QUANTUM LEARNING SEMINAR
LECTURE 1: INTRODUCTION

David A. Meyer

Project in Geometry and Physics, Department of Mathematics
University of California/San Diego, La Jolla, CA 92093-0112
`http://math.ucsd.edu/~dmeyer/`; `dmeyer@math.ucsd.edu`

Introduction

The topic of this seminar is quantum learning—an apparently well-studied subject: searching on Google this morning (1 October 2002) for "quantum learning" returned more than 1500 hits! There even seem to be two companies called Quantum Learning—one in New Zealand and one in Canada. Our plan, however, is not to open a franchise, but rather to explore the connections between quantum computing [1] and machine learning [2]. In this introductory lecture we'll survey some of these connections, leaving the details to be worked out in subsequent lectures.

Motivation

Quantum computation is based on the idea that it might be possible to build a computer that acts at the logical level according to quantum mechanical principles. There is currently a great deal of effort being devoted to doing so [3]. The general question that motivates this seminar is: What might we do with a quantum computer once one is built?

Much of the present excitement about quantum computing is due to the discovery that certain problems appear to be easier to solve (in the computational complexity sense) on a quantum computer than on a classical computer. The most famous example is FACTORING which Shor's quantum algorithm solves in time $O(n^2 \log n \log \log n)$ [4]. The best classical algorithm known for FACTORING is much slower, requiring time $O\big(\exp(cn^{1/3} \log^{2/3} n)\big)$ [5]. FACTORING is important—its hardness is the basis for the security of standard public key cryptosystems [6]—but for quantum computation to be of general interest there need to be more problems for which it provides superior solutions. There are, however, only a handful of quantum algorithms discovered to date. Grover's search algorithm [7] is often the only other one mentioned, and one can argue that all, or at least most known quantum algorithms depend on the two techniques exploited by Shor and Grover: quantum fast transforms and amplitude amplification [8], respectively.

1

A mathematics/computer science challenge in this subject, therefore, is to develop new quantum algorithms/techniques. For reasons that should be clearer by the end of this introduction, and hopefully *much* clearer by the time some of the details have been worked out, there may be inspiration to be drawn from classical results in machine learning [2].

Perhaps the first hints of this possibility appeared in several papers, by different authors, on "quantum neural nets" and related topics [9]. Artificial neural nets (ANNs), of course, constitute a class of methods/algorithms for certain kinds of learning problems. (As the name implies, ANNs were originally inspired by *biological* neural systems, but the mathematical models are abstractions not completely consistent with biology. An interesting question is whether there is more inspiration to be drawn from biology, for either classical or quantum algorithms.)

The challenge of finding new quantum algorithms inspired by results in machine learning and the question of what the right quantum version of ANNs is, provide general and specific motivations for this seminar on quantum learning.

Learning problems

We begin with a brief introduction to classical learning. When we speak of learning, we often mean that there is some task to be accomplished and the problem is to learn what action to take given specific circumstances. Some examples include:

RESPONDING TO PAVLOV: input is ringing bell or not;
action is salivate or not;

DRIVING: lots of input (visual, auditory, sensorimotor, . . .);
actions include accelerate, brake, shift, turn, . . .;

PLAYING GO: input is board position;
action is next move.

In each case we can imagine that there is some function from inputs to best/correct outputs:

RESPONDING TO PAVLOV:

$$\text{ring} \mapsto \text{salivate}$$
$$\text{no ring} \mapsto \text{don't bother}$$

DRIVING: a partial description is

$$\text{red light} \mapsto \text{stop}$$
$$\text{green light} \mapsto \text{go}$$
$$\text{yellow light} \mapsto \text{go faster}$$

—as learned by Jeff Bridges in the movie *Starman* (1984)!

PLAYING GO:

go position $\mapsto$ move with highest positional value

For our purposes, *learning* is defined to be the process of approximating such a function, to which we will sometimes refer as the *target*. Notice that this means that learning has been accomplished when almost correct actions are taken for most inputs. There is no reference to "understanding"—that is, there is no Chinese Room argument [10]: once Searle in his room has mastered the set of rules for correlating input Chinese character strings to output Chinese character strings, he has learned to respond correctly to questions in Chinese. Whether or not he "understands" Chinese is not our concern.

The space of all possible functions, from allowed inputs to allowed outputs, is very, very large. When we program a classical computer to learn (and maybe when we learn, ourselves), we define a *hypothesis space*, by specifying some restricted class of functions. In the case of PLAYING GO, for example, we might decide to consider only positional value functions depending on some, but not all, specific features of a position: areas of live groups, 'thickness', *etc.* Then the problem of approximating the target function is reduced to selecting the best approximation within this hypothesis space. Once a hypothesis space has been specified, a learning process typically occurs *via* a sequence of inputs, outputs, corrections, and adjustments, as shown in Figure 1.1. This has been drawn deliberately to look like a control process for two reasons:

1. At some point we will want to implement quantum learning algorithms as sequences of gate operations, *i.e.*, like an electronic circuit.

2. It may be interesting to determine how much this analogy with control theory has been used in classical machine learning.
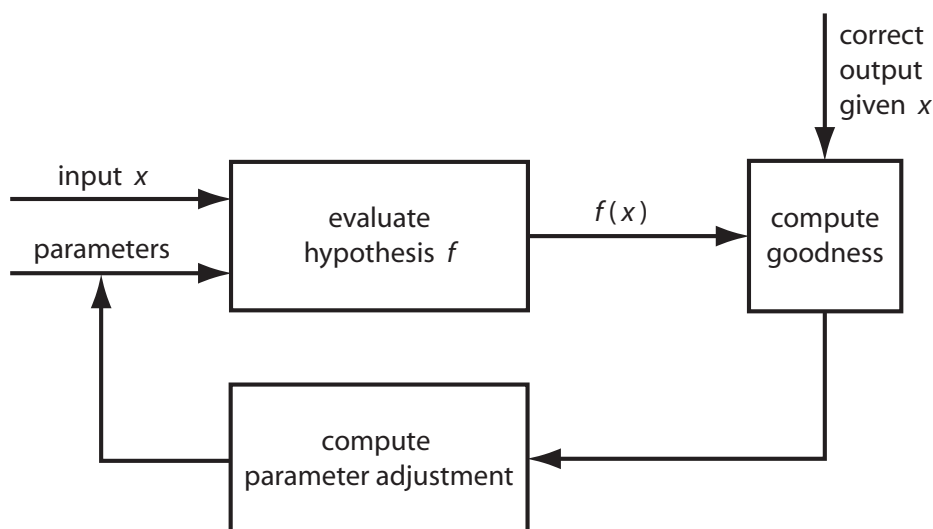

Figure 1.1. A schematic for a typical learning process, formatted to look like a control process.

3

To summarize, there are four ingredients in machine learning:

type of inputs: presented examples, chosen examples, ...
type of outputs: action, real function value, ...
hypothesis space: parameterized subset of possible functions
learning algorithm: how to apply feedback, how to best approximate, ...

We will be interested *not* in the mere existence of quantum learning algorithms, but in quantum *improvements*: requiring fewer examples to learn, requiring less computation time, or maybe even solving different problems. The first two of these measures of a learning algorithm are called the *sample complexity* and the *computational complexity*, respectively, in the branch of machine learning known as *computational learning theory* [11]. To invoke this aspect of machine learning, we should probably call the topic of this seminar "quantum computational learning". This has the added benefit of connoting machine rather than human learning, distinguishing our topic from the majority of those 1500 web pages!

Concept learning

The simplest nontrivial output space defines a restricted class of learning problems:

DEFINITION. A *concept c* is a map from inputs X to $\mathbb{Z}_2 = \{0, 1\}$.

A concept specifies a subset of the possible inputs, namely $c^{-1}(1)$. For example, c could be the map that takes the value 1 when its argument is red. This map is equivalent to the concept of "red". Let $N = |X|$; $n = \log N$ is the number of bits necessary to specify any particular input x. (Unless otherwise indicated, logarithms in these notes are base 2.) There are $2^N = 2^{2^n}$ possible concepts, which is way, way too many to have any hope of approximating the target concept very well given only polynomially many (in n) inputs/examples. A typical concept learning problem, therefore, reduces this superexponential number of concepts by restricting the possible hypotheses to some *concept class*. The problem is thus to search this (still very large) concept class for the best hypothesis. In this simplified setting, Angluin formalized a model for learning [12], part of which Servedio and Gortler noticed can be generalized easily to quantum computation [13]:

DEFINITION. *Concept learning from membership queries* is the problem of identifying a concept c from a concept class $\mathcal{C}$, given a *membership oracle* (MO) that responds to a query $x \in X$ with $\text{MO}(x) = c(x)$. That is, a membership oracle answers the question "Is $x \in c^{-1}(1)$?".

Implicit in the idea of learning from membership queries is the idea that the learner chooses the inputs about which to query the membership oracle. Like the adjustments to the current hypothesis, adjustments to the query are a consequence of the feedback from the oracle about previous queries. Figure 1.2 illustrates such *active learning*.

4

An example of concept learning

Consider the concept class

$$\mathcal{G}^n = \{g_a : \mathbb{Z}_N \to \mathbb{Z}_2 \mid a \in \mathbb{Z}_N \text{ and } g_a(x) = \delta_{xa}\},$$

where $\delta_{xa} = 1$ if $x = a$ and 0 otherwise. A particular concept g_a should be thought of as "the number a"; a is the only element in $g_a^{-1}(1)$. Since there are $N = 2^n$ possible values for a, the number of possible hypotheses is N, much smaller than 2^N, but still very large.

It is obvious, and easy to show, that learning from membership queries has $O(N)$ sample complexity. The following deterministic algorithm—formatted as in Figure 1.2— solves the problem of identifying a target concept in $\mathcal{G}^n$:

Algorithm D$\mathcal{G}^n$.
    1. Select input $x = 0$.
    2. Set hypothesis to g_x.
    3. Evaluate $g_x(x)$.
    4. Query the membership oracle about x.
    5. If $g_x(x) = \mathrm{MO}(x)$ then output g_x; stop, else adjust input by $x \leftarrow x + 1$; go to **2**.

For $x < a$, $1 = g_x(x) \neq \mathrm{MO}(x) = g_a(x) = 0$, so x is incremented by 1 and the algorithm loops. Once $x = a$, however, $1 = g_x(x) = \mathrm{MO}(x) = g_a(a) = 1$, so the conditional is satisfied, the algorithm outputs g_a and stops. This can take no more than N iterations, each of which includes one query of the membership oracle.

It should be plausible that no classical learning algorithm can do better than this (in the worst case). In fact, this concept learning problem has $\Omega(N)$ sample complexity; in a subsequent lecture we will explain how to prove such a lower bound. Our immediate concern, however, is to understand that we can interpret Grover's algorithm as a quantum learning algorithm for this problem.

As we noted above, the membership oracle computes $g_a(x)$ when it is queried about x and the target concept is g_a. Implemented in a digital computer, we might use a data structure consisting of an n bit 'query' register and a 1 bit 'response' register. Then the membership oracle would act by

$$(x, b) \mapsto \big(x, b + g_a(x)\big) \in \mathbb{Z}_N \times \mathbb{Z}_2,$$

where $+$ should be interpreted mod 2, and we would set $b = 0$ for simplicity.

Rather than using the deterministic Algorithm D$\mathcal{G}^n$, we can imagine using a probabilistic algorithm, again in the format of Figure 1.2:

Algorithm P$\mathcal{G}^n$.
    0. Set $X = \mathbb{Z}_N$.
    1. Select input $x \in X$ uniformly at random.

    2. Set hypothesis to g_x.
    3. Evaluate $g_x(x)$.
    4. Query the membership oracle about x.
    5. If $g_x(x) = \text{MO}(x)$ then output g_x; stop, else adjust $X \leftarrow X \setminus \{x\}$; go to **1**.

Using the data structure described in the previous paragraph, at the time of the first query to the membership oracle, the state of a classical computer running this probabilistic algorithm is $(x, 0)$ with probability $1/N$, for each $x \in \mathbb{Z}_N$. That is, the state of the computer is described by the vector $(1/N, \ldots, 1/N, 0, \ldots, 0) \in \mathbb{R}^{2N}$, in the basis defined by the $2N$ possible states: $(0, 0), \ldots, (N-1, 0), (0, 1), \ldots, (N-1, 1)$, in that order, so the components of the vector are the probabilities of each. Writing (x, b) for the corresponding unit basis vector, this probability vector is

$$\sum_x \frac{1}{N}(x, 0) \tag{1.1}$$

and querying the membership oracle transforms it to the vector

$$\sum_x \frac{1}{N}\big(x, g_a(x)\big) = \frac{1}{N}(a, 1) + \sum_{x \neq a} \frac{1}{N}(x, 0).$$

In general, we can describe the state of the probabilistic computer at each timestep by a unit ℓ_1-norm non-negative vector in $\mathbb{R}^{2N}$. Each iteration of Algorithm $\mathrm{P}\mathcal{G}^n$ adjusts this probability vector. *E.g.*, after the second query the state is

$$\frac{2}{N}(a, 1) + \sum_{x \neq a} \frac{1}{N}\Big(1 - \frac{1}{N-1}\Big)(x, 0),$$

where we have added the probability of stopping after the first query to the probability of the basis vector $(a, 1)$. Algorithm $\mathrm{P}\mathcal{G}^n$ also runs in time $O(N)$.

Quantum states and evolution

    Once we accept that the state of a classical computer running a probabilistic algorithm can be described by a unit ℓ_1-norm non-negative vector in $\mathbb{R}^{2N}$, there are only two, apparently small steps to quantum computing. First, the state of the quantum computer is a unit ℓ_2-norm vector in $\mathbb{C}^{2N}$ (for this example problem). Second, the evolution from one state to the next is *via* multiplication by a unitary matrix. Then the quantum analogue of Algorithm $\mathrm{P}\mathcal{G}^n$ would create a state

$$\sum_x \frac{1}{\sqrt{N}}|x, b\rangle$$

with which to query the membership oracle, which will return

$$\sum_x \frac{1}{\sqrt{N}}|x, b + g_a(x)\rangle.$$

Here the classical register states (x, b) again label a basis, now of $\mathbb{C}^{2N}$. Following standard physics notation, we denote them by asymmetrical brackets $|x, b\rangle$ to distinguish them from the corresponding elements $\langle x, b|$ of the dual space. The membership oracle mapping $|x, b\rangle \mapsto |x, b + g_a(x)\rangle$ extends by linearity to a unitary map since it acts as a permutation on the basis vectors. In fact, the data structure is designed so that the membership oracle acts *via* multiplication by a unitary matrix, and hence is an allowed step in a quantum algorithm.

References

[1] A good general reference is M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information* (Cambridge: Cambridge University Press 2000)

[2] For the computer science perspective see, *e.g.*, T. M. Mitchell, *Machine Learning* (San Francisco: McGraw-Hill 1997);
for a recent mathematical perspective see F. Cucker and S. Smale, "On the mathematical foundations of learning", *Bull. Amer. Math. Soc.* **39** (2002) 1–49.

[3] See, *e.g.*, the special issue of *Fortsch. Phys.* **48** no. 9–11 (2000).

[4] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring", in S. Goldwasser, ed., *Proceedings of the 35th Symposium on Foundations of Computer Science*, Santa Fe, NM, 20–22 November 1994 (Los Alamitos, CA: IEEE Computer Society Press 1994) 124–134;
P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer", *SIAM J. Comput.* **26** (1997) 1484–1509.

[5] A. K. Lenstra, H. W. Lenstra, Jr., M. S. Manasse and J. M. Pollard, "The number field sieve", in *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing*, Baltimore, MD, 14–16 May 1990 (New York: ACM Press 1990) 564–572;
A. K. Lenstra and H. W. Lenstra, Jr., eds., *The Development of the Number Field Sieve*, Lecture Notes in Mathematics, vol. 1554 (New York: Springer-Verlag 1993).

[6] R. L. Rivest, A. Shamir and L. M. Adleman, "A method of obtaining digital signatures and public-key cryptosystems", *Commun. ACM* **21** (1978) 120–126.

[7] L. K. Grover, "A fast quantum mechanical algorithm for database search", in *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*, Philadelphia, PA, 22–24 May 1996 (New York: ACM 1996) 212–219;
L. K. Grover, "Quantum mechanics helps in searching for a needle in a haystack", *Phys. Rev. Lett.* **79** (1997) 325–328.

[8] L. K. Grover, "A framework for fast quantum algorithms", in *Proceedings of 30th Annual ACM Symposium on Theory of Computing* (New York: ACM 1998) 53–62;
G. Brassard, P. Hoyer and A. Tapp, "Quantum counting", `quant-ph/9805082`.

[9] R. L. Chrisley, "Quantum learning", in P. Pylkkänen and P. Pylkko, eds., *New Directions in Cognitive Science*, Proceedings of the International Symposium, Saariselka, Finland, 4–9 August 1995 (Helsinki: Finnish AI Society 1995) 77–89;
D. Ventura and T. Martinez, "An artificial neuron with quantum mechanical properties", *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, Norwich, England, April 1997, 482–485;

E. C. Behrman, J. E. Steck and S. R. Skinner, "A spatial quantum neural computer", *IJCNN'99*, vol. 2, Proceedings of the International Joint Conference on Neural Networks, Washington, DC, 10–16 July 1999 (Piscataway, NJ: IEEE 1999) 874–877.

[10] J. Searle, "Minds, brains, and programs", *Behavioral and Brain Sciences* **3** (1980) 417–424.

[11] See, *e.g.*, D. Angluin, "Computational learning theory: Survey and selected bibliography", in *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing* (New York: ACM 1992) 351–369.

[12] D. Angluin, "Queries and concept learning", *Machine Learning* **2** (1988) 319–342.

[13] R. A. Servedio and S. J. Gortler, "Quantum versus classical learnability", in *Proceedings of the Sixteenth IEEE Conference on Computational Complexity* (Los Alamitos, CA: IEEE 2000) 138–148.