires more than an order of magnitude less iterations than using the TD λ learning algorithm.

In many applications of gathering data such as sensor networks [1], users are constrained by communication constraints. There is a cost to transmitting information over sensor networks and a goal is to make good decisions once sufficient information is gathered. As an example consider the PODS project where sensors are placed at remote locations in Hawaii to monitor endangered plant species. Data is gathered to determine if these plant species are healthy or not. For this type of scenario sequential tests are preferable to fixed sample tests. In a fixed sample test, a set amount of data is gathered and then a decision is made. In a sequential test data is gathered until sufficient information is retrieved to make a decision. It has been shown that sequential tests achieve the same performance as fixed sample tests with less than half the data samples [15].

In this paper we consider pattern recognition problems where we learn from training data. The vast majority of work uses fixed sample tests where supervised learning algorithms such as multi-layer feedforward networks, Radial Basis Functions, or Support Vector machines are used [7, 9]. There is very little work on learning training data from sequential detection problems. A goal of this paper is to design a computationally efficient learning algorithm to be able to learn from training sequences. Here we extend the work in [6] to apply the LSTD algorithm to sequential detection problems. A goal is to design a computationally efficient algorithm that can also be implemented on-line and can be applied to more complicated decentralized sequential detection problems.

The paper is organized as follows. In Section 2 we discuss the TD λ learning algorithm and the LSTD λ learning algorithm. Section 3 gives a brief discussion of sequential detection and Wald's SPRT. Section 4 implements the TD λ and the LSTD λ learning algorithm for the sequential detection problem. Section 5 presents some simple simulation results showing the computational advantages of the LSTD λ algorithm. Finally, Section 6 discusses extensions of this work to learning on-line, applying LSTD λ to decentralized sequential detection problems, and modifications of the algorithm using kernels.

2. TD AND LSTD ALGORITHMS

The Temporal Difference (TD) learning algorithm was developed in [12] and has been used in many applications from learning absorption probabilities in a Markov chain to path navigation to learning how to play backgammon to solving the policy iteration step for dynamic programming problems [2, 13]. TD learning is easy to implement for look up tables or for function approximation using linear functions. Here we will consider linear functions. Given an observation $x_k \in \mathcal{R}^n$, we let $\Phi(x_k) = (\phi_1(x_k), \dots, \phi_d(x_k))^T$ denote the feature vector and let

$$\hat{y}(x_k) = w^T \Phi(x_k) \tag{1}$$

denote the estimate of a function $y(x)$ where w is a weight vector that is learned. One iteration cycle of the TD λ learning algorithm can be described as follows [4]:

```
TD $\lambda$ iteration

i) Initialization: set change $\Delta = 0$ and get input sequence
      $x_1, \dots, x_N$. Set $v_1 = \Phi(x_1)$ and $t = 1$.

ii) While $t \leq N$,
        $\Delta \leftarrow \Delta + v_t(R_t + (\Phi(x_{t+1}) - \Phi(x_t))^T w$, update
            weight change with $R_t$ being reward.
        $v_{t+1} = \lambda v_t + \Phi(x_{t+1})$, update modified feature vec-
            tor.
        $t \leftarrow t + 1$.

iii) $w \leftarrow w + \mu\Delta$.
```

Here μ is the step size. This algorithm is a gradient approximation algorithm and we would like to know when it converges. We get mean convergence when

$$\mathrm{E(w) = E(w)} + \mu(\mathrm{b + Aw + u}) \tag{2}$$

where

$$b = \mathrm{E}(\sum_{\mathrm{t}} \mathrm{v_t R_t}) \quad \mathrm{A} = \mathrm{E}(\sum_{\mathrm{t}} \mathrm{v_t}(\Phi(\mathrm{x_{t+1}}) - \Phi(\mathrm{x_t}))^{\mathrm{T}}) \tag{3}$$

and u is zero mean noise with small variance. In order for (2) to be satisfied we must have $b + Aw = 0$ and this is the basis of the least squares algorithm [4] where we approximate both b and A by training samples. A cycle of this algorithm is described as follows [4]

```
LSTD $\lambda$ iteration

i) Initialization: set change $A = 0, b = 0$ and get input se-
      quence $x_1, \dots, x_N$. Set $v_1 = \Phi(x_1)$ and $t = 1$.

ii) While $t \leq N$,
        $A = A + v_t(\Phi(x_t) - \Phi(x_{t+1}))^T$, update data matrix.
        $b \leftarrow b + v_t R_t$, update data vector with $R_t$ being re-
            ward.
        $v_{t+1} = \lambda v_t + \Phi(x_{t+1})$, update modified feature vec-
            tor.
        $t \leftarrow t + 1$.

iii) $w = A^{-1} b$.
```

The LSTD λ algorithm developed in [4] extends the work of [5] who developed LSTD $\lambda = 0$ algorithm. In general, least squares algorithms have higher complexity per iteration than gradient approximation algorithms, but converge in much fewer iterations.

3. SEQUENTIAL DETECTION

Let $\{X_k; k = 1, 2, \dots\}$ be a sequence of independent and identically distributed (iid) random variables with observations drawn according to

$$\begin{array}{lll} & H_0: & X_k \sim f(x_k; \theta_0), \quad k = 1, 2, \dots \\ \text{versus} & & \\ & H_1: & X_k \sim f(x_k; \theta_1), \quad k = 1, 2, \dots \end{array} \tag{4}$$

where $f(x; \theta)$ is the probability density function of X_k given the parameter θ .

Let $\mathcal{X}_t$ be the vector of t observations, $\mathcal{X}_t = (x_1, \dots, x_t)$ ($t \geq 1$), and let z_t be the logarithmic likelihood ratio based on $\mathcal{X}_t$,

$$z_t = \log \frac{f_t(\mathcal{X}_t; \theta_1)}{f_t(\mathcal{X}_t; \theta_0)} \tag{5}$$

where $f_t(\mathcal{X}_t; \theta)$ is the joint density function of $\mathcal{X}_t$ given θ. Let

$$l(x_k) = \log \frac{f(x_k; \theta_1)}{f(x_k; \theta_0)}. \tag{6}$$

For iid observations we have that

$$z_t = \sum_{k=1}^{t} l(x_k). \tag{7}$$

Then Wald's *sequential probability ratio test* [15], denoted by $SPRT(b, a)$, with $b < 0 < a$ for (4) is defined as (initially set $t = 1$),

i) if $z_t \leq b$, accept H_0 and stop;

ii) if $z_t \geq a$, accept H_1 and stop;

iii) if $b < z_t < a$, continue sampling by observing x_{t+1} ($t \leftarrow t + 1$, goto i));

where a and b ($-\infty < b < a < \infty$) are detection boundaries. The detection boundaries are set so that the false alarm rate is α and the miss probability is β where these quantities are defined by

$$\alpha = P(D = H_1 \mid H_0), \qquad \beta = P(D = H_0 \mid H_1), \tag{8}$$

where D is the decision rule. It can be shown [15] that a and b satisfy

$$a \leq \log \frac{1-\beta}{\alpha}, \qquad b \geq \log \frac{\beta}{1-\alpha}. \tag{9}$$

Practically, we choose the following detection boundary where we use a' and b' instead of a and b where

$$a' = \log \frac{1-\beta}{\alpha}, \qquad b' = \log \frac{\beta}{1-\alpha}. \tag{10}$$

These boundary values are chosen because they are easier to compute than a and b and in many practical cases $a' \approx a$ and $b' \approx b$. Wald proved that when $SPRT(b, a)$ is used, the detection procedure terminates and minimizes the average sample size among all tests while keeping the error probabilities at values α and β [15].

4. LEARNING SEQUENTIAL DETECTION TRAINING SEQUENCES

Here we apply TD and LSTD learning algorithms to learn the sufficient statistics of the binary sequential detection problem in Section 3. We are given a set of SPRT sequences and know the values of the false alarm probability α and the miss probability β. Sequential detection can be viewed as dynamic programming problem [15] and we can formulate the two reinforcement learning algorithms of Section 2 to learn the sufficient statistic given the set of training sequences.

The sequential detection problem can be viewed as a Markov process with two absorbing boundaries. We are given sequences drawn from both H_0 and H_1. A reward of a is achieved when we hit the upper boundary and a reward of $-b$ is achieved when we hit the lower boundary. If we have not terminated the sequence no rewards are given. Our goal is to learn the sufficient statistics. For Gaussian iid scalar random variables the sufficient statistic is completely described by

$$s_t = c_0 t + c_1 \sum_{i=1}^{t} x_i + c_2 \sum_{i=1}^{t} {x_i}^2. \tag{11}$$

Many other random variables can also be described by the above sufficient statistic s_t. Here our goal is to learn the weights w such that $w \approx c$. The feature vectors for this problem are given by $\phi_1(t) = t$, $\phi_2(t) = \sum_{i=1}^{t} x_i$, and $\phi_3(t) = \sum_{i=1}^{t} {x_i}^2$. This is a slightly different model than presented in Section 2 as now $\Phi(t)$ depends on $(x_1 \ldots x_t)$ and not just x_t.

For vector random variables, other more complex random variables, and dependent random variables we can use other functions such as kernel functions to approximate the sufficient statistic. The key to the sufficient statistic approximation is that it is a linear function of the feature vector $\Phi(t)$.

5. SIMULATIONS

We conducted simulations for different sequential detection tests. Here we present a simple example illustrating some of the advantages of the LSTD λ algorithm. Under H_1, the conditional random variable is Gaussian with mean 1 and variance 1. Under H_0 the conditional random variable is Gaussian with mean 0 and variance 1. We set $\beta = \alpha = .01$ resulting in $a = -b = 4.5951$. Here a sufficient statistic is given by the log-likelihood ratio with $l(x) = -.5 + x$. In our simulations the quadratic term should be close to 0 or $w_2 \approx 0$.

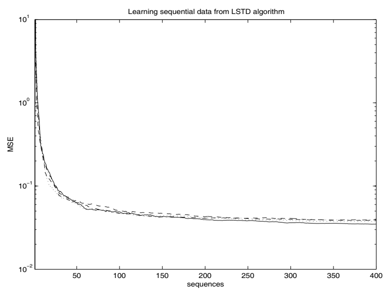

Fig. 1. LSTD algorithm applied to learn Gaussian sequences.

The LSTD algorithm is easy to implement as there are no parameters to set except for λ. Results are shown for $\lambda = 0, .3, .5, 1$. The curves show averages of 100 simulations. Note that the algorithm is not sensitive to the choice of λ as all curves are almost identical. We define convergence loosely as four time constants if we model the curve as an exponentially decaying function. The LSTD algorithm converges very quickly as after 100 sequences it has achieved a mean squared error rate of less than .05 (norm squared value of the weight error).

By contrast the TD algorithm was very sensitive to λ values and a suitable step size must also be chosen. We chose a step size of $\mu = .01/(.01t + 1)$. This seemed to work reasonably well. The TD algorithm also had difficulties converging for values of λ near 1. Even for small values of λ, the algorithm took much longer to converge than LSTD. For $\lambda = .1$ the algorithm needed to learn more than 2000 sequences before it converged. For larger values of λ more than 10000 sequences were needed until convergence was achieved. The TD algorithm also had many more wild fluctuations before converging. Weight magnitudes would often become very large before converging to the desired sufficient statistic values. The TD algorithm also had difficulties with longer sequences. The average sequence length was about 10.5. For sequences longer than 40, weight changes would be large resulting in larger magnitude weights. The TD algorithm improved some when longer sequences were not used. For the simulation conducted we found that the iteration cost for the TD and LSTD were almost the same as the LSTD algorithm only needed to invert 3 by 3 matrices.

6. DISCUSSION AND EXTENSIONS

This paper discussed a computationally efficient method to learn the sufficient statistic of a binary sequential detection problem from seeing SPRT training sequences. The SPRT training sequences cannot be learned using conventional supervised learning algorithms as there is a problem of credit assignment. The reward is only gotten at termination and the sequences are of variable length. In previous work we found that the reinforcement learning algorithm, TD learning was capable of learning sufficient statistics given SPRT sequences [6]. The TD learning algorithm is slow and we found that the LSTD learning algorithm by [4] achieved a speed up of between 10 and 100.

There are many further extensions for this work. An online version of LSTD can easily be applied to update the weights at each update. The algorithm would be similar to recursive least square algorithms and each iteration would find the weight values in $\mathcal{O}(d^2)$ instead of $\mathcal{O}(d^3)$ operations that are required to invert a matrix. An on-line algorithm would also be able to perform other tasks such as tracking and processing of time varying data.

We will also consider using LSTD algorithms for decentralized or distributed detection problems. Here local detectors gather information from sensors and send quantized information to a fusion center which then makes a decision. In [11], a fixed sample sized decentralized sequential detection problem is considered with learning performed using marginalized kernels. An optimization problem is formulated to find the kernels and associated weights. This problem could be extended to sequential detection where the fusion center sends feedback information to the local detectors. For the parametric problem where conditional densities are known the problem is formulated as a dynamic programming problem [14]. When densities are unknown we can possibly solve the problem using LS Policy Iteration (LSPI) as discussed in [8]. We could also formulate suboptimal approaches where local detectors do not receive feedback information from the fusion center. Different forms of the LSTD algorithm could make intelligent decisions under these constraints.

Since the LSTD algorithms give good performance with relatively low complexity there would be interest in applying the algorithms to real decentralized detection problems such as sensor networks. Kernel learning algorithms have been used to perform sensor localization for real sensor networks [10]. LSTD algorithms could easily be applied to sensor networks to perform sequential detection or even estimation of parameters.

7. REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, A survey on sensor networks, *IEEE Communications Magazine* 102-114, Aug. 2002.

[2] D. Bertsekas and J. Tsitlikis. *Neurodynamic programming*, Athena Scientific, Belmont, MA., 1996.

[3] E. Biagioni and K. Bridges. The application of remote sensor tecdhnology to assist the recovery of rare and endangered species. *International Journal of High Performance Computing Applications*, vol. 16, 315-324, Aug. 2002.

[4] J. Boyan. Technical update: Least-squares temporal difference learning. *Machine Learning*, 49(2-3), 233-246, 2002.

[5] S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning, *Machine Learning* 22, 33-57, 1996.

[6] C. Guo and A. Kuh. Temporal Difference Learning Applied to Sequential Detection, *IEEE Trans. on Neural Networks*, 8(2), 278-287, Mar. 1997.

[7] S. Haykin. *Neural Networks, A Comprehensive Foundation*, 2nd. Ed., Prentice Hall, 1998.

[8] M. Lagoudakis and R. Parr. Least-squares policy iteration, *Journal of Machine Learning Research*, Vol. 4, 1107-1149, 2003.

[9] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An Introduction to Kernel-Based Learning Algorithms. *IEEE Trans. on Neural Networks*, Vol. 12, #2, 181-202, 2001.

[10] X. Nguyen, M. Jordan, and B. Sinopoli. A kernel-based learning approach to ad hoc sensor network localization, *ACM Transactions on Sensor Networks*, Vol 1(1), pages 134–152, 2005.

[11] X. Nguyen, M. Wainwright, M. Jordan. Nonparametric decentralized detection using kernel methods, *IEEE Trans. on Signal Processing*, to appear Nov. 2005.

[12] R. Sutton. Learning to predict by the methods of temporal differences, *Machine Learning,* 3, 9-44, 1988.

[13] R. Sutton. and A. Barto. *Reinforcement learning: an introduction*, MIT Press, Cambridge MA., 1998.

[14] V. V. Veeravalli, T. Basar and H. Poor. Decentralized sequential detection with a fusion center performing the sequential test, *IEEE Trans. Inform. Theory,* vol. 39, 433-442, March, 1993.

[15] A. Wald and J. Wolfowitz. Optimum character of the Sequential probability ratio test," *Ann. Math. Statist.*, 19 326-339, 1948.