Randomized Neural Networks for Learning Stochastic Dependences

Vivek S. Borkar, *Senior Member, IEEE*, and Piyush Gupta

***Abstract*—We consider the problem of learning the dependence of one random variable on another, from a finite string of independently identically distributed (i.i.d.) copies of the pair. The problem is first converted to that of learning a function of the latter random variable and an independent random variable uniformly distributed on the unit interval. However, this cannot be achieved using the usual function learning techniques because the samples of the uniformly distributed random variables are not available. We propose a novel loss function, the minimizer of which results in an approximation to the needed function. Through successive approximation results (suggested by the proposed loss function), a suitable class of functions represented by combination feedforward neural networks is selected as the class to learn from. These results are also extended for countable as well as continuous state-space Markov chains. The effectiveness of the proposed method is indicated through simulation studies.**

I. Introduction

THIS paper considers the following basic problem and its variants. Given a finite string $\{(X_k, Y_k), 1 \leq k \leq n\}$ of copies of the random-variable pair (X, Y), we want to learn the dependence of Y_k on X_k. The conventional procedure would be to seek a function h so as to minimize an appropriately defined error between Y and the estimate $h(X)$. A common choice is the well-known *mean square error* given by $E[||Y - h(X)||^2]$. For this error criterion, the best estimate is given by the conditional mean $h(X) = E[Y \mid X]$. Our problem then reduces to learning this $h(\cdot)$ based on observed samples.

Having done so, suppose we also want to know the best estimate of $g(Y)$ given X for a prescribed function $g(\cdot)$. Having estimated $h(X) = E[Y \mid X]$, a natural candidate would be $g(h(X)) = g(E[Y \mid X])$. But this need not equal the true "best estimate" $E[g(Y) \mid X]$. Thus the "functional learning" approach sketched above runs into trouble when one wants to learn more than one function (e.g., the first few conditional moments) of Y given X.

Ideally, what one would really need to know is the conditional law of Y given X. This, of course, is a classical statistical problem and various techniques for its resolution, both parametric and nonparametric, are available. Our aim here is to present an alternative that retains the "functional learning" flavor and allows us to use neural network techniques in a convenient manner.

The key step is to invoke a standard result of probability theory which allows us to write $Y_k = f(X_k, \xi_k)$, where $\{\xi_k\}$ are independent random variables uniformly distributed over $[0, 1]$ and independent of $\{X_k\}$, and f a suitable (measurable) map. The problem of learning the conditional law of Y_k given X_k is then equivalent to that of learning the function f.

But there is a catch here, viz., that the samples of $\{\xi_k\}$ are not available. We work around this by choosing our error measure so that the learning does not require $\{\xi_k\}$. Yet another problem is that "measurable map f" is far too general a search domain to be computationally amenable. One therefore restricts a priori the class of candidate f's, or the "architecture" for learning, keeping in mind the error measure being used. The remainder of this section makes these ideas precise.

To motivate our choice of the function class, consider the evolution of $f(X, \xi)$ as a three step process:

1) take the family $f(\cdot, y)$ indexed by $y \in [0, 1]$;
2) evaluate $f(X, y)$ for $y \in [0, 1]$;
3) pick one of them, $f(X, \xi)$, according to a uniform distribution on $[0, 1]$.

With this picture in mind, consider as a first approximation replacement of the uncountable family $f(\cdot, y)$, $y \in [0, 1]$, by a finite family $f_i, 1 \leq i \leq N$. Correspondingly, we replace $f(X, \xi)$ by its approximation

$$\hat{Y} \triangleq \bar{f}(X, \xi) = \sum_{i=1}^{N} I\{\xi \in A_i\} f_i(X)$$

where $\{A_i\}$ is a partition of $[0, 1]$. (As we argue later in the Appendix, the choice of N can be based upon how many "averages" of (Y, X) and $(\hat{Y}, X)$ we wish to match.) The family $\{f_i\}$ can be picked from a suitable function space, say the space of continuous functions, and chosen so that they are "well spread" in this space. This makes a case for choosing them from a parameterized family known to have good approximation properties. We choose the family of feedforward neural networks, justified by the approximation theorems of [1] and [8], among others. We let β_i denote the parameter vector parameterizing the feedforward neural net $f_i(\cdot)$.

Given the above architecture for $\bar{f}$, suppose one wishes to match $g_m(Y)$ and $g_m(\hat{Y})$, $1 \leq m \leq M$, for some prescribed functions $\{g_m\}$. The latter could be, for example, the first few moments or the first few elements of a complete orthonormal basis of a suitable Hilbert space of functions, or something

Manuscript received April 5, 1998; revised September 16, 1998. This paper was recommended by Associate Editor K. Pattipati.

V. S. Borkar is with the Department of Computer Science and Automation, Indian Institute of Science, Bangalore 560012, India.

P. Gupta is with the Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign, IL 61801 USA (e-mail: piyush@decision.csl.uiuc.edu).

Publisher Item Identifier S 1083-4419(99)05268-1.

else (such as "features") motivated by the application in mind. In any case, these are assumed to be prescribed a priori. The "matching" can then be defined in terms of minimizing the "error"

$$\hat{L}_n(f,\bar{f}) = \frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} a_m \cdot \left(g_m(Y_k) - \sum_{j=1}^{N} \alpha_j g_m(f_j(X_k)) \right)^2 \tag{1}$$

where $\alpha_j = P(\xi_k \in A_j), 1 \le j \le N$. Note that the second term in the difference above is not $g_m(\hat{Y}_k)$ as intended, but its conditional expectation given X_k. This is because the former depends on ξ_k explicitly and is therefore not available. Furthermore, this replacement is justified by an "asymptotic equivalence" (i.e., as $n \to \infty$) between the two, described in the Appendix. Parameters $\{a_m\}$ are a priori weights. Without any loss of generality, these may be taken to be positive with $\sum_m a_m = 1$.

The advantage of the above criterion is that it no longer involves $\{\xi_k\}$ or $\{A_j\}$ explicitly, only the probabilities $\{\alpha_j\}$. The learning problem then amounts to minimizing this error criterion simultaneously over $\{\alpha_j\}$, $\{\beta_i\}$. To ensure that different β_i's do not converge to the same value leading to a degenerate representation, we may introduce lateral inhibition among them. We did not, however, find this necessary in our simulation studies. It seems to suffice to initialize the β_i's randomly.

It is important to note that in the summation in the square brackets of (1), the α_j's appear as multipliers of g_m and not of $f_j(X_k)$'s. Thus they can truly be viewed as probabilities with which we randomize between the f_j's. This is to be distinguished from a mere "weighted average of feedforward nets" which would be the case if we replaced the expression in square brackets by $g_m(\sum_{j=1}^{N} \alpha_j f_j(X_k))$. This is the difference between "conditional expectation of a function" and "function of conditional expectation" that we alluded to in the beginning. With this interpretation in mind, we refer to this architecture as randomized neural networks (RNN's).

The foregoing discussion also underscores the key difference between RNN's and standard multilayer perceptrons. Note that the architecture of RNN appears as though it were just a multilayer perceptron with a linear layer at the output. If we treat it as such and use the usual training methods for the same, we would not be justified in using the weights of the linear layer as probability weights as done above. In particular, an estimate of $h(Y)$, say, can be justifiably taken to be $\sum_j \alpha_j h(f_j(X))$, which is not the case with a standard multilayer perceptron. This difference has come about because of our error criterion and our stochastic realization theoretic formulation. To repeat, RNN's try to learn an approximation to conditional law, not a particular conditional average as a multilayer perceptron would.

The next section describes the training algorithm for the RNN, followed by numerical experiments for the independently identically distributed (i.i.d.) case sketched above, as well as the Markov case in Sections III and IV, respectively. A more detailed mathematical motivation for our formulation is given in the Appendix. We conclude this section with a brief comparison with some representative works related to ours.

Most of the earlier approaches assume that the true conditional distribution comes from a known countable family of distributions [9], [14]. For example, DeSantis *et al.* [9] consider the problem of learning the conditional distribution from a countable class of distributions, so as to minimize the *entropy* of the observed data. In the approach taken here, we do not make any such assumptions. In [14], on the other hand, the f_j's are interpreted as different algorithms and the output is a weighted sum of them where the weights are learned from data. This is the "weighted average" formalism described and contrasted with our approach in an earlier paragraph.

Another approach which is closely related to the issues addressed here, is the generalized probably approximately correct (PAC) model of learning from examples [12]. In this model, the learning system is required to decide on an action, say a_k, after observing the input X_k. Depending on a_k and actual output Y_k, there is a "loss" $l(a_k, Y_k)$ and the objective is to find a decision rule (i.e., a function which maps X_k to a_k) so as to minimize the time-averaged empirical loss. The idea, illustrated by examples, is to make the system learn various features of the conditional distribution, by choosing an appropriate loss function. The formulation a priori is strong enough to subsume the most general problem, viz., that of learning conditional distributions. One simply lets the domain of $\{a_k\}$ to be the space of probability measures on the domain of $\{Y_k\}$ and let a_k denote the estimated conditional distribution of Y_k given X_k. The accent of [12], however, is on the general learning theoretic issues of the abstract model, whereas our aim is to propose a specific learning scheme with theoretical and empirical support.

Another related strand of work is the estimation of conditional distributions based on density mixtures [13]. Here the conditional mean is estimated from the class of weighted sums of a prescribed family of functions such as Gaussian (more generally, radial basis functions). Our approach differs both in our choice of the f_j's and our choice of the loss function which goes for "conditional expectation of function" rather than visa versa, as already argued.

Yet another development with apparent similarity is the techniques of "bagging" and "arcing" [7] wherein one retrains the same architecture several times with data resampled with replacement (the two differ in the choice of sampling distributions) and then takes a weighted average. The concluding comments of the preceding paragraph also apply here.

Finally, we note that function learning with a single feedforward neural network can be viewed as a special case of RNN corresponding to $N = 1$.

II. Learning Algorithm

In this section we describe the learning algorithm for the proposed RNN architecture. The various steps involved in the algorithm are as follows:

- given $\mathcal{T}_n = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, the set of training samples;

- compute $\bar{f}_n \in \mathcal{F}$ such that $\bar{f}_n = \arg\min_{\bar{f}\in\mathcal{F}} \hat{L}_n(f,\bar{f})$, where $\hat{L}_n(f,\bar{f})$ is given by (1);
- output $\bar{f}_n$.

The above algorithm requires that the function $\bar{f}$ is chosen to be a global minimum of $\hat{L}_n(f,\bar{f})$, which is usually difficult to achieve. Thus we can use a variant of the above algorithm which performs local search to find $\bar{f}$. As described in (§I), each $\bar{f} \in \mathcal{F}$ can be interpreted as the output of a linear unit whose inputs are the outputs of N neural nets, and the linear unit weighs the jth input by α_j. We can train the aforementioned neural nets using the backpropagation algorithm [16] so as to minimize the loss function (1).

More specifically, since each $f_j, j = 1,\ldots,N$, is specified by a set of weights of a neural net, $\bar{f}$ can be parameterized by these sets of weights and α_j's. Therefore, we can perform search in the weight space to find the minimizer $\bar{f}$ of $\hat{L}_n(f,\bar{f})$ in $\mathcal{F}$. We can use the gradient descent approach to accomplish the goal in two steps. In the first step, α_j's are updated as

$$\alpha_j := \alpha_j - \eta \cdot \frac{\partial \hat{L}_n}{\partial \alpha_j}$$

where η is the learning rate parameter. In the second step, the weights in the N neural nets are updated using the backpropagation algorithm, i.e., for all weights w_{rs}

$$w_{rs} := w_{rs} - \eta \cdot \frac{\partial \hat{L}_n}{\partial w_{rs}}$$

where the gradients are computed by backpropagating the error term as illustrated below. Let

$$N_{k,m} = \sum_{j=1}^{N} \alpha_j \cdot \bar{g}_m(f_j(X_k)).$$

Then, corresponding to (36), we have

$$\hat{L}_n(f,\bar{f}) = \frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} a_m \cdot (\bar{g}_m(Y_k) - N_{k,m})^2. \tag{2}$$

Differentiating (2) with respect to α_j, we get

$$\begin{aligned}\frac{\partial \hat{L}_n}{\partial \alpha_j} &= \sum_{m=1}^{M} \frac{\partial \hat{L}_n}{\partial N_{k,m}} \cdot \frac{\partial N_{k,m}}{\partial \alpha_j} \\ &= \frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} 2 \cdot a_m \cdot (N_{k,m} - \bar{g}_m(Y_k)) \cdot \bar{g}_m(f_j(X_k)).\end{aligned} \tag{3}$$

Let w_{st}^r be the weight between neural units s and t, and $O_r(k) = f_r(X_k)$ be the output of the rth neural net, then by the chain rule of differentiation, we have

$$\begin{aligned}\frac{\partial \hat{L}_n}{\partial w_{st}^r} &= \sum_{m=1}^{M} \frac{\partial \hat{L}_n}{\partial N_{k,m}} \cdot \frac{\partial N_{k,m}}{\partial O_r(k)} \cdot \frac{\partial O_r(k)}{\partial w_{st}^r} \\ &= \frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} 2 \cdot a_m \cdot (N_{k,m} - \bar{g}_m(Y_k)) \cdot \alpha_r \\ &\quad \cdot \frac{\partial \bar{g}_m}{\partial O_r(k)} \cdot \frac{\partial O_r(k)}{\partial w_{st}^r}\end{aligned} \tag{4}$$

where $\partial O_r(k)/\partial w_{st}^r, r = 1,\ldots,N$, can be computed using the backpropagation algorithm. In the actual implementation of the algorithm, the weights are updated after every input–output pair seen, as done in usual practice.

In the next few sections we give the simulation results for the above algorithm for a number of problems.

III. Simulations for the i.i.d. Case

We first give the simulations for the i.i.d. case. The simulations for the discrete state-space Markov chains are given in §IV. We discuss the extension to the continuous state-space Markov chains in §V.

As motivated in §I and discussed in detail in the Appendix, we consider the following modified loss function:

$$\begin{aligned}&\hat{L}(f,\bar{f}) \\ &\quad = \frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} a_m \cdot \left(\bar{g}_m(Y_k) - \sum_{j=1}^{M+1} \alpha_j \cdot \bar{g}_m(f_j(X_k))\right)^2.\end{aligned} \tag{5}$$

Note that, in the above, we consider just $N = M+1$ f_j's to match M conditional moment functions $\bar{g}_m, 1 \le m \le M$, even though, theoretically, we are required to take $N = M \cdot O + 1$ f_j's to match $M \cdot O$ functions $\bar{g}_m(y) \cdot h_l(x), 1 \le l \le O, 1 \le m \le M$ (derived in 36). The simulation results indicate that we are justified in doing so.

We consider six synthetic problems. For all the problems, we have used two-layer feedforward neural nets with sigmoidal hidden units and linear output unit to learn $f_j, 1 \le j \le M+1$. The neural networks are trained using the backpropagation algorithm so as to minimize the loss function $\hat{L}(f,\bar{f})$, given in (5) (see §II for other details). As in usual practice, we use the *momentum* term to update the weights. The training is stopped when the incremental change in the average loss function (averaged over λ input–output pair) goes below a pre-specified threshold. We give the results in terms of a table which includes the following entries:

- *#Iter*, the number of iterations taken for training. One iteration stands for the number of input–output pairs over which the loss function is averaged;
- L_o, the error (value of the loss function $\hat{L}(f,\bar{f})$) at the start of training for the given moment functions $\{\bar{g}_m\}$;
- TL_o, the error at the start of training for the test moment functions $\{G_m\}$. (Our choice of moments for test functions is in conformity with the long statistical tradition of using moments as key "features" of a probability distribution. As observed in the introduction, alternative choices are possible);
- L_f, the error at the end of training for the given functions $\{\bar{g}_m\}$;
- TL_f, the error at the end of training for the test functions $\{G_m\}$.

The error for test functions is computed using the loss function (5) with $\{\bar{g}_m\}$ being replaced by the given test moments $\{G_m\}$. We include this error so as to indicate that the learned function $\bar{f}$ approximates well the averages of functions other than the ones used for training $\{\bar{g}_m\}$.

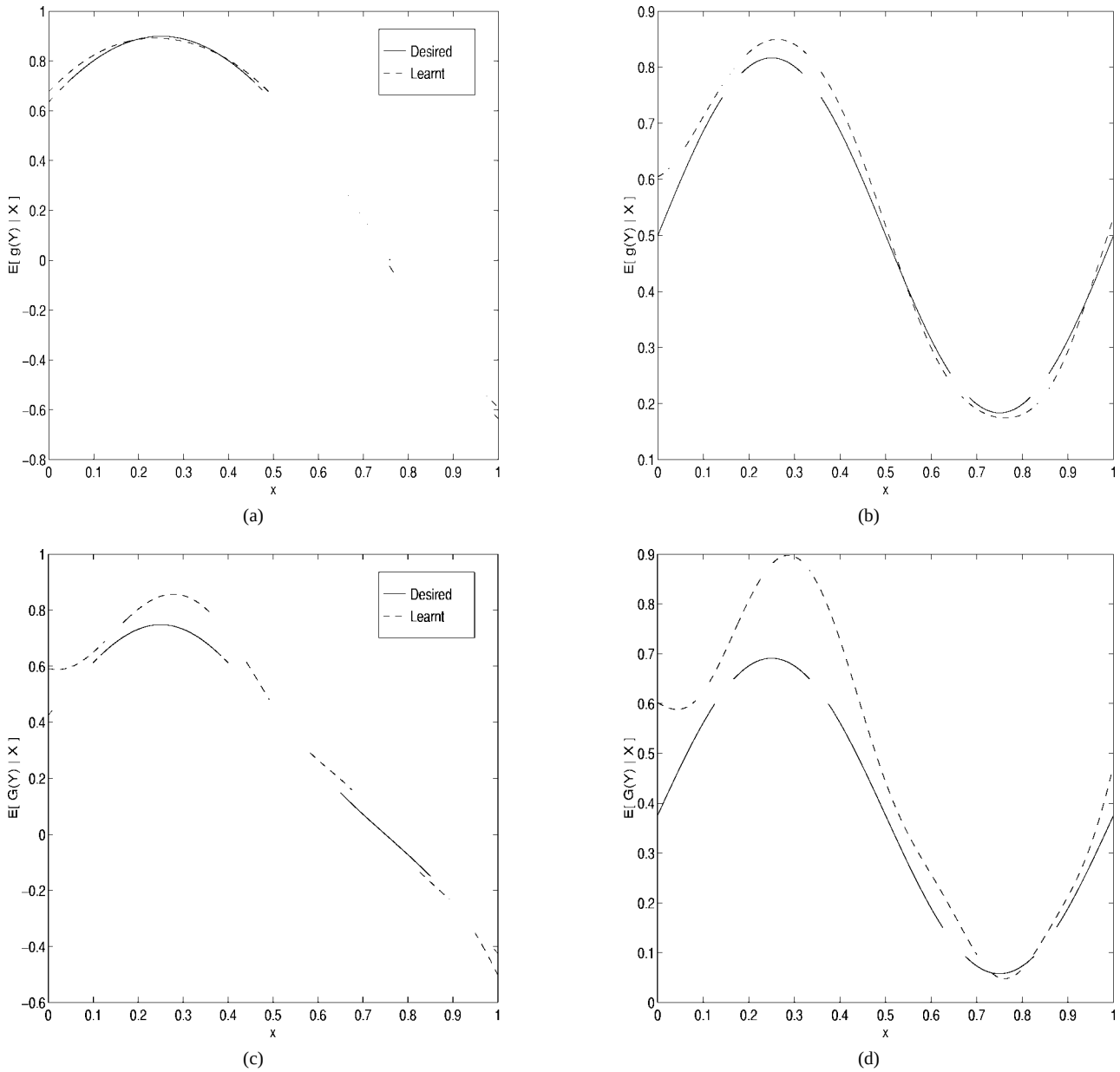

Fig. 1. The desired and the learned moments of the target function of (6): (a) $\bar{g}_1(y) = y$, (b) $\bar{g}_2(y) = y^2$, (c) $G_1(y) = y^3$, and (d) $G_2(y) = y^4$.

In view of Theorem 1, we specify (X_1, Y_1) for the problems by giving the f such that $Y_k = f(X_k, \xi_k)$. The first problem we consider is for the target function

$$f(x, z) = \sin(\pi \cdot (x + 0.5 \cdot z)) \tag{6}$$

where x is uniformly distributed over $[0, 1]$ (recall that $z \in [0, 1]$). We match two $\bar{g}_m(y)$ functions, namely y and y^2, with $a_1 = 0.6$ and $a_2 = 0.4$. The test functions $G_m(y)$ are taken as y^3 and y^4, with $a_1 = 0.6$ and $a_2 = 0.4$. Each of the three neural nets has two hidden nodes. The number of input–output pairs over which the loss function is averaged, λ, is taken to be 200.

Table I gives the performance of the algorithm for the above chosen parameters. Fig. 1 gives the plots of "$E[g_m(Y) \mid X]$ v/s x" for both the training functions, $\bar{g}_m(y)$, and the test functions, $G_m(y)$. As the plots indicate, the algorithm not only performs well on the training functions, but also well-approximates the averages of the test functions. In the next three problems, we take $x \equiv (x_1, x_2) \in \Re^2$, with each component of x being uniformly distributed over $[0, 1]$. The

TABLE I
OUTPUT OF THE ALGORITHM FOR VARIOUS TARGETFUNCTIONS IN THE i.i.d. CASE

S.No.	Target function	#Iter -ations	Initial Error, L_o	Final Error, L_f	Initial test Error, TL_o	Final test Error, TL_f
1	(6)	375	0.185	4.2e-2	0.135	4.41e-2
2	(7)	92	0.343	2.1e-2	0.318	2.75e-2
3	(8)	418	0.150	1.4e-2	0.110	3.23e-2
4	(9)	332	0.178	1.1e-2	6.2e-2	9.60e-3
5	(10)	197	0.108	1.1e-3	1.4e-2	9.04e-4
6	(11)	421	0.328	3.1e-2	0.41	9.50e-2

parameters are taken as in the above problem except that each neural net now has five hidden units. The target functions are

$$f(x,z) = \cos(\pi \cdot x_1 \cdot x_2 \cdot z) \tag{7}$$

$$f(x,z) = \sqrt{(x_1^2 + x_2^2 - 2 \cdot x_1 \cdot x_2 \cdot z)} \tag{8}$$

$$f(x,z) = (x_1 \cdot z)^{x_2}. \tag{9}$$

The performance of the algorithm is summarized in Table I.

Finally, in the last two problems we take $x \equiv (x_1, x_2, x_3) \in \Re^3$. Each component of x has uniform density over $[0,1]$. The parameters are assigned the same values as above. The target functions are

$$f(x,z) = \frac{1}{1+\exp(-\|x\| \cdot z)} \tag{10}$$

$$f(x,z) = \sqrt{\|x\|^2 + z^2}. \tag{11}$$

Again, Table I gives the performance of the algorithm for the above target functions.

IV. Simulations for the Markov Chain Case

In this section we present the simulation results for the discrete state-space Markov chains. As in the i.i.d. case, we consider the loss function having $M+1$ f_m's to match M conditional-moment functions $\{\bar{g}_m\}$, as against $M \cdot O + 1$ f_m's derived in §B; that is, the loss function is taken as

$$\hat{L}(f,\bar{f}) = \frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} a_m \left(\bar{g}_m(X_{k+1}) - \sum_{l=1}^{M+1} \alpha_l \cdot \bar{g}_m(f_l(X_k)) \right)^2. \tag{12}$$

In the following, we consider five problems. The first four problems are for when the state space of the Markov chain is finite, and the last problem is for infinite-but-countable state space. In all the problems, we have used two-layer feedforward neural networks with sigmoidal hidden nodes and linear output nodes, for learning $\alpha_m, f_m, 1 \le m \le M+1$.

Example 1: This example is the *LRU-stack model* for *page referencing behavior of programs* [17, ch. 7]. For $(N+1)$-state model, the transition matrix, $v(i,j), 0 \le i,j \le N$, is defined as follows: Let $0 < b_i < 1$, $\sum_{i=0}^{N} b_i = 1$, and $B_i = \sum_{j=0}^{i} b_j$, then

$$\begin{aligned} v(i,0) &= b_i, \quad 0 \le i \le N \\ v(i,i) &= B_{i-1}, \quad 1 \le i \le N \\ v(i,i+1) &= 1 - B_{i-1}, \quad 0 \le i \le N-1 \\ v(i,j) &= 0, \quad \text{otherwise.} \end{aligned} \tag{13}$$

Here we take $N = 10$ and

$$b_i = \frac{2 \cdot (i+1)}{(N+1) \cdot (N+2)}.$$

As in the i.i.d. case, we match two $\bar{g}_m(y)$ functions, namely, y and y^2, with $a_1 = 0.6$ and $a_2 = 0.4$. The test functions $G_m(y)$ are taken as y^3 and y^4 with $a_1 = 0.6$ and $a_2 = 0.4$. Finally, each of the three neural subnets has six hidden nodes, and the inputs and the outputs of the neural nets lie in [0,1]. That is, the interval [0,1] is divided into N subintervals, and each subinterval stands for one state of the Markov chain.

TABLE II
Output of the Algorithm for the Markov Chain Case

S.No.	Transition Probabilities	#Iter -ations	Initial Error, L_o	Final Error, L_f	Initial test Error, TL_o	Final test Error, TL_f
1	(13)	283	9.6e-2	1.75e-2	2.8e-2	1.0e-2
2	(14)	192	0.19	4.6e-3	0.13	6.4e-4
3	(15)	116	4.4e-2	2.20e-3	8.0e-3	8.7e-4
4	(16)	198	7.5e-2	5.90e-4	1.8e-2	6.7e-4
5	(18)	70	8.1e-2	3.56e-4	6.3e-3	2.2e-5

Table II gives the performance of the algorithm for the above chosen parameters. Fig. 2 gives the plots of "$E[g_m(X_{k+1}) \mid X_k]$ v/s x" for both the training functions, $\bar{g}_m(y)$, and the test functions, $G_m(y)$. As the plots indicate, the algorithm not only performs well on the training functions $\{\bar{g}_m\}$, but also well-approximates the averages of the test functions $\{G_m(y)\}$.

Example 2: Let the transition probabilities be

$$v(i,j) = \frac{\exp\left(-\frac{(i-j)^2}{2\cdot\sigma_i^2}\right)}{\sum_{k=1}^{N}\exp\left(-\frac{(i-k)^2}{2\cdot\sigma_i^2}\right)}, \quad 1 \le i, \quad j \le N. \tag{14}$$

Here we take $N = 20$ and $\sigma_i = 50/i$. The parameters are the same as in the previous example except that the number of hidden nodes in each neural nets is two. The performance of the algorithm is summarized in Table II.

Example 3: The next example corresponds to the *Bernoulli-Laplace diffusion model* [11, ch. 15]. The transition probabilities are given by

$$\begin{aligned} v(i,i-1) &= \left(\frac{i}{N}\right)^2 \\ v(i,i+1) &= \left(\frac{N-i}{N}\right)^2 \\ v(i,i) &= \frac{2\cdot i\cdot(N-i)}{N^2}, \quad 0 \le i \le N. \end{aligned} \tag{15}$$

Here we take $N = 10$. The parameters are assigned the same values as in the previous example. Table II summarizes the performance of the algorithm for this Markov chain.

Example 4: In this example, we consider the *Ehrenfest diffusion model*. The Markov chain has $(N+1)$ states, and the transition probabilities are

$$\begin{aligned} v(i,i+1) &= 1 - \frac{i}{N} \\ v(i,i-1) &= \frac{i}{N}, \quad 0 \le i \le N. \end{aligned} \tag{16}$$

Here we take $N = 50$. The parameters are taken as in the previous examples. The output of the algorithm is given in Table II.

Example 5: Finally, we consider the M/G/1 Queuing system, i.e., a single-server queuing system whose arrival process

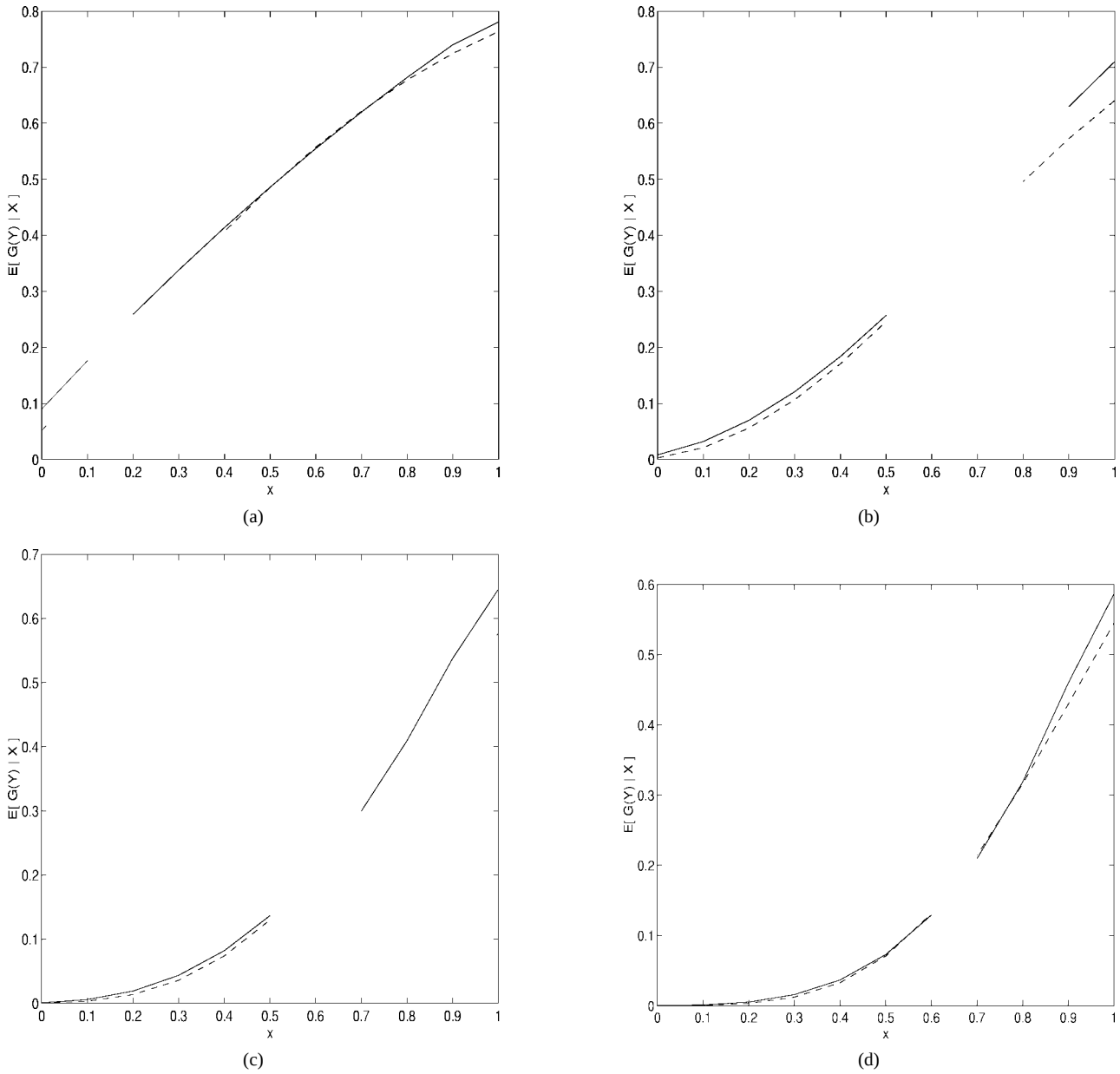

Fig. 2. The desired and the learned moments for the Markov chain with transition probabilities given in (13): (a) $\bar{g}_1(y) = y$, (b) $\bar{g}_2(y) = y^2$, (c) $G_1(y) = y^3$, and (d) $G_2(y) = y^4$.

is Poisson with the average arrival rate λ. The service times are independent and identically distributed with some unknown but fixed distribution. Let $N(t)$ denote the number of customers in the system (those in the queue plus any in service) at time t. Since, in general, $\{N(t), t \geq 0\}$ need not be a Markov chain, we consider the process $\{X_k, k = 0, 1, 2, \ldots\}$, where X_k is the number of customers in the system at the time of departure of the kth customer. Now $\{X_k\}$ can be shown to be a Markov chain. Furthermore, it can be shown that the limiting distribution of the number of customers $N(t)$ observed at an arbitrary point in time is identical to the number of customers observed at times of departures of customers [19, ch. 5], i.e.,

$$\lim_{t\to\infty} P[N(t) = n] = \lim_{k\to\infty} P[X_k = n]. \tag{17}$$

Let Y_k denote the number of arrivals during the service time of the kth customer, then

$$X_{k+1} = \begin{cases} X_k - 1 + Y_{k+1}, & \text{if } X_k > 0 \\ Y_{k+1}, & \text{if } X_k = 0. \end{cases}$$

And the transition probabilities of the Markov chain $\{X_k\}$ are given by

$$\begin{aligned} v(i,j) &= P(X_{k+1} = j \mid X_k = i) \\ &= \begin{cases} P(Y_{k+1} = j - i + 1), & \text{if } i \neq 0, \quad j \geq i - 1 \\ P(Y_{k+1} = j), & \text{if } i = 0, \quad j \geq 0 \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{18}$$

Since the service times are assumed to be i.i.d., then so are $\{Y_k\}$. Let η be the limiting probability measure for P. We are then interested in learning an approximation of η so as

to match a given set of moments $\{\bar{g}_m(\cdot), m = 1, \ldots, M\}$ of the steady-state average number of customers in the system, which by (17) is

$$\begin{aligned}\lim_{t\to\infty} E[\bar{g}_m(N(t))] &= \lim_{k\to\infty} E[\bar{g}_m(X_k)] \\ &= \sum_j \bar{g}_m(j)\cdot\eta(j) \\ &= \sum_j \bar{g}_m(j)\cdot\left(\sum_i v(i,j)\cdot\eta(i)\right) \\ &= \sum_j\sum_i \bar{g}_m(j)\cdot\mu(i,j)\end{aligned}$$

where μ is the occupation measure for (v, η), i.e., $\mu(i,j) = \eta(i)\cdot v(i,j)$. Hence the problem falls in the framework analyzed in §B.

In the following, we consider the service process to be hyperexponential with two components, i.e., if T denotes the service-time random variable, then

$$P[T \le t] = \beta\cdot(1 - e^{-\mu_1 t}) + (1-\beta)\cdot(1 - e^{-\mu_2 t}). \quad (19)$$

Here we take $\beta = 0.3$, $\mu_1 = .6$ and $\mu_2 = 1.5$. The average arrival rate is $\lambda = 1.0$. We consider the first and the second moments of X_k for matching, i.e., $\bar{g}_1 = y$ and $\bar{g}_2 = y^2$; while the test moment functions are $G_1 = y^3$ and $G_2 = y^4$. Finally, each of the three neural subnets in the learning structure has six hidden nodes. The performance of the algorithm is given in Table II.

V. Extension to Continuous State Space

In this section we discuss the simulation results for the continuous state-space Markov chains. The problem formulation remains the same as in the discrete case except that now X_k's take values in a closed bounded subset C of $\Re$; that is, given a finite string $\{X_k, 1 \le k \le m\}$, we desire to learn the transition probability distribution $x \to v(x, dy)$. A result similar to Theorem 3 can be obtained so to convert the problem of learning v to that of learning an appropriate function, i.e., it can be shown that there exist a measurable function $f : C \times [0,1] \to C$, such that

$$X_{k+1} = f(X_k, \xi_k), \quad k \ge 1$$

where ξ_k are i.i.d. random variables, uniformly distributed over $[0,1]$ (for details see [3]). As in the discrete case, we minimize the following loss function:

$$\begin{aligned}&\hat{L}(f,\bar{f}) \\ &\quad = \frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} a_m\left(\bar{g}_m(X_{k+1}) - \sum_{l=1}^{M+1}\alpha_l\cdot\bar{g}_m(f_l(X_k))\right)^2\end{aligned} \quad (20)$$

so as to match M conditional moments $\{\bar{g}_m\}$ of X_{k+1} given X_k. A result similar to Theorem 4 can be proved by identical arguments, which can justify using such a loss function.

In the following, we consider five problems. We use feedforward neural networks with sigmoidal hidden nodes and linear output nodes, for learning $\alpha_m, f_m, 1 \le m \le M+1$.

TABLE III
Output of the Algorithm for Various Target Functions in the Continuous State–Space Markov Chain Case

S.No.	Target function $f(x,z)$	#Iter-ations	Initial Error, L_o	Final Error, L_f	Initial test Error, TL_o	Final test Error, TL_f
1	(21)	117	0.30	4.5e-2	.20	6.3e-2
2	$2/3\cdot\sqrt{x^2+z^2}$	115	6.2e-2	8.8e-3	1.7e-3	5.1e-4
3	$\sin(\pi\cdot x\cdot z)$	124	9.5e-2	2.14e-2	7.9e-2	2.8e-2
4	$1/(1+\exp(\ x\cdot z))$	155	7.3e-2	9.5e-4	1.7e-2	6.8e-4
5	(22)	178	0.11	2.4e-2	1.8e-2	1.5e-2

As indicated above, we specify the transition probability distribution of $\{X_k\}$ for the problems by giving the f such that $X_{k+1} = f(X_k, \xi_k)$. The first problem we consider is for the target function

$$f(x,z) = \sin(\pi\cdot(x + 0.5\cdot z)) \quad (21)$$

where x is uniformly distributed over $[0,1]$. As in the discrete case, we match two $\bar{g}_m(y)$ functions: namely, y and y^2, with $a_1 = 0.6$ and $a_2 = 0.4$. The test functions $G_m(y)$ are taken as y^3 and y^4, with $a_1 = 0.6$ and $a_2 = 0.4$. Each of the three neural nets has two hidden nodes. Table III gives the performance of the algorithm for the above chosen parameters. It also summarizes the results for various other target functions.

Finally, we discuss an example from queuing theory. Consider a GI/G/1 queue. Let $\{\alpha_n\}$ be the sequence of inter-arrival times to the queue, and let $\{\sigma_n\}$ be the sequence of service times. The time spent in the system by the nth customer, denoted by X_n, satisfies the following recurrence relation:

$$\begin{aligned}X_{n+1} = (X_n + \sigma_{n+1} - \alpha_{n+1})I\{\alpha_{n+1} \le X_n\} \\ + \sigma_{n+1}I\{\alpha_{n+1} > X_n\}.\end{aligned} \quad (22)$$

Here we take both the service process $\{\sigma_n\}$ and the arrival process $\{\alpha_n\}$ to be hyperexponential (19). The parameters for $\{\sigma_n\}$ are: $\beta = .7$, $\mu_1 = 6.0$ and $\mu_2 = 3.0$; and for $\{\alpha_n\}$ are: $\beta = .6$, $\mu_1 = 4.0$ and $\mu_2 = 2.0$. The performance of the algorithm is given in Table III.

VI. Conclusion

We have considered the problem of learning the stochastic dependence of one random variable on another, from a finite string of copies of the pair. The problem formulation has been adapted from [4]. We have improved upon some of the results of [4], by suggesting an alternate loss function when the given string consists of i.i.d. copies of the random-variable pair. We have also extended the results for the case when the given string is a Markov chain over a countable state space, as well as for when the state space is continuous. We have given simulation results to indicate that our approach performs well on standard problems.

Some possible extensions we hope to pursue in the future are as follows.

- Choosing $\{f_j\}$ from a mixed family of Neural Network architectures (e.g., some could be based on sigmoids, others on radial basis functions). This may capture different

features of the data better. One can also consider different training algorithms for different f_j's as in [14].
- The RNN architecture may provide a useful parameterization for *Model Reference Adaptive Control* of nonlinear stochastic systems.

APPENDIX
MATHEMATICAL FORMALISM

Here we give a detailed mathematical motivation for our formulation of the problem of learning stochastic dependences.

A. The i.i.d. Case

Let $(X_k, Y_k), k \geq 1$ be independent and identically distributed pairs of random variables taking values in $C_3 = C_1 \times C_2$, where $C_1 \subset \Re^{\mathcal{N}}$ and $C_2 \subset \Re^{\mathcal{d}}$ are closed bounded subsets. Let μ, ν, η denote the laws of $(X_1, Y_1), X_1, Y_1$, respectively. Then we can write $\mu(dx, dy)$ as

$$\mu(dx, dy) = \nu(dx) \cdot v(x, dy) \tag{23}$$

where $x \rightarrow v(x, dy)$ is the regular conditional law of Y_1 given X_1, defined ν-a.s. uniquely [6, ch. 3]. Now the problem is to learn the conditional law $x \rightarrow v(x, dy)$. To this end, the problem is first converted into another equivalent problem via the following theorem [2], [4, Theorem 1].

Theorem 1: Given an i.i.d. sequence (X_k, Y_k), $k \geq 1$, as above on some probability space, there exist a measurable function $f : C_1 \times [0,1] \rightarrow C_2$ and a sequence of i.i.d. random variables $\{\xi_k\}$ defined on a possibly augmented probability space such that each ξ_k is uniformly distributed over $[0,1]$, $\{\xi_k, X_k, k \geq 1\}$ are independent and

$$Y_k = f(X_k, \xi_k), \quad k \geq 1. \tag{24}$$

Hence the problem is reduced to that of learning the function f in (24).

Let $\mathcal{F}_M$ be the set of measurable functions $C_1 \times [0,1] \rightarrow C_2$, and $\mathcal{F} \subset \mathcal{F}_M$ a prescribed subset thereof that will serve as the *hypothesis space* for the learning algorithm, i.e., the approximation of f, denoted $\bar{f}$, is sought from $\mathcal{F}$. Let $L : \mathcal{F} \times \mathcal{F}_M \rightarrow \Re^+$ be a loss function, i.e., $L(f, \bar{f})$ has the interpretation as the loss associated with deciding $\bar{f}$ when the target function is f. Then, the problem is to choose a function $\bar{f} \in \mathcal{F}$ that minimizes $L(f, \bar{f})$.

We next discuss our choices of the loss function $L(f, \bar{f})$ and the hypothesis space $\mathcal{F}$.

1) Choice of Loss Function: This subsection gives some preliminary motivation for the choice of loss function.

If we had a control on the random variables $\{\xi_k\}$ featuring in (24), we could learn f by minimizing the mean-square difference between Y_k and $\bar{f}(X_k, \xi_k)$, where $\bar{f} \in \mathcal{F}$ is the current approximation of f. But $\{\xi_k\}$ are not known except for their distribution. Thus the next best thing to do is to generate $\{\hat{\xi}_k\}$ that mimic $\{\xi_k\}$ in law, i.e., $\{\hat{\xi}_k\}$ are i.i.d. uniformly distributed on $[0,1]$ such that $\{\hat{\xi}_k, X_k, k \geq 1\}$ is an independent family. One may then compare $\{Y_k\}$ with $\{\hat{Y}_k\}$, $\hat{Y}_k = \bar{f}(X_k, \hat{\xi}(k))$, $k \geq 1$, for a proposed $\bar{f}$. This comparison cannot be made sample path-wise, as $\{\hat{\xi}_k\}$, $\{\xi_k\}$ are unrelated except in law. Hence we can compare (X_k, Y_k), $(X_k, \hat{Y}_k)$ only in law, i.e., if $\hat{\mu}$ is the law of $(X_1, \hat{Y}_1)$, we can compare μ, $\hat{\mu}$ as elements of $P(C_3)$(=the space of probability measures on C_3 with the topology of weak convergence). A standard metric on $P(C_3)$ is the Prohorov metric ρ defined by [6, ch. 2]

$$\rho(\mu_1, \mu_2) = \inf\{\epsilon > 0 \mid \mu_1(A) < \mu_2(A^\epsilon) + \epsilon, \\ \mu_2(A) < \mu_1(A^\epsilon) + \epsilon, \text{ for all Borel } A \subset C_3\}$$

where $A^\epsilon = \{x \in C_3 \mid \exists y \in A \text{ such that } \| x - y \| \leq \epsilon\}$. ρ is, however, not computationally tractable, and hence an alternate equivalent metric $\bar{\rho}$ is used, where

$$\bar{\rho}(\mu_1, \mu_2) = \sum_{n=1}^{\infty} 2^{-n} \left| \int g_n \, d\mu_1 - \int g_n \, d\mu_2 \right| \tag{25}$$

where $\{g_n\} \subset \mathcal{C}(C_3)$ (the space of all real-valued continuous functions on C_3) is a bounded countable *convergence determining* class [6, ch. 2], i.e., it satisfies

$$\int g_i \, d\mu_n \rightarrow \int g_i \, d\mu, \quad \forall i \Longrightarrow \mu_n \rightarrow \mu \text{ in } P(C_3).$$

Correspondingly, the loss function $L(f, \bar{f})$ can be taken as

$$L(f, \bar{f}) = \sum_{i=1}^{N} a_i \cdot \left(\int g_i \, d\mu - \int g_i \, d\hat{\mu} \right)^2 \tag{26}$$

where $a_i > 0, \sum_{i=1}^{N} a_i = 1$. (26) differs from (25) in the following: It has only a finite number of terms in the sum (to make the loss function computationally tractable), it has more general *weights* $\{a_n\}$, and finally, the modulus has been replaced by its square (to make it differentiable with respect to certain *design parameters* to be defined later).

A natural approximation for μ and $\hat{\mu}$ is given by the *empirical measures* μ_n and $\hat{\mu}_n$, respectively, defined by

$$\mu_n(A) = \frac{1}{n} \sum_{m=1}^{n} I\{(X_m, Y_m) \in A\}$$
$$\hat{\mu}_n(A) = \frac{1}{n} \sum_{m=1}^{n} I\{(X_m, \hat{Y}_m) \in A\}$$

for A Borel in C_3. Thus the empirical loss is now defined as

$$L_n(f, \bar{f}) = \sum_{i=1}^{N} a_i \cdot \left(\int g_i \, d\mu_n - \int g_i \, d\hat{\mu}_n \right)^2. \tag{27}$$

We shall successively modify the above loss function further after the choice of $\mathcal{F}$ is made in the next subsection.

2) Choice of Hypothesis Space: The choice of the hypothesis space, $\mathcal{F}$, is based on the following observation. Given the loss function (26), we need only to find some $\bar{f}$ such that

$$E[g_i(X_1, f(X_1, \xi_1))] = E[g_i(X_1, \bar{f}(X_1, \xi_1))], \quad 1 \leq i \leq N \tag{28}$$

where $f(\cdot, \cdot)$ is as in (24). The next theorem [4, Theorem 3] gives one such $\bar{f}$.

Theorem 2: There exists a partition $\{A_1, \ldots, A_{N+1}\}$ of $[0,1]$ into intervals (some of them possibly empty) and measurable functions $f_1, \ldots, f_{N+1} : C_1 \to C_2$ such that $\bar{f} : C_1 \times [0,1] \to C_2$ defined by

$$\bar{f}(x,z) = \sum_{i=1}^{N+1} I\{z \in A_i\} \cdot f_i(x) \tag{29}$$

satisfies (28).

Proof [4]: Since $C_1 \times C_2$ is compact, so is $P(C_1 \times C_2)$ by Prohorov's theorem [6, ch. 2]. Let $Q_\nu = \{\mu \in P(C_1 \times C_2) \mid \mu \text{ is of the form } (23)\}$. Then $Q_\nu \subset P(C_1 \times C_2)$ is closed (hence compact) and convex. By Lemma 2.2 of [3], extreme points of Q_ν are precisely those μ for which $x \to v(x, dy)$ as in (23) is a Dirac measure for ν-a.s. x. Now consider

$$\tilde{Q} = Q_\nu \cap \left\{ \bar{\mu} \,\middle|\, \int g_i \, d\bar{\mu} = E[g_i(X_1, Y_1)], 1 \le i \le N \right\}.$$

By a result of [10] (see also [18]), the extreme points of $\tilde{Q}$ can be expressed as a convex combination of at most $N+1$ extreme points of Q_ν. That is, they correspond to $\bar{\mu} \in Q_\nu$ of the form

$$\begin{aligned} \bar{\mu}(dx, dy) &= \sum_{j=1}^{N+1} \alpha_j \cdot \left[\nu(dx) \cdot \delta_{f_j(x)}(dy) \right] \\ &= \nu(dx) \cdot \left[\sum_{j=1}^{N+1} \alpha_j \cdot \delta_{f_j(x)}(dy) \right] \end{aligned}$$

for some $\alpha_j \in [0,1]$ with $\sum_j \alpha_j = 1$, $f_j : C_1 \to C_2$ measurable and $\delta_z =$ the δ-function (for mathematicians, a "Dirac measure") at z for $z \in C_2$. Now partition $[0,1]$ into $A_1, \ldots, A_{N+1}$ with the length of $A_j = \alpha_i, 1 \le j \le N+1$, and define $\bar{f}$ as in (29). The claim follows. □

Thus the search of $\bar{f}$ can now be restricted to those of the form (29). Correspondingly, we can modify the empirical loss function $\tilde{L}(f, \bar{f})$ (via a simple "conditioning") as

$$\begin{aligned} \hat{L}_n(f, \bar{f}) &= \sum_{i=1}^{N} a_i \cdot \left(\int g_i \, d\mu_n - \sum_{j=1}^{N+1} \alpha_j \int (g_i \circ \tilde{f}_j) \, d\nu_n \right)^2 \\ &= \sum_{i=1}^{N} a_i \cdot \left(\frac{1}{n} \sum_{k=1}^{n} g_i(X_k, Y_k) \right. \\ &\quad \left. - \frac{1}{n} \sum_{k=1}^{n} \sum_{j=1}^{N+1} \alpha_j \cdot g_i(X_k, f_j(X_k)) \right)^2 \end{aligned} \tag{30}$$

where

$\circ$	composition of functions;
$\bar{\alpha} = [\alpha_1, \ldots, \alpha_{N+1}]$	satisfies $\alpha_i \in [0,1], 1 \le i \le N+1$, and $\sum_{i=1}^{N+1} \alpha_i = 1$;
$f_j : C_1 \to C_2$	measurable with $\tilde{f}_j : C_1 \to C_1 \times C_2$ defined correspondingly by $\tilde{f}_j(x) = (x, f_j(x))$ (i.e., $\tilde{f}_j(x)$ has two components the first being $x \in C_i$, and the second $f_j(x) \in C_2$);
ν_n	empirical measure for the input process, defined by

$$\nu_n(A) = \frac{1}{n} \sum_{m=1}^{n} I\{X_m \in A\}, A \text{ Borel in } C_1.$$

Note that (30) does not require the knowledge of $\{\hat{\xi}_k\}$. Equation (30) was the loss function used in [4]. We shall modify it further.

If in (28) $\{g_i\}$ are taken to be functions of only the second variable, then only the law of Y_k is learned and not the joint law of (X_k, Y_k). But, suppose we are given M functions, $\bar{g}_1, \ldots, \bar{g}_M : C_2 \to \Re$ (underscores the fact that the domain of $\bar{g}_m$'s is C_2, distinct from that of g_i's, which was C_3), and we are required to match their conditional expectations, i.e.

$$\begin{aligned} E[\bar{g}_m(f(X_1, \xi_1)) \mid X_1] = E[\bar{g}_m(\bar{f}(X_1, \xi_1)) \mid X_1], \\ \nu\text{-a.s.}, \quad 1 \le m \le M \end{aligned} \tag{31}$$

which can be equivalently written as

$$\begin{aligned} E[h_l(X_1) \cdot \bar{g}_m(f(X_1, \xi_1))] = E[h_l(X_1) \cdot \bar{g}_m(\bar{f}(X_1, \xi_1))], \\ l = 1, 2, \ldots; \quad 1 \le m \le M \end{aligned} \tag{32}$$

where $\{h_l\} \subset \mathcal{C}(C_1)$ is a bounded countable convergence determining class (i.e., $\int h_l d\nu_n \to \int h_l d\nu, \forall l \Rightarrow \nu_n \to \nu$ in $P(C_1)$). Again, for computational tractability, we desire to match (32) for only finitely many h_l, i.e.,

$$\begin{aligned} E[h_l(X_1) \cdot \bar{g}_m(f(X_1, \xi_1))] = E[h_l(X_1) \cdot \bar{g}_m(\bar{f}(X_1, \xi_1))], \\ 1 \le l \le O; \quad 1 \le m \le M. \end{aligned} \tag{33}$$

Now, by Theorem 2, there exists a $\bar{f}$ which satisfies (33) and can be expressed as

$$\bar{f}(x,z) = \sum_{i=1}^{M \cdot O + 1} I\{z \in A_i\} \cdot f_i(x). \tag{34}$$

The ideal loss function corresponding to (31) in our setup would be

$$\begin{aligned} L(f, \bar{f}) = \sum_{m=1}^{M} a_m \int \left(\int_0^1 \bar{g}_m(f(x,z)) \, dz \right. \\ \left. - \int_0^1 \bar{g}_m(\bar{f}(x,z)) \, dz \right)^2 \nu(dx). \end{aligned}$$

Let $\hat{\mu}_n(dx, dy) = \nu_n(dx) \cdot \bar{v}_n(x, dy)$. Then $\bar{v}_n$ is the "empirical conditional law" of Y_k given X_k. Now one approximation of the above loss function is

$$\begin{aligned} L_n(f, \bar{f}) = \sum_{m=1}^{M} a_m \int \left(\int \bar{g}_m(y) v(x, dy) \right. \\ \left. - \int \bar{g}_m(y) \bar{v}_n(x, dy) \right)^2 \nu_n(dx). \end{aligned} \tag{35}$$

This is a better choice for the loss function than (30) because whenever (35) is zero, then so is (30), but the converse, in general, is not true. However, this loss function requires the

knowledge of $v(x,\cdot)$, which was the goal we started with. Hence we consider the following loss function instead:

$$\hat{L}_n(f,\bar{f}) = \frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} a_m \cdot \left(\bar{g}_m(Y_k) - \sum_{j=1}^{N+1}\alpha_j \bar{g}_m(f_j(X_k))\right)^2 \tag{36}$$

where $N = M \cdot O$ (notation chosen so that we can give a unified exposition for (30) and (36) in the following). Equation (36) is minimized in the limit (i.e., as $n \to \infty$), by exactly the same set of solutions that minimizes (35). That can be seen as follows: (36) can be rewritten as (by adding and subtracting $\int \bar{g}_m(y)v(x,dy)$ within the square, and then expanding the square appropriately)

$$\begin{aligned}\hat{L}_n(f,\bar{f}) &= \frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} a_m \cdot \left(\left(\bar{g}_m(Y_k) - \int \bar{g}_m(y)v(x,dy)\right)^2\right. \\ &\quad + \left(\int \bar{g}_m(y)v(x,dy) - \sum_{j=1}^{N+1}\alpha_j \bar{g}_m(f_j(X_k))\right)^2 \\ &\quad + 2\left(\bar{g}_m(Y_k) - \int \bar{g}_m(y)v(x,dy)\right) \\ &\quad \left.\times \left(\int \bar{g}_m(y)v(x,dy) - \sum_{j=1}^{N+1}\alpha_j \bar{g}_m(f_j(X_k))\right)\right).\end{aligned} \tag{37}$$

In (37), the first term in the summand is independent of the parameters α_j and $f_j, 1 \le j \le N+1$. The last term goes to 0 as $n \to \infty$, as follows:

$$\begin{aligned}&\lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n}\sum_{m=1}^{M} a_m \cdot \left(\bar{g}_m(Y_k) - \int \bar{g}_m(y)v(x,dy)\right) \\ &\quad \cdot \left(\int \bar{g}_m(y)v(x,dy) - \sum_{j=1}^{N+1}\alpha_j \bar{g}_m(f_j(X_k))\right) \\ &\quad = \sum_{m=1}^{M} a_m \cdot E\left[(\bar{g}_m(Y) - E\left[\bar{g}_m(Y) \mid X\right]) \right. \\ &\quad\quad \left. \cdot \left(E[\bar{g}_m(Y) \mid X] - \sum_{j=1}^{N+1}\alpha_j \bar{g}_m(f_j(X))\right)\right] \\ &\quad = \sum_{m=1}^{M} a_m \cdot [E\left[E[(\bar{g}_m(Y) - E[\bar{g}_m(Y) \mid X]) \mid X\right] \\ &\quad\quad \cdot \left. \left[E[\bar{g}_m(Y) \mid X] - \sum_{j=1}^{N+1}\alpha_j \bar{g}_m(f_j(X))\right)\right] \\ &\quad = 0\end{aligned}$$

The only remaining middle term in (37) corresponds exactly to (35) (after using (34) in the latter). Hence the claim follows.

Now $\hat{L}_n$ is minimized over the choice of $\bar{\alpha}$ from among $(N+1)$-dimensional probability vectors, as well as over $\{f_j\}$ from the set of measurable maps from C_1 to C_2. We can restrict our attention to continuous f_j, as continuous functions are dense in L^p-spaces for $1 \le p < \infty$, and by Lusin's theorem [15], given any measurable function on C_1, there exists a continuous function which agrees with the given function outside a set of arbitrarily small positive measure.

We can now use some parameterized family of continuous functions to learn f_j's. In particular, we can consider the family of continuous functions that can be formulated as a specific type of feedforward neural network: The class of two-layer networks with an unrestricted number of sigmoidal units in the first layer and a linear unit in the second layer. Hence the learning structure consists of $N+1$ two-layer feedforward networks, stacked in parallel and receiving common inputs, and a linear unit in the last layer whose $N+1$ inputs are the outputs of the aforementioned nets, and whose corresponding weights are $\alpha_j, 1 \le j \le N+1$.

B. Markov Chain Case

In this section, we generalize the results of the previous section for Markov chains. It is worth noting that the case of Markov chains is not fundamentally different from the i.i.d. case insofar as we are still being presented pairs (X_k, Y_k) with the task of finding the probabilistic dependence of the second component on the first, except for the additional feature: $Y_k = X_{k+1}$. There is, however, a subtle difference. The expectation in the expected error criterion in i.i.d. case was with respect to a distribution explicitly given as the joint law of the pair, now it is with respect to the stationary distribution of the process defined only implicitly through the dynamics. Nevertheless, beyond this technical issue, the two problems are really the same, but the Markov chain case has the added attraction of linking this theory to the "system identification" problem of electrical engineers of which it is the simplest specimen.

Let $\{X_k\}$ be a stationary Markov chain on a countable state space S. We assume, without loss of generality, that the states are labeled as $\{1,2,3,\ldots\}$. Let $v = ((v(i,j)))_{i,j\in S}$ be the transition probability matrix for $\{X_k\}$, i.e., the probability $P(X_{k+1} = j \mid X_k = i) = v(i,j)$. If $\eta \in P(S)$ is an invariant probability measure under v, we associate with the pair (v,η) an "occupation measure" $\mu \in P(S \times S)$ defined by

$$\mu(i,j) = \eta(i) \cdot v(i,j), \quad i,j \in S. \tag{38}$$

The set of all occupation measures, with prescribed η, will be denoted by G. We are interested in learning the transition probability matrix v. For this purpose, we first convert the problem into that of learning an appropriate function via the following result (analogous to Theorem 1).

Theorem 3 [3]: Given a S-valued Markov chain $\{X_k\}$, there exists a measurable function $f : S \times [0,1] \to S$ and a sequence of i.i.d. random variables $\{\xi_k\}$ such that each ξ_k is uniformly distributed over $[0,1]$, $\{\xi_k, X_k, k \ge 1\}$ are independent and

$$X_{k+1} = f(X_k, \xi_k), \quad k \ge 1. \tag{39}$$

Remark 1: Since the state space S is countable, we can obtain an explicit representation of X_{k+1} in terms of X_k and ξ_k (the latter being as given in the statement of the theorem)

$$X_{k+1} = \sum_{i=1}^{\infty} I\left\{\sum_{j=0}^{i-1} v(X_k, j) < \xi_k \leq \sum_{j=0}^{i} v(X_k, j)\right\} \cdot i$$

where I is the indicator function and $v(\cdot, 0) = 0$.

As before, we would like to learn the function f. But as the samples corresponding to ξ are not available, we instead learn an approximation of f, denoted $\bar{f}$, by matching N moment functions $g_l : S \times S \to \Re, 1 \leq l \leq N$, i.e.

$$E[g_l(X_k, f(X_k, \xi_k))] = E[g_l(X_k, \bar{f}(X_k, \xi_k))], \quad 1 \leq l \leq N \tag{40}$$

or, equivalently, to learn an approximation of μ, denoted $\bar{\mu}$, such that

$$\sum_{i \in S} \sum_{j \in S} g_l(i,j) \cdot \mu(i,j) = \sum_{i \in S} \sum_{j \in S} g_l(i,j) \cdot \bar{\mu}(i,j), \quad 1 \leq l \leq N. \tag{41}$$

We next derive a result analogous to Theorem 2 to show that there exists a $\bar{\mu} \in G$ which satisfies (41) and can be written as a convex combination of at most $N+1$ *conditional* Dirac measures. (See also [5], for some related results.)

Theorem 4: There exists a partition $\{A_1, \ldots, A_{N+1}\}$ of $[0,1]$ into intervals (some of them possibly empty) and measurable functions $f_1, \ldots, f_{N+1} : S \to S$ such that $\bar{f} : S \times [0,1] \to S$ defined by

$$\bar{f}(i,z) = \sum_{m=1}^{N+1} I\{z \in A_m\} \cdot f_m(i) \tag{42}$$

satisfies (40).

Proof: As before, let G be the set of all occupation measures μ with prescribed η (38). Then, by Lemma 2.2 of [3], extreme points of G are precisely those μ for which $i \to v(i,j)$ as in (38) is a Dirac measure for η-a.s. i. Now let

$$\tilde{G} = \left\{ \bar{\mu} \in G \mid \sum_{i \in S} \sum_{j \in S} g_l(i,j) \cdot \bar{\mu}(i,j) = E[g(X_k, X_{k+1})], 1 \leq l \leq N \right\}.$$

Then, by Dubins' Theorem [10] (see also [18]), the extreme points of $\tilde{G}$ can be expressed as a convex combination of at most $N+1$ extreme points of G. That is, they correspond to $\bar{\mu} \in G$ of the form

$$\bar{\mu}(i,j) = \sum_{m=1}^{N+1} \alpha_m \cdot \mu_m(i,j) = \sum_{m=1}^{N+1} \alpha_m \cdot \eta(i) \cdot \delta_{f_m(i)}(j) \tag{43}$$

for some $\alpha_m \in [0,1]$ with $\sum_m \alpha_m = 1$, and $f_m : S \to S$. Now partition $[0,1]$ into intervals $A_1, \ldots, A_{N+1}$ with the length of $A_m = \alpha_m$, and define $\bar{f}$ as in (42). The claim follows. □

As in the i.i.d. case, we learn $f_m, \alpha_m, 1 \leq m \leq N+1$ by minimizing the following loss function:

$$\hat{L}_n(f, \bar{f}) = \sum_{l=1}^{N} a_l \left(\frac{1}{n} \sum_{k=1}^{n} g_l(X_k, X_{k+1}) - \frac{1}{n} \sum_{k=1}^{n} \sum_{m=1}^{N+1} \alpha_m \cdot g_l(X_k, f_m(X_k)) \right)^2. \tag{44}$$

If instead of matching the joint moments $\{g_l(\cdot,\cdot)\}$, we desire to match the conditional moments $\bar{g}_m(Y), m = 1, \ldots, M$, i.e.

$$\sum_{j \in S} \bar{g}_m(j) \cdot \mu(i,j) = \sum_{j \in S} \bar{g}_m(j) \cdot \bar{\mu}(i,j), \quad \eta\text{-a.s.}; \quad 1 \leq m \leq M \tag{45}$$

or, its approximation

$$\sum_{i \in S} \sum_{j \in S} h_l(i) \cdot \bar{g}_l(j) \cdot \mu(i,j) = \sum_{i \in S} \sum_{j \in S} h_l(i) \cdot \bar{g}_l(j) \cdot \bar{\mu}(i,j), \quad 1 \leq l \leq O; \quad 1 \leq l \leq M.$$

Then, as argued in the i.i.d. case, the corresponding loss function is

$$\hat{L}_n(f, \bar{f}) = \frac{1}{n} \sum_{k=1}^{n} \sum_{l=1}^{M} a_l \left(\bar{g}_l(X_{k+1}) - \sum_{m=1}^{N+1} \alpha_m \cdot \bar{g}_l(f_m(X_k)) \right)^2 \tag{46}$$

where $N = M \cdot O$.

As before, we consider a parameterized family of functions to learn f_m from. Here, neural networks, in general, may not be the best choice as the functions are maps over a countable space. But, if the chain is on an integer lattice (i.e., on a subset of Z^d for some $d \geq 1$) whereby it can be embedded in a Euclidean space, and the functions to be learned have some "nice" properties, e.g., $|f(i) - f(j)| \leq K \cdot |i - j|, \forall i, j \in S$, for some $K < \infty$, then we can indeed use neural networks in conjunction with an appropriate quantizer, to learn the functions f_m's. In such a case, a learning algorithm similar to one for the i.i.d. case, can be used to train the neural nets so as to minimize the loss function $\hat{L}_n(f, \bar{f})$.

REFERENCES

[1] A. Barron, "Neural net approximation," in *Proc. 7th Yale Workshop Adaptive Learning Systems*, New Haven, CT, 1992, pp. 69–72.

[2] R. Blumenthal and H. Corson, "On continuous collections of measures," *Ann. Inst. Fourier*, Grenoble, France, vol. 20, no. 2, pp. 193–199, 1970.

[3] V. Borkar, "White noise representation in stochastic realization theory," *SIAM J. Contr. Optim.*, vol. 31, no. 3, pp. 1093–1102, 1993.

[4] V. Borkar, K. Rajaraman, and P. Sastry, "Learning stochastic dependences by matching empirical averages," in *3rd Symp. Intelligent Systems*, Bangalore, India, Dec. 1993.

[5] V. Borkar, "Ergodic control of Markov chains with constraints—The general case," *SIAM J. Contr. Optim.*, vol. 32, no. 1, pp. 176–186, 1994.

[6] V. Borkar, *Probability Theory: An Advanced Course*. New York: Springer-Verlag, 1995.

[7] L. Breiman, "Bias, variance and arcing classifiers," Tech. Rep. 460, Dept. Statistics, Univ. California, Berkeley, 1996.

[8] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Contr., Signals, Syst.*, vol. 2, pp. 303–314, 1989.
[9] A. DeSantis, G. Markowsky, and M. Wegman, "Learning probabilistic prediction functions," in *Proc. 1988 Workshop Computational Learning Theory*. San Mateo, CA: Morgan Kaufmann, 1988.
[10] L. Dubins, "On extreme points of convex sets," *J. Math. Anal. Applicat.*, vol. 5, pp. 237–244, 1962.
[11] W. Feller, *An Introduction to Probability and Its Applications*, 3rd ed. New York: Wiley, vol. 1, 1972.
[12] D. Haussler, "Decision theoretic generalizations of the PAC model for neural net and learning applications," *Inf. Comput.*, vol. 100, pp. 78–150, 1992.
[13] R. Jacobs and M. Jordan, "A competitive modular connectionist architecture," in *Advances in Neural Information Processing Systems 3*, R. Lippman, J. Moody, and D. Touretzky, Eds. San Mateo, CA: Morgan Kaufmann, 1991, pp. 767–773.
[14] N. Littlestone and M. K. Warmuth, "The weighted majority algorithm," Tech. Rep. UCSC-CRL-91-28, Univ. California, Santa Cruz, 1992.
[15] W. Rudin, *Real and Complex Analysis*. New York: McGraw-Hill, 1966.
[16] D. Rumelhart and J. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol. 1: Foundations*. Cambridge, MA: MIT Press, 1986.
[17] K. Trivedi, *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. New Delhi, India: Prentice-Hall, 1988.
[18] H. Witsenhausen, "Some aspects of convexity useful in information theory," *IEEE Trans. Inform. Theory*, vol. IT-26, pp. 265–271, 1980.
[19] R. Wolff, *Stochastic Modeling and the Theory of Queues*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

Vivek S. Borkar (SM'95) was born in Mumbai, India, in 1954. He received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Mumbai, in 1976, the M.S. degree in systems and control from Case Western Reserve University, Cleveland, OH, in 1977, and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley, in 1980.

After spending a year at Technische Hogeschool Twente, The Netherlands, he joined T.I.F.R. Centre, Bangalore, India, in 1981. He moved to the Indian Institute of Science, Bangalore, in 1989, where he is currently an Associate Professor with the Department of Computer Science.

Dr. Borkar was a cowinner of IEEE Control Systems Society's Best Transactions Paper Award, the S. S. Bhatnagar Award for Engineering and Technology awarded by the Council of Scientific and Industrial Research, Government of India, and the Homi Bhabha Fellowship in 1994–1995. He is a member of American Mathematical Society, life member of the Indian Society for Probability and Statistics, and a Fellow of the Indian Academy of Sciences and the Indian National Science Academy.

Piyush Gupta received the B.Tech. degree in electrical engineering from the Indian Institute of Technology, Bombay, in 1993, and the M.S. degree in computer science and automation from the Indian Institute of Science, Bangalore, in 1996. He is currently pursuing the Ph.D. degree in the Department of Electrical and Computer Engineering, University of Illinois, Urbana-Champaign.

From 1993 to 1994, he was a Design Engineer at the Center for Development of Telematics, Bangalore. His current research interests are in wireless communication networks, queuing theory, and learning and intelligent systems.