"Quantum" Classification of Malware

John Seymour
seymour1@umbc.edu

Charles Nicholas
nicholas@umbc.edu

August 24, 2015

Abstract

Quantum computation has recently become an important area for security research, with its applications to factoring large numbers and secure communication. In practice, only one company (D-Wave) has claimed to create a quantum computer that can solve relatively hard problems, and that claim has been met with much skepticism. Regardless of whether it is using quantum effects for computation or not, the D-Wave architecture cannot run the standard quantum algorithms, such as Grovers and Shors. The D-Wave architecture is instead purported to be useful for machine learning and for heuristically solving NP-Complete problems.

We'll show why the D-Wave and the machine learning problem for malware classification seem especially suited for each other. We also explain how to translate the classification problem for malicious executables into an optimization problem that a D-Wave machine can solve. Specifically, using a 512-qubit D-Wave Two processor, we show that a minimalist malware classifier, with cross-validation accuracy comparable to standard machine learning algorithms, can be created. However, even such a minimalist classifier incurs a surprising level of overhead.

1 Introduction

The D-Wave architecture is a unique approach to computing that utilizes quantum annealing to solve discrete optimization problems. At the time of this writing, the extent to which the D-Wave machines utilize quantum effects for computational purposes is a hotly debated topic. Regardless, the D-Wave machine is not a general purpose quantum computer; it cannot run well-known quantum algorithms such as Shors or Grovers algorithms. Applications for D-Wave machines instead include binary classification, complex protein-folding models, and heuristics for solving intractable problems such as the Traveling Salesman Problem. We focus on one method for binary classification, QBoost, first explained in [2]. This method has been shown to outperform several standard techniques for classification, especially

in contexts where instances may be labeled incorrectly. As malware datasets often have this characteristic, the D-Wave might be especially suited for the problem of malware classification.

Recently, D-Wave has released the D-Wave Two, a quantum annealer with up to 512 qubits, and a 1000 qubit machine is currently being tested in D-Wave's lab. D-Wave claims that the number of qubits will continue to scale for the forseeable future. More qubits means more difficult problems can be embedded onto the chip directly, extending the problem space that the D-Wave system can natively support. At UMBC, we have access to a D-Wave Two processor with 496 working qubits, called SYSTEM 6, and software for embedding problems onto the chip. We also have access to software which simulates a D-Wave chip on a classical machine.

The D-Wave chip consists of niobium loops that act as qubits, and couplers which affect both individual loops and pairs of loops. Programming the D-Wave consists of choosing the weights for these couplers. The D-Wave natively solves problems of the following form:

$$\sum_i a_i q_i + \sum_{i,j} b_{ij} q_i q_j \qquad (1)$$

where the a_i and the b_i values are given, and the D-Wave returns the list of $q_i \in \{-1, 1\}$ that minimize the above summation. Translating a real-world problem into this form reduces to the Graph Minor Embedding problem, which is NP-Complete in the general case. However, several heuristics exist that may be able to embed real-world problems onto D-Wave chips for specific instances. In particular, QBoost involves a dialogue between a classical Tabu search and the D-Wave chip. Figure 1 gives a graphical depiction of this equation, with weights set, based on the SYSTEM 6 processor.

One major difference between the D-Wave machines and the D-Wave simulator is the presence of dead qubits in the actual machines. In Figure 1, there are several nodes that are absent from the graph. Programmers cannot interact with them, as they are defects in the actual chip. This influences the possible values for variables in equation 1, hence, it limits the potential problems that the chip can solve. D-Wave chips can have different numbers and placements of dead qubits.

Boosting is a machine learning technique which takes a set of weak classifiers, or classifiers with only a slightly-better-than-random accuracy, and combines them into a strong classifier with much higher accuracy. QBoost differs from standard boosting algorithms as each weak classifier has the same weight, and the final strong classifier is created simply by taking the majority vote from the weak classifiers comprising it. QBoost searches over the subsets of weak classifiers and attempts to minimize the error of the strong classifier through inclusion or exclusion of weak classifiers. This error is represented through a loss function: the smaller the loss, the better the quality of the classifier. [1] has one example of a useable loss function, which can be found in the following equation.

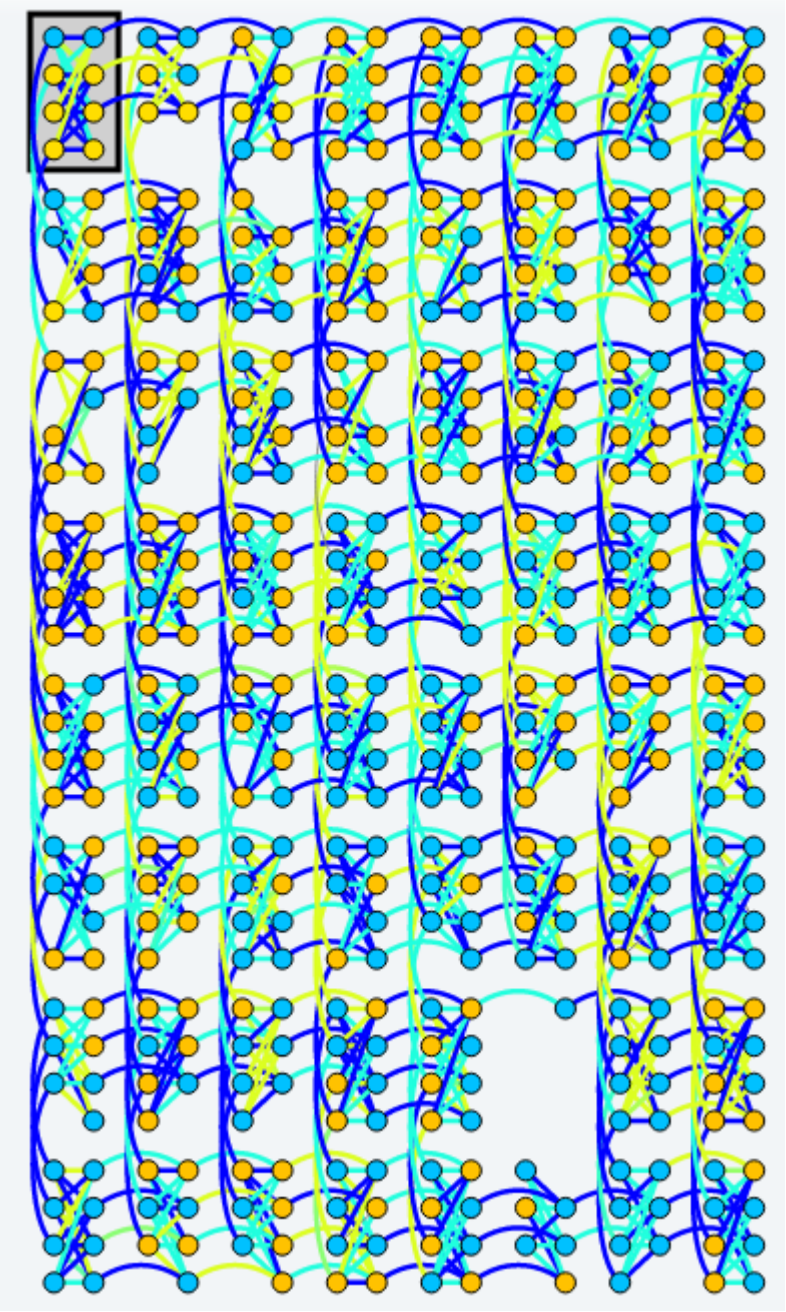

Figure 1: Graphical depiction for the SYSTEM 6 processor, known as the Chimera graph.

$$G(w) = \frac{1}{4}\sum_{s=1}^{S}(sign[\sum_{j=1}^{D} w_j F_j(x_s)] - y_s)^2 + \lambda \sum_{j=1}^{D} w_j \quad (2)$$

Briefly, the $\frac{1}{4}\sum_{s=1}^{S}(sign[\sum_{j=1}^{D} w_j F_j(x_s)] - y_s)^2$ corresponds to the number of errors a given strong classifier will make, and $\lambda \sum_{j=1}^{D} w_j$ serves as a regularization constant to prioritize strong classifiers that use smaller numbers of features.

2 Methods

There are a few publically available corpora for malicious executables. We use VxHeaven, which consists of 65 gigabytes of malware, labeled by type (e.g. banking trojans). There is, however, no standard dataset for benign software. We supplement Vx Heaven with Windows XP, Windows 7, Cygwin, and Sourceforge executables as in previous work.[5] We then resample the corpus, because the raw corpus consists of many more malicious executables than benign executables. Thus, a classifier that simply classifies all executables as malware would have a near-perfect accuracy on the raw corpus, but it would not be useful at all in practice. Resampling also has the side-effect of reducing the time to build classifiers on the corpus. We resample with replacement, meaning it is possible to select an executable multiple times and have multiple copies of that executable in the resampled corpus. Resampling with replacement, as opposed to without replacement, has several good statistical properties in terms of the resulting distribution.

We chose to use 3-grams as the basis of our classifier. We specifically chose 3-grams because they are easy to generate, because similar features have been used before for classification of malware, and because it is easy to obtain a large list of binary features which can be trivially translated into weak classifiers. A classifier built using only 3-grams will not have accuracy comparable to malware classifiers currently used in industry. However, our goal here is to compare QBoost to standard machine learning algorithms, and the classifiers we build even with these simplistic features will be complex enough for comparison.

Blackbox is software, written by D-Wave, which implements the QBoost algorithm. Though Blackbox has been evaluated before, evaluations have primarily focused on solving intractable problems. In [3], Blackbox is used with a timeout of 30 minutes and using at most 10^7 state evaluations. We would like to tighten this bound, both because the standard algorithms we compare against complete in under a second, and because our allotted time on the D-Wave machine is limited. A pilot study, based on minimizing the sum function for a number of variables, gave guidance on setting these parameters. In particular, we found that the D-Wave was incapable of finding an optimal solution to a problem with 12 variables given the default timeout of 10 seconds. This means that our D-Wave classifier will likely need more time to build than standard classifiers. We press

on, in case the accuracy increase justifies the increased time cost of the D-Wave system.

Based on the pilot study, we collect the top 16 3-grams from the benign executables and 16 from the malicious executables to use as features. We then create a vector of weak classifiers: the first 32 weak classifiers classify instances in which the 3-gram is present as malware, and the next 32 weak classifiers classify instances in which the 3-gram is present as benign. For a given selection of weak classifiers, we calculate the loss using Equation 2 and return this loss as the value of the objective function. For comparison, we use the same features to create multiple classifiers in WEKA, a popular tool for machine learning.

3 Results

We wish to test the effectiveness of the malware classifier produced by the D-Wave machine. To do so, we perform 10-fold cross-validation: we build the classifier on each set of 9 folds and evaluate the D-Wave classifier on the remaining fold, and then average these accuracies together. For each fold, we record the accuracy of the classifier, the time to build the classifier, and the number of features selected in the final classifier.

We compare the D-Wave classifier to the same classifier using the D-Wave simulator, which is classical in nature. We also compare to three classical models built using WEKA: Adaboost, J48 (Decision Tree), and Random Forest. We choose Adaboost as it and QBoost have been compared before, and we chose J48 and Random Forest as they have been shown to produce decent results in the field of malware analysis. Again, we expect accuracies lower than state-of-the-art classification systems, as we have restricted the classification problem significantly in order to embed it onto the D-Wave chip. Table 1 compares the accuracies and time taken to build each of the different classifiers. Unlike in [4], the timing in Table 1 for the D-Wave machine is underreported; we chose to only include time that the D-Wave was running to remove the latency caused by the network, and thus the time that the classical system was creating D-Wave instructions is not accounted for in the table.

We were able to achieve a cross-validation accuracy of 80% using the actual D-Wave machine, which outperformed WEKA's Adaboost and underperformed WEKA's RandomForest. However, given the substantial time to build the classifier, we were not able to perform multiple runs on the D-Wave machine to know whether this run was an outlier; as such, these accuracies should not be used directly as benchmarks for comparison. This accuracy comes at a great cost: the D-Wave classifier took roughly 10,000 times as long to create. Further, the standard machine learning algorithms scale, but the D-Wave algorithm must be greatly restricted in order to create a classifier in a reasonable amount of time.

It is interesting to note that the simulator needed less time to train than the actual chip. This might be because the simulator uses the maximum number of nodes in the Chimera graph, whereas the actual chip has

Classifier	Cross-fold Accuracy	Average Time to Build (Seconds)
D-Wave	0.80	536.32
D-Wave Simulator	0.802	451.62
Adaboost	0.768	0.02
J48	0.796	0.03
RandomForest	0.814	0.05

Table 1: Cross-fold accuracy and time to build classifiers.

dead qubits it must work around.

4 Conclusions

We have shown it is possible to create a malware classifier using a D-Wave machine along with the Blackbox embedding software. Furthermore, we have shown this classifier has 10-fold cross-validation accuracy comparable to classical classifiers using the same features. However, there is significant overhead in building such a classifier using Blackbox. Our results show that, at this time and for this domain, this method for classification does not outperform other methods enough to justify the cost. Whether the D-Wave will outpace classical speedup remains to be seen.

There are, however, potentially other uses for this method. We noticed during our experiment that the D-Wave often achieved the same accuracy as the classical methods, but using a fewer number of features. This is consistent with previous work.[2] It is possible that Blackbox is best suited for preprocessing of data. The question of why the D-Wave and the simulator seem to use less features in their classifiers should be investigated further; exploiting this characteristic may provide a new use for the system in feature and instance selection.

Some interesting paths for malware research are introduced in this paper as well. There are few public standards for classification in the malware domain. There are several malware datasets (even if potentially flawed), but there is no standard for benign datasets, and the features for classification are generally not public. Creation of a standard benchmarking corpus of malicious and benign executables is long overdue.

References

[1] Binary classification using a d-wave one system. `http://www.dwavesys.com/en/dev-tutorial-qbc.html`. Accessed: 2013-06-13.

[2] V. S. Denchev. *Binary classification with adiabatic quantum optimization.* PhD thesis, Purdue University, 2013.

[3] C. C. McGeoch and C. Wang. Experimental evaluation of an adiabiatic quantum system for combinatorial optimization. In *Proceedings of the ACM International Conference on Computing Fron-*

tiers, CF '13, pages 23:1–23:11, New York, NY, USA, 2013. ACM.

[4] J. Seymour. Quantum classification of malware. Master's thesis, University of Maryland, Baltimore County, 2014.

[5] J. Seymour and C. Nicholas. Overgeneralization in feature set selection for classification of malware. Technical report, UMBC CSEE Technical Report, TR-CS-14-06, September, 2014, 2014.