Assessing the Quantum Signature of the D-Wave One Machine

Andrew Guo[1] and Brian Wai[2]

***Abstract*— In our project, we will explore the controversial question of whether the D-Wave One machine is a true quantum computer. D-Wave Systems claims that their machine, a quantum annealer obtains a speed-up over classical computers on combinatorial optimization problems. Using machine learning tools, we hoped to determine features that would classify a problem as 'easy' or 'hard' for the D-Wave One. By comparing these features to those of classical simulations, we would obtain a metric to measure the quantum signature of the D-Wave One machine. Our Naive Bayes classifiers yielded inconclusive results, due to a lack of adequate features and insufficient data.**

I. INTRODUCTION

The nascent field of quantum computation aims to create quantum devices that possess computational capabilities that far surpass those of their classical peers. By utilizing the exponentially larger parameter space of coherent quantum systems, quantum computer scientists aim to achieve "quantum" speed-ups. They have shown exponential speed-up in the factoring of large numbers via Shor's algorithm, as well as other promising applications such as the efficient simulation of quantum systems and a quadratic speed-up in searching via Grover's algorithm [4].

Quantum annealing methods comprise a proper subset of quantum computational techniques, utilizing such quantum behaviors as tunneling to obtain a quantum speed-up. Finite-distance quantum tunneling - a phenomenon whereby a system can overcome costly energy barriers surrounding local minima by passing through them - has proven useful for the discovery of local minima of binary optimization problems. Simulated quantum annealing has been shown to be more efficient than (classical) thermal annealing for certain problems that can be modeled by 2D Ising spin glasses. The goal is to find the ground state of a Hamiltonian (i.e. a cost function) given by [3].

$$H_{ising} = \sum_{i<j} J_{ij}\sigma_i^z\sigma_j^z - \sum_i H_i\sigma_i^z$$

For our project, we assume that each initialization of the Chimera graph uniquely determines the probability that the D-Wave machine will succeed on a given configuration. Initially, we treat the input as the starting state for the chimera graph, and then take the output to be the probability of success (that is, the probability of getting the lowest energy state). We then train a Naive Bayes model on the edges energies to obtain an estimate of the probability of success.

We have also noticed that the D-wave machine tends to do very well on some problems and rather poorly on others. That is, it finds the lowest energy state fairly often given some initial configurations and very infrequently given other initial configurations. We used this information to reapply the given problem as a classification problem; e.g. whether we could classify certain initial configurations as easy or hard for the D-wave machine.

II. RELATED WORK

Our inspiration for assessing the quantum signature of the D-Wave One was inspired and facilitated by vigorous academic debate on the arXiv. In March of 2011, D-wave Systems released a paper claiming to have achieved quantum annealing with over one hundred qubits [1]. They justified their claim of a quantum speed-up not via the D-Wave Ones speed which, at the time, still lagged behind simulated annealers run on laptop computers - but by the qualitative aspects of its performance. When run on a variety of problems, D-Wave One found certain problems hard (average probability of finding the optimal solution near 0), and other problems easy (average probability of finding the optimal solution near 1). This meant that the D-Wave One had a bimodal distribution of success probabilities, which seemed categorically different from classical models. The simulated annealer generated a Gaussian distribution of success probabilities, while D-Wave's distribution consisted of two mixed Gaussians. By finding good correlation between the easy and hard problems with that of a simulated quantum annealer, and poor correlation with a classical annealer, D-Waves supporters argued that D-Wave was indeed behaving in a quantum manner.

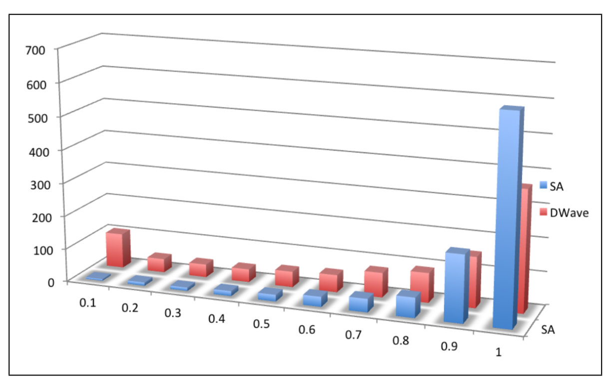

Fig. 1. Distribution of Success Probabilities of D-Wave and Simulated Annealing

*This work was supported by Stanford University

[1]Andrew Guo is a B.A. Candidate in Physics at Stanford University, Class of 2016. `aguoman@stanford.edu`

[2]Brian Wai is a B.A./M.S. Candidate in Math/Computer Science at Stanford University, Class of 2016. `brianwai@stanford.edu`

D-waves critics, however, offered differing explanations for the D-Wave Ones supposed deviation from existing classical algorithms [2]. A paper by John Smolin & Graeme Smith argues that the comparison between D-wave and a generic simulated annealer was specious, as the wrong classical model was used. To test this, we decided to apply machine learning to test a home-brewed classical simulated annealer on the data, which also generated a bimodal distribution for success probabilities. By using a learning algorithm that could learn which problems were easy or hard for the D-Wave and for the classical annealers, we could test whether certain features determined the difficulty of the problem. By comparing the most important features for D-Wave and for the classical annealer, we hoped to find another source of data to support or refute the hypothesis that the D-Wave One is a quantum computer.

III. Datasets and Features

Our dataset for the D-wave behavior was the 1000 training examples released publicly by D-wave. The data consists of the starting energy and the energy between edges. The data for the classical model is generated by a C program, which we run on the same initial starting states as the D-Wave data to generate our simulated thermal annealing data. The features we used all derived from the initial edge energies of the Chimera graph, which is shown in the graph below.

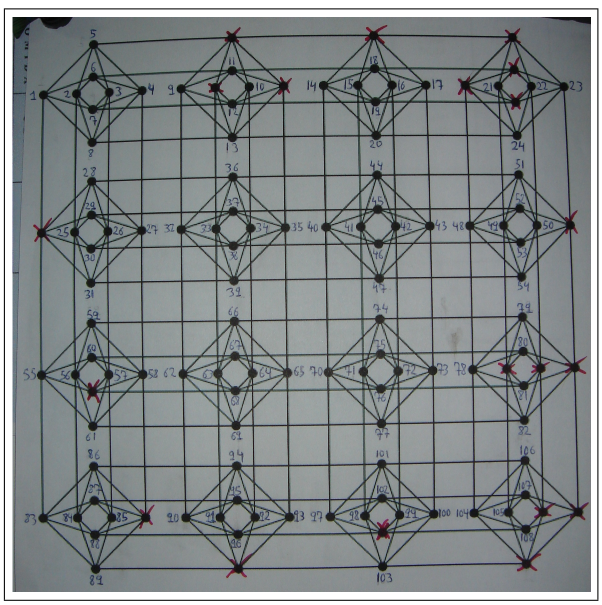

Fig. 2. Portrayal of the Chimera Graph

We attempted to find a subspace of our high-dimensional (n=256) feature space using PCA, but since our features were sampled from (correlated) Bernoulli random variables with parameter $\phi = 0.5$, they already possessed maximum variance. We used sums of the energies pairs of edge that coincide at a vertex in order to model first-order correlations between edge.

Data examples for the D-Wave machine were obtained from the ancillary files on the arXiv [3]. The code and results for the simulated classical annealer came courtesy of Tomas Navarro, a student in the EdX course "CS-191x: Quantum Mechanics and Quantum Computation," taught by Professor Umesh Vazirani in 2013. [5]

TABLE I

Sample of D-Wave Data (# Rows = 255)

First Vertex	Second Vertex	Edge Energy
1	3	-1
2	5	1
3	8	-1
3	5	1
⋮	⋮	⋮
107	108	1

IV. Methods

The learning algorithms we used were Nave Bayes, (Weighted) Linear Regression, and SVM. Initially, we began with Linear Regression and SVM. Linear regression formulates the problem by training the classifier as a linear function of the features (the edges of the chimera graph). Each answer, in this case the probability of success, is modeled as a linear function of the edges, and we aim to minimize the average squared error. A support vector machine works as a linear classifier by weighing the vectors with the best possible supports for the data. That is, although the SVM uses infinitely many possible vectors, it features a regularization component that allows it to not overfit the data set given a sufficient amount of data. Generally, a support vector machine will do better than Naive Bayes given a sufficient amount of data because it does not assume each feature is independent of the others. However, when not given enough data, a support vector machine will overfit the training set and not generalize well to other examples. Nave Bayes works by modeling each feature as an independent variable. Given that we have either a easy or hard problem, we can find the probability that each edge is labeled 1 or -1 given whether the problem is easy or hard. Given the prior distribution on easy/hard problems, given a problem and its edge specification, we can calculate the probability of the problem being easy or hard, and output the answer with the greater probability. Unfortunately, both methods gave us unacceptable error values of the success probabilitywe had around .1 error, which was unacceptable. Therefore, we decided to reformulate our problem as a classification problem. This was inspired by the fact that the D-wave success probabilities form a somewhat bimodal graph, as D-wave does very well on some types of problems and not well on others, as seen in the graph below.

To establish a baseline, we used a classifier that picked the majority value; this baseline was .3. Using linear regression and SVM to classify our models did not pass the baseline of .3 error. Therefore, we decided to use Nave Bayes. In order to do this, we first ran Nave Bayes on all possible edges. This gave us an error of around .25. However, we knew the Nave

Bayes assumption is inapplicable here because the difficulty of the problem mainly depends on the interaction between the edges. Because we have difficulty finding which pairs of edges affect the function the most, we ran Nave Bayes all pairs of edges. Unfortunately, this failed to converge because we lacked the required data. Therefore, because we could not do this, we looked for a way to compress the number of features. Because the previous Nave Bayes algorithm on each edge weighted each edge about equally, we decided to try running Nave Bayes on the sum of the edges; e.g. the total energy. When we did this, we found out that we got the test error down to 20, because we are no longer overfitting the data. The test error and the training error were about equal in this case; in fact, in some cases, the test error was actually lower than the training error.

V. Experiments, Results, and Discussion

First, we tried linear regression and an SVM. The SVM has low training error and high test error because we did not have enough training examples to fit the features we wanted. Linear regression did converge, but the mean squared error, .11, was too high to predict the probability of success accurately. Next, we tried to train Nave bays on the initial energy of each edge as our features. Using a training set of 800 examples and a test set of 200 examples, our results can be seen in the graph below.

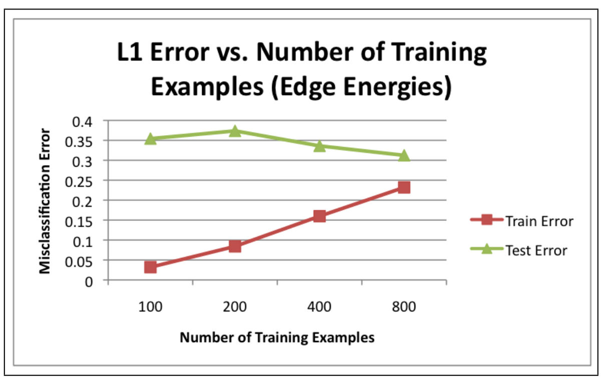

Fig. 3. Misclassification error of Naive Bayes trained on single edge energies as features. Test size was 200 examples.

Here, the training error and test error converge to around 0.3, but we have yet to converge because we lack the required data. In other words, even with only 255 features, we are overfitting the training set, and therefore, we should compress our features. Because of this, we implemented Nave Bayes on the energy alone. Because the energy was a number between 0 and 255, we encoded the energy as 8 bits of "separate" observations. This gives us the graph below. Here, we see that when we train only given the energy, the test error converges almost immediately; in fact, the fact that the test error is lower than the training error shows that we have in fact converged and are not overfitting. We then attempted to run Nave Bayes on pairs of edges, but the algorithm predictably failed to converge due to the lack of available data.

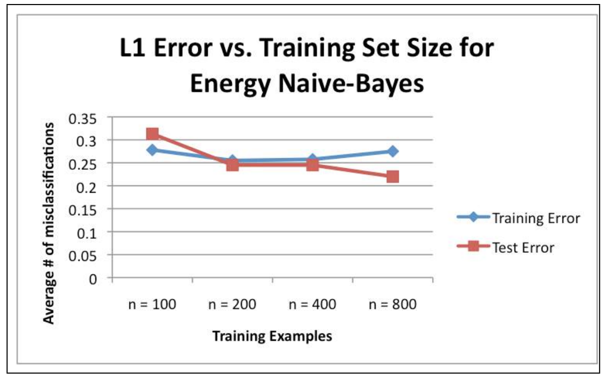

Fig. 4. Misclassification error of Naive Bayes trained on the total energy as the feature. Test size was 200 examples.

We noticed that the Naive Bayes classifier for the D-wave Machine classified the classical data with a similar error of 0.20, while the Naive Bayes classifier trained on the classical data classified the D-wave data with similar error (below 0.30). However, because both error values are still high, further research is necessary to draw a conclusion.

VI. Conclusion and possible Future work

For this project, the main roadblock was the lack of data. Only so many inferences can be made with 1000 data points. This is a problem because we have 255 possible edges, so even if we only count edges alone as features gives us 255 of them, let alone accounting for interaction between edges. We simply do not have enough data to train all our parameters. Another problem was the lack of a quantum algorithm to formulate quantum data, as we cannot compare the D-wave data to data we do not have.

Nave Bayes was the best-working algorithm due to the low number of data points we have, as Nave Bayes does not require much data to function effectively. However, Naive Bayes makes the assumption that all the features work independently, which eliminates any interaction between the features. This is important because the interaction between the edge energies have great physical importance in the Chimera Graph. Therefore, in order to advance this project further, we would need either more data, or better features. Unfortunately, we lack the physics knowledge required to construct better features by hand. Therefore, we can only hope to obtain more data from DWave.

With more data, the next step would be to run PCA or a SVM if possible on the increased data. If this is not possible, we would run Nave Bayes on pairs of edges, and extract out the edges which have the highest significance. This would allow us to find the pairs of edges which correlate best with the success probabilities, and then we can extract features given these edges.that is, we could look at pairs of these edges and train a neural network model on them.

To formulate the quantum algorithm, we would code up a Quantum Monte Carlo algorithm to mimic what the current classical algorithm does, except it would take into account entangled states to get to the lowest possible energy state.

References

[1] S. Boixo, et al. Quantum annealing with more than one hundred qubits. arXiv:1304.4595

[2] J.A. Smolin, G. Smith. Classical signature of quantum annealing. arxiv:1305.4904

[3] L. Wang, et al. Comment on: "Classical signature of quantum annealing" arXiv:1305.5837

[4] A. Harrow. Why now is the right time to study quantum computing. arXiv:1501.00011

[5] Tomas Navarro's Dropbox link can be found under the Class Project discussion page for CS-191x: https://courses.edx.org/courses/BerkeleyX/CS-191x/2013_August/courseware/