Proceedings of the Joint Conference on Information Sciences, pp. 750-752, 2000

Learning Quantum Operators

Dan Ventura
Applied Research Laboratory
Penn State University
dav8@psu.edu
http://www.personal.psu.edu/dav8

Abstract

Consider the system $\hat{F}|\chi\rangle = |\psi\rangle$ where $\hat{F}$ is unknown. We examine the possibility of learning the operator $\hat{F}$ inductively, drawing analogies with ideas from classical computational learning.

1 Introduction

The field of quantum computation views computation as effected by the time evolution of a physical system, usually described mathematically as

$$\hat{F}|\chi\rangle = |\psi\rangle \tag{1}$$

where $|\chi\rangle$ represents the initial state of the system, $\hat{F}$ the system's (unitary) evolution, and $|\psi\rangle$ the final state of the system. From a computational point of view, we might say that $|\chi\rangle$ is the input to a "function" $\hat{F}$ and that $|\psi\rangle$ is the output of that function. In other words, we might draw a mathematical analogy between Eq. (1) and

$$f(x) = y \tag{2}$$

The field of (classical) computational learning deals with the situation where we do not know the function f but instead have only a set $\mathcal{P}$ of example functional points of the form (x, y) such that $f(x) = y$. The challenge is to find a learning algorithm that uses these functional examples to hypothesize a function g such that $g \sim f$ (g approximates f). In other words, the learning algorithm takes $\mathcal{P}$ as input and returns g as output. The relationship $g \sim f$ may be defined in many ways. For example, we might say that $g \sim f$ if $\forall x(|g(x) - f(x)| < \epsilon)$ for some suitably small ϵ; or we might say that $g \sim f$ if $\exists \mathcal{A}(g(a) = f(a), a \in \mathcal{A}, |\mathcal{A}| > n)$ for some suitably defined n. In any case, much of the field of computational learning involves discovering how we can suitably hypothesize the function g.

Taking our analogy a little further, we may consider the situation in quantum computation in which we do not know the operator $\hat{F}$ and thus would like to hypothesize an operator $\hat{G}$ such that $\hat{G} \sim \hat{F}$. If we are to continue the analogy, this hypothesis step requires the availability of a set $\mathcal{S}$ of "functional" examples and some learning algorithm that uses $\mathcal{S}$ to produce $\hat{G}$. The remainder of this paper will discuss this idea, explore what the set $\mathcal{S}$ might consist of, and discuss a proposal for an algorithm for learning quantum operators.

2 Quantum Learning

From the preceding discussion it appears that the most natural candidate for examples in the quantum setting is a pair of quantum states $(|\chi\rangle, |\psi\rangle)$, where $|\psi\rangle$ is the result of the operator $\hat{F}$ operating on $|\chi\rangle$. Given a set of such pairs and a random (unitary) operator $\hat{G}^0$, we would like to discover an algorithm Λ that will iteratively produce $\hat{G}^1$, $\hat{G}^2$, ..., $\hat{G}^n$ such that $\hat{G}^n \sim \hat{F}$.

One well-known classical learning algorithm is the delta learning rule used in some implementations of neural networks. The basic idea is to evaluate a neural network using an example input and compare the network's output with that of the target output. The weights of the network are then adjusted so that the output will more closely approximate the target. A similar concept can be developed as a simple learning algorithm for quantum operators.

Given a set of quantum examples, $\mathcal{S} = \{(|\chi\rangle_i, |\psi\rangle_i)\}$ and a random initial (unitary) operator $\hat{G}^0$, the algorithm proceeds as in Fig. 1. First, the operator $\hat{G}$ is applied to the quantum state $|\chi\rangle$ and the result is compared to the target state $|\psi\rangle$.

$$\hat{G}|\chi\rangle = |\tilde{\psi}\rangle \tag{3}$$

The operator $\hat{G}$ is then updated according to the

1. for each time step t
2. for each example pair $(|\chi\rangle, |\psi\rangle)$
3. calculate $\hat{G}^t|\chi\rangle = |\tilde{\psi}\rangle$
4. Update $\hat{G}$ as $g_{jk}^{t+1} = g_{jk}^t + \delta(\psi_j - \tilde{\psi}_j)\chi_k$

Figure 1: Inductive learning algorithm for quantum operators

rule

$$g_{jk}^{t+1} = g_{jk}^t + \delta(\psi_j - \tilde{\psi}_j)\chi_k \quad (4)$$

where t represents time or iteration number and g_{jk} are the matrix entries indexed by row j and column k. Intuitively, a matrix entry is modified according to how much it contributes to the state evolution error (the difference between the target state $|\psi\rangle$ and the actual state $|\tilde{\psi}\rangle$); δ is a traditional learning constant that scales how quickly the error-based modification affects the operator; and multiplying the error by χ_k weights the change according to how significantly g_{jk} affects the state error.

Note that in the limited experimentation performed to date the learning rate δ does not appear to be a sensitive parameter and thus in the interest of parsimony was set to $\delta = 1$.

As a simple example, suppose we want to learn the operator $\hat{F}$ represented by the two examples

$$\mathcal{S} = \left\{ \begin{array}{c} \left(\frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \frac{1}{\sqrt{10}} \begin{bmatrix} 3 \\ 1 \end{bmatrix} \right) \\ \left(\frac{1}{\sqrt{20}} \begin{bmatrix} -2 \\ 4 \end{bmatrix}, \frac{1}{\sqrt{40}} \begin{bmatrix} 2 \\ -6 \end{bmatrix} \right) \end{array} \right\}$$

Further suppose the operator $\hat{G}$ is initialized as

$$\hat{G}^0 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

which corresponds to Fig. 2(a). We then evaluate the first example using $\hat{G}^0$

$$\hat{G}^0|\chi\rangle_1 = |\tilde{\psi}\rangle_1 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \frac{1}{\sqrt{5}} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \frac{1}{\sqrt{5}} \begin{bmatrix} 1 \\ -2 \end{bmatrix}$$

Next, we update the matrix entries. For example,

$$g_{00}^1 = g_{00}^0 + \delta(\psi_0 - \tilde{\psi}_0)\chi_0 = 0 + (\frac{3}{\sqrt{10}} - \frac{1}{\sqrt{5}})\frac{2}{\sqrt{5}} \approx 0.449$$

Calculating the rest of the matrix entries in a similar fashion results in an updated version of $\hat{G}$,

$$\hat{G}^1 \approx \begin{bmatrix} 0.449 & 1.224 \\ 0.083 & 0.541 \end{bmatrix}$$

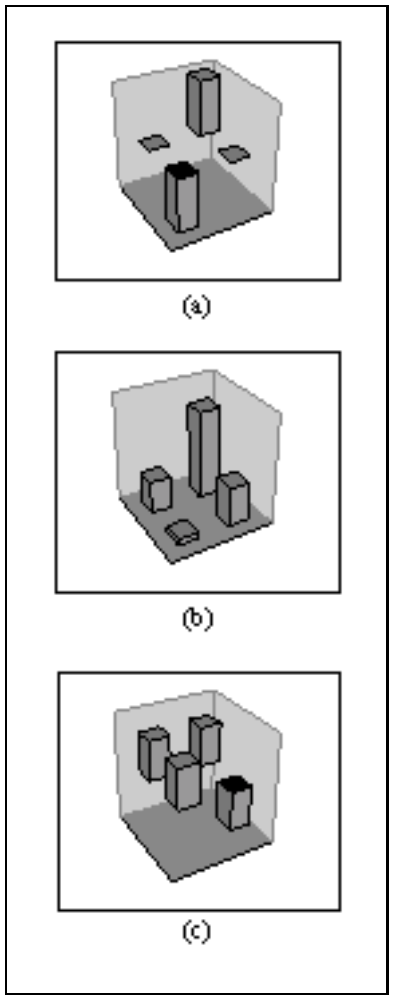

Figure 2: Evolution of the operator $\hat{G}$: (a) initial random (unitary) state (b) state after processing first example (c) state after processing second example – the Hadamard transform

corresponding to Fig. 2(b). Repeating the process for the second example results in the final version of $\hat{G}$, which is shown in Fig. 2(c) and is, in fact, the well-known Hadamard transform

$$\hat{G}^2 \approx \begin{bmatrix} 0.707 & 0.707 \\ 0.707 & -0.707 \end{bmatrix} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

3 Generalization

In computational learning, once the function g has been produced the issue becomes one of generalization. This ill-defined term refers to how reasonably the function g performs for new inputs not used during learning. In other words, it is an attempt to

qualify the relationship $g \sim f$. Similarly, we will be interested in how well $\hat{G}$ generalizes to new quantum states.

A quantum state can be thought of as a vector in a Hilbert space, and a quantum operator as simply a description of a rotation in that Hilbert space. Therefore, the algorithm presented here is simply using a set of vector pairs as a partial description of how such a rotation should be performed. The pertinent question is whether the algorithm produces an operator that describes parsimonious rotation in the Hilbert space, such that new vectors are rotated in a way that may be described as interesting, or useful or reasonable. In the simple example above, learning the Hadamard transform would certainly qualify as interesting, useful and reasonable, and therefore we can say at least that the algorithm generalizes well for some problems.

4 Conclusion

In the newly emerging field of quantum computational intelligence, approaches to date have concentrated on implementing intelligent algorithms using quantum computation [1] [4] or on modeling intelligent systems using quantum dynamics [2]. In other words, they have tried to produce quantum operators that affect quantum systems in such a way as may be characterized as intelligent. Here we have taken a different tact entirely, attempting to apply a classical approach to learning to the quantum realm. Therefore, instead of trying to produce quantum algorithms that mimic or exhibit or explain intelligent behavior, we have produced an (in fact classical!) computational intelligence approach to producing quantum operators. The algorithm described here can be thought of as a delta rule for learning quantum operators.

The matrix representation of a quantum operator grows exponentially with the size of the associated Hilbert space. There are at least two approaches to addressing this problem. One obvious possibility is the development of a quantum version of the algorithm. In other words, we might solve the problem by producing quantum operators for implementing an algorithm for learning quantum operators. For example, one could consider using density matrices as representations of the operator to be learned and then manipulating the quantum system described by the density matrix. Another possibility for addressing the tractability issue is the decomposition of the problem of learning general unitary operators into a problem of learning a series of elementary unitary operators. Related work in this area has been done on decomposing a general operator into a sequence of elementary operators [3]. If, in fact, the problem can be effectively decomposed, then the classical version of the learning algorithm presented here will be sufficient. Finally, learning theoretic properties of the algorithm need to be explored. These include, for example, convergence properties, upper bounds on number of examples required, and quantification of the relationship $\hat{G} \sim \hat{F}$.

References

[1] Behrman, E., J. Niemel, J. Steck and S. Skinner, "A Quantum Dot Neural Network", *Proceedings of the Workshop on Physics of Computation*, pp. 22-24, 1996.

[2] Perus, Mitja, "Neuro-Quantum Parallelism in Brain-Mind and Computers", *Informatica*, vol. 20, pp. 173-183, 1996.

[3] Tucci, Robert R., "A Rudimentary Quantum Compiler (2nd Ed.)", *Los Alamos Preprint Archive, quant-ph/9902062*, 1999.

[4] Ventura, Dan and Tony Martinez, "Quantum Associative Memory" *Information Sciences*, in press.