NIPS 2009 Demonstration: Binary Classification using Hardware Implementation of Quantum Annealing

Hartmut Neven
Google
neven@google.com

Vasil S. Denchev
Purdue University
vdenchev@purdue.edu

Marshall Drew-Brook
D-Wave Systems Inc.
marshall@dwavesys.com

Jiayong Zhang
Google
jiayong@google.com

William G. Macready
D-Wave Systems Inc.
wgm@dwavesys.com

Geordie Rose
D-Wave Systems Inc.
rose@dwavesys.com

December 7, 2009

Abstract

Previous work [NDRM08, NDRM09] has sought the development of binary classifiers that exploit the ability to better solve certain discrete optimization problems with quantum annealing. The resultant training algorithm was shown to offer benefits over competing binary classifiers even when the discrete optimization problems were solved with software heuristics. In this progress update we provide first results on training using a physical implementation of quantum annealing for black-box optimization of Ising objectives. We successfully build a classifier for the detection of cars in digital images using quantum annealing in hardware. We describe the learning algorithm and motivate the particular regularization we employ. We provide results on the efficacy of hardware-realized quantum annealing, and compare the final classifier to software trained variants, and a highly tuned version of AdaBoost.

1 Introduction

In a recent series of publications [NDRM08, NDRM09] we have developed a new approach to train binary classifiers. The new training procedure is motivated by the desire to capitalize on improvements in heuristic solvers (both software and hardware solvers) for a class of discrete optimization problems. The discrete optimization problem is used to identify a subset of weak classifiers from within a dictionary that vote on the class label. Explicit regularization favors small voting subsets. In [NDRM08] the general approach was outlined, and in [NDRM09] the algorithm was adapted to allow for scaling to arbitrarily large dictionaries while solving discrete optimization problems of bounded size. Theoretical advantages of the proposed approach have been considered, and when tested using heuristic software solvers the resulting procedure was shown to offer benefits in both sparsity and test set accuracy over competing alternatives like AdaBoost.

In this progress update we report first results on the use of an optimization heuristic embodied in hardware realizing a implementation of quantum annealing (a simulated-annealing-like heuristic that uses quantum instead of thermal fluctuations). The learning objective, which identifies the sparse voting set of weak classifiers, is formulated as a Quadratic Unconstrained Binary Optimization (QUBO), and mapped to an Ising model which models the low-energy physics of hardware in development by D-Wave. Time evolution according to the Ising model with quantum interactions present in the hardware allows the hardware to realize a physical implementation of quantum annealing (QA). The optimizing ability of this hardware heuristic is harnessed to allow training of a large scale classifier for the detection of cars in digital images.

A more detailed study will be published elsewhere, but this report provides details on the algorithm, the experiment on hardware, and compares the results to classical (non-quantum) alternatives. We begin by introducing QUBOs and their relation to the Ising model that is realized in hardware. We show how QUBOs can be optimized by mapping to Ising form, and evolving the the dynamics to realize a physical implementation of quantum annealing. We note some of the constraints that the hardware implementation imposes as these motivate the design choices in the construction of a classification algorithm. With this background to quantum annealing on hardware, we apply QUBO-solving to binary classification. We describe how a strong classifier is constructed from a dictionary of weak confidence-weighted classifiers showing how the primary limitations imposed by the hardware, namely small sparsely-connected QUBOs, can be overcome. We validate the resultant algorithm with an experiment where the classifier is trained on hardware consisting of 52 sparsely connected qubits. We compare the final classifier to one trained with AdaBoost and other software solvers. Preliminary results are encouraging – we obtain a classifier that employs fewer weak classifiers that AdaBoost and whose training set error is only marginally worse. As this work is but the first step at implementing the algorithm in hardware, we conclude with a discussion of next steps.

2 Background

2.1 QUBOs and the Ising model

The QUBO problem seeks the minimizer of a quadratic function of N Boolean variables:[1]

$$\text{QUBO}(\boldsymbol{Q}): \qquad \boldsymbol{w}^\star = \operatorname*{argmin}_{\boldsymbol{w}} \; \boldsymbol{w}'\boldsymbol{Q}\boldsymbol{w} \quad \text{where } w_i \in \{0,1\}$$

Terms linear in $\boldsymbol{w}$ are captured on the diagonal of the $N \times N$ matrix $\boldsymbol{Q}$. QUBO is NP hard, and is related to the Ising problem

$$\text{ISING}(\boldsymbol{h},\boldsymbol{J}): \qquad \boldsymbol{s}^\star = \operatorname*{argmin}_{\boldsymbol{s}} \left\{\boldsymbol{s}'\boldsymbol{J}\boldsymbol{s} + \boldsymbol{h}'\boldsymbol{s}\right\} \quad \text{where } s_i \in \{-1,+1\}$$

through the change of variables $\boldsymbol{s} = 2\boldsymbol{w} - \mathbf{1}$. The $\boldsymbol{s}$ variables are called spins. D-Wave hardware approximates $\boldsymbol{s}^\star$ through a hardware realization of quantum annealing. The hardware is based on a system of interacting qubits realized as loops of superconductor called Josephson junctions. The physical description of the qubits is well captured at low energies by the Ising model augmented with additional terms arising from quantum interactions. These quantum interactions are harnessed

[1] Vectors are indicated in lowercase bold, and matrices in uppercase bold.

in the hardware to realize a quantum annealing algorithm for addressing ISING. Unlike previous studies of quantum annealing [KTM09, SKT$^+$09] in machine learning, the process is not simulated in software but embodied in physical quantum dynamics.

2.2 Quantum annealing

In 2000 [FGGS00] proposed a model of quantum computation based on adiabatic quantum evolution. The basic idea relies on the transformation of an initial problem (that is easy to solve) into the problem of interest (which may be difficult to solve). If the transformation is sufficiently slow then we are guaranteed to find the solution of the problem of interest with high probability. In the present context we are interested in solving an optimization problem (specifically ISING), and the method in this case is known as quantum annealing. As the D-Wave hardware relies on quantum annealing we outline the basics of the quantum annealing when applied to ISING.

Quantum mechanics requires the representation of spin states $s = \pm 1$ as orthogonal vectors in a two-dimensional vector space. We indicate the two possibilities by $|+\rangle = \begin{bmatrix}1 & 0\end{bmatrix}'$ and $|-\rangle = \begin{bmatrix}0 & 1\end{bmatrix}'$. The extension to vectors is necessary as linear combinations of these state vectors exist as real descriptions of nature. An example combination state, called a superposition, is $\alpha_-|-\rangle + \alpha_+|+\rangle = \begin{bmatrix}\alpha_+ & \alpha_-\end{bmatrix}'$. The state can always be normalized so that $|\alpha_-|^2 + |\alpha_+|^2 = 1$.[2] This quantum description of a bit is called a qubit. The amplitudes α_- and α_+ of the superposition state are given meaning through measurement of the qubit. When measured, the qubit is seen as -1 with probability α_-^2, and seen as $+1$ with probability α_+^2. Larger quantum systems are constructed through tensor product of the individual qubit vector spaces, so that for example $|-+\rangle \equiv |-\rangle \otimes |+\rangle = \begin{bmatrix}0 & 0 & 1 & 0\end{bmatrix}'$ represents the 2 spin state $s_1 = -1, s_2 = +1$. Superpositions of N qubit states are also possible with the associated amplitudes representing the probability of observing the respective N spin state, e.g. $\alpha_{-,+}^2$ is the probability of measuring $s_1 = -1$ and $s_2 = +1$.

To make contact with ISING define the single qubit operators $I = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$ and $\sigma^z = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$. With this definition $\sigma^z|-\rangle = -|-\rangle$ and $\sigma^z|+\rangle = +|+\rangle$ so that the operator σ^z leaves the qubit state unchanged (σ^z is diagonal), and simply multiplies the state by it's classical spin value, either -1 or $+1$. Similarly, on a 2 qubit state the operator $\sigma^z \otimes I$ extracts the classical spin of the first qubit, $I \otimes \sigma^z$ extracts the classical spin of the second qubit, and $\sigma^z \otimes \sigma^z$ extracts the product $s_1 s_2$ of the two classical spin variables.

With this notation the ISING objective on N spins is represented as a $2^N \times 2^N$ diagonal operator (called the Hamiltonian)

$$\mathcal{H}_I = \sum_{i,j} J_{i,j}\sigma_i^z\sigma_j^z + \sum_i h_i\sigma_i^z$$

where σ_i^z is shorthand for the operator σ^z acting on qubit i alone[3], and $\sigma_i^z\sigma_j^z$ is shorthand for the operator giving the product of spins on qubits i and j. $\mathcal{H}_I$ acting on the classical state $|\boldsymbol{s}\rangle = |s_1 \cdots s_N\rangle$ gives $\mathcal{H}_I|\boldsymbol{s}\rangle = E(\boldsymbol{s})|\boldsymbol{s}\rangle$ where $E(\boldsymbol{s}) = \boldsymbol{s}'\boldsymbol{J}\boldsymbol{s} + \boldsymbol{h}'\boldsymbol{s}$. The state with the lowest objective value corresponds to the eigenvector of $\mathcal{H}_I$ having the lowest eigenvalue. The quantum effects in the D-Wave hardware manifest themselves with the addition of another single qubit operator. This

[2] Most generally α_- and α_+ are complex valued. However, in what follows α_- and α_+ may be taken to be real.

[3] Explicitly $\sigma_i^z = \underbrace{I \otimes \cdots \otimes I}_{i-1 \text{ times}} \otimes \sigma^z \otimes \underbrace{I \otimes \cdots \otimes I}_{N-i \text{ times}}$.

operator is given by $\sigma^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, and acts to flip the qubit state, i.e. $\sigma^x(\alpha_+|+\rangle + \alpha_-|-\rangle) = \alpha_+|-\rangle + \alpha_-|+\rangle$. Note that unlike the σ^z term, this operator is off-diagonal and alters the quantum state when applied. The eigenstates of this operator are $(|+\rangle + |-\rangle)/\sqrt{2}$ (eigenvalue +1) and $(|+\rangle - |-\rangle)/\sqrt{2}$ (eigenvalue -1) with both giving equal probability to a measurement yielding either $s = +1$ or $s = -1$. The many qubit version of this operator acts upon each qubit individually as $\mathcal{H}_0 = \Delta \sum_i \sigma_i^x$. The lowest eigenstate of $\mathcal{H}_0$ is $\bigotimes_i (|+_i\rangle - |-_i\rangle)/\sqrt{2}$ which gives all 2^N states the same amplitude. This is a uniform superposition making all joint spin states equally likely to be observed.

The quantum description of the D-Wave hardware is well approximated at low energies by the Hamiltonian operator $\mathcal{H} = \mathcal{H}_0 + \mathcal{H}_I$ where all parameters Δ, $\boldsymbol{h}$, and $\boldsymbol{J}$ can be user defined and adjusted dynamically. Given sufficient time, dissipative quantum evolution at zero temperature evolves an initial quantum state into a state in the subspace corresponding to the lowest energy eigenvalue of $\mathcal{H}$. However, just as classical systems may become trapped in metastable higher energy states, the quantum dynamics may spend long periods of time at higher energy states. Quantum annealing is a technique to ameliorate this problem. Consider a convex combination of the Ising and $\mathcal{H}_0$ portions of the Hamiltonian:

$$\mathcal{H}(\tau) = (1 - \tau)\mathcal{H}_0 + \tau\mathcal{H}_I \qquad \text{for } 0 \leq \tau \leq 1.$$

When $\tau = 0$ the lowest energy state is easily discovered and gives all classical configurations equal probability. $\tau = 1$ corresponds to the ISING problem that we want to solve. It is natural then to evolve from the initial $\tau = 0$ lowest energy state to the $\tau = 1$ minimal Ising state. Just as simulated annealing gradually eliminates the entropic exploration effects at higher temperatures to focus on low energy configurations at low temperatures, quantum annealing gradually reduces the exploration effects of quantum fluctuations at large Δ to focus on the classical Ising problem at $\Delta = 0$. The adiabatic theorem indicates how rapidly the quantum term can be annealed to maintain high probability of locating the globally lowest classical configuration. The difference between thermal annealing (simulated annealing) and quantum annealing can be significant. There are problems where quantum annealing conveys no speedups [Rei04, FGGN05], and others where quantum annealing is exponentially faster than simulated annealing [FGG02]. A more complete theoretical characterization of QA awaits.

Previous studies of quantum annealing in machine learning [KTM09, SKT$^+$09] have necessarily relied on classical simulations of QA. While such simulations are more complex than simulated annealing better minima are often obtained. In the experiments reported here no simulation is necessary, quantum annealing is performed by dynamically adjusting τ in the D-Wave hardware. We rely on the fast physical timescales within the D-Wave chip and the parallel dynamics of all qubits to realize QA natively.

2.3 Hardware constraints

The idealized sketch of quantum annealing presented above omits complications arising from hardware implementation. We mention the most important of these complications that affect both the types of Ising problems that can be solved, and the efficacy of hardware optimization on early generation hardware.

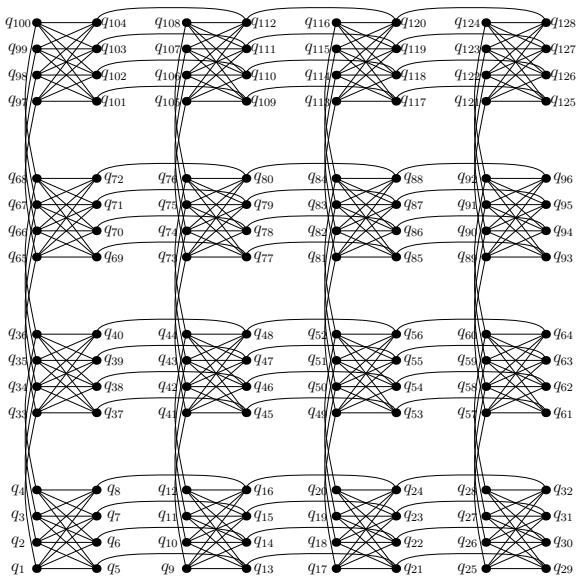


Figure 1: Connectivity between the 128 qubits in the current D-Wave chip. The pattern of connectivity used is driven by hardware design considerations.

Qubit connectivity To represent any Ising problem on N variables requires complete connectivity between qubits. Physically realizing all $\binom{N}{2}$ connections is difficult, and the D-Wave hardware is architected with a sparser pattern of connectivity. The current chip on 128 qubits utilizes the connectivity given in Fig. 1 The classification algorithm we develop accounts for this particular hardware connectivity.

Non-zero temperature While the quantum hardware operates at a rather frigid 20-40mK, this temperature is significant when compared to the relevant energy scales of the chip. In units where the largest h_i or $J_{i,j}$ magnitude is 1, the temperature is $T \approx 1/4 - 1/3$. Temperature effects are well approximated by a Boltzmann distribution so that states with energy E are approximately seen with probability $\exp(-E/T)$. Thus, thermally excited states will be seen in many sample runs on the hardware.

Parameter variability Early generations of the hardware are unable to provide highly precise parameter values. Instead, when a particular h_i, $J_{i,j}$ is requested, what is realized in hardware is a sample from a Gaussian centered on the desired value. The standard deviation is large enough that the mis-assigned value can generate candidate solutions of higher energy (according to the correct parameter values).

To mitigate both the temperature and precision difficulties we solve the same ISING$(\boldsymbol{h}, \boldsymbol{J})$ problem multiple times, and select the lowest energy result by evaluating each returned configuration using the desired problem parameters. Currently, obtaining m candidate minima on the hardware takes time $t = 900 + 100m$ ms. In section 4 we provide a characterization of the optimization efficacy of the this hardware optimization procedure.

3 QBoost

3.1 Baseline System

We want to build a classifier which exploits the ability to effectively solve QUBO. The basic idea is straightforward: given a dictionary of weak classifiers, boost these into a strong classifier by selecting a small subset which vote on the class label. The solution of a sequence of QUBOs identifies the best performing subset of weak classifiers. The resultant algorithm is called QBoost to bring to mind the quantum-inspired QUBO-based boosting procedure.

In [NDRM08] we studied a binary classifier of the form

$$y = H(\boldsymbol{x}) = \operatorname{sign}\left(\sum_{i=1}^{N} w_i h_i(\boldsymbol{x})\right) \equiv \operatorname{sign} \boldsymbol{w}'\boldsymbol{h}(\boldsymbol{x}) \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^M$ are the input patterns to be classified, $y \in \{-1, +1\}$ is the output of the classifier, the $h_i : \mathbb{R}^M \mapsto \{-1, +1\}$ are so-called weak classifiers or features detectors, and the $w_i \in \{0, 1\}$ are a set of weights to be optimized during training. $H(\boldsymbol{x})$ is known as a strong classifier.

Training, the process of choosing $\boldsymbol{w}$, proceeds by minimizing two terms. The loss $L(\boldsymbol{w})$ term measures the error over a set of S training examples $\{(\boldsymbol{x}_s, y_s)|s = 1, \ldots, S\}$. To give a QUBO objective we use quadratic loss. The regularization $R(\boldsymbol{w})$ term ensures that the classifier does not become too complex, and mitigates over-fitting. We employ a regularization term based on the L0-norm, $R(\boldsymbol{w}) = \|\boldsymbol{w}\|_0$ which for Boolean-valued variables is simply $R(\boldsymbol{w}) = \sum_i w_i$. This term encourages the strong classifier to be built with as few weak classifiers as possible while maintaining a low training error. Thus, training is accomplished by solving the following discrete optimization problem:

$$\begin{aligned}\text{Baseline:} \qquad \boldsymbol{w}^\star &= \operatorname*{argmin}_{\boldsymbol{w}} \Bigg(\underbrace{\sum_{s=1}^{S} \Big(\frac{1}{N} \boldsymbol{w}'\boldsymbol{h}(\boldsymbol{x}_s) - y_s \Big)^2}_{L(\boldsymbol{w})} + \lambda \underbrace{\|w\|_0}_{R(\boldsymbol{w})} \Bigg) \\ &= \operatorname*{argmin}_{\boldsymbol{w}} \Bigg(\boldsymbol{w}' \underbrace{\left(\frac{1}{N^2} \sum_{s=1}^{S} \boldsymbol{h}(\boldsymbol{x}_s)\boldsymbol{h}'(\boldsymbol{x}_s) \right)}_{\mathbb{E}_{P_0(\boldsymbol{x},y)}\left(\boldsymbol{h}(\boldsymbol{x})\boldsymbol{h}'(\boldsymbol{x})\right)/N} \boldsymbol{w} + \boldsymbol{w}' \Bigg(\lambda \mathbf{1} - 2 \underbrace{\sum_{s=1}^{S} \frac{\boldsymbol{h}(\boldsymbol{x}_s) y_s}{N}}_{\mathbb{E}_{P_0(\boldsymbol{x},y)}\left(\boldsymbol{h}(\boldsymbol{x}) y\right)} \Bigg) \Bigg)\end{aligned} \tag{2}$$

The parameters of this QUBO depend on correlations between the components of $\boldsymbol{h}$ and with the output signal y. These correlations are measured with respect to P_0, the empirical distribution over $(\boldsymbol{x}, y)$. Note that the weights $\boldsymbol{w}$ are binary and not positive real numbers as in AdaBoost. Even though discrete optimization could be applied to any bit depth representing the weights, we found that a small bit depth is often sufficient [NDRM08]. Here we deal with the simplest case in which the weights are chosen to be binary.

3.2 Comparison of the baseline algorithm to AdaBoost

In the case of a finite dictionary of weak classifiers $\boldsymbol{h}(\boldsymbol{x})$ AdaBoost can be seen as a greedy algorithm that minimizes the exponential loss [Zha04],

$$\boldsymbol{\alpha}^{opt} = \underset{\boldsymbol{\alpha}}{\operatorname{argmin}} \left(\sum_{s=1}^{S} \exp\left(-y_s \boldsymbol{\alpha}' \boldsymbol{h}(\boldsymbol{x}_s)\right) \right), \tag{3}$$

with $\alpha_i \in \mathbb{R}^+$. There are two differences between the objective of our algorithm (Eq. (2)) and the one employed by AdaBoost. The first is that we added L0-norm regularization. Second, we employ a quadratic loss function, while AdaBoost works with the exponential loss.

It can easily be shown that including L0-norm regularization in the objective in Eq. (2) leads to improved generalization error as compared to using the quadratic loss only. The proof is as follows. An upper bound for the Vapnik-Chernovenkis dimension of a strong classifier H of the form $H(\boldsymbol{x}) = \sum_{t=1}^{T} h_t(\boldsymbol{x})$ is given by

$$\mathrm{VC}_H = 2(\mathrm{VC}_{\{h_t\}} + 1)(T+1)\log_2(e(T+1)), \tag{4}$$

where $\mathrm{VC}_{\{h_t\}}$ is the VC dimension of the dictionary of weak classifiers [FS95]. The strong classifier's generalization error $\mathrm{Error}_{\text{test}}$ has therefore an upper bound given by [VC71]

$$\mathrm{Error}_{\text{test}} \leq \mathrm{Error}_{\text{train}} + \sqrt{\frac{\mathrm{VC}_H \ln(1 + 2S/\mathrm{VC}_H) + \ln(9/\delta)}{S}}. \tag{5}$$

It is apparent that a more compact strong classifier that achieves a given training error $\mathrm{Error}_{\text{train}}$ with a smaller number T of weak classifiers (hence, with a smaller VC dimension VC_H), comes with a guarantee for a lower generalization error. Looking at the optimization problem in Eq. (2), one can see that if the regularization strength λ is chosen weak enough, i.e. $\lambda < 2/N + 1/N^2$, then the effect of the regularization is merely to thin out the strong classifier. One arrives at the condition for λ by demanding that the reduction of the regularization term $\Delta R(w)$ that can be obtained by switching one w_i to zero is smaller than the smallest associated increase in the loss term $\Delta L(w)$ that comes from incorrectly labeling a training example. This condition guarantees that weak classifiers are not eliminated at the expense of a higher training error. Therefore the regularization will only keep a minimal set of components, those which are needed to achieve the minimal training error that was obtained when using the loss term only. In this regime, the VC bound of the resulting strong classifier is lower or equal to the VC bound of a classifier trained with no regularization.

AdaBoost contains no explicit regularization term and it can easily happen that the classifier uses a richer set of weak classifiers than needed to achieve the minimal training error, which in turn leads to degraded generalization. Fig. 2 illustrates this fact. The regions outside the positive cluster bounding box of the final AdaBoost-trained classifier occur in areas in which the training set does not contain negative examples. This problem becomes more severe for higher dimensional data. The optimal configuration is not found despite the fact that the weak classifiers necessary to construct the ideal bounding box are generated. In fact AdaBoost fails to learn higher dimensional versions of this problem altogether with error rates approaching 50%. See section 6 for a discussion on how global optimization based learning can handle this data set.

In practice we do not operate in the weak λ regime, but rather determine the regularization strength λ by using a validation set. We measure the performance of the classifier for different

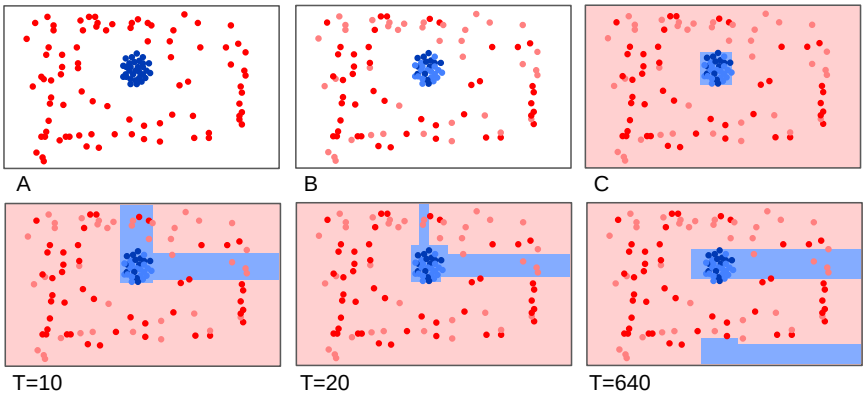


Figure 2: AdaBoost applied to a simple classification task. (A) shows the data, a separable set consisting of a two-dimensional cluster of positive examples (blue) surrounded by negative ones (red). (B) shows the random division into training (saturated colors) and test data (light colors). The dictionary of weak classifiers is constructed of axis-parallel one-dimensional hyperplanes. (C) shows the optimal classifier for this situation, which employs four weak classifiers to partition the input space into positive and a negative areas. The lower row shows partitions generated by AdaBoost after 10, 20, and 640 iterations. The $T = 640$ configuration does not change with further iterations.

values of λ on a validation set and then choose the one with the minimal validation error. In this regime, the optimization performs a trade-off and accepts a higher empirical loss if the classifier can be kept more compact. In other words it may choose to misclassify training examples if it can keep the classifier simpler. This leads to increased robustness in the case of noisy data, and indeed we observe the most significant gains over AdaBoost for noisy data sets when the Bayes error is high. The fact that boosting in its standard formulation with convex loss functions and no regularization is not robust against label noise has drawn attention recently [LS08, Fre09].

The second difference between AdaBoost and our baseline system, namely that we employ quadratic loss while AdaBoost works with exponential loss is of smaller importance. In fact, the discussion above about the role of the regularization term would not have changed if we were to choose exponential rather than square loss. Literature seems to agree that the use of exponential loss in AdaBoost is not essential and that other loss functions could be employed to yield classifiers with similar performance [FHT98, Wyn02]. From a statistical perspective, quadratic loss is satisfying since a classifier that minimizes the quadratic loss is consistent. With an increasing number of training samples it will asymptotically approach a Bayes classifier i.e. the classifier with the smallest possible error [Zha04].

3.3 Large scale classifiers

The baseline system assumes a fixed dictionary containing a number of weak classifiers small enough, so that all weight variables can be considered in a single global optimization. This approach needs to be modified if the goal is to train a large scale classifier. Large scale here means that at least one of two conditions is fulfilled:

1. The dictionary contains more weak classifiers than can be considered in a single global opti-

mization.

2. The final strong classifier consists of a number of weak classifiers that exceeds the number of variables that can be handled in a single global optimization.

Let us take a look at typical problem sizes. The state-of-art solver ILOG CPLEX can obtain good solutions for up to several hundred variable QUBOs depending on the coefficient matrix. The quantum hardware solvers manufactured by D-Wave currently can handle 128 variable problems. In order to train a strong classifier we often sift through millions of features. Moreover, dictionaries of weak learners are often dependent on a set of continuous parameters such as thresholds, which means that their cardinality is infinite. We estimate that typical classifiers employed in vision based products today use thousands of weak learners. Therefore it is not possible to determine all weights in a single global optimization, but rather it is necessary to break the problem into smaller chunks.

Let T designate the size of the final strong classifier and Q the number of variables that we can handle in a single optimization. Q may be determined by the number of available qubits, or if we employ classical heuristic solvers such as ILOG CPLEX or tabu search [Pal04], then Q designates a problem size for which we can hope to obtain a solution of reasonable quality in reasonable time. We consider two cases. The first with $T \leq Q$ and the second with $T > Q$.

We first describe the "inner loop" algorithm used when the number of variables we can handle exceeds the number of weak learners needed to construct the strong classifier. This algorithm may be

Algorithm 1 $T \leq Q$ (inner loop)

Require: Training and validation data, dictionary of weak classifiers
Ensure: Strong classifier

Initialize weight distribution d_{inner} over training samples as uniform distribution $\forall s : d_{inner}(s) = 1/S$
Set $T_{inner} \leftarrow 0$ and $K \leftarrow \emptyset$
repeat
    Select the set $\mathcal{K}$ of $Q - T_{inner}$ weak classifiers h_i from the dictionary that have the smallest weighted training error rates
    for $\lambda = \lambda_{min}$ to λ_{max} **do**
        Run the optimization $\boldsymbol{w}^\star(\lambda) = \operatorname{argmin}_{\boldsymbol{w}} \big(\sum_{s=1}^{S} \{ \sum_{h_t \in K \cup \mathcal{K}} w_t h_t(x_s)/Q - y_s \}^2 + \lambda \|\boldsymbol{w}\|_0 \big)$
        Set $T_{inner}(\lambda) \leftarrow \|\boldsymbol{w}^\star(\lambda)\|_0$
        Construct $H(\boldsymbol{x}; \lambda) \leftarrow \operatorname{sign}\big(\sum_{t=1}^{T_{inner}} w_t^\star(\lambda) h_t(\boldsymbol{x}) \big)$
        Measure validation error $\text{Error}_{\text{val}}(\lambda)$ of $H(\boldsymbol{x}; \lambda)$ on unweighted validation set
    end for
    $\lambda^\star \leftarrow \operatorname{argmin}_\lambda \text{Error}_{\text{val}}(\lambda)$
    Update $T_{inner} \leftarrow T_{inner}(\lambda^\star)$ and $K \leftarrow \{h_t | w_t^\star(\lambda^\star) = 1\}$
    Update weights $d_{inner}(s) \leftarrow d_{inner}(s) \big(\sum_{h_t \in K} h_t(x)/T_{inner} - y_s \big)^2$
    Normalize $d_{inner}(s) \leftarrow d_{inner}(s) / \sum_{s=1}^{S} d_{inner}(s)$
until validation error $\text{Error}_{\text{val}}$ stops decreasing

seen as an enrichment process. In the first round, the algorithm selects those T_{inner} weak classifiers out of subset of Q that produce the optimal validation error. The subset of Q weak classifiers has been preselected from a dictionary with a cardinality possibly much larger than Q. In the next step the algorithm fills the $Q - T_{inner}$ empty slots in the solver with the best weak classifiers drawn from a modified dictionary that was adapted by taking into account for which samples the strong classifier constructed in the first round is already good and where it still makes errors. This is the boosting

idea. Under the assumption that the solver always finds the global minimum, it is guaranteed that for a given λ the solutions found in the subsequent round will have lower or equal objective value i.e. they achieve a lower loss or they represent a more compact strong classifier. The fact that the algorithm always considers groups of Q weak classifiers simultaneously rather than incrementing the strong classifier one by one and then tries to find the smallest subset that still produces a low training error allows it to find optimal configurations more efficiently.

If the validation error cannot be decreased any further using the inner loop, one may conclude that more weak classifiers are needed to construct the strong one. In this case the "outer loop" algorithm "freezes" the classifier obtained so far and adds another partial classifier trained again by the inner loop.

Algorithm 2 $T > Q$ (outer loop)

Require: Training and validation data, dictionary of weak classifiers
Ensure: Strong classifier

1: Initialize weight distribution d_{outer} over training samples as uniform distribution $\forall s : d_{outer}(s) = 1/S$
2: Set $T_{outer} \leftarrow 0$ and $c(\boldsymbol{x}) \leftarrow 0$
3: **repeat**
4: Run **Algorithm 1** with d_{inner} initialized from current d_{outer} and using an objective function that takes into account the current $c(\boldsymbol{x})$:
$\boldsymbol{w}^\star = \operatorname{argmin}_{\boldsymbol{w}} \big(\sum_{s=1}^{S} \big\{ \big(c(\boldsymbol{x}_s) + \sum_{i=1}^{Q} w_i h_i(\boldsymbol{x}_s) \big) / (T_{outer} + Q) - y_s \big\}^2 + \lambda \|\boldsymbol{w}\|_0 \big)$
5: Set $T_{outer} \leftarrow T_{outer} + \|\boldsymbol{w}^\star\|_0$ and $c(\boldsymbol{x}) \leftarrow c(\boldsymbol{x}) + \sum_{i=1}^{Q} w_i^\star h_i(\boldsymbol{x})$
6: Construct a strong classifier $H(\boldsymbol{x}) = \operatorname{sign}\big(c(\boldsymbol{x})\big)$
7: Update weights $d_{outer}(s) = d_{outer}(s) \big(\sum_{t=1}^{T_{outer}} h_t(x) / T_{outer} - y_s \big)^2$
8: Normalize $d_{outer}(s) = d_{outer}(s) / \sum_{s=1}^{S} d_{outer}(s)$
9: **until** validation error $\text{Error}_{\text{val}}$ stops decreasing

3.4 Mapping to hardware

The QBoost algorithm circumvents the need to solve QUBOs with many variables, and thus is suitable for execution on D-Wave hardware. However, one important obstacle remains before the learning algorithm can be executed on hardware. As described, QBoost requires complete coupling between all pairs of qubits (optimization variables). The coupling between qubits i and j representing variables w_i and w_j is proportional to $\mathbb{E}_{P_0(\boldsymbol{x})}\big(h_i(\boldsymbol{x}) h_j(\boldsymbol{x})\big)$, which typically will not be zero. Thus, when realized in hardware with sparse connectivity, many connections must be discarded. How can this be done in a way which minimizes the information lost?

Let $G = (V, E)$ indicate the graph of connectivity between qubits. As the QUBOs are translated to Ising problems we embed an Ising objective of N variables into the qubit connectivity graph (with $|V| \geq N$) with a mapping $\varphi : \{1, \ldots, N\} \mapsto V$ such that $(\varphi(i), \varphi(j)) \in E \Rightarrow J_{i,j} \neq 0$. The mapping φ can be encoded with a set of binary variables $\phi_{i,q}$ which indicate that problem variable i is mapped to qubit node q. For a valid mapping we require $\sum_q \phi_{i,q} = 1$ for all problem variables i, and $\sum_i \phi_{i,q} \leq 1$ for all qubit nodes q. To minimize the information lost we maximizing the total magnitude of $J_{i,j}$ mapped to qubit edges, i.e. we seek

$$\boldsymbol{\phi}^\star = \operatorname*{argmax}_{\boldsymbol{\phi}} \sum_{i>j} \sum_{(q,q') \in E} |J_{i,j}| \phi_{i,q} \phi_{j,q'}$$

where $\boldsymbol{\phi}$ is subject to the mapping constraints.[4] This problem is a variant of the NP hard quadratic assignment problem. Since this embedding problem must be solved with each invocation of the quantum hardware, we seek a fast $\mathcal{O}(N)$ heuristic to approximately maximize the objective.

As a starting point, let $i_1 = \text{argmax}_i \sum_{j<i} |J_{j,i}| + \sum_{j>i} |J_{i,j}|$ so that i_1 is the row/column index of $\boldsymbol{J}$ with the highest sum of magnitudes ($\boldsymbol{J}$ is upper-triangular). We map i_1 to the one of the qubit vertices of highest degree. For the remaining variables, assume that we have defined φ on $\{i_1, \ldots, i_k\}$ such that $\varphi(i_j) = q_j \in V$. Then assign $\varphi(i_{k+1}) = q_{k+1}$, where $i_{k+1} \notin \{i_1, \ldots, i_k\}$ and $q_{k+1} \notin \{q_1, \ldots, q_k\}$ to maximize the sum of all $|J_{i_{k+1},i_j}|$ and $|J_{i_j,i_{k+1}}|$ over all $j \in \{1, \ldots, k\}$ for which $\{q_j, q_{k+1}\} \in E$. This greedy heuristic performs well. As an example, the greedy heuristic manages to map about 11% of the total absolute edge weight $\sum_{i,j} |J_{i,j}|$ of a fully connected random Ising model into the hardware connectivity of Fig. 1 in a few milliseconds. For comparison, a tabu heuristic designed for this matching quadratic assignment problem performs marginally better attaining about 11.6% total edge weight after running for 5 minutes.

4 Experimental results

We test QBoost to develop a detector for cars in digital images. The training and test sets consist of 20 000 images with roughly half the images in each set containing cars and the other half containing city streets and buildings without cars. The images containing cars are human-labeled ground truth data with tight bounding boxes drawn around each car and grouped by positions (e.g. front, back, side, etc.) For demonstration purposes, we trained a single detector channel only using the side-view ground truth images. The actual data seen by the training system is obtained by randomly sampling subregions of the input training and test data to obtain 100 000 patches for each data set. Before presenting results on the performance of the strong classifier we characterize the optimization performance of the hardware.

4.1 Optimization efficacy

The hardware used in the experiment is an early generation Chimera chip where 52 of the 128 qubits were functioning. The connectivity of the functioning qubits is shown in Fig 3. All Ising problems defined on even the fully functional 128 qubit array can be solved exactly using dynamic programming as the treewidth of the fully functional 128 qubit graph is 16. Thus, we can easily obtain global minima for problems defined on the function 52 qubit subset. Fig. 4(a) shows the results on a typical problem generated at late stages in QBoost. We see that although there is certainly room for improvement, good results can be obtained with as few as $m = 32$ samples. Fig. 4(b) shows the distribution of the hardware-obtained energy difference from the global minimum in units of the standard deviation of Ising objective value across all configurations. We see clear improvement with increasing numbers of samples m. For the runs of QBoost we used $m = 32$ and found satisfactory performance. Larger values of m were tried but the performance gains did not warrant the time cost.

[4] The linear terms of the original ISING problem pose no problem, and can be mapped without loss into the hardware.

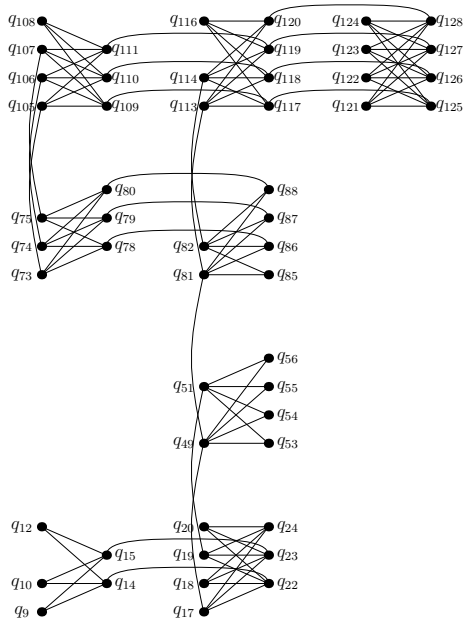
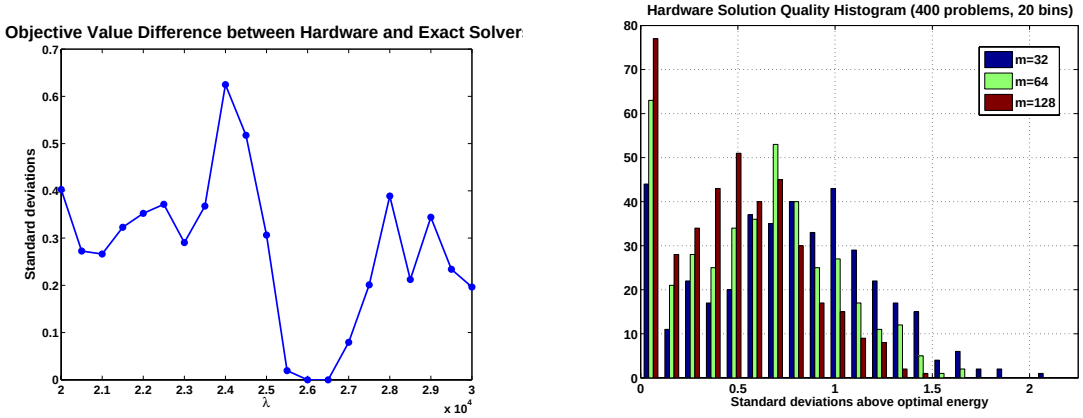


Figure 3: Functioning qubits used in the car detector training. Compare with Fig. 1.



(a) (b)

Figure 4: (a) Comparison of hardware solution (obtained using $m = 32$ samples) to the known globally optimal result for an "inner loop" optimization across a set of regularization weights λ. We plot the objective value after the problem is translated to Ising format. Differences of objective values are in units of the standard deviation of objective values across all configurations. (b) Histogram of differences of hardware objective values and the globally optimal result (in units of the problems standard deviation) across a random sampling of 400 Ising problems taken at different stages of QBoost, and for randomly selected λ. Results are shown for $m = 32$, 64, and 128 samples.

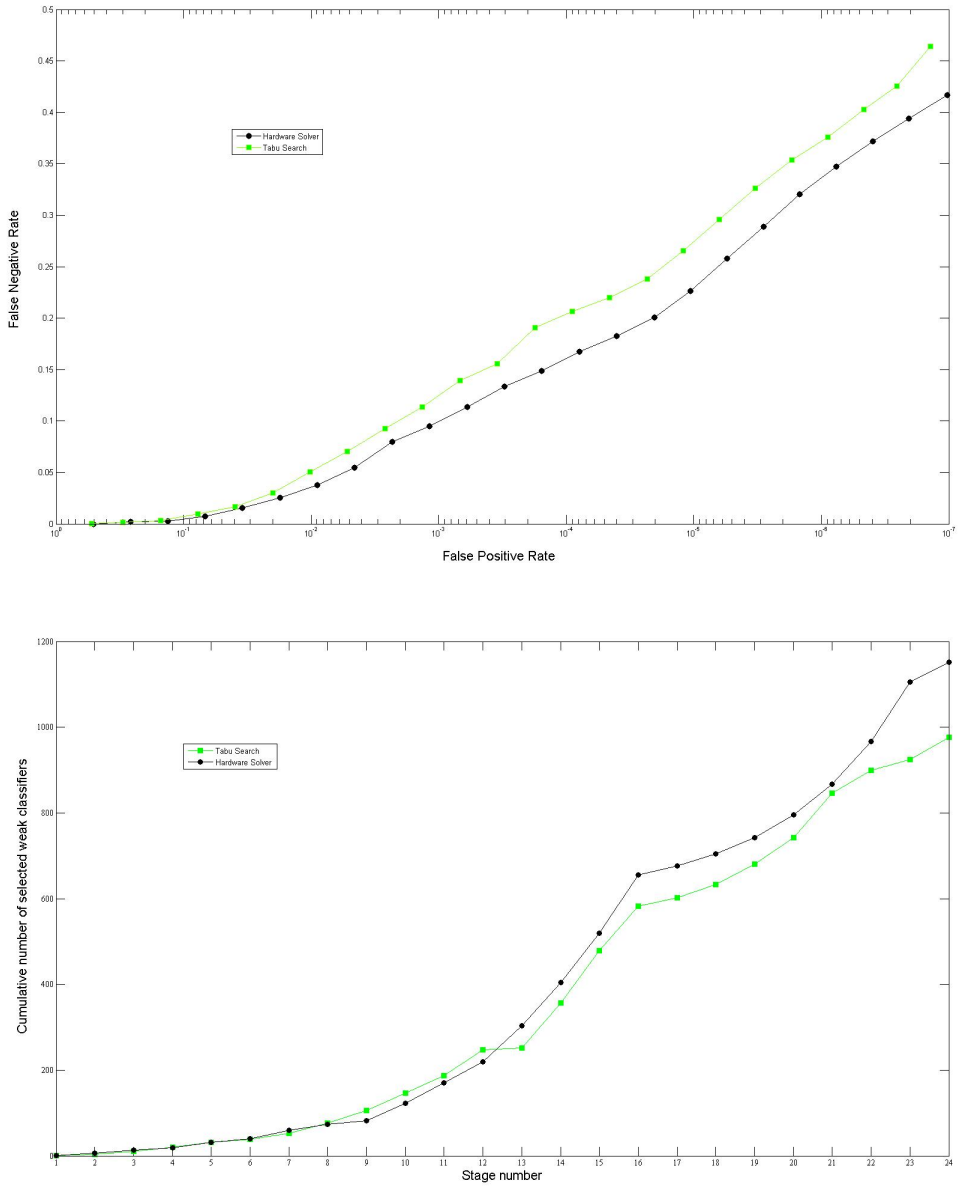


Figure 5: Error rates and sparsity levels for Tabu Search as compared to the quantum hardware solver. The training consisted of training 24 cascaded stages that employ three different feature types. Each stage is trained on the false negatives of the previous stage as its positive samples complemented with true negative samples unseen so far [VJ04]. The error rates of the hardware solver are superior to the ones obtained by Tabu search across all stages. Please note that the error rates shown are per pixel errors. The cumulative number of weak classifiers employed at a certain stage does not suggest a clear advantage for either method.

4.2 Classifier results

We use three types of weak classifiers. Each weak classifier takes a rectangular subregion of the scanned image patch as input, and outputs a continuous value encoding the log likelihood ratio of positive and negative classes. The three types of weak classifiers work similarly in the sense that they all divide the input rectangular subregion into grid cells, map multiple raw grid cell values linearly into a single scalar response, quantize this scalar value into multiple bins, and output the confidence score associated with that bin. They differ only in grid sizes and mapping weights, leading to different complexity and discrimination power. We group weak classifiers into multiple stages. New bootstrap negative samplers are created at the end of each stage. For speed consideration, more complex yet more powerful weak classifiers are only considered in later stages.

We compare 3 different variants of the baseline algorithm all of which use a window of size $Q = 52$. The first variant allows for complete connectivity between all 52 optimization variables, and is solved using a tabu heuristic designed for QUBOs [Pal04]. The two remaining variants utilize the subgraph of K_{52} given in Fig. 3. The only difference between these two variants is that one solves all QUBOs using an exact software solver while the other solves them approximately using the quantum hardware. The exact solver performs equivalently to the hardware version if additive zero mean Gaussian noise is added to Ising parameters before solving. All three variants are compared to a highly tuned state-of-the-art system that blends a variety boosting techniques on the same dictionary of weak learners [SS98][VPZ06].

Starting from the highly tuned state-of-the-art system developed for Google's MapReduce infrastructure, we replaced its feature-selection routine by QBoost. For each stage, the original system calls the feature selection routine, requesting a fixed number T of features for that stage. The numbers of features for different stages and different cascades are determined by experience and generally involve smaller numbers for earlier stages and larger numbers for later stages in order to achieve an optimal tradeoff between accuracy and processing time during evaluation of the final detector.

Consequently, QBoost is invoked with a request for selecting T features out of the dictionary (size up to several million) of the current cascade. We keep the window size Q fixed at 52, which is the maximum problem size that the current quantum hardware can take. Since T can be both smaller and larger than 52, early stages utilize only the inner loop of QBoost, whereas later stages take advantage of the outer loop as well. However, regardless of the particular value of T for a given stage, we obtain the top 52 features using the current weight distribution on the training data and deliver to the quantum hardware an optimization problem of that size. In order to ensure our ability to make a feature selection of size at most T when $T < 52$, we send to the hardware multiple optimization problems with varying regularization strengths. After that, we use cross-validation to evaluate all of the returned solutions of size at most T and select the one that gives the lowest validation error. One of the advantages of QBoost is that it could very well be the case that the best validation error is obtained for a solution of size smaller than T, thereby achieving better sparsity in the final detector.

QBoost iterates in this manner until it determines that it cannot lower the validation error any further, at which point it returns the selected features for a single stage to the original system. The original system then finalizes the current stage with them and proceeds to the next one.

We found that training using the quantum hardware led to a classifier that produces useful classification accuracy as visitors to our demonstration stand will be able to experience first hand. But

more importantly as shown in more detail in Fig. 5 the accuracy obtained by the hardware solver is better than the accuracy of the classifier trained with Tabu search. However, the classifier obtained with the hardware solver even though it is sparser by about 10% has a false negative rate approximately 10% higher than the boosting framework we employed as a point of departure. This can have several reasons. One possibility is that the weak learners used here have 16 different output values as opposed to just two as studied in our previous studies [NDRM08, NDRM09]. This could imply that we need more than a single bit to represent the weights. Another reason may be the drastically changed ratio of Q relative to the number of weak learners as here we employ several million weak classifiers.

5 Discussion and future work

Building on earlier work [NDRM08] we continued our exploration of QUBO-based optimization as a tool to train binary classifiers. The present focus was on developing algorithms that run on an existing hardware QUBO solver developed at D-Wave. The proposed QBoost algorithm accommodates the two most important constraints imposed by the hardware:

- QUBOs with a great many variables cannot be directly addressed in hardware, consequently large scale QUBOs must be decomposed into a sequence of smaller QUBOs
- the hardware only allows for sparse but known connectivity between optimization variables (qubits), thus each problem must discard some interactions before solving

The first constraint is overcome by using a boosting-inspired greedy algorithm which grows the set of voting classifiers. The second constraint is solved by dynamically mapping QUBOs into the fixed connectivity by minimizing the loss of pairwise interactions. The resultant algorithm was found to generate a strong classifier that, while not as accurate as a highly-tuned AdaBoost-based classifier, did perform well with fewer weak classifiers.

Being but the first run on hardware QUBO solvers there are a great many questions that are unanswered and control experiments still need to be performed. We highlight a number of important issues to explore.

The classifier trained using a tabu heuristic on the fully connected graph of all pairwise interactions performed noticeably worse than both the Boosting, and the variants that threw away edges in the mapping down to hardware connectivity. Is this poor performance the result of poor optimization of the fully connected 52 variable problem, or does inclusion of all pairwise interactions actually lead the algorithm away from classifiers with the lowest generalization error? If we were able to exactly solve the 52 variable QUBOs would performance have been better or worse than the solvable sparsely connected problems? On a related note there is good reason to suspect that even with exact optimization, the fully connected problem arising from quadratic loss may not be the best choice. Our choice for quadratic loss is pragmatic – quadratic loss leads directly to a QUBO. However, zero-one loss or a margin-based loss may be much more effective. These other loss functions can be represented as QUBOs, but require the introduction of additional variables to represent higher-than-quadratic interactions.

The embedding of the fully connected quadratic objective into the sparsely connected hardware throws away a great many interactions. The current scheme which seeks to minimize the magnitude

of discarded interactions is but one choice. Are there other embedding criteria that can be greedily optimized, and what is the effect of these alternate embeddings on generalization error?

This first experiment provides no insight into the importance of the sizes of QUBOS Q that can be solved. Certainly as Q increases we expect that we can better approximate the globally best sparse voting subset from the large dictionary. How rapidly does the identification of this optimal improve as Q increases? Does a closer approximation to the regularized sparse set translate into lower generalization error? Initial insights into these questions are easily obtained by experiments with differing Q. Improved versions of the current hardware will allow for explorations up to $Q = 128$.

Lastly, previous experiments on other data sets using software heuristics for QUBO solving have shown that performance beyond AdaBoost can typically be obtained. Presumably the improvements are due to the explicit regularization that QBoost employs. We would hope that QBoost can be made to outperform even the highly tuned boosting benchmark we have employed here.

Finally, we mention that the experiments presented here were not designed to test the quantumness of the hardware. Results of such tests will be reported elsewhere.

Acknowledgments

We would like to thank Johannes Steffens for his contributions to the baseline detector and Bo Wu for preparing the training data.

References

[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`.

[FGGN05] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Daniel Nagaj. How to make the quantum adiabatic algorithm fail. 2005. Available at `http://arxiv.org/abs/quant-ph/0512159`.

[FGGS00] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. 2000. arXiv: quant-ph/0001106v1.

[FHT98] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[Fre09] Yoav Freund. A more robust boosting algorithm. 2009. arXiv: stat.ML/0905.2138v1.

[FS95] Yoav Freund and Robert E. Shapire. A decision-theoretic generalization of online learning and an application to boosting. *AT&T Bell Laboratories Technical Report*, 1995.

[KTM09] K. Kurihara, S. Tanaka, and S. Miyashita. Quantum annealing for clustering. In *The 25th Conference on Uncertainty in Artificial Intelligence*, 2009. Available at `http://www.cs.mcgill.ca/~uai2009/papers/UAI2009_0019_71a78b4a22a4d622ab48f2e556359e6c.pdf`.

[LS08] Philip M. Long and Rocco A. Servedio. Random classification noise defeats all convex potential boosters. *25th International Conference on Machine Learning (ICML)*, 2008.

[NDRM08] Hartmut Neven, Vasil S. Denchev, Geordie Rose, and William G. Macready. Training a binary classifier with the quantum adiabatic algorithm. 2008. arXiv: quant-ph/0811.0416v1.

[NDRM09] H. Neven, V. Denchev, G. Rose, and W. Macready. Training a large scale classifier with the quantum adiabatic algorithm. 2009. Available at `http://arxiv.org/abs/0912.0779`.

[Pal04] Gintaras Palubeckis. Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. *Ann. Oper. Res.*, 131:259–282, 2004.

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

[SKT+09] I. Sato, K. Kurihara, S. Tanaka, S. Miyashita, and H. Nakagawa. Quantum annealing for variational bayes inference. In *The 25th Conference on Uncertainty in Artificial Intelligence*, 2009. Available at `http://www.cs.mcgill.ca/~uai2009/papers/UAI2009_0061_3cf8f564c6b5b6a22bd783669917c4c6.pdf`.

[SS98] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *COLT*, pages 80–91, 1998.

[VC71] Vladimir Vapnik and Alexey Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

[VJ04] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[VPZ06] Paul Viola, John Platt, and Cha Zhang. Multiple instance boosting for object detection. In Bernhard Schölkopf Yair Weiss and John C. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18, pages 1417–1424. MIT Press, 2006.

[Wyn02] Abraham J. Wyner. Boosting and the exponential loss. *Proceedings of the Ninth Annual Conference on AI and Statistics*, 2002.

[Zha04] Tong Zhang. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85, 2004.