CURRENT IMPLEMENTATIONS OF FUTURE TECH: HOW QUANTUM ANNEALING MACHINES SOLVE PROBLEMS

by

SAMUEL CALVERT

A THESIS

Presented to the Department of Mathematics
and the Robert D. Clark Honors College
in partial fulfillment of the requirements for the degree of
Bachelor of Science

May 2018

An Abstract of the Thesis of

Samuel Calvert for the degree of Bachelor of Science
in the Department of Mathematics to be taken May 2018

Title: Current Implementations of Future Tech: How Quantum Annealing Machines Solve Problems

Approved: _____

Benjamin Young

As processors in classical computers continue to keep pace with Moore's law an eventuality has emerged. As the number of transistors on a microchip continues to increase we approach physical limitations and enter a world dominated by quantum processes. For processing density to continue to increase we need to find a new method of processing or new physical rules for the universe. In conjunction with UC Davis and D-Wave Systems I explore an adiabatic quantum annealing machine. Specifically, I look at the mathematical model Quantum Machines use to solve problems now, methods for error reduction, scalability, and the influence of quantum computers on the P vs NP Millennial Prize Problem. I have this conversation in the context of the Traveling Salesman Problem as well as a method I developed for training Support Vector Machines using quantum computing techniques.

Acknowledgements

I would like to thank Professor Benjamin Young of the University of Oregon for guiding me through all stages of the thesis process as well as my final year at University. He was always willing to learn with me and lend advice. I would also like to thank Professor Ellen Eischen of the University of Oregon through whose tutelage I learned about the kind of mathematician I should strive to be, and Professor Kelly Sutherland who also served on my Thesis Committee.

I also need to thank Dr. Mike Hogarth of UC San Diego, Dr. David Chin of University of Massachusetts, and Jupinder Parmar soon to be of Stanford University for trusting me, getting excited with me, making mistakes with me, and ultimately succeeding with me. Finally, I want to thank the University of California, Davis for securing computation time with D-Wave Systems and the largest quantum annealing machine in the world at the time of writing.

Table of Contents

Introduction to the Topic

Background

The advent of computers has changed how we view knowable information. Mathematics has undergone many changes throughout its history in terms of what tools are available to the mathematician and, more importantly, the acceptance of those tools. Mathematics is built on the idea that some logical operations are universal and, accepting that, we can build from these simple ideas to powerful conclusions. In order for this process to work, all mathematicians need to agree on what constitutes a viable operation. Without a base of acceptable tools we cannot build on another's work and we would all have to start from scratch. From the discovery of zero, to negative numbers, to complex numbers, mathematics has changed and that change took a while to be accepted.

To get an idea of this concept, consider the map coloring problem. This problem was posed in 1852 and asks, in two dimensions, what is the minimum number of colors needed to fill in an arbitrary map such that no two adjacent regions are the same color (regions that touch at only one point, i.e. corners, are not considered adjacent). Percy Heawood (1890) proved that it could be done in five colors, but not that this was the minimum. His proof was readily accepted by the mathematical community as a valid proof because it used traditional methods and could be verified by peer review. However, it turns out that only four colors are needed to solve this problem which led to the advent of the four color theorem by Kenneth Appel and Wolfgang Haken (1976). Their proof was not immediately heralded as a success, however. The problem the

mathematical community had is that a computer did the bulk of the computations involved in the proof. There where so many computations needed that it was infeasible to peer review by hand. The proof itself was similar to the method of exhaustion. Appel and Haken found the smallest set of possible counterexamples and showed that none of them worked; concluding that if there exists no counter example then their theorem must be true. Today, some still question the validity of this proof, although not of the theorem as other proofs have come out since then. This issue is going to become more complicated in the next ten years as quantum computers start to achieve widespread use in research.

Purpose of Study: UC Davis

I was granted the Edmondson Fellowship in 2017 by the University of California, Davis in order to explore a related question to the one just discussed. My colleagues at UC Davis wanted to know what use quantum computing could be in medical research. To that end, Dr. Mike Hogarth secured computation time with D-Wave Systems, a company which, at the time of writing, has the largest quantum annealing machine in the world. UC Davis was interested not only in what the machine could do, but also in how justifiable any resulting research was. In the medical field especially there needs to be a high degree of confidence in results if those results are ever going to affect patient treatment. For that reason we stayed away from any example calculations that could replace the human element of healthcare. Ensuring that a human always had the final say would increase trust in whatever system we developed.

The research team consisted of myself, an Undergraduate Mathematician and Physicist; Dr. Mike Hogarth, a pathologist; Dr. David Chin, a statistician; and Jupinder Parmar, a programmer. Together we not only had to learn how to encode problems on a quantum computer, but we also had to determine the best test problems to solve and present to the University as they had no specific problems in mind. Rather it was the technology and the challenges it presented to new research that interested UC Davis.

Purpose of Study: Personal

As the only member of the research team with a background in physics, much of the analysis of the quantum computer itself fell to me. Before our team convened I spent three months studying the published works on quantum computing as well as the propriety software of D-Wave Systems. As a result, I had the largest sway over what problem we would set out to solve because I had the best understanding of what a quantum computer does differently. As we went through our research I tried to keep in mind the question I wanted answered: how do current quantum computing techniques impact the field of mathematics and computer science and what, if anything, do we need to change about how we view problems to take advantage of this technology.

Literature Review

Quantum computing is a vast and quickly evolving field, but understanding some base terms and concepts is easy and provides a consistent basis for viewing new information. The first necessary term to understand is "quantum computing" itself. Up until this point I have been using this term as a catch-all for any computing mechanism that uses the principle of superposition. Erwin Schrodinger (1926) proposed a linear model of waves with his time dependent wave equation. In quantum mechanics superposition refers to the fact that, as a linear equation, if both A and B are solutions to the Schrodinger Equation, then A + B is as well. This leads to some interesting consequences in computing as noted by Richard Feynman (1982).

In a keynote speech at the MIT Physics of Computation Conference Feynman remarked that we do not have a complete understanding of the physical laws of the universe and that current (to his day) computing techniques could only scale so well. This is because classical computers are linear machines. They are very good at solving linear problems, but take a long time to solve more complex problems. Feynman noted that many physical problems are not linear and that simulations of these problems could be useful to physicists. He then proposed an interesting idea, that of the quantum computer. A quantum computer is a computer where each bit takes on a superposition state and then operations are applied to that bit (or qubit: quantum bit). Because classical computers have bits that take on either a true or false value, increasing the number of bits, N, in a classical computer scales the number of simulated states linearly (complexity $\sim N$). In this new quantum computer complexity would scale with $\sim 2^N$, drastically increasing the simulating power of a computer. This idea was interesting but

purely theoretical as, at that time, researchers had no idea how to even create a bit in a superposition state or how to apply operations to something with both a value of true and false. It was this second idea that motivated Peter Shor (1994) to invent an algorithm that used this superposition logic.

Shor's Algorithm can singlehandedly be pinpointed as the genesis of the current boom in quantum computing research as he came up with an algorithm that can factor numbers in polynomial time. Most current cyber security is based on the fact that this cannot be done. Now, Feynman had proposed a new type of computer and Shor has shown that it has an immediate and pressing use. These two facts led to a race to build a quantum computer. The main problem facing researchers is called decoherence (Moore, 1996). This is the idea that interacting with something in a superposition state will cause it to collapse to a classical state. This makes it difficult to perform operations on, and read the results of, quantum computers. Currently the largest quantum computer using Feynman's original vision, which we will call a Universal Quantum Computer, is owned by IBM and has 50 qubits (quantum bits) that were stable for 90 microseconds (Galeon 2017).

Quantum computers are fast, but decoherence is as well and it is difficult to perform operations on systems with such short life spans. Frustration with these limitations led D-Wave Systems to a novel take on quantum computing. Instead of creating a Universal Quantum Computer like Feynman, had envisioned D-Wave set out to create a quantum annealing machine as first proposed by Kadowaki and Nishimori (1998). Quantum annealing is different from universal quantum computing because it uses decoherence as its main computational force. Instead of applying a logic gate to

each qubit, quantum annealing sets up the interactions between qubits to govern the most likely state (true or false) that the qubit decoheres into. If the interactions are set up properly then the resulting state can be a specific solution to a problem. We will discuss this process in detail later.

Even quantum computers cannot escape the P vs NP question. I have alluded to the concept of P vs NP before with my discussion of "fast" and "slow" computational times. P stands for polynomial time, and NP stands for nondeterministic polynomial time. These times refer to how long it takes to find a solution to a problem with respect to the number of inputs. The problem, officially stated by Stephen Cook (1971), asks whether it is always possible to find a polynomial time solution to a class of problems, or if some problems are inherently more complex. The reason I bring this idea up will become clear in my discussion of P vs NP later in the paper. It is important to note that this is an unsolved problem and one of the Millennium Prize Problems (Clay, 2000) whose solution is worth one million USD.

There is a particular problem I want to highlight that is frequently examined when looking at the P vs NP problem, called the Traveling Salesman Problem. This problem has many practical applications and is intuitive to understand. The problem itself has been around for a long time, but it was first proved NP- complete (a subset of NP hard problems) by Held and Karp (1970). The problem itself asks what path provides the minimum distance between N cities given the following constraints: the path needs to visit each city exactly once and end in the starting city. Just as a traveling salesman would want to find the shortest route to peddle their wares, mathematicians are interested in the optimal path between several nodes on a graph.

An unrelated, but important, concept to cover is Support Vector Machines. Support Vector Machines, or SVMs, are a machine learning method used for classification created in their current form in 1992 by Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik (Boser et. a 1992). The method was invented to work for binary classification but can classify data points into an arbitrary number of classes by grouping classes several times and then looking at the overlap of these "super classes". This process does increase the training time of a SVM, but has little effect on the classification of new data points. To train the machine on M classes, M "super classes" are created. Each one contains all the classes except one. A binary classification is then made between the super class that does not contain class A, and class A. Looking at the intersection of all such classifications produces a single class that a point belongs to, assuming a perfect classifier. This is a linear increase in the time it takes to classify a new point, but a larger increase in the training time because of the greater degree of accuracy needed.

Quantum Strengths

To understand the problem I eventually settled on tackling and my proposed solution we need to talk about the strengths and weaknesses of Quantum Annealing Machines. The first thing to keep in mind is that a Quantum Annealing Machine is a physical solver using the base rules of the universe as a computational force. The issue that arises is that the universe is a fundamentally complicated place. To give you an idea let's examine the Ising Model that forms the basis for Quantum Annealing Machines.

Ising Model

The Ising Model, despite being named for Ernst Ising, was proposed by Wilhelm Lenz. Wilhelm Lenz was Ising's thesis advisor and proposed the model to him as a topic for his thesis. Ising found a general solution to the 1-D model in his unpublished thesis in 1924 (Ising). The Ising Model was originally a model of ferromagnetism. Ising's model calculates the energy of a system, E, (down to some constants) by:

$$E \sim \sum_{i,j} \mathrm{J}_{ij} x_i x_j + \sum_i h_i x_i$$

Where J_{ij} is the strength of the interaction between x_i and x_j and h_i is the strength of an external field at x_i. For magnets, each x_i represents an electron that has either positive or negative spin. Finding all the valid configurations of electrons for a given energy level, E, is an NP problem. This is because of the J_{ij} term. If $J_{ij} \equiv 0$ then we have a P problem. This can be seen by noting that each electron has no influence on those around it and the double sum goes to zero. Physically however, J_{ij} is non-zero because each electron wants all of its neighbors to either spin the same way, $J_{ij} < 0$, or have opposite

spin, $J_{ij} > 0$. Ising's original model was concerned only with ferromagnetism, which constrains $J_{ij} < 0$. The reason that Ising's model has become so famous is because it turned out not to only work for ferromagnetism, but also for anti-ferromagnetism, or indeed any system governed by pairwise interacting particles.

In physics it is often enough to look at a simpler case consisting of only nearest neighbor interactions. This simplification is what many people are familiar with, but the Ising Model itself describes a complete graph. The model is composed of a sum of pairwise interactions over all the elements in a system and the sum of the strength exerted by each individual element. The model considers both local solutions and global solutions. Armed with this knowledge we can create an artificial system that is governed by Ising's Model.

Isothermal and Adiabatic Systems

Thermodynamics tells us that a system always wants to be in equilibrium with connected systems. In other words, energy should be evenly distributed (for an interesting consequence of this look at the "inevitable heat death of the universe"). A more relevant consequence of this is the idea of a ground state. The ground state is the lowest energy state of a system possible. This energy is described by with the Ising Model. In fact, by setting the model equal to the ground state energy we can see the possible ground states (arrangements of electrons in the ferromagnetic case). However, solving this problem in general is NP complete. This is a specific case of the Ising Model. In general, the Ising Model describes the energy of a system given a particular

arrangement. We can now swap the inputs and outputs of the model to calculate possible systems given a particular energy level, E.

Let's tie this back to Quantum Annealing Machines. The Ising Model described one of the situations Feynman was talking about: a physical system whose simulation was difficult. However, observing this system is not that difficult. The Quantum Annealing Machine was created to take advantage of this fact. If it was possible to take a system governed by the Ising Model and put it in its ground state we could observe a specific solution to the Ising Model. There are two methods for doing this: Isothermally and Adiabatically. The first is by messing with what I call the "background energy" of the system. It is an isothermal process, i.e., the temperature of the system remains constant, but the heat changes. This comes in the form of modifying h_i in the Ising Model. More specifically another function, $C(t)$, can be added to h_i . This term is, in its simplest form, independent of h_i but dependent on time. The function starts large at $t = 0$ and $\lim_{t \to t_{final}} C(t) = 0$. The purpose of this term is to start many times the magnitude of the J_{ij} as to make the local interactions have very little impact on the energy of the overall system. Then as time goes on $t \to t_{final}$ and the local interactions (as well as h_i terms) are the governing forces. This ideally leads to a solution that has the lowest possible global energy state. However, depending on how fast *C(t)* converges to zero the system can become stuck in a local minimum. This can occur both if *C(t)* converges too quickly (particles don't have the time to react to other particles changing their value), or too slowly (particles don't have the energy to force a change in orientation).

Additionally, the fact that this is a time dependent process implies that it takes some amount of time to complete.

To avoid this problem altogether we took an adiabatic approach. Adiabatic is a term that means we keep the heat of the system constant i.e. no messing with the Ising Model at all during each computation. To do this, each x_i is placed in the superposition state that we talked about earlier. This is an "energetically favorable state" which means that it has a lower total energy than either of the two classical states. However, the superposition state is an unstable equilibrium. To find the classical ground state we choose J_{ij} and h_i terms such that each x_i has a preferred classical state. Then, when we "shock" the system out of its unstable equilibrium there is a high chance that it decoheres into a global minimum. We do not need to choose specific values of J_{ij} and h_i to cause this to happen, but by setting specific ones we can use this decoherence to produce a set of orientations that satisfy the Ising Model, which itself is an NP problem. This process is faster than the isothermal one because it is time independent, but also has a higher chance for error compared to a perfect tuned $C(t)$.

Quantum Problems

There are a few limitations to the Quantum Machine that we need to touch on now. The first was alluded too in the previous section, and this is one of error. Using the Quantum Machine is a probabilistic process, so how do we weight the dice in our favor? A related problem is how do we encode problems that have constants? As a probabilistic machine we cannot guarantee a specific value for a given x_i. For instance, in the traveling salesman problem we know that we want to start and end in the same city. The last problem is one that will seem novel to those accustomed to modern classical computers, and that is the number of available qubits.

Methods for Error Reduction

The most obvious method for error reduction is to simply run many trials. The fact that the ground state is the most likely state of the system means that we can reduce variance by increasing the number of trials. The adiabatic computing method we are using theoretically runs instantly. The part of the computation that takes time is resetting the system after each trial. Even so, we ran each of our trials 1000 times. An added benefit to this is that not only is the ground state the arrangement that will appear most often, but additionally it will have the lowest energy level (value calculated by the Ising Model). This is good because we do not necessarily know beforehand what that value is. If we do not put any constraints on the system (J_{ij} and h_i constant with respect to i, j) then the ground state would be zero (for an even number of qubits). However, that is an uninteresting case and in general the ground state will be a non-zero number. Because the ground state is the lowest value, we do not need to run enough trials for the

value to show up significantly more often than any other result. Instead we just need to run enough trials to have a reasonable chance of the expected value to appear once. We chose to run 1000 trials because the ground state would show up around 10 times on an average run with a poor encoding. It is faster to run the Quantum Machine many extra times than it is to compute the chance that the ground state will show up in any given run given an encoding and adjust our number of trials accordingly.

While increasing the number of trials is a purely mathematical method for reducing error, there are physical constraints on the system as well. The largest one has to do with the "background energy" I mentioned when describing the isothermal method. This background energy is still present in the adiabatic method, but it is a constant. A large background energy would outweigh the local interactions and arrange the system randomly. Essentially what happens is that $C(t)$ might have a large percent difference with the average J_{ij} and h_i terms compared to the percent difference between any two given J_{ij} or h_i terms. This causes differences in J_{ij} and h_i to be on the same scale as the variance of the system and essentially be treated as constants. Recall, when J_{ij} and h_i are constants then the ground state of the system will be zero (plus $C(t)$).

To combat this we can do two things. The first is to use quantum entanglement: take each x_i and make it into a pair of qubits. This process creates a link between two qubits independence of distance. The link either forces them to anti-align or to align. In this case, we force them to align. This doubles the energy required to change the state of the qubit because now you have to physically flip two qubits. Twice as much energy implies that the background energy would have to be twice as high to have the same effect. We can also tackle this problem from the other side and reduce the background

energy of the system. D-Wave Systems does this by lowering the temperature of the machine to almost absolute zero.

Encodings

Let's tackle the elephant in the room head on: how in the world do we actually get useful information out of an Adiabatic Quantum Annealing Machine? The machine is governed by the Ising Model which describes a complete graph. Each node on the graph can have a value associated with it and each edge can have a length. Every node x_i is weighted with an h_i and each edge from x_i to x_j is weighted with a J_{ij} . These h_i and J_{ij} terms are our inputs to the machine as well as a scalar V. The equation we use looks like this:

$$E = \left| \sum_{i,j} \mathrm{J}_{ij} x_i x_j \; + V \sum_i h_i x_i \right|$$

and is called the Hamiltonian of the system. V is a scalar that we use to control how much influence each sum has on the finial energy level. Because E is always positive we will also take the absolute value of the system. This is purely a physical constraint and I include it so that we don't forget the machine cannot differentiate between positive and negative final energies. The values we have not specified are E and x_i. E is the total energy of the system and the dependent value in our equation. We will use E to determine the ground state of the system as discussed earlier. x_i represents the value of the qubit at a specific node. It starts in a superposition state and ends in a classical state (we have no direct control over which one) Up until now we have been ignoring what that actual value of x_i is beyond that it is a Boolean variable. That's because it really

doesn't matter. Physically we say that $x_i \in \{-\frac{1}{2}, +\frac{1}{2}\}$ in reference to the spin of an electron, but there exists a simple isomorphism between these, $\{true, false\}$, $\{1, 0\}$, or $\{-1, +1\}$. For the most part I will use either $\{0, +1\}$ or $\{-1, +1\}$. The specific choice depends on which problem you need to encode.

Let's revisit the traveling salesman problem. It is immediately clear that being able to solve a minimization problem on a complete graph could generate a solution to the traveling salesman problem. The difficulty is in the details. The first problem is that the outputs of the quantum machine are: an energy value, which tell us which solution is best but nothing about the solution itself; and a series of x_i's, which are Boolean variables. That means that we need a sequence of true/false questions such that the answers to those questions tell us the shortest path the salesman can take between several cities. A simple set of questions would be "Does the salesman travel from c_i to c_j?" for each unique pair of cities c_i and c_j. To answer those questions we can assign a qubit, x_i, to each of those edges.

The next problem that arises is that not all paths are equal. This problem has a simple fix as we can assign h_i equal to the distance along the path x_i. If we stop to do a sanity check now we might notice that either of our potential choices for x_i still work. x_i true as +1 (in both potential encodings) still disincentivizes the system to take a given path and x_i false as -1 or 0 still give a lower overall energy than a true choice. We are forced to make a choice however, when we notice that each has a separate problem. If $x_i \in \{-1, +1\}$ then the system has no incentive to only visit each city once. Energy is always a positive value so the system will want a roughly equal number of true and

false responses. The opposite problem occurs with a $x_i \epsilon \{0, +1\}$ as now the system has no incentive to visit any cities at all!

In general, for my test cases I chose $x_i \epsilon \{-1, +1\}$. I do this because of two facts. First, ${x_i}^2 \equiv 1$, a constant. Second, the Hamiltonian is made up of two finite sums and as such is a linear equation. That means that if I generate multiple values for J_{ij} or h_i I can simply add the constituent values together and get a single scalar again. In this case I can use both of these facts to incentivize the system to visit each city only once. I can add to the Hamiltonian a term for each city c_i that looks like

$$(\sum_{i_z}(x_{i_z}) + n)^2,$$

where x_{i_z} is the collection of all paths going into or out of the city c_i and n = number of cities – *2*. This additional term is smallest when exactly two x_i are true and the rest are false. We want each city to have two true paths because each city needs to be arrived at and departed from. When this sum is expanded we generate J_{ij} values (as well as constants that we can ignore). You may note that the path is a circle and so should be independent of starting location. This is the simplest encoding that can be done. For instance, the problem can be further simplified by noting that by specifying a starting city we are also specifying an ending city and can reduce the complexity of the problem.

Scaling

Scaling is one of the largest problems facing quantum computing in its current state. There are physical barriers to creating an arbitrarily large quantum machine.

However, unlike a classical computer if a quantum computer does not have enough space to solve a given problem it cannot break it into linear chunks and solve them independently. The machine I was using at D-Wave Systems has 2000 qubits. Even with the simplification I mentioned above, the traveling salesman problem requires $(n-1)(n-2)/2$ qubits to encode (one for each unique path out of each of the N cities), which means that 64 cities (requiring 3,906 qubits) is the largest traveling salesman problem that can be solved directly on current day quantum computers. Even then the results are more prone to error than solving a smaller system. That is because the error reducing method of quantum entanglement I had mentioned before can be generalized to chains of qubits (not with quantum entanglement, just very high J_{ij} values). Using all available qubits increases the effect of the background noise.

It is possible to approximate larger systems by using a method similar to the isothermal process I described earlier. You randomly chose local sections of the graph to minimize and then stich them together to find a global minimum. This process loses the main strength of the quantum computer however. It is slow (must look at each node several times as the surrounding area changes), inaccurate (as it is harder to keep global constraints like only visiting each city once in mind), and worst of all only produces one result. As a probabilistic machine, what makes the quantum computer viable is that it can produce so many potential solutions so quickly.

Problems in Healthcare

The problem of scalability is solvable in the future, but UC Davis was interested in the current capabilities of quantum computing in addition to projections about the future. There are physical complications to work out to effectively scale a quantum computer but not physical impossibilities. Classical processors adherence to Moore's Law is a testament to the rate at which technology is progressing, but soon classical computing models will run into these physical limitations and quantum computing is an exciting alternative. When choosing a problem to demonstrate the power of an Adiabatic Quantum Annealing Machine I wanted something that was useful currently and scaled well with the size of quantum machines. In addition, any problem I wanted to tackle would need to be solvable by classical methods, or at least well approximated, to validate any solutions. The last feature I wanted in my healthcare specific example was usefulness without a quantum computer. I only had so many hours on D-Wave's machine and I wanted something to show for that time that was not only cool, but also useful.

Support Vector Machines

With these requirements in mind I chose to build a support vector machine for UC Davis. Training as well as classification of new data points on a support vector machine (SVM) is a linear process. As such classical computers can perform these operations without much difficulty. However, like all machine learning methods, data must undergo a significant amount of preprocessing before this linear process can

begin. During the preprocessing of the data we can leverage the quantum machine to significantly speed the training of the SVM.

The idea is simple: in healthcare regulations have driven hospitals and doctors to record and keep on file huge amounts of data. One example I worked with first hand was O_2 levels recorded every 5 seconds for each patient in surgery, except that "O_2 levels" is not an adequate term to describe the five data points that go into that measurement. Additionally, because recording continuous measurements is impossible, variance between the last five data points is reported at each point as well as a running average and variance measurement. One measurement of a patient taken every five seconds produces twenty data points. With all of this data available the question is: which points are useful? The example data set I was working with (MIMIC III) cleaned up, aggregated, averaged, and deidentified the data of sixty thousand ICU patients. This relational database was some of the cleanest data I have ever seen and it was still a beast to work with.

Support Vector Machines are one of the best machine learning methods to use when each measurement is complex compared to the number of measurements taken. In other words they look unsuited for working with something like the MIMIC database where the number of datapoints is vast. The problem that arises is that even polynomial time processes can start to take an incredible amount of time as the number of inputs grows. To solve this I wanted to use the quantum machine to sort through data sets and, quite literally, identify edge cases.

To demonstrate the value of identifying edge cases let me note a few facts about support vector machines. Support vector machines are supervised learners which means

that they need a training set and a validation set of data. The training set is a group of data points that we assign a classification and the SVM iterates through trying to create a matching classification. Once it achieves a high enough accuracy in training set, the SVM attempts to classify data in the validation set. These data points are new to the machine and are not used to modify it, but rather to check that it is learning in a way that generalizes to new data points. If all of the data in the training set was correctly identified edge cases then the SVMs performance on the validation set will be better as it is less likely to encounter a new scenario.

The next facet of support vector machines I want to highlight is their method of classifying data. SVMs find the hyperplane that evenly separates two classes of data in n-dimensional space. Edge cases are literally edge cases for SVMs. Once these points have been identified the rest of the points are increasingly irrelevant to finding the hyperplane, called the decision boundary. Often there is only one relevant point in each class of data and it will always be on the periphery. The challenge that arises is that SVMs don't map each data point into n-dimensional space as that is a costly process. Instead they take the inner product of pairs of data points in sperate classes using the kernel trick. SVMs do not know where each data point is, just how far it is to all the others. Calculating these values still takes a while and is the most computationally intense part of training a SVM. Assuming that both classes have an equal number of data points, N, there are N^2 inner products to calculate. If the SVM only had to consider edge cases when applying the kernel trick the number of effective inputs would diminish significantly. How significantly depends on the ratio of edge cases to what I call "textbook" cases (those that sit in the center of each class).

Method

The fact that reducing the number of effective inputs to a SVM is useful is ultimately irrelevant if we cannot identify edge cases faster than the SVM could train around them. Recall that we could find the minimum path between a set of cities in the traveling salesman problem without knowing precisely where those cities were located. Instead we only needed to know how far each city was from the others. This is precisely the type of information calculated by a SVM and the kernel trick. Only now we want to use this information to identify edge cases. I do that by pairing data points with their nearest neighbor and then removing one of them. I remove the one that maximizes the total distance between the remaining points. This is a process that could be accomplished procedurally with a classical computer, but the quantum computer lends something special. Instead of choosing a pair of points to optimize first and then iterating through all the remaining pairs, an Adiabatic Quantum Annealing Machine can act on all pairs of points at once.

Before I talk about the specific encoding I used to do this I want to talk about the cost of this procedure. The quantum machine is fast and will perform its job almost instantaneously, but to get to that point we need to calculate a lot of inner products. My stated objective for this method was to reduce the number of inner products that needed to be calculated. The reason we calculate inner products to feed the quantum machine is that these are different inner products than we talked about before and there are less of them. Instead of calculating the inner product pairwise between all x_i in class A and all y_i in class B we can calculate the inner product inside of each class. Keeping our earlier assumption that class A and B are the same size, calculating the distance between each

point inside a class gives information about the reverse relationship as well. Because of this we only have to calculate $\frac{N(N-1)}{2}$ inner products for each class.

The encoding itself uses $x_i \in \{0, +1\}$ unlike the traveling salesman problem. Also unlike the traveling salesman problem we are looking for a maximum value instead of a minimum one. There are two ways to achieve this. The first is to invert each term in the Hamiltonian and then proceed as normal finding the minimum. The second, is to multiply each term by negative one and continue as normal. Unfortunately, the Hamiltonian is a measurement of a physical quantity that cannot go negative, so while using a negative scalar is a valid mathematical solution to the problem, it is not one that works on a quantum computer if it would force the Hamiltonian below zero. To avoid that problem I inverted all the inner products after applying a thresholding program to them to ensure I did not get error introduced by inverting floating point numbers. Where I could avoid it I used a negative scalar instead of inverting. Recall the scalar V in the Hamiltonian. By using this value to ensure that one sum was ~10x larger than the other, I could use a negative scalar in front of the smaller sum to look for a maximum over a minimum.

The kernel I used in the SVM was the radial bias kernel, represented below by $K_{i,j}$. Note that this method is independent of kernel choice. The kernel gives us the distance between any pair of points, but to use that to eliminate points we need to make a choice about how we order them. In this case the quantum machine lets us choose not to order the points if we constrain ourselves to only eliminating half of the remaining points each iteration. We can do this by pairing nearest neighbors. To eliminate error, I applied another thresholding program here to ensure that the two points weren't too far

away from each other. If they were then the points where eliminated from consideration and the computer kept both. The threshold was adjustable, but one of the reasons we chose the radial bias kernel is that the difference between points is predictable and so the same threshold can be used for all runs in a data set.

We need to bias the quantum machine to keep only one point from each pair, and we want it to keep the point that is the furthest from all the remaining points. To do this we can multiply $K_{i,j}$ with x_i and x_j. This value is zero when either one or both of the points take on their false value. What is cool about the quantum computer is that x_i represents a physical measurement and so it must take on the same value every time it appears. All N products that contain x_i will have the same truth value. This is beneficial to us because the quantum computer looks at all the pairs of points simultaneously when looking at eliminating a data point. It would take a classical computer a very long time to do this because it would have to look at the results of x_i taking on both values at each point (and do that N times for each x_i). The problem we run into now is that the computer has no motivation to ever choose a true value for a given x_i. To solve this we use a second sum in the Hamiltonian and use linearity add it to the first. This sum is:

$$\sum_{i,j}(x_i + x_j - 1)^2$$

Note that this sum is zero exactly when one of the values is true and the other is false. Although we cannot eliminate any ${x_i}^2$ term here like we could with $x_i \epsilon \{-1, +1\}$ we can simplify by $x_i = {x_i}^2 \; \forall \; x_i$.

Anemia

In the healthcare setting we wanted to use the SVM to perform differential diagnosis. To that end we built a smaller scope machine focused around anemia. Anemia is a medical condition diagnosed by a low complete blood count (CBC). The low CBC can be thought of as a symptom because there are several potential causes of a low CBC such as low blood cell production, a bleed, or an autoimmune response destroying blood cells. Tests for one cause (such as a bleed) can be dangerous if something else is causing the problem (like an autoimmune disorder). Anemia is common, and easy to diagnose in general, but the specific sub types can become a problem. We chose to limit ourselves to just these three categories as they span all types of anemia, but are large enough that we would have a number of data points from each. As we are now looking at building a four way classifier (no anemia, low production, loss of blood, and breaking of blood) instead of a binary classifier we have to modify the SVM. Our research into this area is ongoing. However, initial results are promising. The largest barrier to success is incomplete data. We made the assumption that the appropriate data needed to differentiate between the types was already collected. This may not be the case as doctors do often perform follow up tests to determine the type of anemia a patient has.

Dangers of the Method

This method of simplification has limited real world application. It requires a data set large enough such that a reduction in the number of points in necessary. This is not an uncommon occurrence. What is, is the fact that the data points also need to be differentiated enough such that choosing a subset randomly to do analysis on is

unacceptable. This feature of a data set appears more often in things like medical data as the variance between different people is high. This variance could lead to a subset of much more important points and is something we saw while exploring anemia.

However, looking at anemia also exposed two other weaknesses of the method. The first is that a SVM built with the remaining data points is much more sensitive to outliers or misclassified data points. In the medical field, outliers are common and, unfortunately, many data sets that are differentiated well enough to require the application of this method will also contain outliers. To combat this problem we use a soft-margin SVM as opposed to the traditional hard-margin. This gives us some "wiggle room" to make sure that outliers do not skew the entire machine while still respecting the stratification of the data. The second problem is that the method works best when all classes have an equal number of points. This does not happen with our anemia data set because not all types of anemia appear with equal frequency. In addition when building anything more than a binary classifier the size difference between the classes grows with the number of classes. This has to do with the fact that we are calculating pairwise interactions inside of each class as opposed to between them. When we can make all the classes an equal size, N, this method saves time if we can reduce the original number of data points to less than $\sqrt{N}$. If the classes are too uneven in size then this method will never save time. Technically, each class could be reduced to a very small subset without losing accuracy. The problem is that we do not know where each of these points are in space. As such we don't know where it is in relation to the other class. We have to assume that the other class can be positioned on any side of these points. To that end, we have to keep the "edge cases" on every side which increases the minimum number

of points we need to keep. Each iteration of the method eliminates at most half of the remaining points. If the thresholding term I mentioned earlier is set too high then repeated applications of the method can lose the key data points and the SVM will become a very poor classifier. To solve this problem we have the thresholding term I already discussed. In addition we can also see that the SVM becomes very bad at classification. Because we have been eliminating data points, the remaining data is not as dense and the decision boundary will shift substantially if an important data point is eliminated. This is an indicator to go back a step and look at what happens with slightly more points.

I wanted to make a final note on the efficacy of the method. My method for training a SVM does not get around some of the problems inherent to SVMs. As such, I wanted to make sure that it was also compatible with other methods of error reduction commonly used with SVMs. I already used a soft margin classifier, but bagging and boosting are also compatible with the quantum machine. We have research ongoing into using less energetically favorable results from the quantum computer to boost.

Analysis

Why I'm Not a Millionaire

I said at the beginning of this paper that solving the traveling salesman problem was worth a million dollars. I also went on to claim that I had generated some solutions for it. The natural question is then: why am I not a millionaire? This question hits at a discussion that needs to be held in the near future about polynomial time equations. Quantum computing in general is going to change what kinds of problems we see as solvable. A true solution to the traveling salesman problem is a general solution. That means that it can produce all correct answers to a given set of initial conditions in polynomial time. What the quantum machine is doing is producing one correct answer in linear time. Linear time is how I have been thinking about the "speed" of a quantum machine. Because the machine is essentially a physical solver, it takes the same amount of time to run regardless of what problem you give it. The snag is that if the problem has too many variables then it simply cannot be done. This fact makes the traditional classifications of polynomial and nonpolynomial time irrelevant as there is no scaling with the number of inputs, just a cutoff point.

Right now quantum machines are so small that this hardly matters because the problems they solve could be polynomial or nonpolynomial and still possible to compute on a classical machine. As they scale in size though this will change. Take note of the method I used to eliminate data points. Maximizing the distance between all those points would take a classical computer 2^N steps and yet the quantum computer can do it in one. Ultimately quantum computers will not provide the answer to P vs NP

as that is a purely mathematical question asking if one can be reduced to the other. What they might do is make that distinction irrelevant.

Current Implementations of Future Tech

As they stand right now, quantum machines are not large enough to be useful. However, they have enough potential to be useful that they cannot be ignored. The largest problem I faced when trying to encode problems on the quantum machine was how linearly oriented my mind was. I have been trained to reduce problems to a linear space because that is what computers are good at solving (and people too). We have a huge library of linear approximations that speaks to how ingrained that mentality is. The ease at which the quantum machine could solve problems traditionally considered quadratic or exponential was amazing. The hard part was putting new problems into that framework. As quantum machines become more common and quantum algorithms start to be taught in schools this mentality will slowly change.

The area of research I am interested in right now is hierarchical models. These are a common method of modeling systems, especially in healthcare. Using a quantum machine to include interaction between variables will make these models much more accurate without impacting computation time significantly.

The method I employed to train a SVM was a good example of the power a quantum machine holds, but not a good example of an implementation of it. Ultimately, there are few data sets that would benefit from this form of preprocessing, but the fact that these maximization problems can be solved is huge. The main use that I foresee for quantum computing is not a speed up of existing methods as it does not offer a significant advantage with linear problems. Rather it is a refinement of existing

problems as we are able to eliminate some of the approximations we were forced to make previously. People always say that quantum machines are fast, but they leave off the fine print. Quantum machines only solve the problems they are good at solving. Adiabatic Annealing Machines will never replace classical computers, although universal quantum computers might if we can fix decoherence.

Glossary

Quantum Machine: In this paper an abbreviation for Adiabatic Quantum Annealing Machine. A type of quantum computer developed by D-Wave Systems that uses thermodynamics to solve problems.

Quantum Computer: A term used generally to refer to a problem-solving method that uses the principal of superposition or quantum entanglement. Traditionally refers to the Universal Quantum Computer.

Universal Quantum Computer: Also called a Three-State Quantum Computer. A type of quantum computer that stores bits in three states (on, off, superposition).

Superposition Principal: The fact that systems need not be found in their binary states, but instead can be found in a more energetically favorable state that encompasses both on and off simultaneously.

Ising Model: A model of magnetic materials that generalizes to any field of pairwise interacting particles. Typically, (in physics) only neighboring interactions are considered, but the model describes a complete graph.

Polynomial Time Problem: A problem where every possible solution can be generated in an amount of time that varies with the number of inputs in a polynomial (usually linear) fashion. ie a problem that can be solved quickly relative to the size of the problem.

Nondeterministic Polynomial Time Problems (NP): A problem that has no known solution in polynomial time. ie a problem that cannot be solved quickly (or often feasibly) with respect to the number of inputs.

NP-Complete: A subset of NP problems that are said to encompass all NP problems. This is because any NP problem can be reduced to a NP-Complete problem in polynomial time. A polynomial time solution to an NP-complete problem is a Millennium Problem and is worth 1 million dollars (or a proof that such a solution does not exist).

Boolean Variables: A variable that takes a value of either True or False. This can also be represented by any pair of numbers. A binary variable.

Encoding: A fixed J_{ij} and h_i for all values of i that defines a problem on the quantum machine.

Anemia: A medical condition diagnosed by a low complete blood count (CBC). The low CBC can be thought of as a symptom because there are several potential causes of a low CBC such as low blood cell production, a bleed, or an autoimmune response destroying blood cells. Tests for one cause (such as a bleed) can be dangerous if something else is causing the problem (like an autoimmune disorder).

Bagging: A machine learning method that builds many iterations of a machine with randomly selected training sets.

Boosting: A machine learning method that builds many iterations of a machine using information from the previous run to weight the next selection of data points.

Soft Margin: a commonly used method to make SVMs more resistant to error in data points. Essentially looks at a ball around each point instead of just the point itself.

Hard Margin: The default state of a SVM, does not employ a Soft Margin.

Isothermal: a term used to in thermodynamics to refer to a system that stays at a constant temperature.

Adiabatic: a term used in thermodynamics to refer to a system that stays at a constant heat.

Kernel Trick: the mathematical trick that makes SVMs useful. Calculates the inner product between 2 values in N dimensional space.

Absolute Zero: The theoretical coldest temperature possible. Particles cease to move due to thermal forces.

Bibliography

Boser, Bernhard E.; Guyon, Isabelle M.; Vapnik, Vladimir N. (1992). "A training algorithm for optimal margin classifiers". *Proceedings of the fifth annual workshop on Computational learning theory*. pp. 144.

Clay Mathematics Institute. (2000), "The Millennial Prize Problems". *Coll`ege de France.*

Cook, Stephen (1971). "The complexity of theorem proving procedures". *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. pp. 151–158.

Ernst Ising. (1925), "Contribution to the Theory of Ferromagnetism." *Z. Phys*. 31. pp. 253- 258

Feynman, Richard and Shor, Peter. (1982), "Simulating Physics with Computers" *SIAM Journal on Computing*, 26, pp. 1484-1509.

Heawood, P. J. (1890), "Map-Colour Theorems", *Quarterly Journal of Mathematics*, Oxford, 24, pp. 332-338

Held, M.; Karp, R.M. (1970). "The Traveling Salesman Problem and Minimum Spanning Trees". *Operations Research*. 18: 1138–1162.

K. Appel and W. Haken. (1976), "Every map is four colourable", *Bulletin of the American Mathematical Society* 82, 711-712.

Monroe, C., & Wineland, D. (1996). "Future of quantum computing proves to be debatable." *Physics Today*, 49, 107-108.

Schrodinger, Erwin (1926). "Quantisierung als Eigenwertproblem". *Annalen der Physik.* 384 (4): 273-376.

Shor, P. W. (1994). "Algorithms for quantum computation: Discrete logarithms and factoring." *Foundations of Computer Science*, 1994 Proceedings, 35th Annual Symposium, pp. 124-134.

T. Kadowaki and H. Nishimori. (1998), "Quantum annealing in the transverse Ising model" *Phys. Rev.* 58, 5355.