The Ising model: teaching an old problem new tricks

Zhengbing Bian, Fabian Chudak, William G. Macready[*], Geordie Rose

D-Wave Systems

August 30, 2010

Abstract

In this paper we investigate the use of hardware which physically realizes quantum annealing for machine learning applications. We show how to take advantage of the hardware in both zero- and finite-temperature modes of operation. At zero temperature the hardware is used as a heuristic minimizer of Ising energy functions, and at finite temperature the hardware allows for sampling from the corresponding Boltzmann distribution. We rely on quantum mechanical processes to perform both these tasks more efficiently than is possible through software simulation on classical computers. We show how Ising energy functions can be sculpted to solve a range of supervised learning problems. Finally, we validate the use of the hardware by constructing learning algorithms trained using quantum annealing on several synthetic and real data sets. We demonstrate that this novel approach to learning using quantum mechanical hardware can provide significant performance gains for a number of structured supervised learning problems.

1 Background

Not all computational problems are created equal. Some problems are so simple that even large instances may be solved rapidly. Other problems are intractable – once the problem is large enough there is essentially no hope of solving it exactly. Computer scientists have formalized this observation, and divided problems into complexity classes of varying difficulty. To enable this classification, both the kinds of problems we are interested in solving and the tools used to attack these problems must be defined. We consider two kinds of problems: decision problems having a yes or no answer, and optimization problems which seek to minimize a cost or energy measure. For both kinds of problems we consider their solution with either classical or quantum resources.

Consider the familiar Sudoku puzzle from the daily newspaper. The goal is to assign numbers 1 to 9 to the empty cells of a partially filled 9×9 grid such that a variety of constraints are satisfied (the numbers in each row, column, and in each of nine 3×3 sub-squares must be distinct). The decision or yes/no version of this problem asks whether there is an assignment

[*]Author to whom correspondence should be addressed.

of numbers to cells satisfying all the constraints. Intuitively, answering this yes/no question is no easier than solving the Sudoku in the sense that a yes answer likely requires exhibiting a solution to the puzzle. Many *constraint satisfaction problems* like Sudoku share the feature that a candidate solution to the yes/no question can be checked rapidly, but the finding of the solution which generates a yes response can be exceedingly difficult. For Sudoku this means verifying that in the solution all rows, columns, and subsquares consist of distinct numbers. The solution acts as witness to the fact that a given Sudoku is solvable. For these kinds of problems it is the finding of a witness, and not the checking that makes the problem hard. This class of problems where we can verify a yes answer quickly are called NP.[1]

The difficulty of a problem also depends upon the tools that can be brought to bear to attack it. Rather than dealing with every possible kind of computer that might be used to solve a problem, computer scientists have abstracted the essence of the computers that sit on our desks, and in our phones. Classes of difficulty are defined relative to this abstracted essence so that if a problem is hard for your cell phone, then it is hard for your desktop. This clever avoidance of the details is accomplished by considering how the difficulty of the yes/no question changes as the size of the problem is increased (imagine solving Sudoku on an $n \times n$ grid rather than just a 9×9 grid). The class of polynomial problems, called P, requires a solution time which increases as a polynomial of the problem size n, e.g. n^2. In contrast, other problems get harder much more quickly, and require exponentially more time as n grows, e.g. $\exp(n)$. As the problem gets larger the effort to solve an exponential problem dwarfs the effort needed to solve a polynomial problem.

In 1971 Stephen Cook [Coo71], and Leonid Levin [Lev73] independently showed that there is a class of NP problems having a simplifying feature. If we can solve any problem in this class then we can solve all problems in NP with only a little extra work. This subclass of NP problems is called NP-complete. NP-complete problems are the hardest in NP – if you can solve an NP-complete problem, then you can solve any problem in NP. The amount of extra work is polynomial so that if any one of the problems in NP-complete is in P, then all the problems in the NP class are in P. It is natural then to ask how hard NP-complete problems are. The overwhelmingly dominant position amongst computer scientists is that the NP complete problems are dramatically harder than the P class problems, though no proof of this conjecture has yet been found.

In this work we are concerned with combinatorial optimization problems which are more general than decision problems. However, optimization and decision problems are closely related. Given a finite set of objects and a measure which assigns a cost to each object in the set, a combinatorial optimization problem seeks the element of the set having minimal cost or energy. For example, Sudoku can be cast as an optimization problem. The objects in the set of possibilities represent the possible assignments of numbers to empty cells. The cost of any such object might be the number of constraints that the assignment violates. With these definitions a Sudoku is solved by finding a zero cost assignment so that no constraints are violated. Other optimization problems are more complex, and are not simply recastings of constraint satisfaction. As an example the famous traveling salesman problem asks for an ordering of cities to visit such that the total distance traveled is minimized.

[1]NP stands for non-deterministic polynomial. Similarly to the class NP, co-NP problems are the decision problems where the no answer can be checked quickly.

An optimization problem can be converted into a decision problem by asking if there is an object from the set having cost less than some value. By answering a sequence of these decision problems the minimal cost solution to the optimization problem may be obtained (albeit the number of queries needed may not be polynomial). If the decision versions of the optimization problem are NP-complete, then we say that the optimization problem is NP-hard.

1.1 Boolean satisfiability

The first problem identified by Stephen Cook in 1971 [Coo71] to be NP-complete was Boolean satisfiability; SAT for short. The input to a satisfiability problem is a Boolean formula involving logical variables $\{y_1, \ldots, y_n\}$ each taking the value **true** or **false**, and connected by the propositional operators $\neg$ (negation), $\wedge$ (and), and $\vee$ (or). A formula is satisfiable if the variables can be assigned values for which the formula evaluates to **true**. For example, $\neg y_1 \vee (y_2 \wedge \neg y_3)$ is a formula which evaluates to a truth value for each of the 8 possible joint assignments for y_1, y_2, y_3. This formula is satisfiable by $y_1 =$ **false** (with y_2, y_3 taking arbitrary values), and by $y_2 =$ **true** and $y_3 =$ **false** (with y_1 taking either value). For all other combinations of y_1, y_2, y_3 the value of the formula is **false**. In contrast, the formula $y_1 \wedge \neg y_1$ is clearly not satisfiable. Boolean satisfiability asks the yes/no question: is a given formula satisfiable? In simple terms, Boolean satisfiability is NP-complete because for some formulae involving n variables there is no known method which is significantly better than enumerating all 2^n combinations of inputs looking for a witness or showing that one does not exist.

The practitioner interested in solving decision or optimization problems must be aware of a significant limitation in the theory of computational complexity. As hinted above, the theory says only that there are SAT problems which are difficult to solve. However, the theory is silent on whether a typical SAT problem is hard or not. In fact, many industrially relevant SAT problems with millions of variables are routinely solved with modern DPLL (see [DGP04]) clause-learning SAT solvers. The restriction to worst case complexity is done for good reason. Average case complexity requires the specification of what 'typical' means (often a challenging problem in itself), and average case analysis is technically more demanding than worst case analysis.

If restrictions are placed on the types of Boolean formulae under consideration then the resultant problems may simplify. For example, k-SAT restricts attention to formulae of the form

$$C_1 \wedge C_2 \wedge \cdots \wedge C_M$$

where each clause C_m is the logical **or** of k variables or their negations, e.g. $C_m = y_1 \vee y_{12} \vee \neg y_{13}$. k-SAT is NP-complete for $k \geq 3$, but is polynomial solvable for $k \leq 2$. There are optimization variants of satisfiability as well. Given a set of clauses on k literals (variables or their negations) MAX-k-SAT asks for a variable assignment with the maximal number of satisfied clauses. Weighted MAX-k-SAT allows for positive weights associated with each clause and asks for the assignment maximizing the total weight of all satisfied clauses. Weighted and unweighted MAX-k-SAT is NP-hard even for $k = 2$ (more on this later).

Since Cook and Levin's pioneering work, a great many other problems have been shown to be NP-complete. A problem other than SAT, call it problem A, is shown to be in NP-complete by reducing Boolean satisfiability to A. Reducing Boolean satisfiability to A means providing a polynomial time algorithm for solving Boolean satisfiability assuming that we can solve A. The solution of A is used as a subroutine we can call polynomially many times to solve Boolean satisfiability. If A is easy (can be solved in polynomial time) then Boolean satisfiability is easy. Since we know Boolean satisfiability is hard and we can construct a reduction, then A itself must also be hard. Using this technique Richard Karp [Kar72] showed in 1972 that a great many problems are NP-complete.

2 Thermodynamics and NP-hardness

Modern physics contains at its core the assumption that there are simple, knowable rules that govern the behavior of the world around us. There are many of these, some believed to be more fundamental than others. Many physicists believe that the laws of thermodynamics, and in particular the second law of thermodynamics, are more likely to remain true as we discover more and more about our universe than any other set of laws. Quantum mechanics or general relativity could be shown to be facets of some grander underlying theory, but thermodynamics will likely remain as the bedrock upon which all other physical theories must sit.

The second law of thermodynamics has many formulations. One way to think about the law is that when you put two different physical systems in contact, they will attempt to thermally equilibrate with each other – heat flows from the hotter system to the colder one, until the whole system reaches some equilibrium temperature in the middle. This simple common sense observation has very deep roots. If it were to fail, much of our understanding of how the universe works would be shown to be false. As just one example, perpetual motion machines would become feasible.

2.1 The Ising model

There are an enormous number of physical systems we might want to model, from individual electrons to collisions of galaxies. The most useful models of nature are those that can be used to represent a large number of completely different systems. Understanding how such models work then leads to understanding all of the physical systems the model can be used to represent.

One of the most widely used models in physics is called the Ising model. It was initially proposed in the mid 1920s by Ernst Ising and Wilhelm Lenz as a way to understand how magnetic materials work. The approach modeled a magnetic material as a collection of molecules, each of which has a spin which can align or anti-align with an applied magnetic field, and which interact through a pairwise term with each other [Bru67]. Let $s_i \in \{-1, +1\}$ represent the spin of the ith molecule, V represent the set of all molecules, and $|V|$ represent the number of molecules. The energy of a collection of spins $\boldsymbol{s} = [s_1, \cdots, s_{|V|}]$ is

$$E(\boldsymbol{s}) = \sum_{(i,j)\in \text{neigh}} J_{i,j} s_i s_j + \sum_{i \in V} h_i s_i \equiv \langle \boldsymbol{s}, \boldsymbol{J}\boldsymbol{s} \rangle + \langle \boldsymbol{h}, \boldsymbol{s} \rangle \tag{1}$$

where h_i is the strength of the applied field at molecule i and $J_{i,j}$ acts as interaction field between neighbouring spins i and j. $\boldsymbol{h} = [h_1, \cdots, h_{|V|}]$ is the $|V|$-vector of magnetic fields, and the $|V| \times |V|$ matrix $\boldsymbol{J}$ has i, j element $J_{i,j}$. At low temperatures where the system is strongly biased towards low energy states, a positive $J_{i,j}$ favors anti-aligned neighbouring spins ($s_i s_j = -1$) since the product $J_{i,j} s_i s_j$ will be negative. If $J_{i,j}$ is negative the opposite occurs, s_i and s_j tend to align ($s_i s_j = 1$) so that $J_{i,j} s_i s_j$ is again negative. Though Ising did not realize it at the time this simple model provides excellent experimental agreement with the properties of many magnetic materials.

Over time it was realized that Eq. (1) could be used to model many different physical systems. Any system that describes a set of individual elements (modeled by the spins s_j) interacting via pairwise interactions (the quadratic terms $s_i s_j$) can be described in this framework. In the period 1969 to 1997, more than 12,000 papers were published using the Ising model to describe systems in fields ranging from artificial intelligence to zoology.

2.1.1 Thermodynamics of the Ising model

Even in 1920, it was well-known that if a system described by Eq. (1) was in thermal equilibrium at temperature T (using units where $\hbar = k_B = 1$), the probability of observing any particular configuration $\boldsymbol{s}$ follows a Boltzmann distribution, i.e., the probability of $\boldsymbol{s}$ is proportional to $\exp(-E(\boldsymbol{s})/T)$ where T is the temperature of the system. At high temperatures almost all configurations have the same probability as $T \gg E(\boldsymbol{s})$ for all $\boldsymbol{s}$. However, at low temperatures the $\boldsymbol{s}$ having the lowest energy becomes the most likely state. At exactly $T = 0$ only the state(s) $\boldsymbol{s}$ having the lowest energy will be observed.

This thermal equilibration process is driven by the second law of thermodynamics. Any system that can be described as an Ising model of the form Eq. (1), if placed in contact with any other system in thermal equilibrium, will attempt to thermalize with this other system. If this thermalization can be achieved, the most likely state of the system in Eq. (1) will be its ground state.

This leads to a fascinating conundrum, because finding the global ground state of the Ising model is NP-hard. This fact was established in 1982 by Francisco Baharona [Bar82] by constructing Ising equivalents of the logical operators $\neg$, $\wedge$, and $\vee$ of SAT. Treating $s_i = -1$ as **false** and $s_i = +1$ as **true** the truth tables for the logical operators can be encoded as minima in an Ising energy function. The Ising equivalents for $z = \neg y$, $z = y_1 \vee y_2$ and $z = y_1 \wedge y_2$ are given in Figure 1. From these basic elements an arbitrary SAT formula may be encoded as an Ising model so that the SAT formula is satisfiable if and only if the lowest energy state of the Ising model is 0.

Therefore it seems as though nature, via its fundamental statistically-driven tendency to thermalize all systems to the same temperature, is applying an enormous amount of computing power attempting to solve a large number of NP-hard problems. Thermalizing any of the physical systems to which Eq. (1) has been applied requires 'solving' an NP-hard problem, in the sense that if equilibration can occur the most likely state of the system is its ground state, which encodes the solution to an NP-hard optimization problem.

This is a peculiar observation – that perhaps the most fundamental aspect of our universe, embodied in the second law of thermodynamics, can be largely frustrated by the fact that its desired outcome – thermalizing physical systems – may be computationally intractable for a large number of systems.

Exploring these ideas is beyond the scope of this paper. For our purposes, we will simply use

Logical operator	Ising penalty
$z = \neg y$	$1 + s_z s_y$
$z = y_1 \wedge y_2$	$3 - (s_{y_1} + s_{y_2}) + 2s_z + s_{y_1} s_{y_2} - 2(s_{y_1} + s_{y_2}) s_z$
$z = y_1 \vee y_2$	$3 + s_{y_1} + s_{y_2} - 2s_z + s_{y_1} s_{y_2} - 2(s_{y_1} + s_{y_2}) s_z$

Table 1: Ising penalty functions representing the logical $\neg$, $\wedge$, and $\vee$ operators. The penalty functions have minimal energy equal to zero at configurations satisfying the constraint imposed by the corresponding logical operator. Thus, energy minimization realizes the logical operations.

the observations that physical systems described by Eq. (1) exist, and are driven by fundamental statistical physical laws to thermally equilibrate with their environments. If the system can thermalize, the ground state of the physical system will be the most likely state. This fact can be used to construct novel algorithms for attacking optimization problems.

2.2 Equivalence of the Ising and weighted Max-2-SAT models

The proof of NP hardness of the Ising model can also be used to show that weighted Max-2-SAT and the Ising model are two guises of the same problem.

Rather than converting weighted Max-2-SAT to the Ising model, we convert it to a quadratic unconstrained binary optimization problem (QUBO) defined over Boolean variables.[2] QUBOs are Ising models where the spin variables $s_i \in \{-1, +1\}$ are transformed to binary-valued variables $y_i \in \{0, 1\}$. This transformation is easily realized through $s_i = 2y_i - 1$. If the QUBO objective is written as $E(\boldsymbol{y}) = \sum_{i,j} y_i Q_{i,j} y_j$ (the QUBO terms linear in y arise from the diagonal elements $y_i Q_{i,i} y_i$ as $y_i^2 = y_i$ for binary valued variables) then

$$E(\boldsymbol{y}) = \langle \boldsymbol{y}, \boldsymbol{Q}\boldsymbol{y} \rangle = \langle \boldsymbol{y}, \tilde{\boldsymbol{Q}}\boldsymbol{y} \rangle + \langle \tilde{\boldsymbol{q}}, \boldsymbol{y} \rangle = \gamma + \langle \boldsymbol{s}, \boldsymbol{J}\boldsymbol{s} \rangle + \langle \boldsymbol{h}, \boldsymbol{s} \rangle$$

where $\gamma = \langle \mathbf{1}, \tilde{\boldsymbol{Q}}\mathbf{1} \rangle / 4 + \langle \mathbf{1}, \tilde{\boldsymbol{q}} \rangle / 2$, $\boldsymbol{J} = \operatorname{triu}(\tilde{\boldsymbol{Q}} + \tilde{\boldsymbol{Q}}^t)/4$, and $\boldsymbol{h} = \tilde{\boldsymbol{q}}/2 + \langle \tilde{\boldsymbol{Q}} + \tilde{\boldsymbol{Q}}^t, \mathbf{1} \rangle / 4$. In this expression $\tilde{\boldsymbol{q}}$ is the vector of diagonal elements of $\boldsymbol{Q}$, $\tilde{\boldsymbol{Q}}$ is the matrix of off-diagonal elements of $\boldsymbol{Q}$ (the diagonal elements are zeroed), for a square matrix A the operation $\operatorname{triu}(A)$ zeroes the lower triangular part of A, and $\mathbf{1}$ is the vector all of whose components are 1. Thus, up to an irrelevant constant there is a simple relationship between the $\boldsymbol{h}, \boldsymbol{J}$ of an Ising model and the $\boldsymbol{Q}$ of a QUBO. In the rest of this paper we will freely use both the QUBO and Ising forms as different representations of the same underlying problem.

Weighted Max-2-SAT to QUBO: Weighted Max-2-SAT seeks to maximize the weight of satisfied clauses, or equivalently to minimize the weight of unsatisfied clauses. A weighted Max-2-SAT problem is converted to a QUBO by translating each weighted clause into a corresponding QUBO energy contribution. By adding the contributions of each clause we derive a QUBO whose energy function $E(\boldsymbol{y})$ gives the total weight of violated clauses. Consider a

[2]QUBOs are a special case of pseudo-Boolean optimization discussed at length in [BH02].

weighted clause C_m with weight $w_m > 0$ given by $l_{m,1} \vee l_{m,2}$ where $l_{m,1}$ and $l_{m,2}$ are literals (variables or their negation). We represent C_m as $(l_{m,1}, l_{m,2}; w_m)$. C_m is unsatisfied if $\neg l_{m,1} \wedge \neg l_{m,2}$ where negation is realized as $\neg l = 1 - l$ for binary valued literals. We use $\overline{l}$ as shorthand for $\neg l$. The cost of this violation is w_m so that the energy penalty due to C_m can be written as $w_m \overline{l}_{m,1} \overline{l}_{m,2}$. There are 3 possibilities to consider for the signs of the literals in the clause:

1. no negative literals: $(y_{m,1}, y_{m,2}; w_m) \to w_m(1 - y_{m,1})(1 - y_{m,2})$
2. one negative literal: $(y_{m,1}, \overline{y}_{m,2}; w_m) \to w_m(1 - y_{m,1}) y_{m,2}$
3. two negative literals: $(\overline{y}_{m,1}, \overline{y}_{m,2}; w_m) \to w_m y_{m,1} y_{m,2}$

For length 1 clauses we have

1. no negative literals $(y_{m,1}; w_m) \to w_m(1 - y_{m,1})$
2. one negative literal: $(\overline{y}_m; w_m) \to w_m y_m$

We apply this conversion to all weighted clauses in the weighted Max-2-SAT instance and add the resultant contributions to obtain the QUBO representation.

QUBO to weighted Max-2-SAT: In this case we translate each QUBO contribution into a weighted clause. Adding QUBO contributions simply adds weighted clauses. First consider the bilinear term $Q_{i,j} y_i y_j$. There are two cases to consider according to the sign of the quadratic terms:

1. $Q_{i,j} > 0$: $Q_{i,j} y_i y_j \to (\overline{y}_i, \overline{y}_j; Q_{i,j})$
2. $Q_{i,j} < 0$: $-|Q_{i,j}| y_i y_j \to \{(y_i, \overline{y}_j; |Q_{i,j}|), (y_j; |Q_{i,j}|)\} - |Q_{i,j}|$

The negative $Q_{i,j}$ adds two clauses, one involving two variables and the other involving a single variable. Similarly, for the linear terms

1. $Q_{i,i} > 0$: $Q_{i,i} y_i \to (\overline{y}_i; Q_{i,i})$
2. $Q_{i,i} < 0$: $-|Q_{i,i}| y_i \to (y_i; |Q_{i,i}|) - |Q_{i,i}|$

Note that the conversion of negative QUBO terms to clausal form requires the energy to be shifted downward since all clausal contributions are positive. We add the weighted clause for all linear and quadratic terms in the QUBO. Some of the clauses resulting from this process may be combined and simplified. For example clauses $(l_1, l_2; w_1)$ and $(l_1, l_2; w_2)$ can be combined into $(l_1, l_2; w_1 + w_2)$. As an example

$$\begin{aligned} \text{Ising:} &\quad -s_1 + 2s_1 s_2 - 3s_2 s_3 \\ \text{QUBO:} &\quad -6y_1 + 2y_2 + 6y_3 + 8y_1 y_2 - 12 y_2 y_3 \\ \text{Weighted MAX-2-SAT:} &\quad -12 + \{(y_1; 6), (\overline{y}_2; 2), (y_3; 6), (\overline{y}_1 \vee \overline{y}_2; 8), (y_2 \vee \overline{y}_3; 12)\} \end{aligned}$$

all represent the same problem. Note that the Weighted Max-2-SAT representation is not unique.

3 Quantum optimization

As discussed in the introduction, the difficulty of a problem depends on the tools used to solve it. Quantum mechanics provides a superset of the classical resources used by existing computers, and thus offers the possibility to solve certain problems faster. The complete quantum mechanical description of a set of n quantum bits (qubits) requires the specification of 2^n complex numbers. These 2^n amplitudes are operated upon by Nature in parallel. As early as 1982 it was suggested that this parallelism might be exploited to speed computation [Fey82, Ben82]. A number of quantum algorithms are now known that are more efficient than the best known (and in some cases the best possible) classical algorithms. Arguably the best known quantum algorithm is Shor's Algorithm [Sho94], which factors products of prime numbers in time growing polynomially in the size of the integer to be factored.

Here we consider the specialization of a general model for quantum computation [FGGS00] to the solution of discrete optimization problems. This specialization, called *quantum annealing*, is conceptually analogous to the well-known simulated annealing heuristic used for minimization in discrete optimization.

From one perspective the hardness of optimization arises from deception. Consider a search space Y and a function $E(\boldsymbol{y})$ to be minimized that associates a cost or energy with each $\boldsymbol{y}$ in Y. For large optimization problems one of the most effective solution strategies is to improve upon an initial guess $\boldsymbol{y}_t$ by search locally amongst neighbouring configurations similar to $\boldsymbol{y}_t$. Thus, the next solution $\boldsymbol{y}_{t+1}$ is obtained by finding the best solution within the local neighborhood of $\boldsymbol{y}_t$; this smaller optimization can be usually carried out exactly. $\boldsymbol{y}_{t+1}$ then serves as the starting point for a new local search in the neighborhood centered at $\boldsymbol{y}_{t+1}$. This process stagnates at a local minimum where the final configuration has lower energy than all of its neighbours. For hard problems the greedy exploitation of local improvement may deceive the algorithm and lead it into a local minima whose energy may be much higher than the globally minimum value. For effective search, local exploitation must be tempered with global exploration. Sometimes things have to get worse before they can get better.

Simulated annealing: Simulated annealing (SA), proposed in [KJV83] and inspired by a physical process, is an attempt to balance exploration with exploitation. SA searches for the global minimizer $\boldsymbol{y}^\star$ in an indirect and stochastic way. One view of this process simplifies the picture by removing the stochastic elements of the algorithm. Rather than seeking the global minimizer $\boldsymbol{y}^\star$ directly we seek a probability distribution p over the search space. $p(\boldsymbol{y})$ defines the probability associated with element $\boldsymbol{y}$ in the search space. The highest probability p should be associated with $\boldsymbol{y}^\star$. p is determined by minimizing a function balancing explorative and exploitative terms. Physicists call this function free energy $F(p)$, and it is given by

$$F(p) = \mathbb{E}_p(E) - TS(p).$$

$\mathbb{E}_p(E) \equiv \sum_y p(\boldsymbol{y})E(\boldsymbol{y})$ is the expected energy under the distribution p and is minimized by the probability distribution[3]

$$p(\boldsymbol{y}) = \begin{cases} 1 & \text{if } \boldsymbol{y} = \boldsymbol{y}^\star \\ 0 & \text{for all other } \boldsymbol{y} \end{cases}.$$

[3]Assuming the globally minimal state is unique.

The term $S(p) = -\sum_{y} p(y) \ln p(y)$ is the entropy; the negative entropy is minimized when equal probability is assigned to all y. The inclusion of this term favors exploration of the entire search space. T (for temperature) is a parameter controlling the relative importance of these two terms. Simulated annealing begins at large T which favors widespread exploration of the search space. As the algorithm progresses T is decreased to narrow the focus down to promising regions where $\mathbb{E}_p(E)$ is low. Eventually, T is driven to zero so that the global minimum can be obtained.

For any T the free energy is convex (bowl shaped) as a function of p so that gradient descent on p is guaranteed to locate the globally minimal $p^\star$. In fact, the optimal $p^\star$ at T can be written down immediately as the Boltzmann distribution $p_T(y) = \exp\bigl(-E(y)/T\bigr)$. This observation is not directly helpful because exponentially many numbers are required to specify p_T (we need specify the probability of every y). As a result, approximations to p_T need to be used. Most commonly, a collection of samples drawn approximately from p_T are used to estimate p_T. It is at this point the stochastic nature of SA returns. Markov chain Monte Carlo (MCMC) is a method that allows for samples to be drawn from a probability distribution. It bears much in common with local search methods for optimization. A proposal distribution $v_T(y|y')$ is defined which for each y' defines a probability over all possible next states y. Typically, $v_T(y|y')$ is non-zero only for y in the neighborhood of y'. An initial distribution $\pi_0(y)$ is evolved by v_T into $\pi_1(y) = \sum_{y'} v_T(y|y')\pi_0(y')$. MCMC equilibrates (reaches a fixed point) at a distribution $\pi_\infty(y)$ satisfying

$$\pi_\infty(y')v_T(y|y') = \pi_\infty(y)v_T(y'|y)\,,$$

where the flow of probability out of y' into y is the same as the reverse flow. For SA, $v_T(y|y')$ is chosen so that the equilibrium distribution is $\pi_\infty = p_T$. The rate of convergence to π_∞ is governed by the gap between the largest and second largest eigenvalues of $v_T(y|y')$. The smaller the gap the slower the convergence.

The hardness of a problem for SA manifests itself in the sampling problem. The local nature of the MCMC proposal distribution can make large scale changes in y (i.e. far ranging exploration) difficult. This is why the solution p_{T^+} at a slightly higher temperature $T^+ > T$ is used to bootstrap the sampling at p_T. In hard problems there is typically a particular T where sampling becomes particularly difficult as the character of p_T can change dramatically over a small range of T. This abrupt change in the character of p_T is called a phase transition. The transition in the Ising model from a non-magnetic to a magnetic state at a particular temperature T (the Curie temperature for ferromagnets, and the Neel temperature for antiferromagnets) is an example of a phase transition.

Quantum annealing: From a computational point of view there is nothing special about using entropy as the objective to force exploration. Any function that smoothes the probability over the search space can serve the same purpose. In quantum annealing quantum processes are used to encourage exploration.

For concreteness, suppose that we are trying to solve problem P,[4] represented as a QUBO or its equivalent Ising model on n bits. Classical mechanics is generalized to quantum mechanics

[4]Here P refers to a specific problem instance and not the complexity class P.

by generalizing bits to qubits, classical states of 2^n binary configurations to quantum states, and the energy function E to a Hermitian operator H.

The 2^n classical states $\boldsymbol{y}$ form an orthonormal basis of a 2^n-dimensional vector space $\mathscr{V}$ and are denoted by $|\boldsymbol{y}\rangle$ in standard notation. When $n = 1$, the vector space $\mathscr{V}$ has dimension 2, and we can assume that the two classical states, 0 and 1, map to the canonical vectors $|0\rangle = [1\ \ 0]^\top$ and $|1\rangle = [0\ \ 1]^\top$. When $n = 2$, the 4 classical states map to the 4 canonical vectors in $\mathscr{V}$ written as $|00\rangle = |0\rangle \otimes |0\rangle$, $|01\rangle = |0\rangle \otimes |1\rangle$, $|10\rangle = |1\rangle \otimes |0\rangle$, and $|11\rangle = |1\rangle \otimes |1\rangle$, that is, the tensor product of the classical states of each of the 2 qubits. In general, the classical state $\boldsymbol{y} = b_1 \dots b_n$, where b_i is the ith bit of $\boldsymbol{y}$, can be written as $|y\rangle = |b_1\rangle \otimes |b_2\rangle \dots \otimes |b_n\rangle$. A quantum state is simply a normalized vector of $\mathscr{V}$. Thus, an arbitrary quantum state ϕ in the basis of classical states can be written as $|\phi\rangle = \sum_y \alpha_y |\boldsymbol{y}\rangle$, where the 2^n complex numbers α_y give the amplitudes in each of the 2^n basis vectors $|\boldsymbol{y}\rangle$, and satisfy $\sum_y |\alpha_y|^2 = 1$. $\langle\phi|$ is the Hermitian conjugate.[5] Quantum states cannot be measured directly, but when qubits are measured in state $|\phi\rangle$ each classical state $|\boldsymbol{y}\rangle$ will be seen with probability $|\alpha_y|^2$. For example, the single qubit state $(|0\rangle + |1\rangle)/\sqrt{2}$ is equally likely to be measured as either 0 or 1.

The operator H maps quantum states into quantum states, and for a system of n qubits can be represented as a matrix of size $2^n \times 2^n$. The energy of quantum state $|\phi\rangle$ is $\langle\phi|H|\phi\rangle$. Minimizing the energy $\langle\phi|H|\phi\rangle$ is accomplished by finding the smallest eigenvalue of H, and the corresponding normalized eigenvector(s). For our Ising problem P, each classical state $\boldsymbol{y}$ has energy $E_P(\boldsymbol{y})$. The operator H_P representing this problem is diagonal in the $\{|\boldsymbol{y}\rangle\}$ basis and satisfies $H_P|\boldsymbol{y}\rangle = E_P(\boldsymbol{y})|\boldsymbol{y}\rangle$. Thus, the classical states that have smallest energy $E_P(\boldsymbol{y})$ are also minimizers of $\langle\phi|H_P|\phi\rangle$ over all quantum states $|\phi\rangle$. Furthermore, the minimum value of the energy $\langle\phi|H_P|\phi\rangle$ is equal to the smallest eigenvalue of H_P which in turn is equal to $\min_y E_P(\boldsymbol{y})$. Even though the matrix representing the diagonal operator H_P is exponentially large, the operator H_P can be easily configured in hardware, because it decomposes by qubits (linear terms) and pairs of qubits (quadratic terms).

The goal of quantum annealing is to find a minimizer of H_P through a physical quantum evolution. This is accomplished in a similar way as SA. The process starts by considering a second "energy"-like operator H_X whose minimal energy state is the quantum state that has equal squared amplitude for all bit configurations so that all classical states are equally likely to be observed when measured. The operator H_X is easily configurable in hardware because it can be applied qubit by qubit (we make each qubit equally likely to be measured as 0 or 1). Furthermore, the quantum system can easily attain this minimal energy state. Note that unlike H_P, H_X is not diagonal.

We can, then, define a quantum version of a free-energy-like objective

$$\mathscr{F}(\rho) = \mathrm{tr}(H_P\rho) + \tau\, \mathrm{tr}(H_X\rho)$$

where $\rho = |\phi\rangle\langle\phi|$ and $\mathrm{tr}(H\rho) = \langle\phi|H|\phi\rangle$. ρ is a $2^n \times 2^n$ matrix, but acts like a probability in that all its eigenvalues are positive and sum to 1. With this definition the parallels to SA are direct. τ is initialized at a very large value, and minimizing $\mathscr{F}$ with respect to ρ is the same as minimizing H_X alone which makes all $|\boldsymbol{y}\rangle$ equally likely to be observed. Similarly, when $\tau = 0$

[5] If $|\phi\rangle$ is a column vector then $\langle\phi|$ is the row vector of complex conjugated elements. Also, note that $\langle\phi|\phi\rangle = \sum_y |\alpha_y|^2 = 1$.

minimization yields the solution to the optimization problem encoded by H_P. However, just as we faced a problem in SA due to the exponential size of p, here ρ is exponentially large. The solution to this problem in the present context however is vastly more satisfying, and relies on Nature to implicitly represent ρ.

The minimization of $\mathscr{F}$ (which amounts to diagonalizing $H_P + \tau H_X$ to find the lowest eigenvector) is accomplished by Nature which natively operates on exponentially large quantum states ϕ. Thus, by setting initial conditions to define $\mathscr{F}(\rho)$ and relying on physics we can circumvent the problems associated with the exponential size of ρ. Given that Nature solves our problem why do we need to bother with the H_X contribution encouraging exploration? The answer to this is that it may take an exponentially long time for the physical system to relax to its lowest energy state. We recall that the Ising optimization problems facing Nature are difficult, and even physical evolution can get stuck in local minima for long periods of time. Fortunately, the adiabatic theorem [AR04] can be applied to the system to speed the relaxation to lowest energy states. In simple terms, if the system is initialized at a minimum energy state at large initial τ, and the decrease of τ over time is sufficiently slow, the system will remain at minimum energy state throughout the evolution. In other words, by annealing τ from large values to zero we speed the relaxation to the globally lowest energy state of H_P. The adiabatic theorem relates the rate at which τ can be decreased to the eigenvalues of $H_P + \tau H_X$. More conventionally τ is expressed as $\tau = (1-s)/s$ so that the free-energy-like objective can be restated as $\mathscr{F}(\rho) = \operatorname{tr}\big((sH_P + (1-s)H_X)\rho\big)$ where s now increases from 0 to 1 during quantum annealing.[6]

The parallelism inherent to quantum mechanical evolution can translate into significantly improved optimization. The first quantitative result was derived for unstructured search [Gro96]. Consider a classical energy function $E(\boldsymbol{s}) = -1$ for $\boldsymbol{s} = \boldsymbol{s}^\star$, and $E(\boldsymbol{s}) = 0$ for all other configurations. This problem is extremely difficult as the energy function provides no hints as to what $\boldsymbol{s}^\star$ might be. For strings of length n the best classical search (sampling without replacement) requires $\mathscr{O}(2^n)$ time to locate $\boldsymbol{s}^\star$. Quantum mechanically this can be improved to $\mathscr{O}(2^{n/2})$ using a specially tuned interpolation $g(s)$ from H_X to H_P, i.e. $H = (1-g(s))H_X + g(s)H_P$ [RC02]. While unstructured search remains exponentially hard, the scaling exponent is halved. Carefully engineered optimization problems can be constructed for which quantum annealing is exponentially faster than its classical counterpart SA [FGG02]. Other energy functions with structure can also be constructed in which QA offers no speed-ups beyond the unstructured search square root increase [Rei04]. In general, the efficacy of QA is governed by the minimum gap $\min_s E_1(s) - E_0(s)$ where $E_0(s)$ is the lowest energy eigenvalue of $H(s) = (1-s)H_X + sH_P$, and $E_1(s)$ is the second lowest eigenvalue. However, determining the minimum gap is as difficult as the original problem, and theory offers little practical guidance.

Exploration combining both thermal and quantum mechanisms is possible. In this case the appropriate free energy function is

$$\mathscr{F}(\rho) = \operatorname{tr}(H\rho) - TS(\rho) \tag{2}$$

where $H = sH_P + (1-s)H_X$ and $S(\rho) = -\operatorname{tr}(\rho \ln \rho)$ is the generalized definition of entropy. The limits $T \to 0$ and $s \to 1$ remove the explorative effects of the thermal and quantum annealing processes respectively.

[6]The annealing parameters s should not be confused with $\boldsymbol{s}$ the vector of spin values.

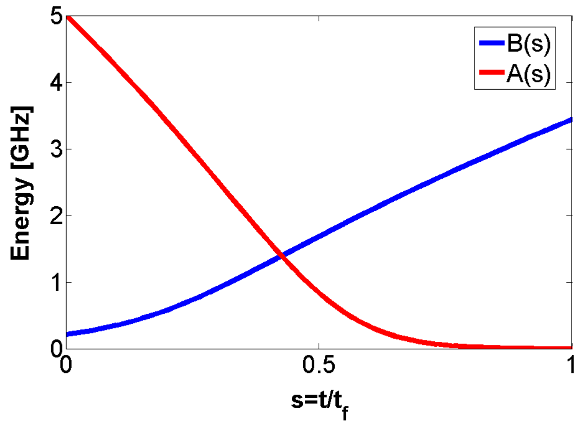


Figure 1: Experimentally measured envelope functions $A(s)$ and $B(s)$ for the 128-qubit processor used to perform the experiments reported on here.

3.1 Quantum annealing realized in hardware

We test the effectiveness of a physical realization of quantum annealing where we exploit the parallelism inherent in quantum evolution. The hardware we use embodies a particular operator for H_X, and allows for the expression of a class of Ising/QUBO/weighted Max-2-SAT problems for H_P. A detailed physical description of the underlying hardware components can be found in [HJB+10, JBM+10, BJB+10]. Due to the underlying physics the weighting of H_X and H_P is not linear in s, but has the form $A(s)H_X + B(s)H_P$, where $A(s)$ and $B(s)$ are referred to as *envelope functions*.[7] This still allows for controlled damping of the exploratory effects of H_X through s. Figure 1 shows the functional form of these functions. The hardware however, is less flexible regarding changes in temperature T. Though slight changes in T are possible, we shall consider the temperature fixed at a low but non-zero value. Thus, runs of the hardware will not deterministically return the global minimum of $E_P(\boldsymbol{y})$, but sample from a distribution centered at low energy configurations.

The class of Ising problems realizable in the current hardware design is limited in the connectivity allowed between variables. Due to the local nature of physical interactions, every qubit is limited to interacting with a small subset of the other qubits. The hardware we use has a connectivity that is driven by engineering feasibility. The connectivity is scalable to very large qubit arrays, and the test problems solved here were solved on hardware consisting of 128 qubits. The connectivity of allowed interactions for the 128 qubit chip used in the experiments reported here is shown in Figure 2(a). Not all qubits had equivalent functionality, and for the experiments presented here we restricted ourselves to the subset of 52 qubits shown in Figure 2(b).

Arbitrary qubit connectivity can be simulated by constraining the values of certain neighbouring qubits to be equal to effectively build paths between qubits that must interact. For example, if we required q_1 to interact with q_{13} we can define a q_1–q_5 interaction which constrains the values taken by q_1 and q_5 to satisfy $q_1 = q_5$. This constraint is easily realized as an

[7]Our convention will be that $A(s)$ and $B(s)$ will have units of energy, and H_X and H_P will be dimensionless.

Ising penalty $-Jq_1q_5$ which contributes J if $q_1 \neq q_5$ and $-J$ if $q_1 = q_5$. In low energy states q_5 becomes a copy of q_1, and now q_5 can interact with q_{13} as if they are neighbors.

3.2 Experimental validation of optimization efficacy

As validation of hardware quantum annealing we generated a variety of Ising problems where the parameters $\boldsymbol{h}$ and $\boldsymbol{J}$ are independently sampled from a zero mean Gaussian distribution. We plot the energy relative to the exact global minimal energy (which can be obtained with dynamic programming for these small lattices).

Figure 3 shows the time required to reach a certain level of optimization. For a given optimization problem defined by $\boldsymbol{h}$ and $\boldsymbol{J}$ we find the global minimum E_{min} and global maximum E_{max}. If the hardware returns a state having energy e we define optimality as $\alpha(e) = (e - E_{\text{min}})/(E_{\text{max}} - E_{\text{min}})$ so that $\alpha = 0$ corresponds to the globally best energy and $\alpha = 1$ corresponds to the globally worst energy. A point (t, α) of Figure 3 is the time (in seconds measured on the left hand y-axis) taken by the hardware to find a state of optimality α. The bands of color denote a decomposition of the total hardware time into its constituent components. Blue shaded regions denote a fixed setup overhead required to program the hardware. After this preprogramming time the problem is solved 2000 times. The gold region is the cumulative annealing time taken to find the solution. The red readout time is the time taken to read out qubits, and the brown thermalization time is the time taken for the hardware to cool after energy is dumped into the chip for programming and qubit readout. The results are the average of 300 problems where the problems were defined so that there were either 3, 31 or 511 distinct h and J values uniformly spaced between -1 and +1. Each of the h_i and $J_{i,j}$ parameters is selected independently and uniformly from the set of allowed values. The red line near zero is the fraction of times out of 300 where the given value was not attained in any of the 2000 samples.

For comparison, the vertical black line shows the performance of an algorithm yielding the lowest energy from hill descents starting from 50 random starting configurations. This particular chip is seen to be an effective and reasonably fast optimizer. The dominant error mechanism is not the optimization itself, but rather the precision to which the problem can be specified. An Ising parameter programmed to a particular h_i or $J_{i,j}$ is experimentally realized as a sample from a Gaussian centered on the desired value, but with a significant standard deviation. Thus, problems with well-spaced h/J values are more likely to be insensitive to small parameter errors and thus more likely to be solved correctly in hardware. This is validated in the experimental results. Problems with 3 distinct h/J values $\{-1, 0, 1\}$ (Fig. 2(a)) are solved to optimality much more rapidly than problems with 511 values $\{-1, -1+2^{-8}, \cdots, 0, \cdots, 1-2^{-8}, 1\}$ (Fig. 2(c)).

4 Putting the Ising model to work

4.1 Structured learning

We have seen how Ising energy functions can be minimized by hardware-realized quantum annealing. As hardware scales to larger qubit lattices this offers the promise of fast large-scale

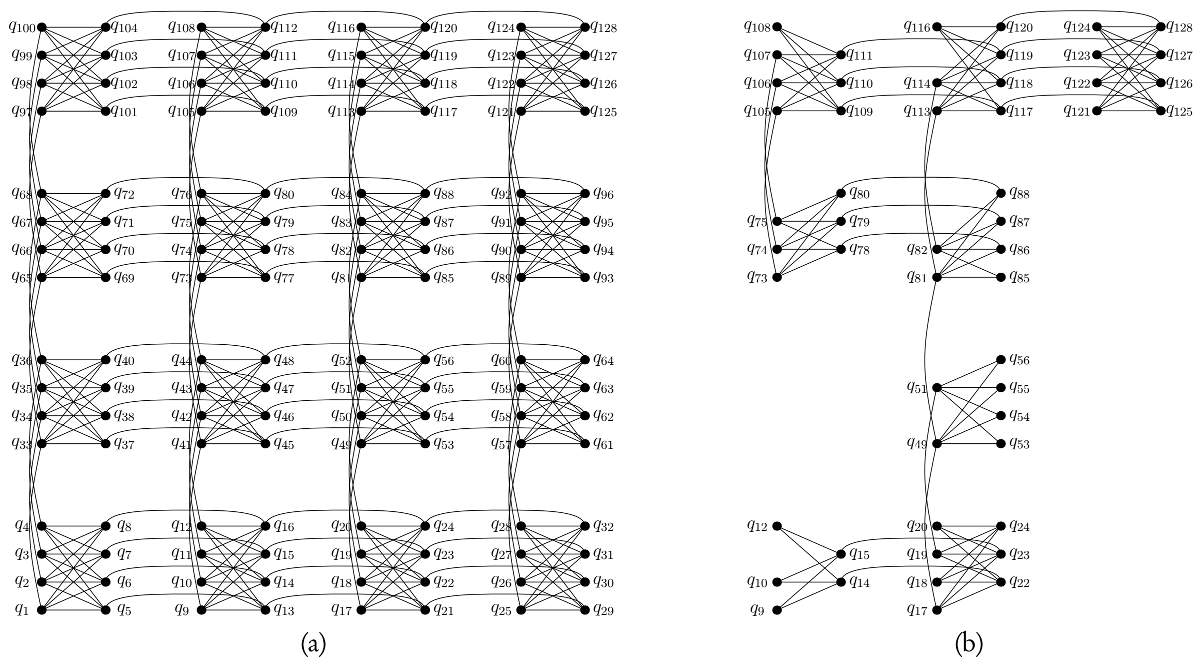
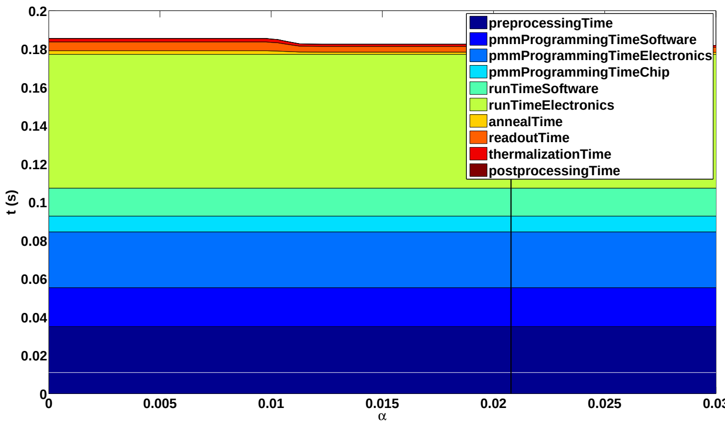
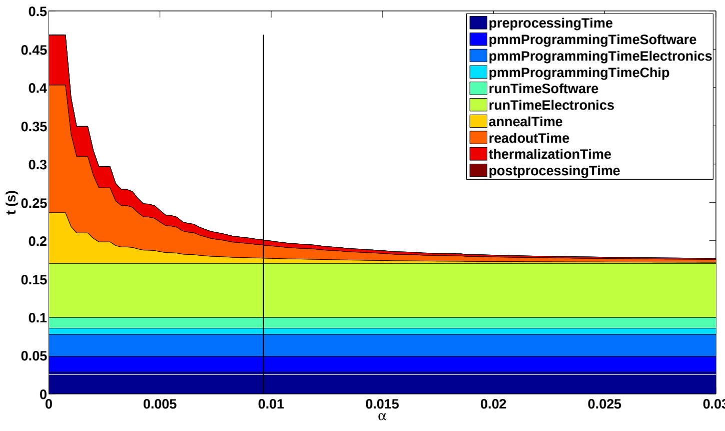


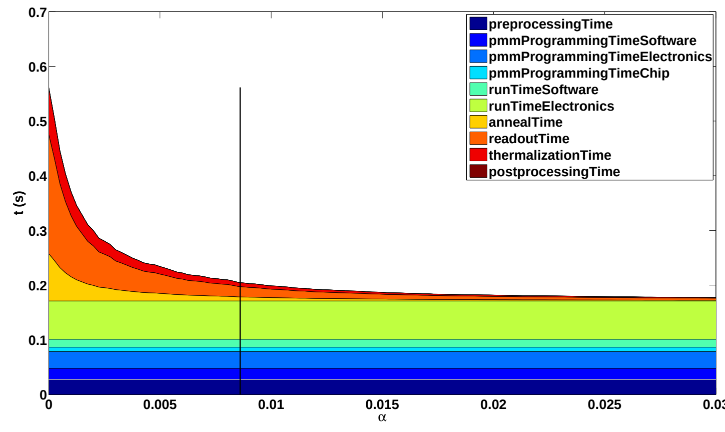
Figure 2: Edges between qubits q_i and q_j indicate an $J_{i,j}q_iq_j$ of tunable strength $J_{i,j}$. Each qubit yields a value of either -1 or +1 when measured. In addition to these quadratic interactions all tunable linear terms h_iq_i are also available. (a) Connectivity of the full 128 qubit chip. (b) Connectivity between the 52 functioning qubits.



(a) 3 values



(b) 31 values



(c) 511 values

Figure 3: Average run time decomposed into its constituent parts to solve randomly generated Ising problems with 3, 31 and 511 distinct h/J values. Note the different *y*-axis scales in each on (a), (b), and (c).

optimization of Ising energy objectives, and opens the door to applications built upon this capability. In this section we explore an application of Ising optimization to machine learning.

The NP-hardness proof for Ising optimization relies on the modeling of hard logical relationships between Boolean variables. Additionally, by varying clause weights (as in weighted Max-2-SAT) we may also model softer relationships between variables. Thus, large scale Ising models offer the potential to unify both symbolic logic-based and analogical connectionist approaches to artificial intelligence [Min91]. As an illustrative example we consider the problem of learning a mapping from arbitrary inputs $\boldsymbol{x}$ to a sequence of output bits $\boldsymbol{y}$. The output $\boldsymbol{y}$ may be a binary description of a complex object. For example, given a sentence $\boldsymbol{x}$ as input, $\boldsymbol{y}$ might label the words in the sentence with their corresponding part of speech (e.g. noun, verb, article, adjective, preposition, etc). Labels can be represented with a collection of binary-valued variables $\{y_{j,p}\}$ defined so that

$$y_{j,p} = \begin{cases} 1 & \text{if the } j\text{th word has part of speech } p \\ 0 & \text{otherwise .} \end{cases}$$

There are hard relationships that must be satisfied, e.g. every word must be labeled with a single part of speech tag so that $\sum_p y_{j,p} = 1$ for all words j, or every sentence must contain a verb $\sum_j y_{j,\text{verb}} \geq 1$. These logical relationships can be encoded as QUBO terms through the use of penalty functions so that, for example, the constraint requiring that every word must be labeled can be enforced by adding $M\bigl(1 - \sum_p y_{j,p}\bigr)^2$ for each word j, where M is a large positive weight that makes labelings violating the constraint costly.[8] However, some relationships are not hard, and are best treated probabilistically. Since nouns are often preceded by adjectives it might be useful to add a weighted clause of the form $(\bar{y}_{j,\text{noun}} \vee y_{j-1,\text{adjective}}; w)$ which says that if word j is labeled as a noun then word $j-1$ should be labeled as an adjective. However, this is clearly not always true, but is an effective rule of thumb. Thus, the weight w assigned to this clause should not be unduly high in order to allow for violations of this heuristic. This approach to softening logic has been applied with much success recently in various learning algorithms [RD06], and is a natural fit to the Ising model.

The weight of the rule might be set once and for all, but more flexibility is provided by letting the weight be determined by the input sentence $\boldsymbol{x}$, i.e. $w = w(\boldsymbol{x})$. Deferring for the moment the precise manner by which the weights are determined how do we label a sentence? It is natural to ask for the labeling that minimizes the weight charged by violating the rules of thumb expressed in the weighted clauses. As we have seen, weighted Max-2-SAT is equivalent to a QUBO so the labeling $\boldsymbol{y}$ of a sentence $\boldsymbol{x}$ can be determined as

$$\boldsymbol{y}(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}} \sum_{i,j} y_i Q_{i,j}(\boldsymbol{x}) y_j \ ,$$

where $\boldsymbol{Q}$ results from the translation to a QUBO and depends on $w = w(\boldsymbol{x})$. Written this way the parsing of a sentence is nothing but minimization of an appropriate Ising model!

How do we define the parameters of the Ising model so that Ising minimization yields good labelings? This problem is solved by utilizing a set of training examples $\mathscr{D} = \{\boldsymbol{x}_d, \boldsymbol{y}_d\}_{d=1}^{|\mathscr{D}|}$. We

[8]Inequality constraints can be treated similarly with the introduction of slack variables.

provide a learning algorithm with examples of sentences and their correct labelings (generated by human grammarians). The clause weights buried in $\boldsymbol{Q}$ are defined so that the labels assigned by Ising minimization agree with those assigned in the training set. Given a parametric dependence of $Q_{i,j}$, say $Q_{i,j} = \sum_k w_{i,j}^k \phi_k(\boldsymbol{x})$ where $\phi_k(\boldsymbol{x})$ are supplied (e.g., rules of thumb), then the $w_{i,j}^k$ are set by requiring that the known labels $\boldsymbol{y}_d$ be obtained as the results of the QUBO minimizations for each labeled sentence $(\boldsymbol{x}_d, \boldsymbol{y}_d) \in \mathscr{D}$.

Our focus here is not on a particular learning problem (e.g. part of speech labeling), but rather validating the fact that we can sculpt $\boldsymbol{x}$-dependent Ising energy functions, and optimize these energy functions using quantum annealing to solve learning problems. We consider two types of learning algorithms, deterministic algorithms that output a single prediction $\boldsymbol{y}(\boldsymbol{x})$, and probabilistic algorithms that output a distribution over possible outputs $P(\boldsymbol{y}|\boldsymbol{x})$. The deterministic algorithms effectively run at zero temperature ($T = 0$), while the probabilistic algorithms run at finite temperate ($T > 0$).

4.1.1 Learning at zero temperature

We follow an approach proposed in [TGK03]. As discussed above, the mapping from $\boldsymbol{x}$ to $\boldsymbol{y}$ is modeled as

$$\boldsymbol{y}(\boldsymbol{x}) = \arg\min_{\boldsymbol{y}} \mathscr{E}(\boldsymbol{x}, \boldsymbol{y}) \qquad \text{with} \qquad y_i \in \{0, 1\}$$

where $\mathscr{E}(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{y}, \boldsymbol{Q}(\boldsymbol{x})\boldsymbol{y} \rangle$ is an $\boldsymbol{x}$-parameterized QUBO model. The minimization over $\boldsymbol{y}$ is accomplished using hardware quantum annealing. In order to account for the finite temperature of the hardware multiple candidate minima are measured and the configuration having the lowest energy is used. The order statistics of this process lowers the effective temperature to near zero.

To determine how $\boldsymbol{Q}$ depends on $\boldsymbol{x}$ we write $\boldsymbol{Q}$ as a linear combination of user supplied matrices (the rules of thumb expressed in matrix form)

$$\boldsymbol{Q}(\boldsymbol{x}; \boldsymbol{w}) = \sum_{\alpha} w_\alpha \boldsymbol{Q}_\alpha(\boldsymbol{x})$$

so that $\mathscr{E}(\boldsymbol{x}, \boldsymbol{y}) = \mathscr{E}(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{w}) = \sum_\alpha w_\alpha \mathscr{E}_\alpha(\boldsymbol{x}, \boldsymbol{y}) \equiv \langle \boldsymbol{w}, \tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}, \boldsymbol{y}) \rangle$ with $\mathscr{E}_\alpha(\boldsymbol{x}, \boldsymbol{y}) = \langle \boldsymbol{y}, \boldsymbol{Q}_\alpha(\boldsymbol{x})\boldsymbol{y} \rangle$ and $\tilde{\boldsymbol{\mathscr{E}}}$ being the vector whose components are $\mathscr{E}_\alpha$. The functions $\mathscr{E}_\alpha$ (and thus $\boldsymbol{Q}_\alpha$), called *features* in the machine learning literature, are assumed to be known and provide predictive hints which are boot-strapped into effective predictors through the identification of the optimal linear combination.

The optimal weighting $\boldsymbol{w}$ of features is determined by minimizing an objective balancing two contributions. One contribution $R(\boldsymbol{w})$ seeks to minimize the errors on the training set, and the second contribution $\Omega(\boldsymbol{w})$ ameliorates problems of overfitting by favoring "simpler" models. Consequently, we write the best $\boldsymbol{w}$ as

$$\arg\min_{\boldsymbol{w}} \{ \lambda \Omega(\boldsymbol{w}) + R(\boldsymbol{w}) \}.$$

The λ parameter balances the relative importance of the two contributions. Since we do not know λ it is customarily set by *cross-validation*. In cross-validation the training data $\mathscr{D}$ is partitioned into two disjoint sets. A model is learned from one partition of the data using a given

λ, and its accuracy is tested on the remaining partition. By averaging over many partitions we can estimate how well the models generalize off the partition on which they were trained as a function of λ. We select the λ having the smallest generalization error across a number of randomly selected partitions.

The regularization term $\Omega(\boldsymbol{w})$ favors simpler models having smaller $\boldsymbol{w}$. Typically this is measured using either 1- or 2-norms. In this work we most often use the 2-norm:

$$\Omega(\boldsymbol{w}) = \langle \boldsymbol{w}, \boldsymbol{w} \rangle / 2. \tag{3}$$

This regularization favors $\boldsymbol{w} \approx 0$ so that features not useful for prediction are zeroed out.

To define $R(\boldsymbol{w})$, the training set error, we first quantify the cost of errors. Since $\boldsymbol{y}$ is a bit-string it is often natural to use Hamming error. The Hamming error

$$\Delta(\boldsymbol{y}, \hat{\boldsymbol{y}}) = \sum_i (y_i - \hat{y}_i)^2 = \sum_i (y_i + \hat{y}_i - 2y_i\hat{y}_i)$$

sums the number of bits which differ in bit-strings $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$. With this measure the average error on the training set is

$$\frac{1}{|\mathscr{D}|} \sum_{(\boldsymbol{x}_d, \boldsymbol{y}_d) \in \mathscr{D}} \Delta\big(\boldsymbol{y}_d, \boldsymbol{y}(\boldsymbol{x}_d; \boldsymbol{w})\big) \tag{4}$$

where $\boldsymbol{y}(\boldsymbol{x}_d; \boldsymbol{w}) = \arg\min_{\boldsymbol{y}} \langle \boldsymbol{y}, \boldsymbol{Q}(\boldsymbol{x}_d; \boldsymbol{w}) \boldsymbol{y} \rangle$. This training set error, which measures the number of bits differing in the predictions and training data, results in a difficult optimization problem that may be discontinuous in $\boldsymbol{w}$. Consequently, we instead measure training set error as

$$R(\boldsymbol{w}) = \frac{1}{|\mathscr{D}|} \sum_{(\boldsymbol{x}_d, \boldsymbol{y}_d) \in \mathscr{D}} \max_{\boldsymbol{y}} \big\{ \Delta(\boldsymbol{y}_d, \boldsymbol{y}) + \mathscr{E}(\boldsymbol{x}_d, \boldsymbol{y}_d; \boldsymbol{w}) - \mathscr{E}(\boldsymbol{x}_d, \boldsymbol{y}; \boldsymbol{w}) \big\} \tag{5}$$

where the optimization over $\boldsymbol{y}$ has been moved out from inside Δ.[9] $R(\boldsymbol{w})$ upper bounds Eq. (4), and has been found to work well in practice. $R(\boldsymbol{w})$ is minimized when $\mathscr{E}(\boldsymbol{x}_d, \boldsymbol{y}_d; \boldsymbol{w})$ is at least $\Delta(\boldsymbol{y}_d, \boldsymbol{y})$ lower in energy than the nearest alternative labeling $\boldsymbol{y}$. Figure 4 illustrates the improvement in the objective obtained by using $R(\boldsymbol{w})$. This plot is obtained using training data for a MAX-3-SAT problem described in Section 4.2.2 using $\boldsymbol{w}$ vectors obtained at iterations 10 and 50.

Combining Eqs. (3) and (5) the learning of $\boldsymbol{w}$ is done by minimizing

$$F(\boldsymbol{w}) \equiv \frac{\lambda}{2} \langle \boldsymbol{w}, \boldsymbol{w} \rangle + \sum_{(\boldsymbol{x}_d, \boldsymbol{y}_d) \in \mathscr{D}} \max_{\boldsymbol{y}} \big\{ \Delta(\boldsymbol{y}_d, \boldsymbol{y}) + \langle \boldsymbol{w}, \tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d, \boldsymbol{y}_d) - \tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d, \boldsymbol{y}) \rangle \big\}. \tag{6}$$

By construction $\mathscr{E}(\boldsymbol{x}_d, \boldsymbol{y}; \boldsymbol{w})$ is quadratic in $\boldsymbol{y}$ and $\Delta(\boldsymbol{y}_d, \boldsymbol{y})$ is linear so that the maximization over $\boldsymbol{y}$ inside $F(\boldsymbol{w})$ can be solved by quantum annealing of a QUBO (Ising) model. Additionally, $F(\boldsymbol{w})$ may be shown to be strictly convex so that it has a single minimum as a function of

[9] A better objective function which is invariant to changes in the scale of Δ is described in [TJHA05], but this results in problems which involve triplets of variables rather than pairs as required for the Ising model. While triplet interactions may be simulated in an Ising model we do not pursue that extension here.

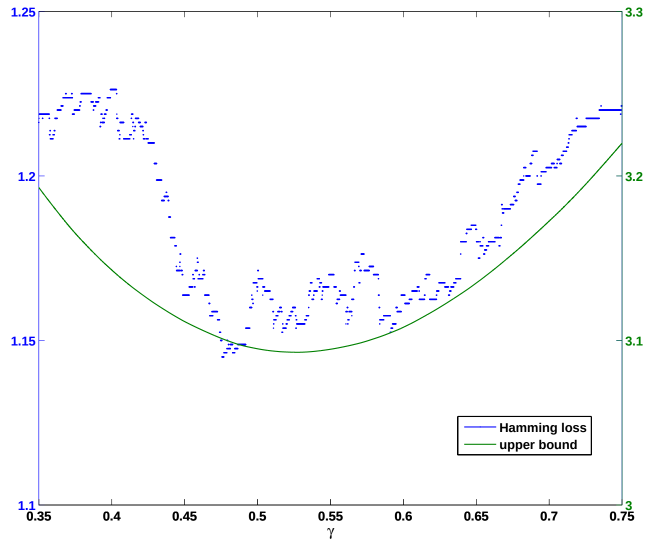


Figure 4: The average Hamming loss (blue) and its upper bound $R(\boldsymbol{w})$ (green) measured across a set of $\boldsymbol{w}$ given by $\gamma\boldsymbol{w}_{10}+(1-\gamma)\boldsymbol{w}_{50}$. Notice the different scales on the separate y axes for the Hamming and upper bound values. The jumps in Hamming loss are discontinuities. Though the upper bound is continuous, there are discontinuities in the derivative which are not observable at this scale.

$\boldsymbol{w}$. However, complicating matters is the fact that $F(\boldsymbol{w})$ is not differentiable everywhere. Nevertheless, optimization methods based on subgradients (generalized gradients that are defined even at points of non-differentiability) can be successfully applied to the minimization of $F(\boldsymbol{w})$.

Subgradient methods rely on subgradients $\partial_{\boldsymbol{w}}F(\boldsymbol{w})$ which lower bound F and satisfy

$$F(\tilde{\boldsymbol{w}}) \geq F(\boldsymbol{w}) + \langle \partial_{\boldsymbol{w}}F(\boldsymbol{w}), \tilde{\boldsymbol{w}} - \boldsymbol{w}\rangle \qquad \text{for all } \tilde{\boldsymbol{w}}.$$

In the present case a subgradient at $\boldsymbol{w}$ is given by

$$\partial_{\boldsymbol{w}}F(\boldsymbol{w}) = \lambda\boldsymbol{w} + \sum_{(\boldsymbol{x}_d,\boldsymbol{y}_d)\in\mathcal{D}} \Big[\tilde{\boldsymbol{\mathcal{E}}}(\boldsymbol{x}_d,\boldsymbol{y}_d) - \tilde{\boldsymbol{\mathcal{E}}}(\boldsymbol{x}_d,\boldsymbol{y}_d^\star(\boldsymbol{w}))\Big]$$

where $\boldsymbol{y}_d^\star(\boldsymbol{w}) = \arg\max_{\boldsymbol{y}}\big\{\Delta(\boldsymbol{y}_d,\boldsymbol{y}) + \langle \boldsymbol{w}, \tilde{\boldsymbol{\mathcal{E}}}(\boldsymbol{x}_d,\boldsymbol{y}_d) - \tilde{\boldsymbol{\mathcal{E}}}(\boldsymbol{x}_d,\boldsymbol{y})\rangle\big\}$. Thus, optimal learning of $\boldsymbol{w}$ can be done by a subgradient method which relies on QUBO solving to generate subgradients. Having determined the optimal $\boldsymbol{w}^\star = \arg\min F(\boldsymbol{w})$, the mapping from $\boldsymbol{x}$ to QUBO parameters $\mathbf{Q}$ is realized with the energy function $\mathcal{E}(\boldsymbol{x},\boldsymbol{y}) = \langle \boldsymbol{w}^\star, \tilde{\boldsymbol{\mathcal{E}}}(\boldsymbol{x},\boldsymbol{y})\rangle$ which gives $\mathbf{Q}(\boldsymbol{x}) = \sum_\alpha w_\alpha^\star \mathbf{Q}_\alpha(\boldsymbol{x})$.

In summary, Algorithm 1 describes the learning procedure used to identify a quadratic function $\mathcal{E}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{w}^\star)$. The $\boldsymbol{y}$ predicted to be associated with a novel input $\boldsymbol{x}$ is $\arg\max_{\boldsymbol{y}}\mathcal{E}(\boldsymbol{x},\boldsymbol{y})$.

4.1.2 Learning at finite temperature

Algorithms for supervised

Algorithm 1 Zero temperature learning using quantum annealing hardware

```
Require: training data $\mathscr{D}$, features $\{\mathbf{Q}_\alpha(\boldsymbol{x})\}$
  Initialize $t \leftarrow 0$, and $\boldsymbol{w}_t \leftarrow \mathbf{0}$
  while $\boldsymbol{w}$ not converged do
      Evaluate $R(\boldsymbol{w}_t)$ by applying quantum annealing to maximize the QUBO $\Delta(\boldsymbol{y}_d,\boldsymbol{y}) + \langle \boldsymbol{w}_t, \tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,\boldsymbol{y}_d) - \tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,\boldsymbol{y})\rangle$ for each training data point $(\boldsymbol{x}_d,\boldsymbol{y}_d) \in \mathscr{D}$; record each maximizer $\boldsymbol{y}_d^\star(\boldsymbol{w}_t)$.
      Find a subgradient at $\boldsymbol{w}_t$: $\partial_{\boldsymbol{w}} F(\boldsymbol{w}_t) \leftarrow \lambda \boldsymbol{w}_t + \sum_{(\boldsymbol{x}_d,\boldsymbol{y}_d)\in\mathscr{D}} \big[\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,\boldsymbol{y}_d) - \tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,\boldsymbol{y}_d^\star(\boldsymbol{w}_t))\big]$.
      Update $\boldsymbol{w}_t$ to $\boldsymbol{w}_{t+1}$ using any convex minimization algorithm relying on the history of function values $\{F(\boldsymbol{w}_0),\cdots,F(\boldsymbol{w}_t)\}$ and subgradients $\{\partial_{\boldsymbol{w}}F(\boldsymbol{w}_0),\cdots,\partial_{\boldsymbol{w}}F(\boldsymbol{w}_t)\}$ (we used the level method described in [LNN95]).
      $t \leftarrow t+1$.
  end while
  return the learned function $\mathscr{E}(\boldsymbol{x},\boldsymbol{y}) = \sum_\alpha w_t(\alpha)\langle \boldsymbol{y}, \mathbf{Q}_\alpha(\boldsymbol{x})\boldsymbol{y}\rangle$.
```

learning at finite temperature are called conditional random fields in the machine learning literature [LMP01]. Here we describe conditional random fields constructed from the Ising model. To learn probabilistic mappings from $\boldsymbol{x}$ to bit-strings $\boldsymbol{y}$ we model the conditional probability as an exponential model:

$$P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w}) = \frac{1}{Z(\boldsymbol{x},\boldsymbol{w})}\exp\big(-\mathscr{E}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{w})\big)$$

where the normalization Z depends on $\boldsymbol{x}$, $\boldsymbol{w}$. Explicitly

$$Z(\boldsymbol{x},\boldsymbol{w}) = \sum_{\boldsymbol{y}} \exp\big(-\mathscr{E}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{w})\big).$$

Evaluating the normalization (partition function) is exponential in the length of the bit-string $\boldsymbol{y}$, and thus we require methods that work even when the partition function is approximated using a polynomial amount of effort. We use the probabilistic nature of quantum annealing on the Ising model to obtain good approximations with a small number of calls to the annealing hardware.

The learning algorithm learns the $\boldsymbol{x}$ dependence of $\mathscr{E}(\boldsymbol{x},\boldsymbol{y};\boldsymbol{w}) = \langle \boldsymbol{y}, \mathbf{Q}(\boldsymbol{x})\boldsymbol{y}\rangle$ by finding the $\mathbf{Q}(\boldsymbol{x}) = \sum_\alpha w_\alpha \mathbf{Q}_\alpha(\boldsymbol{x})$ that gives the highest conditional likelihood of the observed training data $\mathscr{D}$. If the elements of the training set $\mathscr{D}$ are independent then the conditional likelihood of $\mathscr{D}$ is

$$\mathrm{CL}(\boldsymbol{w}) = \prod_{(\boldsymbol{x}_d,\boldsymbol{y}_d)\in\mathscr{D}} P(\boldsymbol{y}_d|\boldsymbol{x}_d,\boldsymbol{w}).$$

Rather than maximizing $\mathrm{CL}(\boldsymbol{w})$ directly, it is convenient (and equivalent) to minimize its negative logarithm, $\mathrm{LCL}(\boldsymbol{w}) \equiv -\ln \mathrm{CL}(\boldsymbol{w})$ to determine $\boldsymbol{w}$:

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \mathrm{LCL}(\boldsymbol{w}) = \arg\min_{\boldsymbol{w}} \sum_{(\boldsymbol{x}_d,\boldsymbol{y}_d)\in\mathscr{D}} \big\{\mathscr{E}(\boldsymbol{x}_d,\boldsymbol{y}_d;\boldsymbol{w}) + \ln Z(\boldsymbol{x}_d,\boldsymbol{w})\big\}. \tag{7}$$

This choice for $\boldsymbol{w}^\star$ makes the observed training data most likely amongst all possible choices of $\boldsymbol{w}$. LCL($\boldsymbol{w}$) is a convex and differentiable function of $\boldsymbol{w}$. The gradient of LCL($\boldsymbol{w}$) is

$$\begin{aligned}\nabla_{\boldsymbol{w}}\mathrm{LCL}(\boldsymbol{w}) &= \sum_{(\boldsymbol{x}_d,\boldsymbol{y}_d)\in\mathscr{D}} \Big\{\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,\boldsymbol{y}_d) - \sum_{\boldsymbol{y}} \tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,\boldsymbol{y})P(\boldsymbol{y}|\boldsymbol{x}_d,\boldsymbol{w})\Big\} \\ &= \sum_{(\boldsymbol{x}_d,\boldsymbol{y}_d)\in\mathscr{D}} \Big\{\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,\boldsymbol{y}_d) - \mathbb{E}_{P(Y|\boldsymbol{x}_d,\boldsymbol{w})}\big(\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,Y)\big)\Big\}\end{aligned}$$

where Y is a random variable that takes values on the set of $\boldsymbol{y}$'s with probability $P(\boldsymbol{y}|\boldsymbol{x}_d,\boldsymbol{w})$ and $\mathbb{E}_{P(Y|\boldsymbol{x}_d,\boldsymbol{w})}$ denotes the expectation with respect to this probability distribution. Evaluating the gradient at any $\boldsymbol{w}$ exactly requires effort which is exponential in the length of $\boldsymbol{y}$. We use samples from the annealing hardware to rapidly obtain a Monte Carlo approximation of the required expectations. Having obtained the noisy gradient estimates we use a simple gradient descent algorithm with updates $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma_t \nabla_{\boldsymbol{w}}\mathrm{LCL}(\boldsymbol{w}_t)$ where γ_t determines the step size to improve $\boldsymbol{w}_t$. Assuming that the Monte Carlo estimate of $\mathbb{E}_{P(Y|\boldsymbol{x}_d,\boldsymbol{w}_t)}\big(\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,Y)\big)$ is not too noisy so that the estimated gradient has significant alignment with the true gradient, then the stochastic approximation convergence theorem [You89] guarantees that choosing $\gamma_t = 1/t$ will result in convergence to the global minimum $\boldsymbol{w}^\star$ of LCL($\boldsymbol{x}$).

Approximating expectations For long bit-strings $\boldsymbol{y}$ we approximate the expectation with samples (indexed by k). If we could directly obtain samples $\boldsymbol{y}^{(k)}$ from $P(\boldsymbol{y}|\boldsymbol{x}_d,\boldsymbol{w}_t)$ then we could approximate the required expectation as

$$\mathbb{E}_{P(Y|\boldsymbol{x}_d,\boldsymbol{w}_t)}\big(\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,Y)\big) \approx \frac{\sum_{\boldsymbol{y}^{(k)}} \tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,\boldsymbol{y}^{(k)})}{\sum_{\boldsymbol{y}^{(k)}} 1}. \tag{8}$$

However, due to the effects of quantum mechanics the annealing hardware does not return samples from the Gibbs distribution $\exp(-\langle \boldsymbol{y},\boldsymbol{Q}\boldsymbol{y}\rangle)/Z(\boldsymbol{x},\boldsymbol{w})$ when fed input parameters $\boldsymbol{Q}$. Instead, there are two types of deviations. The first deviation is due to the effects of fixed hardware temperature. Thus, instead of being at $T=1$ (as required by $P(\boldsymbol{y}|\boldsymbol{x}_d,\boldsymbol{w}_t)$) there is an unknown but fixed temperature T_0. Due to this effect the distribution governing samples from the annealing hardware is approximated as

$$P^0_{\mathrm{eff}}(\boldsymbol{y}|\mathbf{Q}) = \frac{\tilde{P}^0_{\mathrm{eff}}(\boldsymbol{y}|\mathbf{Q})}{Z} = \frac{\exp\big(-\langle \boldsymbol{y},\mathbf{Q}\boldsymbol{y}\rangle/T_0\big)}{Z(\mathbf{Q},T_0)}, \tag{9}$$

where $\tilde{P}^0_{\mathrm{eff}}(\boldsymbol{y}|\mathbf{Q}) = \exp\big(-\langle \boldsymbol{y},\mathbf{Q}\boldsymbol{y}\rangle/T_0\big)$ is the unnormalized version of $P^0_{\mathrm{eff}}(\boldsymbol{y}|\mathbf{Q})$. Consider the Monte Carlo approximation Eq. (8) we require. By using importance sampling we can write this expectation as

$$\mathbb{E}_{P(Y|\boldsymbol{x}_d,\boldsymbol{w}_t)}\big(\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,Y)\big) \approx \frac{\sum_{\boldsymbol{y}^{(k)}} \zeta(\boldsymbol{y}^{(k)})\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d,\boldsymbol{y}^{(k)})}{\sum_{\boldsymbol{y}^{(k)}} \zeta(\boldsymbol{y}^{(k)})}, \tag{10}$$

where $\boldsymbol{y}^{(k)}$ are samples obtained from the hardware having distribution $P^0_{\text{eff}}(\boldsymbol{y}|\mathbf{Q}_t(\boldsymbol{x}_d))$ where $\mathbf{Q}_t(\boldsymbol{x}_d) = \sum_\alpha w_{\alpha,t}\mathbf{Q}_\alpha(\boldsymbol{x}_d)$, and where the importance weights ζ are given by

$$\zeta(\boldsymbol{y}) = \frac{\tilde{P}(\boldsymbol{y}|\boldsymbol{x}_d, \boldsymbol{w}_t)}{\tilde{P}^0_{\text{eff}}(\boldsymbol{y}|\mathbf{Q}_t(\boldsymbol{x}_d))},$$

where $\tilde{P}(\boldsymbol{y}|\boldsymbol{x}_d, \boldsymbol{w}_t) = \exp(-\langle \boldsymbol{y}, \mathbf{Q}_t(\boldsymbol{x}_d)\boldsymbol{y}\rangle)$ is the unnormalized form of the probability we need, and $\tilde{P}^0_{\text{eff}}(\boldsymbol{y}|\mathbf{Q}_t(\boldsymbol{x}_d))$ can be computed directly if T_0 is known. T_0 can be set by maximizing the likelihood of the observed samples $\{\boldsymbol{y}^{(k)}\}$ with respect to T_0. The second type of deviation is due to quantum mechanical details that result in an effective $\mathbf{Q}_{\text{eff}}$ differing slightly from the input $\mathbf{Q}$. We write $\mathbf{Q}_{\text{eff}} = \mathbf{Q} + \boldsymbol{\mathcal{Q}}$. In principle $\boldsymbol{\mathcal{Q}}$ is a deterministic function of $\mathbf{Q}$ dictated by quantum physics, but in practice is unavailable from first principles, and has to be estimated. Thus, samples returned from the annealing hardware tuned to $\mathbf{Q}$ are actually samples from an effective distribution

$$P_{\text{eff}}(\boldsymbol{y}|\mathbf{Q}) = \frac{\tilde{P}_{\text{eff}}(\boldsymbol{y}|\mathbf{Q})}{Z} = \frac{\exp\left(-\langle \boldsymbol{y}, (\mathbf{Q}+\boldsymbol{\mathcal{Q}})\boldsymbol{y}\rangle / T_0\right)}{Z(\mathbf{Q}+\boldsymbol{\mathcal{Q}}, T_0)} \tag{11}$$

where $\tilde{P}_{\text{eff}}$ is the unnormalized version of P_{eff}. Thus, a better approximation to the required expectations is obtained with importance sampling as in Eq. (10) but with importance weights given by

$$\zeta(\boldsymbol{y}) = \frac{\tilde{P}(\boldsymbol{y}|\boldsymbol{x}_d, \boldsymbol{w}_t)}{\tilde{P}_{\text{eff}}(\boldsymbol{y}|\mathbf{Q}_t(\boldsymbol{x}_d))}$$

where again $\tilde{P}(\boldsymbol{y}|\boldsymbol{x}_d, \boldsymbol{w}_t)$ is a known function, and if T_0 and $\boldsymbol{\mathcal{Q}}$ are estimated from the samples $\{\boldsymbol{y}^{(k)}\}$ then $\tilde{P}_{\text{eff}}(\boldsymbol{y})$ is also known and thus the importance weights $\zeta(\boldsymbol{y}^{(k)})$ can be easily evaluated.[10] $\boldsymbol{\mathcal{Q}}$ is typically small so that $P_{\text{eff}}(\boldsymbol{y}|\mathbf{Q}_t(\boldsymbol{x}_d))$ is close to $P(\boldsymbol{y}|\boldsymbol{x}_d, \boldsymbol{w}_t)$ and the Monte Carlo estimate is typically quite accurate even with only a small number of hardware samples $\boldsymbol{y}^{(k)}$.

In the experimental results section we will examine gradient descent algorithms using estimates of the required expectations based upon both P_{eff} and the simpler variant P^0_{eff}.

Regularization Just as regularization is needed in the zero temperature case, we also regularize the finite temperature case. Regularization can be interpreted as a prior $p(\boldsymbol{w})$ over beliefs for $\boldsymbol{w}$. The effects of the prior are accommodated by minimizing the posterior probability $P(\boldsymbol{w}|\mathcal{D}) \propto P(\mathcal{D}|\boldsymbol{w})p(\boldsymbol{w})$ over weights $\boldsymbol{w}$ when conditioned on the training data $\mathcal{D}$. In this case the optimization objective determining $\boldsymbol{w}^\star$ is given by

$$\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} \{\text{LCL}(\boldsymbol{w}) - \ln p(\boldsymbol{w})\}. \tag{12}$$

Common choices for the (unnormalized) prior are the 1- and 2-norms $p(\boldsymbol{w}) \approx \exp(-\lambda\|\boldsymbol{w}\|_1)$ and $p(\boldsymbol{w}) \approx \exp(-\lambda\|\boldsymbol{w}\|_2^2)$.[11] In either case the objective in Eq. (12) remains convex, but the

[10] T_0 and $\boldsymbol{\mathcal{Q}}$ are easily estimated by setting them to maximize the log likelihood of the observed samples $\{\boldsymbol{y}^{(k)}\}$. The resultant optimization is convex and easily solved.

[11] The λ parameter is again determined by cross validation.

1-norm introduces points of non-differentiability; fortunately though, the optimization for $\boldsymbol{w}^\star$ can still be carried out efficiently.

In this case, the gradient descent step is replaced by the composite gradient mapping [Nes07], defined by

$$\boldsymbol{w}_{t+1} = \arg\min_{\boldsymbol{w}} \left\{ \mathrm{LCL}(\boldsymbol{w}_t) + \langle \nabla_{\boldsymbol{w}} \mathrm{LCL}(\boldsymbol{w}_t), \boldsymbol{w} - \boldsymbol{w}_t \rangle + \frac{1}{2\gamma} \|\boldsymbol{w} - \boldsymbol{w}_t\|^2 - \ln p(\boldsymbol{w}) \right\}, \tag{13}$$

where $\gamma > 0$ determines the step size. For the case of 1- and 2-norms solving (13) is straightforward because it decomposes by coordinates.

In summary, Algorithm 2 describes the learning procedure used to identify the conditional distribution $P(\boldsymbol{y}|\boldsymbol{x}) = \exp\big(-\mathscr{E}(\boldsymbol{x},\boldsymbol{y})\big)/Z(\boldsymbol{x},\boldsymbol{w})$. The $\boldsymbol{y}$ predicted to be associated with a novel

Algorithm 2 Finite temperature learning using quantum annealing hardware

```
Require: training data $\mathscr{D}$, features $\{\mathbf{Q}_\alpha(\boldsymbol{x})\}$
  Initialize $t \leftarrow 0$, and $\boldsymbol{w}_t \leftarrow 0$
  while $\boldsymbol{w}$ not converged do
      For each training data point $(\boldsymbol{x}_d, \boldsymbol{y}_d) \in \mathscr{D}$ estimate the expectation $\mathbb{E}_{P(\boldsymbol{Y}|\boldsymbol{x}_d,\boldsymbol{w}_t)}\big(\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d, \boldsymbol{Y})\big)$ using importance sampling and samples obtained from the hardware; depending on the model used for $P(\boldsymbol{Y}|\boldsymbol{x}_d, \boldsymbol{w}_t)$ some parameters may need to be fit from the set of sampled values.
      Determine the gradient at $\boldsymbol{w}_t$: $\nabla_{\boldsymbol{w}} \mathrm{LCL}(\boldsymbol{w}_t) = \sum_{(\boldsymbol{x}_d,\boldsymbol{y}_d)\in\mathscr{D}} \{\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d, \boldsymbol{y}_d) - \mathbb{E}_{P(\boldsymbol{Y}|\boldsymbol{x}_d,\boldsymbol{w}_t)}\big(\tilde{\boldsymbol{\mathscr{E}}}(\boldsymbol{x}_d, \boldsymbol{Y})\big)\}$.
      Update $\boldsymbol{w}$ using $\boldsymbol{w}_{t+1} \leftarrow \arg\min_{\boldsymbol{w}} \{\mathrm{LCL}(\boldsymbol{w}_t) + \langle \nabla_{\boldsymbol{w}} \mathrm{LCL}(\boldsymbol{w}_t), \boldsymbol{w} - \boldsymbol{w}_t\rangle + \|\boldsymbol{w} - \boldsymbol{w}_t\|^2/(2\gamma) - \ln p(\boldsymbol{w})\}$ using an appropriate step size $\gamma$.
      $t \leftarrow t + 1$.
  end while
  return the conditional distribution $P(\boldsymbol{y}|\boldsymbol{x}) = \exp\big(-\sum_\alpha w_t(\alpha)\langle \boldsymbol{y}, \mathbf{Q}_\alpha(\boldsymbol{x})\boldsymbol{y}\rangle\big)/Z(\boldsymbol{x}, \boldsymbol{w})$.
```

input $\boldsymbol{x}$ is distributed according to $P(\boldsymbol{y}|\boldsymbol{x})$. If a single prediction is required we might take the most probable $\boldsymbol{y}$ by maximizing $P(\boldsymbol{y}|\boldsymbol{x})$ (itself a QUBO), or by minimizing some expected loss measure.

4.2 Experimental results

We test the deterministic and probabilistic learning algorithms using quantum annealing to evaluate the required gradients and subgradients. We consider two types of problems: synthetic data based on weighted MAX-3-SAT, and standard test data arising from the labeling of digital images. A summary of the results of our experiments is in Table 2.

Data sets MAX-3-SAT (I) and Scene are learned using the zero-temperature algorithm and data set MAX-3-SAT (II) using the finite-temperature conditional random field (CRF). The MAX-3-SAT (II) data learned as a CRF is solved under two approximations (P^0_{eff} and P_{eff}) for the required expectations with the more accurate P_{eff} approximation providing better results. For comparison, the SVM loss is obtained under a model where the interaction terms $J_{i,j}$ of the

Dataset	dim($\boldsymbol{x}$)	dim($\boldsymbol{y}$)	dim($\boldsymbol{w}$)	Train	Test	SVM loss	QA loss
MAX-3-SAT (I)	20	34	2226	800	800	12.0%	8.6%
Scene	294	6	4425	1211	1196	10.4%	9.4%
MAX-3-SAT (II)	20	8	504	800	800	16.1%	9.8% / 9.1%

Table 2: Summary of problems and experimental results. dim($\boldsymbol{x}$) and dim($\boldsymbol{y}$) are the dimensions of inputs and outputs, dim($\boldsymbol{w}$) is the number of features in the parametric model, Train/Test is the number of examples used for training and testing. The rightmost two columns measure the relative Hamming error between true test set data (that was *not* used during training), and model predictions. The SVM loss is the error under a model where each output component is predicted independently of other components, and QA loss is the error measured by an algorithm which accounts for correlations amongst output components that is trained and tested using quantum annealing in hardware.

Ising model are constrained to be zero so that each component y_i is predicted independently of all other components. This model is called a support vector machine (SVM).

4.2.1 Zero temperature learning

Synthetic data: MAX-3-SAT (I) In this experiment we generate training and test data consisting of 1600 instances of positive 20-dimensional real-valued inputs, and output bit-strings of length 34, i.e. $(\boldsymbol{x},\boldsymbol{y}) \in \mathbb{R}_+^{20} \times \{0,1\}^{34}$. We model the 34 output bits using the 34 qubits of Figure 2(b) with index 73 or larger. We consider MAX-3-SAT instances with clauses $(l_1, l_2, l_3; r)$ where $r > 0$ is the weight of the clause, and l_1, l_2, l_3 are its literals whose corresponding variables, y_1, y_2, y_3, map to three qubits out of the given 34. Furthermore, to strengthen our quadratic features, we also require that exactly two of the edges between the qubits y_1, y_2, y_3 be present in our hardware graph (from Figure 2(b)). Thus, the number of possible clauses with all-positive literals is 243 and by considering all positive/negative variations of the literals in the clause, the total number of clauses is $8 \times 243 = 1944$. To make the data generation more tractable, we select uniformly at random 1500 clauses out of the 1944. We also generate a matrix $\boldsymbol{V} \in \mathbb{R}^{20\times 1500}$ with uniformly independent random entries from [0,1].

Then a data point $(\boldsymbol{x},\boldsymbol{y})$ is generated by choosing $\boldsymbol{x} \in \mathbb{R}^{20}$ with entries uniformly and independently at random from $[0,10]$; the corresponding labels $\boldsymbol{y} \in \{0,1\}^{34}$ are obtained by solving the MAX-3-SAT instance [12] with the 1500 clauses selected as above with clause i having weight equal $\langle \boldsymbol{x}, \boldsymbol{v}_i \rangle$, where $\boldsymbol{v}_i$ is the ith column of V. The 1600 data points $(\boldsymbol{x},\boldsymbol{y})$ generated in this way are broken into two sets of 800; the first set of 800 data points is used for training and the second for testing. This problem offers rich and complex couplings between the components of $\boldsymbol{y}$.

We consider a set of basis 34×34 matrices $\{\boldsymbol{Q}_\alpha\}$ each of which has a 1 as the only non-zero entry. There is one $\boldsymbol{Q}_\alpha$ with a 1 in each of the possible diagonal entries (these matrices will be used to generate all the linear terms), and also one upper triangular matrix for each one of the

[12]We solved the MAX-3-SAT instance using the specialized solver maxsatz of Chu Min Li from http://www.laria.u-picardie.fr/ cli/maxsatz2009.c .

72 edges between the used qubits of Figure 2(b) (these matrices will be used to generate all the quadratic terms). Thus, the set $\{\boldsymbol{Q}_\alpha\}$ consists of $34+72=106$ matrices. The features can now be written as $q_\alpha(\boldsymbol{x})\boldsymbol{Q}_\alpha$, where the dependence on $\boldsymbol{x}$ is linear on the parameters, that is,

$$q_\alpha(\boldsymbol{x}) = \langle \boldsymbol{w}_\alpha, \boldsymbol{x} \rangle + w_\alpha^0 .$$

As $\boldsymbol{x}$ is 20-dimensional there is a total of $106 \times (20+1) = 2226$ parameters w. The optimal parameters $\boldsymbol{w}^\star$ are learned by minimizing Eq. (6) of Section 4.1.1 using a subgradient method with hardware quantum annealing used to evaluate the required subgradients.

To test the predictive quality of the learned classifier we measure the Hamming error between the predicted and true values. We quote results in terms of the relative Hamming error which is the fraction of predicted output bits which are incorrect. For any $\boldsymbol{w}$ we can form the corresponding $\boldsymbol{Q}(\boldsymbol{x}) = \sum_\alpha w_\alpha \boldsymbol{Q}_\alpha(\boldsymbol{x})$ and minimize the resultant QUBO to obtain the prediction $\boldsymbol{y}(\boldsymbol{x})$. It is instructive to look at this error on both the training set and the the test set as the algorithm runs and an initial guess $\boldsymbol{w}_0$ is evolved towards $\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} F(\boldsymbol{w})$ (see Eq. (6)). Typically, we find convergence within 400 updates of $\boldsymbol{w}_t$. At each $\boldsymbol{w}_t$ we can determine the relative Hamming error at points in the training and test sets. Results are presented in Fig. 5. The Hamming error on the training set does not monotonically decrease because we do not minimize this Hamming error directly but rather an upper bound to the Hamming error, and because the QUBO minimizations carried out by quantum annealing do not always return the global minimizer. The red curves show results when an exact software solver is used to provably minimize the QUBO globally. At each $\boldsymbol{w}_t$ we also measure the relative Hamming error on the test set. This error is usually higher than test set error because parameters $\boldsymbol{w}$ are optimized to minimize an upper bound to the Hamming error measured on the training set, and not on the test set.

As a reference, we consider the smallest Hamming loss obtained using the linear classifier that ignores all the quadratic features. The lack of features which couple output bits means that each output bit is predicted separately (and the resultant algorithm is the well known support vector machine (SVM) with linear kernel). In these experiments, the SVM classifier has 12.04% relative Hamming error, compared to the hardware learning/testing which has 8.62% relative Hamming error.

We see for this problem that

- hardware quantum annealing is currently effective enough to be used within an inner loop optimization for convex structured learning algorithms
- the improvement in predictive accuracy by incorporating pairwise interactions between output bits is significant, and reduces errors rates from 12% to less than 9%.

Keeping the quadratic terms in the optimization objective function can lead to significantly better learning when multiple correlated output bits must be predicted. Unfortunately, these better performing algorithms come at a high cost – the underlying computational problems become much more difficult. As quantum annealing hardware improves both in terms of hardware speed and the sizes of problems that can be addressed, we see an opportunity to continue harnessing these better learning algorithms in regimes where conventional approaches begin to fail due to increasing computational difficulty.

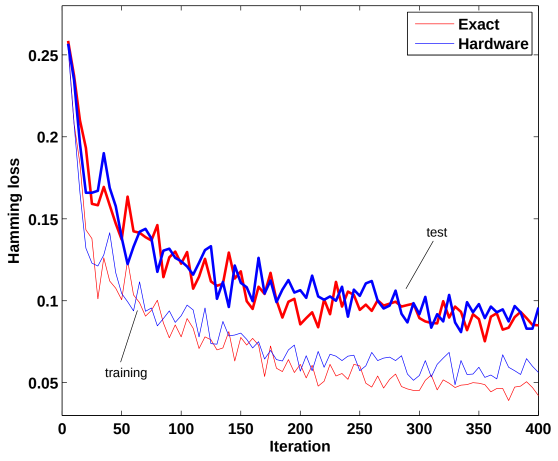


Figure 5: Comparing the performance on training and test data for the exact (red curves) and hardware (blue curves) solvers during the learning algorithm for MAX-3-SAT (I). Training and test set errors were measured on a subset of 100 examples drawn randomly from the 800 test and training examples.

Labeling the scene type of digital images As a second example of zero-temperature learning we address a standard benchmark problem first presented in [BLSB04]. The data arises from labeling digital photographs with one or more of the 6 possible annotations: Beach, Sunset, Fall Foliage, Field, Mountain, and Urban. The annotations to an image are encoded in 6 bits $[y_{\text{Beach}}, \cdots, y_{\text{Urban}}]$ where the bit being on indicates the presence of the label annotation. Images are converted into 294 real inputs by generating features based on color histograms within sub-blocks of the images. Further details are found in [BLSB04]. This dataset has 1211 training images and 1196 testing images.

The mapping from image features to labels is modeled by using the set of 6 qubits $q_{121} = y_{\text{Beach}}$, $q_{122} = y_{\text{Sunset}}$, $q_{123} = y_{\text{Fall}}$, $q_{125} = y_{\text{Field}}$, $q_{126} = y_{\text{Mountain}}$, and $q_{127} = y_{\text{Urban}}$ (see Figure 2(b)) connected as the complete bipartite graph $K_{3,3}$. The dataset consists of datapoints $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{294} \times \{0,1\}^6$. Features for this problem are defined exactly as for the MAX-3-SAT example above; that is, for each linear and edge term the associated weight is learned as a linear function of $\boldsymbol{x}$ plus an offset. Since there are 6 linear terms and 9 edges, the total number of parameters is $(6+9) \times (294+1) = 4425$.

Here, the Hamming error of the linear classifier is 10.4%, while the learning algorithm run using the quantum processor has 9.4% Hamming error. The gains here by including corealtions between components of $\boldsymbol{y}$ are more modest due to the weaker interactions between the scene types.

This result is comparable with the current state-of-the-art found in the literature.

4.2.2 Finite temperature learning

Learning at finite temperature utilizes the entire distribution over bit-strings $\boldsymbol{y}$, and not just the minimal energy bit-string. For any given set of QUBO parameters $\boldsymbol{Q}$ our learning algorithm assumes the distribution of the strings $\boldsymbol{y}$ is Boltzmann with $p(\boldsymbol{y}) \propto \exp\big(-\langle \boldsymbol{y}, \boldsymbol{Q}\boldsymbol{y}\rangle\big)$. However, as described in Section 4.1.2 to obtain such samples using hardware requires an importance sampling correction. To understand how accurate these corrections are we performed two types of experiments.

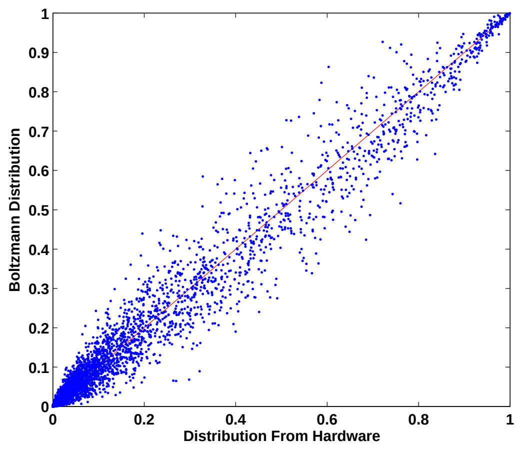
We generated 800 problems with randomly generated $\boldsymbol{Q}$ defined over 8 variables and connected as $K_{4,4}$ (we used qubits q_{121}, q_{122}, q_{123}, q_{124}, q_{125}, q_{126}, q_{127} and q_{128} from Figure 2(b)). Each matrix element $Q_{i,j}$ is independently drawn uniformly at random from the interval $[-1,1]$. For each problem (each $\boldsymbol{Q}_k$, $k = 1,\ldots,800$) 100 samples were drawn from the hardware, 100 times. We considered the two possible models discussed in Section 4.1.2 for the probability distribution determining the samples returned from the hardware — P^0_{eff} which allows for an effective temperature T_0, and P_{eff} which allows for problem-dependent temperatures and shifts $\mathcal{Q}$ on $\boldsymbol{Q}$.

To fit a single problem-independent temperature T_0 for the model P_{eff} we maximized the likelihood of all the observed data (800 problems together) with respect to T_0. This resulted in $T_0 = 21\ mK$ which is in reasonable agreement with the true physical temperature of the hardware. To assess the goodness of this fit we compare the empirical probabilities (defined by frequency counts) with the probabilities defined by the Boltzmann distribution with $T_0 = 21\ mK$. The results are summarized in Figure 6(a). As it can be seen, many data points are close to the diagonal line of the plot, but there is still significant spread indicating deviations from this simple model. This observation prompted the more refined model which allows for both problem-dependent temperatures and shifts to $\boldsymbol{Q}$. For each of the 800 problems $\{\boldsymbol{Q}_k\}$ we fit an effective temperature T_k and shift $\mathcal{Q}_k$. As it can be seen in Figure 6(b), the more complex model P_{eff} is an almost perfect fit to the observed data. Note that the introduction of problem-specific parameters T_k and $\mathcal{Q}_k$ introduces 1+8+16=25 parameters so the quality of the model is bound to improve. However, with these 25 parameters we manage to accurately assign *all* 256 probabilities of each of the 8 qubit configurations.

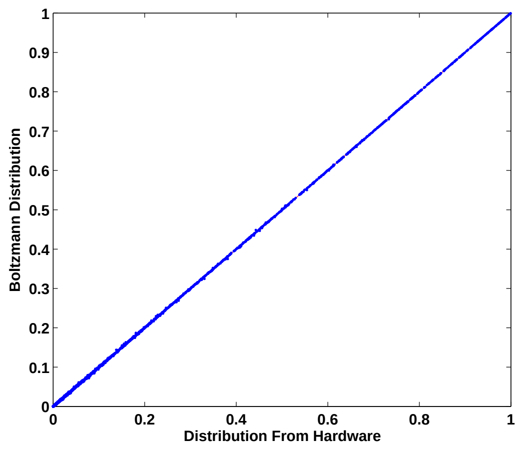
Synthetic data: MAX-3-SAT (II) For this set of experiments we generated MAX-3-SAT data points as before, except for the following differences: we considered problems with 8 bits rather than 34, but allowed all possible clauses of 3 literals to appear (that is, $2^3 \times \binom{8}{3} = 448$). Thus, a data point is of the form $(\boldsymbol{x},\boldsymbol{y}) \in \mathbb{R}^{20} \times \{0,1\}^8$, and the number of training and test points is again 800. We map the 8 bits into a $K_{4,4}$ subgraph in the hardware using qubits $q_{121} - q_{128}$. Thus, the number of features/parameters is $21 \times (8+16) = 504$. We used 1-norm regularization that favors sparse solutions and ameliorates over fitting.

For learning we used the algorithm described in Section 4.1.2 using both P^0_{eff} and P_{eff} to evaluate the importance weights. The P^0_{eff} approximation uses $T_0 = 21\ mK$ determined above, and the P_{eff} model uses a value obtained from fitting to a set of 100 hardware samples. As a baseline the SVM model (which predicts all 8 output bits independently) gives a relative Hamming error of 16.1%. The learning algorithm run using the quantum processor improves upon this to give relative Hamming errors of 9.8% and 9.1% respectively for models using importance weights based on P^0_{eff} and P_{eff} respectively. Both the P^0_{eff} and P_{eff} used the same number of hardware samples.

The improved hardware model based upon P_{eff} pays benefits both in terms of test set prediction accuracy, and also in terms of minimizing the convex objective function $\text{LCL}(\boldsymbol{w})$ (see



(a) P^0_{eff}



(b) P_{eff}

Figure 6: (a) Scatter plot of empirical probabilities and Boltzmann probabilities defined by the best fitting temperature $T_0 = 21\ mK$. Each point corresponds to one of the 256 states of 8 qubits and one of the matrices $\mathbf{Q}_k$; its x-coordinate is the observed frequency in the hardware, while its y-coordinate is the probability using the Boltzmann distribution $P^0_{\text{eff}}(\cdot|\mathbf{Q}_k)$ (see Eq. (9)) with the fitted temperature $T_0 = 21\ mK$. (b) Scatter plot resulting from a fit to the P_{eff} model which additionally allows for a problem dependent offset $\mathcal{Q}$ to $\mathbf{Q}$. In this case the y axis is the probability using the problem dependent Boltzmann distribution $P_{\text{eff}}(\cdot|\mathbf{Q}_k)$ (see Eq. (11)).

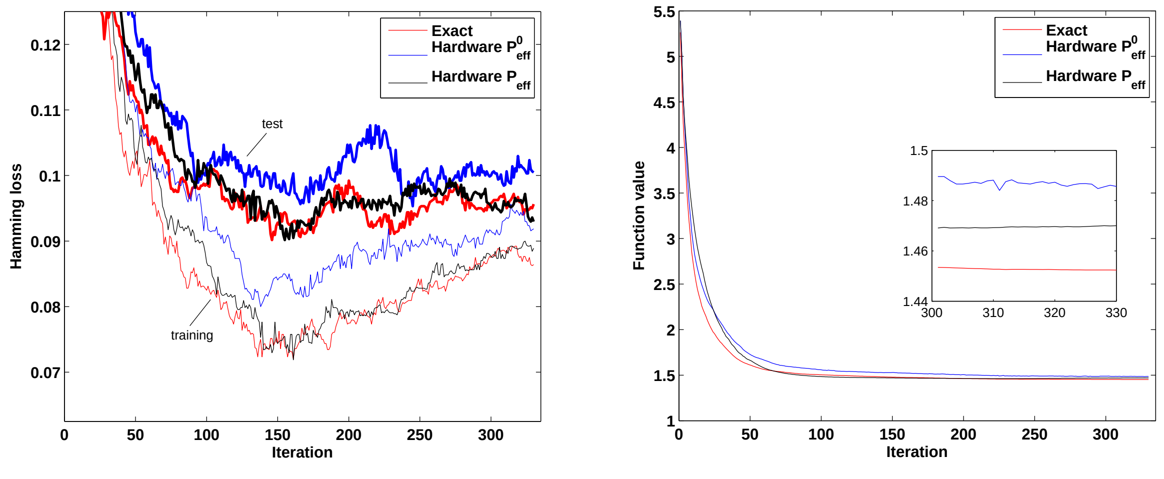


Figure 7: (a) Training and test errors of conditional random fields runs for MAX-3-SAT (II) throughout the learning algorithm. (b) The decrease of the function LCL throughout the run of the learning algorithm. The importance weights associated with P_{eff} better approximate the true objective value.

Eq. (7)). Fig 7(b) shows the objective value obtained under both models and compares this with an exact evaluation in software. Fig 7(a) shows the evolution of relative Hamming error on both test and training sets. Though the model based on P_{eff} is slightly worse in terms of the optimization objective, it performs essentially equivalently in both training and test set error. We further notice that though the objective function has converged there remains variation in the Hamming error on the training set. This is due to the flatness of the objective around its minimum. For this reason the learning algorithm returns the $\boldsymbol{w}$ with the lowest Hamming on the training set, rather than the final $\boldsymbol{w}$ value which may have worse Hamming error.

5 Conclusions

In this paper we provide algorithms and experimental validation for the use of quantum annealing hardware in machine learning applications. The focus was on structured supervised learning where multiple output bits must be predicted as a function of some input. The Ising model is used to enable learning of interactions between output bits for improved prediction over techniques which predict output bits independently. Without constraints on the allowed types of output interactions this problem is NP-hard. We imposed no such constraints and relied upon quantum annealing as a heuristic for the resulting optimization and sampling problems.

We showed how existing algorithms for both zero-temperature and finite-temperature learning can exploit hardware which approximately minimizes and samples Ising energy functions. On both synthetic and real world problems we demonstrated the improvements that are possible by exploiting correlations in output bits. In some cases the gains can be quite significant.

Due to the small scale of the current generation of annealing hardware all of the experiments reported here can also be carried out in software. However, as larger hardware becomes available we anticipate rapidly reaching the point where exact software solution becomes impossible. The hardware utilized here having 128 qubits has treewidth 16, and is almost at the limit of what can be done exactly in software. We have characterized the behavior of the hardware for both optimization and sampling. With current hardware the bulk of time is spent on programming, thermalization, and readout, and not on the actual combinatorial task. Consequently, the potential for improvement is significant.

There are many directions for future work. Most important is the application to larger problems and larger hardware. We are also actively exploring more complex learning models with hidden variables that enable the discovery of novel features. We also comment that the restriction to pairwise interactions is not limiting, but that the best way to overcome this constraint is likely to vary from problem to problem. Thus, models with higher-order features are also under active exploration.

References

[AR04] Andris Ambainis and Oded Regev. An elementary proof of the quantum adiabatic theorem. Technical report, 2004. Available at http://arxiv.org/PS_cache/quant-ph/pdf/0411/0411152v1.pdf.

[Bar82] Francisco Barahona. On the computational complexity of ising spin glass models. *J. Phys. A*, 15:3241–3253, 1982.

[Ben82] Paul Benioff. Quantum mechanical hamiltonian models of turing machines. *J. Stat. Phys.*, 29(3):515–546, 1982.

[BH02] Endre Boros and Peter L. Hammer. Pseudo-boolean optimization. *Discrete Appl. Math.*, 123:155–225, 2002. Available at http://rutcor.rutgers.edu/~boros/Papers/2002-DAM-BH.pdf.

[BJB+10] A. J. Berkley, M. W. Johnson, P. Bunyk, R. Harris, J. Johansson, T. Lanting, E. Ladizinsky, E. Tolkacheva, M. H. S. Amin, and G. Rose. A scalable readout system for a superconducting adiabatic quantum optimization system, March 2010. Available at http://arxiv.org/abs/0905.0891.

[BLSB04] Matthew R. Boutell, Jiebo Luo, Xipeng Shen, and Christopher M. Brown. Learning multi-label scene classification. *Pattern Recognition*, 37(9):1757–1771, 2004. Available at http://www.rose-hulman.edu/~boutell/publications/boutell04PRmultilabel.pdf.

[Bru67] Stephen G. Brush. History of the Lenz-Ising model. *Reviews of Modern Physics*, 39(4):883 – 893, 1967. Available at http://personal.rhul.ac.uk/UHAP/027/PH4211/PH4211_files/brush67.pdf.

[Coo71] Stephen Cook. The complexity of theorem proving procedures. In *Proceedings Third Annual ACM Symposium on Theory of Computing*, pages 151 – 158, May 1971. Available at http://www.cs.toronto.edu/~sacook/homepage/1971.pdf.

[DGP04] Heidi E. Dixon, Matthew L. Ginsberg, and Andrew J. Parkes. Generalizing boolean satisfiability i: Background and survey of existing work. *J. Artif. Intell. Res. (JAIR)*, 21:193–243, 2004.

[Fey82] Richard Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6/7):467–488, 1982.

[FGG02] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. Quantum adiabatic evolution algorithms versus simulated annealing. 2002. Available at http://arxiv.org/abs/quant-ph/0201031.

[FGGS00] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. 2000. Available at http://arxiv.org/abs/quant-ph/0001106.

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *28th Annual ACM Symposium on the Theory of Computing*, page 212, 1996. Available at http://arxiv.org/abs/quant-ph/9605043.

[HJB$^+$10] R. Harris, J. Johansson, A. J. Berkley1, M. W. Johnson, T. Lanting, Siyuan Han, P. Bunyk, E. Ladizinsky, T. Oh, I. Perminov, E. Tolkacheva, S. Uchaikin, E. M. Chapple, C. Enderud, C. Rich, M. Thom, J. Wang, B. Wilson, , and G. Rose. Experimental demonstration of a robust and scalable flux qubit. *Phys. Rev. B*, 81:134510, 2010. Available at http://arxiv.org/abs/0909.4321.

[JBM$^+$10] M W Johnson, P Bunyk, F Maibaum, E Tolkacheva, A J Berkley, E M Chapple, R Harris, J Johansson, T Lanting, I Perminov, E Ladizinsky, T Oh, and G Rose. A scalable control system for a superconducting adiabatic quantum optimization processor. *Superconductor Science and Technology*, 23(6), 2010. Available at http://arxiv.org/abs/0907.3757.

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*. Plenum, 1972. Available at http://www.cs.berkeley.edu/~luca/cs172/karp.pdf.

[KJV83] Scott Kirkpatrick, C. D. Gelatt Jr., and Mario P. Vecchi. Optimization by simulated annealing. *Science*, 220:671Ű680, 1983.

[Lev73] Leonid Levin. Universal search problems. *Problems of Information Transmission*, 9(3):265 – 266, 1973.

[LMP01] John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *18th*

International Conf. on Machine Learning, pages 282 Ű– 289. Morgan Kaufmann, 2001. Available at http://www.cis.upenn.edu/~pereira/papers/crf.pdf.

[LNN95] Claude Lemaréchal, Arkadii Nemirovskii, and Yurii Nesterov. New variants of bundle methods. *Math. Program.*, 69:111–147, 1995.

[Min91] Marvin L. Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine*, 12(2):34 – 51, 1991. Available at http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/894.

[Nes07] Yurii Nesterov. Gradient methods for minimizing composite objective function. Technical report, Center for Operations Research and Econometrics (CORE), 2007. Available at http://www.ecore.be/DPs/dp_1191313936.pdf.

[RC02] Jérémie Roland and Nicolas J. Cerf. Quantum search by local adiabatic evolution. *Phys. Rev. A*, 65(4):042308, Mar 2002.

[RD06] Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006. Available at http://www.cs.washington.edu/homes/pedrod/papers/mlj05.pdf.

[Rei04] Ben Reichardt. The quantum adiabatic optimization algorithm and local minima. In *Annual ACM Symposium on Theory of Computing*, 2004.

[Sho94] Peter W. Shor. Algorithms for quantum computation: discrete log and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. Institute of Electrical and Electronic Engineers Computer Society Press, 1994. Available at http://arxiv.org/abs/quant-ph/9508027v2.

[TGK03] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max margin markov networks. In *Neural Information Processing Systems – 14*, 2003. Available at http://www.seas.upenn.edu/~taskar/pubs/mmmn.pdf.

[TJHA05] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005. Available at http://jmlr.csail.mit.edu/papers/volume6/tsochantaridis05a/tsochantaridis05a.pdf.

[You89] Laurent Younes. Parametric inference for imperfectly observed gibbsian fields. *Probability Theory and Related Fields,*, 82:625 Ű– 645, 1989. Available at http://cis.jhu.edu/~younes/Papers/1989/PTRF89.pdf.