Reproducing Infinite Boolean Sequences: an Application of Hidden Markov Models to Connectionist Learning

A. Kehagias
Division of Applied Mathematics
Brown University
Providence, RI 02912

E-Mail Address: st401843@brownvm.bitnet

Abstract

Given the first few terms of an infinite sequence of 0's and 1's, build a network that reproduces the rest of the sequence. To accomodate this learning task, a framework is developed for learning, general enough to include learning of finite-length input, finite-length output as well. The similarities between the finite and infinite learning tasks are considered and parallels are drawn to the speech recognition problem. Several properties of infinite length learning are obtained, using Hidden Markov Model theory.

1. Introduction: Consider the following learning problem: build a network that, given the first few terms of an infinite sequence of $0's$ and $1's$ (call such a sequence a **Boolean** sequence), will output the rest of the sequence. The network, will be trained on one or several finite (but possibly quite long) valid sequences.

This problem is similar to a standard connectionist learning problem: given input/output pairs train a network that will, when presented with an input, produce the correct output. In this paper I will propose a learning framework that accommodates both the finite and infinite case; also stochastic and deterministic learning by connectionist networks (henceforth CN) can both be tretated. I will argue that the stochastic case is more general and interesting.

An apology is due: because of the very limited space, I have only presented the general rationale behind the theory and some of the theoretical results, but without proof. The proofs and very encouraging computational results will be reported separately.

2. Learning by Computing Probabilities: The pattern learning problem can be phrased as follows: given a **learning universe** $L \subset U \times Y$ (i.e. a set of input/output pairs (u, y)), build a network (and train it) that, when given an input $u \in U$, will output the 'correct' $y \in Y$.

The concept of 'correctness' is generally not unambiguous and, in fact, this is an advantage. E.g., given an input, it is possible that more than one outputs would pass as 'correct' in some sense. Also, given an unambiguous training set $T = \{(u, y)\}$, such that we can build a network that will reproduce the training pairs exactly but will generalize poorly [Huy]. In fact, the power of generalization that CN's exhibit can be partly attributed to their tolerance of errors.

Let us then change the learning problem so that more than one correct answer is possible. The entity to be learned is a conditional probability measure $P(\mathbf{y} \mid \mathbf{u})$, where $\mathbf{u}$ is a $\mathcal{U}$-measurable random variable $\mathbf{y}$ is a $\mathcal{Y}$-measurable random variable, $\mathcal{U}$ is a σ-algebra on U, $\mathcal{Y}$ is a σ-algebra on Y. $\mathbf{u}$, $\mathbf{y}$ are the input and output random variables; the idea is that, given some input, the output is a random variable with probability conditioned on the input. This prompts the following:

Definition 1 *Given random variables* $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ *on an appropriate probability space.* $\mathbf{x}$ *and* $\mathbf{y}$ *are said to be* $\mathbf{z}$ **input equivalent** *iff* $Prob(\mathbf{x} \mid \mathbf{z}) = Prob(\mathbf{y} \mid \mathbf{z})$.

3. Network Architecture: We will consider connectionist networks, that is, networks of numbers of simple, identical, interconnected units. The building units will be **Stochastic Threshold Units** (STU). These units have the following input/output law:

$$Prob(\mathbf{y} = 0 \mid \mathbf{u} = u) = Prob(\eta > u).$$

Here η is a random variable distributed uniformly in $[a, b]$. By proper choice of a, b we can get deterministic output. A network built of such units accepts an input u and produces output y with probability $Q(\mathbf{y} \mid \mathbf{u})$. As the output is exclusively 0/1, call such a network a **Boolean** CN. The network is, in general, stochastic. Of course, nothing precludes that Q is a trivial measure, concentrating mass 1 to one element y and zero to all others (deterministic case). It is useful to think of the connectionist network itself as equivalent to the conditional probability measure $Q(\mathbf{y} \mid \mathbf{u})$.

4. Learning Finite Output: The finite input/output problem has been studied by many researchers, usually as a deterministic problem. (But see [Bar], [Pea], [Lin], [Sol] for probability learning approaches.) In essence we want to learn a Boolean function from a subset of its graph. The introduction of deterministic **Linearly Graded Units** (LGU's) offers more flexible solutions but still an exponentially large number of units and exemplars, is needed for perfect learning [Huy]. Once again, the argument for stochastic learning presents itself: if exact learning requires oversized networks, learning the statistical properties of the output (which ispossible on a smaller network) is preferrable. Indeed, I submit that the most succesful case of discrete input/output learning is done by connectionist methods but outside the traditional connectionist research community. The example I have in mind is speech recognition by Hidden Markov Models (henceforth HMM) [Lev]. Here exact learning of the input/output relationship is out of the question (because the input/output function is unknown and the training set is too large for exact, deterministic learning). The HMM modellers have resorted to a probabilistic model, which can produce rich input/output behavior with few connections exactly because each unit can produce probabilistically many different outputs. On the other hand, I have argued elsewhere [Keh] that the HMM is exactly analogous to a connectionist network and can be implemented as one. Set the intial state of a HMM to $\mathbf{u}$ and let it run for a finite time N. The state sequence of the underlying Markov Chain $\mathbf{y} = \mathbf{y}_1, ..., \mathbf{y}_N$ is a random variable. By identifying sequences of outputs of a Boolean CN with the (finite) number of states of a HMM, we can easily prove the following:

Theorem 1 *To every Hidden Markov Model corresponds a* $\mathbf{u}$ *input equivalent connectionist network of STU's.*

As for the training algorithm: back propagation is an option, the backward-forward HMM algorithm is another [1]. In fact any kind of descent algorithm that utilizes gradient information will benefit from a forward-backward derivative-computation. This follows from the staged character of the architectures we have proposed and the use of Lagrange multipliers optimization [Keh]. It is also true of time-evolving networks which we will use for the infinite output case.

5. Learning Infinite Output: Consider the infinite output case. The STU-networks that we have been considering can be in an on or off state. It follows that the complete state of an M-unit network can be characterized at any given moment by an M-long **state** vector of 0's and 1's, indicating which units are on and which off. Evolution of the state vector in time is described by the network acting on the initial state vector[2]. Time is not of great importance in the finite case, because the network is **static**. However, if we want an infinite output, we must build an infinitely large network or else trade space for time and obtain the output sequence as the output of a time evolving network[3]. The problem of learning an infinite sequence is the following: we are given the initial part of one or more valid sequences and we want to build a network that when presented with the beginning of a sequence will produce correct (infinitely long) output. The natural way to think about the problem is that of building a network that will accept an input and then will run on its own until infinity, producing output. This is like a dynamical network where we specify the

[1] Generalization of the Backward-Forward algorithm to CN's is possible: results will be reported elsewhere.

[2] Another useful state-network characterization is the one based on the state probability vector. This will not be presented here, for brevity.

[3] Such networks are called **recurrent**; they have been considered in the literature [Pin], but usually only as they evolve to a steady state, that is, as all units settle down to either 0 or 1 output for all time.

initial conditions[4]. In the proposed general framework, the input is the initial states. The learning task is, in general, to learn a probability measure $P(\cdot \mid u)$, where u is the input. It would be nice if the output y_n at time n depended on a finite number of past outputs.

$$Prob(\mathbf{y}_n = \tau \mid \mathbf{y}_{n-1} = \sigma_1, ..., \mathbf{y}_1 = \sigma_{n-1}) = f(\sigma_1, ..., \sigma_m) \tag{1}$$

Here f is a Boolean function of m arguments. This is a highly desirable situation, as we can take this sequence as the output of a STU-network with, at most, However its also conceivable that

$$Prob(\mathbf{y}_n = \tau \mid \mathbf{y}_{n-1} = \sigma_1, ..., \mathbf{y}_1 = \sigma_{n-1}) = f(\sigma_1, ..., \sigma_{n-1}) \tag{2}$$

The network as described by equation (1) is an m-th order Markov chain, equivalent to an HMM, which can be trained by the powerful Backward-Forward algorithm. It can also be implemented as a STU-network with at most $O(2^m)$ units. We will also use the term **Boolean Dynamic Network**. The next question is whether the sequence is determinstic or stochastic; consider the case where we are given a finite part of a sequence and must compute the next terms. We do not know whether it is stochastic or deterministic. Of course the following fact is true:

Theorem 2 *For every deterministic Boolean dynamical network with output $y_1, y_2, ...$, $\exists$ N such that y_{N+1}, y_{N+2}, ... is periodic*[5]*.*

However, the period may turn to be exponentially long, so given a relatively short training sequence lack of periodic behavior does not determine whther the inifinite sequence is deterministic or stochastic. That is, we may see any finite substring of 0's and 1's followed by either a 0 or a 1; this would still not show that the sequence is stochastic. In short, we have no basis to decide whether the sequence is deterministic or stochastic and the reasonable assumption is to try to model it as stochastic.

Definition 2 *A* **aggregate** *of a stochastic process $\{X_n\}$ is another process $\{Y_n\}$ s.t. $Y_n = f(X_n)$.*

A HMM is a aggregate of a Markov chain; the output of a connectionist network is a aggregate of the state vector and, since the state vector is a Markov chain, the output of the connectionist network is the observable of a Hidden Markov Model.

Definition 3 *A* **Boolean stochastic process** *is a probability measure on the set of sets of infinite sequences of 0's and 1's (call every such sequence a* **realization** *of the stochastic process), that is, $P(\mathbf{x} = s_1 s_2 ....)$.*

With each realization we can associate a number s in the interval $[0, 1]$; we do this by taking $s = .s_1 s_2 ...$ in binary notation. [6] The measure P is thus well defined on dyadic intervals and can be extended by Kolmogorov's Theorem on the Borel sets in $[0, 1]$. Hence the stochastic process $\mathbf{x}$ is equivalent to a a random variable $\mathbf{x}$ taking values in $[0, 1]$, or, which is really the same thing, to a probability measure on $[0, 1]$. As the CN produces a stochastic process depending on the initial conditions, by the same process as the one outlined above, we have a conditional probability measure $P(\cdot \mid u)$ associated with each input u.

Now we can apply the same steps as in the finite learning task: we must find a probability measure $Q(\cdot \mid \cdot)$ that will approximate $P(\cdot \mid \cdot)$. Many approximation schemes are available, but it is useful to establish first that an arbitrarily close approximation is possible.

Theorem 3 *Given a Stochastic Process, that is, a measure on $[0, 1]$, it can be weakly approximated by a sequence of Hidden Markov Models.*

---

[4] We can go up one degree of generality: Imagine an infinite input as well an infinite output. This would correspond to a network where we have input at every time instant. We can certainly build such a network. I do not give details here, for lack of space, but also because it turns out many of the properties of the input network will be similar to the no-input network. However there are interesting peculiarities, too. Roughly, the no-input network corresponds to a free dynamical network and the input network corresponds to a control network.

[5] There is a subtlety here: the actual evolution will depend on the initial conditions. So for different initial conditions the network may get attracted to a different periodic sequence .This is similar with limit cycles in real-valued nonlinear dynamical systems, as well as with non-ergodic Markov chains.

[6] The correspondence is not one-to-one. However, for all interesting cases, we can turn it to one-to-one by dropping out realizations of probability 0.

Given that there exist HMM's and hence CN's that can approximate a stochastic Boolean sequence arbitrarily close, how does one go about finding them and, probably most important, how does one train them? I have used several schemes; one, call it **structured**, is the following: the training sequence is used to estimate the joint probabilities of x_1, x_2, ..., x_{m+1} , for some prechosen m; then from these the transition probabilities $P(x_{n+m} \mid x_{n+m-1}, ..., x_n)$ are computed. It is easy to implement a CN that has $2^m + m$ STU's and the above computed transition probabilities. The training of such a network consists in updating the estimates of the transition probabilities as new data is collected; when these probabilities are known the weights of the connection can be immediately computed. In this respect it is important to have ergodicity of the teaching sequence, or else samples from all the ergodic elements. This architecture yields large but easily trainable networks; yet output is pretty good even with rather small values of m. Another possibility explored is less structured architectures where a more or less random choice of connectivity is made and then weights are computed by some backpropagating algorithm. The actual criterion used was either cross entropy between desired and actual probability function, or the following weighted quadratic criterion:

$$J = \sum_{n=1}^{N} \frac{(P(z_n) - Q(z_n))^2}{P(z_n)}. \tag{3}$$

Results were again quite good, with no noticeable differences across learning algorithms and criteria.

One last point must be raised: we are essentially looking at high order sequences and attempt to approximate them by aggregates. This alleviates certain problems (estimation, size of networks) but, inevitably, produces some faulty sequences. The degradation in terms of spurious sequences being produced by the approximating network can be quantified in terms of the entropy H of the model as compared to the entropy of the original sequence. This is the subject of the following Theorem; remember that an approximation is a aggregate of the original sequence.

Theorem 4 *If a stochastic process* $\mathbf{x}$ *is a aggregate of a stochastic process* $\mathbf{y}$*, then* $H(\mathbf{x}) \geq H(\mathbf{y})$*. If* $\mathbf{y}_n \to \mathbf{y}$*, then* $H(\mathbf{y}_n) \downarrow H(\mathbf{y})$*.*

This is yet another way of saying that there are approximating networks that will approximate the original sequence arbitrarily well.

6. Conclusion: I have presented an outline of a framework for learning that involves infinitely large learning universes. The basic idea is that as the learning(and training) universe get large it is not feasible to learn deterministically, especially when it is not clear that the objects being learned behave deterministically. Therefore stochastic learning methods are proposed; several theoretical results indicate that stchastic learning is feasible and practical. Computational results are also very encouraging, but were not presented for lack of space.

REFERENCES

[**Bar**] Barto A.G. et al., *Pattern-Recognizing Stochastic Learning Automata*, IEEE SMC, Vol.15, No.3, May 1985.

[**Huy**] Huy, K.A. and Horowitz, M.A., *Generalization in Connectionist Networks that Realize Functions*, Proc. of the Pittsburgh Connectionist School, Morgan Kauffman, 1988

[**Keh**] Keh, Ath., *Optimal Control for Training: The missing Link between Hidden Markov Models and Connectionist Training*, Brown University, Div. of Appl. Math. Preprint, 1989

[**Lev**] Levinson, S.E. et al., *An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Automatic Speech Recognition*, Bell Sys. Tech. J., Vol. 62, No.4, April 1983

[**Lin**] Linsker, R. *Self-Organizing in Perception Networks*, IEEE Computer,Vol.21,No.3, March 1988

[**Pea**] Pearlmutter,B. and Hinton G., *G-Maximization: An Unsupervised Learning Procedure*, Neural Networks for Computing, AIP Conf. Proc. 151, 1986

[**Pin**] Pineda, F., *Generalization of Back-Propagation to Recurrent Neural Networks*, Phys. Rev. Let., Vol.59, No.19

[**Sol**] Solla, S. *Accelerated Learning Experiments in Layered Neural Networks*, Complex Systems, Vol. 2, 1988