RESEARCH ARTICLE

Nonnegative/Binary matrix factorization with a D-Wave quantum annealer

Daniel O'Malley[1,2]*, Velimir V. Vesselinov[1], Boian S. Alexandrov[3], Ludmil B. Alexandrov[4,5]

1 Computational Earth Science (EES-16), Los Alamos National Laboratory, Los Alamos, New Mexico, United States of America, **2** Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Maryland, United States of America, **3** Physics and Chemistry of Materials (T-1), Los Alamos National Laboratory, Los Alamos, New Mexico, United States of America, **4** Theoretical Biology and Biophysics (T-6), Los Alamos National Laboratory, Los Alamos, New Mexico, United States of America, **5** Department of Cellular and Molecular Medicine, University of California San Diego, San Diego, California, United States of America

* omalled@lanl.gov

OPEN ACCESS

Citation: O'Malley D, Vesselinov VV, Alexandrov BS, Alexandrov LB (2018) Nonnegative/Binary matrix factorization with a D-Wave quantum annealer. PLoS ONE 13(12): e0206653. https://doi.org/10.1371/journal.pone.0206653

Editor: Ivan Olier, Liverpool John Moores University, UNITED KINGDOM

Received: October 27, 2017

Accepted: October 14, 2018

Published: December 10, 2018

Data Availability Statement: All software and data underlying the results are available at https://github.com/lanl/Origami.jl.

Funding: The authors acknowledge the support from a Los Alamos National Laboratory (LANL) Laboratory Directed Research and Development project 20180481ER and a LANL Information Science & Technology Rapid Response project. DO was partially supported by the National Nuclear Security Administration's Advanced Simulation and Computing program. VVV was partially supported by the DiaMonD project (An Integrated

Abstract

D-Wave quantum annealers represent a novel computational architecture and have attracted significant interest. Much of this interest has focused on the quantum behavior of D-Wave machines, and there have been few practical algorithms that use the D-Wave. Machine learning has been identified as an area where quantum annealing may be useful. Here, we show that the D-Wave 2X can be effectively used as part of an unsupervised machine learning method. This method takes a matrix as input and produces two low-rank matrices as output—one containing latent features in the data and another matrix describing how the features can be combined to approximately reproduce the input matrix. Despite the limited number of bits in the D-Wave hardware, this method is capable of handling a large input matrix. The D-Wave only limits the rank of the two output matrices. We apply this method to learn the features from a set of facial images and compare the performance of the D-Wave to two classical tools. This method is able to learn facial features and accurately reproduce the set of facial images. The performance of the D-Wave shows some promise, but has some limitations. It outperforms the two classical codes in a benchmark when only a short amount of computational time is allowed (200-20,000 microseconds), but these results suggest heuristics that would likely outperform the D-Wave in this benchmark.

Introduction

Single-core computational performance relentlessly improved for decades, but recently that progress has begun to slow [1]. As a result, alternative computational architectures have sprung up including multi-core processors [2], graphic processing units [3], neuromorphic computing [4], and application-specific integrated circuits to name a few. Here we explore the use of another new architecture: quantum annealing [5]. In particular, we utilize the form of quantum annealing realized with D-Wave hardware [6, 7], which has been used to solve a

Multifaceted Approach to Mathematics at the Interfaces of Data, Models, and Decisions, U.S. Department of Energy Office of Science, Grant #11145687). LBA was supported by a LANL J. Robert Oppenheimer Fellowship. The funders had no role in the study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Competing interests: The authors have declared that no competing interests exist.

number of machine learning problems [8–10]. We focus on a machine learning problem based on matrix factorizations, and describe an algorithm for computing these matrix factorizations that leverages D-Wave hardware. We apply the algorithm to learn features in a set of facial images.

Matrix factorization has been a fundamental problem in applied mathematics for many years with methods such as singular value decomposition, LU decomposition and QR decomposition forming the building blocks of many computational algorithms [11, 12]. More recently, low-rank matrix factorization methods have been utilized to extract features from datasets in unsupervised machine learning analyses. In the context of facial images, matrix factorization has been used to extract "eigenfaces" [13] and parts (e.g., mouths, noses, eyes, *etc*.) of faces [14]. These methods enable each face in a large database of facial images to be accurately and concisely represented as a linear combination of a small number of vectors. The coefficients of these vectors can be used as low-dimensional inputs for facial recognition algorithms [13, 15, 16]. This discussion has focused on using matrix factorization to extract features from facial images and we will apply our method to a database of facial images.

From a machine learning perspective, matrix factorization can be used for dictionary learning [17] or to aid feature selection [18]. There are a variety of feature selection methods [19–23] that can be used to help supervised machine learning methods avoid overfitting and the curse of dimensionality (since they reduce the dimension). In the context of image analysis, machine learning is used for a variety of tasks including pattern recognition [24], object detection [25], and deconvolution [26, 27] among many other possibilities. The matrix factorization methods that we describe here can be considered as a member of this set of machine learning methods for image analysis. However, matrix factorization can be used to extract features from a wide variety of datasets and is largely agnostic to the content of the data. That is, it is not restricted to image analysis.

There is an ongoing back-and-forth regarding whether or not D-Wave's hardware provides performance benefits over classical single-core computing [28–31]. Here, we benchmark the performance of two classical approaches against the performance of the D-Wave. Much of the debate about the D-Wave's performance has centered on problems that are custom-tailored to fit D-Wave's hardware. A component of the matrix factorization problems that we study here can be solved on the D-Wave, but is not customized for the D-Wave and represents a real rather than synthetic problem. The D-Wave outperforms the two classical approaches in a benchmark, but the performance of the two classical approaches suggests a heuristic that could be implemented on a classical computer to outperform the D-Wave in this benchmark. This mixed result on the performance of the D-Wave compared to classical tools is in agreement with the most recent results [31] showing that, even for custom-tailored problems, the D-Wave does not outperform the best classical heuristics. Despite this, our results show that the D-Wave can outperform very good classical tools when only a short amount of computational time is allotted to solve these real-world problems. We also provide a discussion of how future improvements to the algorithm presented here and D-Wave's hardware could improve performance for these matrix factorization problems.

The remainder of this manuscript is organized as follows. First, we describe the methods used to solve the matrix factorization problems and perform the benchmarking. Next, we describe the results that we obtained in solving a matrix factorization problem for a set of facial images and the benchmark results. Finally, we discuss the results and indicate how future developments might improve the performance of the D-Wave for this problem.

Methods

We seek to represent an $n \times m$ matrix, V, as the product of two matrices, W and H, where W is an $n \times k$ matrix and H is a $k \times m$ matrix. That is, we wish to find W and H such that

$$V \approx WH \tag{1}$$

We impose constraints on W and H. In particular, the components of W must be nonnegative (i.e., $W_{ij} \geq 0$) and the components of H must be binary (i.e., $H_{ij} \in \{0, 1\}$). Since W is a nonnegative matrix and H is a binary matrix, we describe this matrix factorization as Nonnegative/Binary Matrix Factorization (NBMF). This is in contrast to Nonnegative Matrix Factorization (NMF) [14] where H is allowed to take on any nonnegative value, not just 0 or 1. To satisfy Eq 1, we utilize an alternating least squares algorithm [32, 33] (see Algorithm 1).

Algorithm 1 A high-level description of the alternating least squares algorithm that we employ to perform NBMF.

```
Require: $V$, $k$
Ensure: $W$, $H$
  Randomly initialize each element of $H$ to be either 0 or 1
  while not converged do
    $W := \arg\min_{X \in \mathbb{R}^{+n \times k}} ||V - XH||_F + \alpha ||X||_F$
    $H := \arg\min_{X \in \{0,1\}^{k \times m}} ||V - WX||_F$
  end while
```

From algorithm 1, we focus on utilizing a D-Wave 2X quantum annealer to compute

$$H = \underset{X \in \{0,1\}^{k \times m}}{\arg\min} ||V - WX||_F \tag{2}$$

where $||\cdot||_F$ is the Frobenius norm. Note that Eq 2 can be solved by solving a set of independent optimization problems (one for each column of H). This is because the variables in the i^{th} column of H impact only the i^{th} column of WX, and the variables outside the i^{th} column of H do not impact the i^{th} column of WX—see S1 Appendix for details. That is, if we denote the i^{th} columns of H and V by H_i and V_i, respectively, then

$$H_i = \underset{\mathbf{q} \in \{0,1\}^k}{\arg\min} ||V_i - W\mathbf{q}||_2 \tag{3}$$

for $i = 1, 2, \ldots, m$. This means that we can solve for H by solving a series of linear least squares problems in binary variables. This type of problem can be readily solved on D-Wave hardware [34], as long as the number of binary variables is small. Eq 2 involves km binary variables, but Eq 3 involves only k binary variables. Since the D-Wave 2X imposes severe limitations on the number of binary variables that can be dealt with, this reduction in the number of variables is crucial. In practice, this means that relatively large datasets can be analyzed (i.e., n and m can be large). Since the number of variables the D-Wave works with at any given time is determined only by k, the D-Wave imposes restrictions only on the number of features (i.e., k). This is not a major imposition, because it is usually desirable for k to be small. That is, the rank of the factorization is usually supposed to be low.

We also compare the performance of the D-Wave to solve Eq 3 with two classical approaches. One utilizes the JuMP [35] modeling language and a mathematical programming tool called Gurobi [36]. The other, called qbsolv [37], utilizes is an efficient, open-source implementation of tabu search [38, 39]. Qbsolv can be used in two different modes: a tabu search which uses purely classical computing and another which uses a combination of classical computing and the D-Wave. We use only qbsolv's purely classical tabu mode here. The performance is compared using a cumulative time-to-targets benchmark that is a variation of the time-to-target benchmark [40]. In the course of executing algorithm 1, Eq 3 must be solved many times for different values of i and W. Each time this equation is solved, the D-Wave is given a fixed number of annealing cycles to minimize $||V_i - W\mathbf{q}||_2$ (we look at examples where the number is fixed at 10, 10^2, 10^3, and 10^4). Each annealing cycle results in one approximate solution to Eq 3, and we denote the best of these approximate solutions by H_i^*. The best solution, is used to generate a target value for the objective function, $||V_i - WH_i^*||_2$, that the classical approaches (qbsolv and Gurobi) must match. The cumulative time-to-targets benchmark computes the cumulative amount of time it takes for qbsolv or Gurobi to find a solution that is at least as good (in terms of $||V_i - WH_i||_2$) as the best solution found by the D-Wave for each instance of Eq 3 that is encountered in executing algorithm 1. This cumulative time-to-targets is then compared with the amount of annealing time used by the D-Wave. If qbsolv or Gurobi take more than 10 minutes to reach an individual target, the time to reach that target is set to 10 minutes. The 10 minute limit is an expedient that enables the analysis to be run in a reasonable amount of time. Gurobi never reached the 10 minute limit, but qbsolv did in a number of cases.

Programming the D-Wave

D-Wave quantum annealers deal natively with quadratic, unconstrained, binary optimization (QUBO) problems [41]. These problems are associated with objective functions that have the form

$$f(\mathbf{q}) = \sum_i a_i q_i + \sum_{i<j} b_{ij} q_i q_j \tag{4}$$

where $\mathbf{q} = (q_1, q_2, \ldots, q_n)$. One might think of a 0^{th} order approximation of the D-Wave's behavior as being that each anneal returns a vector, $\mathbf{q}$, so that $f(\mathbf{q})$ is minimized. A 1^{st} order approximation of the D-Wave's behavior is that each anneal returns a sample, $\mathbf{q}$, from a Boltzmann distribution where $f(\mathbf{q})$ is the energy. Both of these approximations are inexact, but highlight the basic behavior of the D-Wave: each annealing cycle returns a sample, $\mathbf{q}$, which tends to make $f(\mathbf{q})$ small. Eq 3 can be readily put into the form of a QUBO by setting [34]

$$a_j = \sum_l W_{lj}(W_{lj} - 2V_{ij}) \tag{5}$$

$$b_{jk} = 2\sum_l W_{lj} W_{lk} \tag{6}$$

Having reformulated Eq 3 in this way, the quadratic coefficients in the QUBO, b_{ij}, are generally all nonzero. However, the D-Wave's hardware imposes sparsity constraints on the b_{ij}. These constraints can be overcome via embedding [42–44], where multiple physical qubits are used to represent a single binary variable. The D-Wave's hardware is composed of bits that are coupled via a "Chimera" graph, $\mathcal{C}_{M,N,L}$, which consists of an M-by-N grid of $K_{L,L}$ bipartite graphs [44]. The design for the D-Wave 2X is based on $\mathcal{C}_{12,12,4}$. It has been shown that a complete graph with LM nodes can be embedded in $\mathcal{C}_{M,M,L}$ using $M + 1$ physical bits per binary variable [44].

Since the b_{ij}'s in our problems are never exactly zero, a complete graph with the number of nodes equal to the number of binary variables must be embedded in the graph imposed by the D-Wave 2X chip. If a D-Wave 2X chip had no defects (i.e., if all the qubits and couplers were operational), the maximum number of binary variables that could be used for these problems is 49 (if the matrix formed by the elements b_{ij} had some natural sparsity to it, this number could increase). However, some of the qubits and couplers on these chips are not available for use, and as the number of binary variables increases, the number of physical qubits required to represent each variable increases. Since using a larger number of physical qubits to represent a single binary variable is associated with poor performance, we limit our study to 35 binary variables. In this case, we found an embedding where each binary variable is represented by at most 19 physical qubits.

We analyze the algorithmic complexity of approximately solving Eq 2 using the algorithm described here. Approximately solving Eq 2 requires the solution of m instances of Eq 3. Approximately solving each instance of Eq 3 comes with two computational components—one on the classical computer and another on the quantum annealer. The computational time on the quantum annealer is determined by the number of samples, N, obtained from the quantum annealer. Note that N is fixed throughout the execution of the algorithm. The total computational time associated with the quantum annealer is thus $O(mN)$.

The time required to use a classical computer to formulate the input to the quantum annealer can be understood via Eqs 5 and 6. The sum in Eq 5 contains n terms and this sum must be computed k times (to fill out the vector elements of the QUBO, a_j) for each instance of Eq 3, resulting in a complexity of $O(nk)$. The sum in Eq 6 contains n terms and this sum must be computed k^2 times (to fill out the matrix elements of the QUBO, b_{ij}) for each instance of Eq 3, resulting in a complexity of $O(nk^2)$. Therefore, for each instance of Eq 3, the computational complexity on the classical computer is $O(nk^2)$. Since Eq 3 must be solved approximately m times in order to approximately solve Eq 2, the computational complexity on the classical computer for approximately solving Eq 2 is $O(mnk^2)$.

The total computational complexity (including both the quantum annealing time and classical time) is thus $O(mnk^2 + mN)$. This is in contrast to exact algorithms [45] that can perform the factorization in $O(nk2^k + mnk + mk^2)$. The key trade-off here is to exchange an approximate solution that can be obtained in polynomial time in k for an exact solution whose run time grows exponentially with k. Once the rank of the factorization, k, becomes sufficiently large the exact algorithm becomes intractable while the approximate algorithm remains performant.

Results

We analyzed the same set of 2,429 facial images [46] (available at http://www.ai.mit.edu/courses/6.899/lectures/faces.tar.gz) that was previously analyzed to learn the parts of faces using nonnegative matrix factorization [14]. To represent this set of images as a matrix, each column of the matrix corresponds to a different facial image and each row corresponds to a different pixel in the image. That is, V_{ij} contains the value of pixel i from image j. Fig 1 shows the features that were learned using algorithm 1 with 10,000 anneals per solve of Eq 3 and how those features are used to reconstruct the image of a face. Some of the features in Fig 1 may appear to be all black, but they actually contain subtle features such as a bright spot in the lower-left corner or a shiny cheek/nose (see Fig 2). Unlike NMF where the parts of faces are learned [14], the features learned by NBMF are holistic. One can view NBMF as being a method that is somewhere in between the NMF and vector quantization methods considered in [14]. Like NMF, it imposes the nonnegativity constraints on W, but, unlike NMF, imposes binary constraints on H. Vector quantization imposes binary constraints on H, but adds an additional constraint that each column of H can only contain one nonzero entry. This additional constraint causes vector quantization to learn holistic, prototypical faces. NBMF appears to learn features that are holistic like the features that come out of vector quantization, but, unlike vector quantization, NBMF's features are not necessarily prototypical faces. Many of

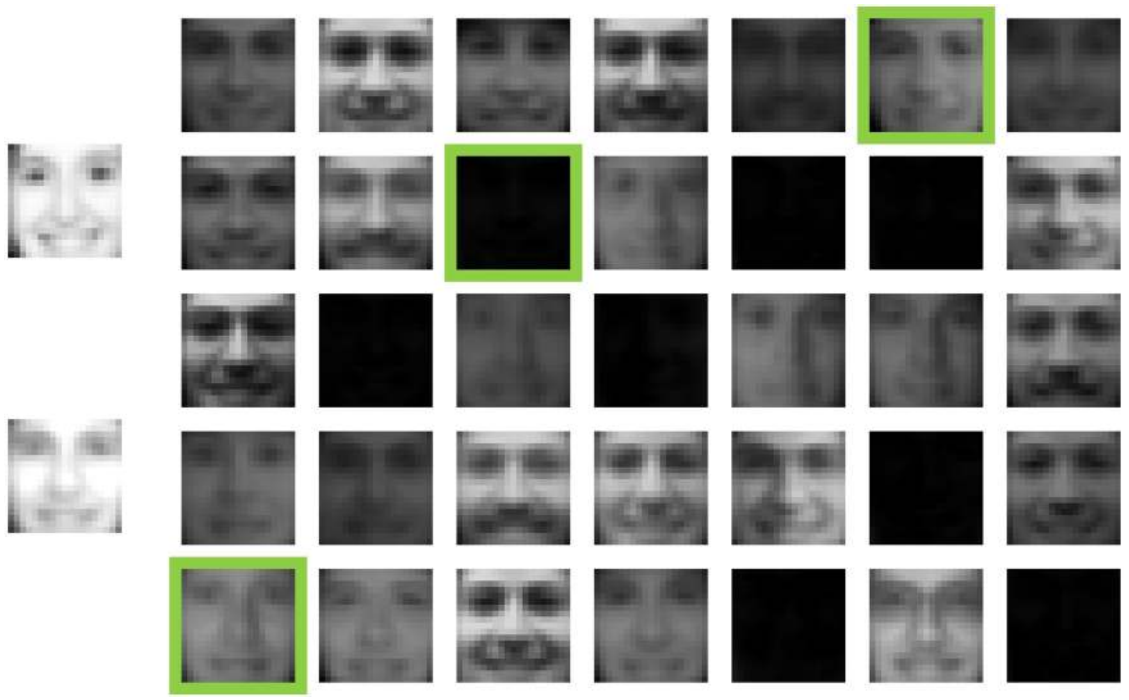

Fig 1. Face image reconstruction using features learned by NBMF. The five-by-seven matrix of images on the right shows the features that were learned. The two images on the left show the original image (top) and the reconstruction (bottom). The reconstruction is obtained by summing the features that are boxed in green. Note that although some of the features appear to be all black, they actually contain facial features that are small in magnitude (black corresponds to 0, white corresponds to 1).

https://doi.org/10.1371/journal.pone.0206653.g001

them appear ghostly and the ones that are mostly black are even more subtle. As in Fig 1, these subtle and ghostly features can be combined to reproduce a face.

The NBMF method employed here has some advantages and disadvantages compared to NMF. One advantage of NBMF is that H is more sparse when NBMF is used than when NMF is used. Analyzing this database of facial images using 35 features ($k = 35$), the H produced by NBMF is approximately 85% sparse (i.e., 85% of the elements of H are zero) whereas the one produced by NMF is approximately 13% sparse. Further, the storage requirements for each component of H are less for NBMF (1 bit) than NMF (e.g., 64 bit floating point numbers were used here). A disadvantage is that $||V - WH||_F$ is larger for NBMF than NMF. For this database of facial images, $||V - WH||_F$ using NMF was about 46% of this norm when using NBMF. In layman's terms, NMF had about half as much error as NBMF. NMF has the additional advantage that W is about 41% sparse, whereas W is dense for NBMF in the images analyzed here.

Fig 3 shows the results of the cumulative time-to-targets benchmark. The cumulative time-to-targets for qbsolv always exceeds the cumulative annealing time by a factor of 20-50 depending on the number of anneals. In the test with 10, 100, and 1,000 annealing cycles, Gurobi's cumulative time-to-targets exceeds the cumulative annealing time by factors of about 61, 7 and 1.2, respectively. In the test with 10,000 anneals, Gurobi's cumulative time-to-targets was less than the cumulative annealing time by a factor of about 6.4.

Gurobi and qbsolv show different performance trends in this benchmark that we explore in more detail with Fig 4. Qbsolv's performance is characterized by frequently reaching the target before the annealing time for individual problems, but, when it fails to reach the target before the annealing time, it can take a comparatively long time to reach the target. These problems where qbsolv takes a very long time to reach the target make up a large portion of

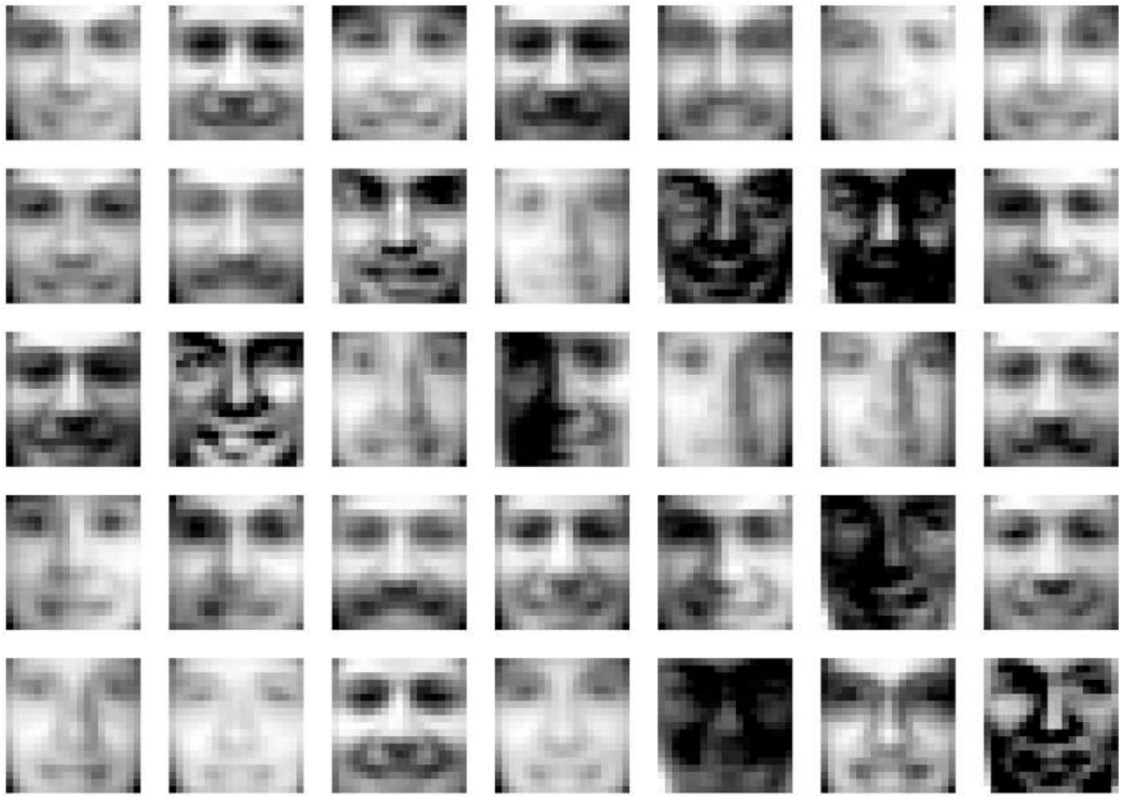

Fig 2. The same features as shown in Fig 1 are shown. Here they are rescaled to maximize contrast so that the darkest pixel is black and the brightest pixel is white.

https://doi.org/10.1371/journal.pone.0206653.g002

the cumulative time-to-targets. As the D-Wave takes more and more samples, the fraction of the problems where the time-to-target exceeds the annealing time increases, as can be seen from the increasing number of red dots above the orange line in Fig 4. When 10 anneals are used, qbsolv's time-to-target exceeds the annealing time in less than 1% of the problems, whereas when 10,000 anneals are used, qbsolv's time-to-target exceeds the annealing time in more than 28% of the problems. Gurobi rarely has problems for which it takes a very long time to reach the target set by the D-Wave. In the more than 90,000 problems considered here, Gurobi took more than a second to reach the target set by D-Wave only 21 times with the maximum time being 13.6 seconds. By contrast, qbsolv took more than a second to reach the target 5,509 times and hit the 10 minute maximum in 24 cases. While Gurobi rarely takes a long time to reach the target set by the D-Wave, it also rarely solves the problems very quickly. Gurobi reached the target set by the D-Wave in under a millisecond in only 57 cases, whereas qbsolv did this in almost 80,000 cases.

In short, Gurobi's performance is characterized by consistency and qbsolv's performance is characterized by solving many problems very quickly and some problems slowly. Qbsolv's tabu search implements a lean heuristic: "no U-turn." That is, qbsolv performs a random walk through solution space and attempts to avoid visiting the same solution twice. The leanness of this heuristic enables qbsolv to find good solutions quickly, but relying on a single heuristic leaves it prone to longer run times when the heuristic does not work well. Gurobi, on the other hand, implements a wide array of heuristics. Gurobi's complexity prevents it from achieving the very fast solution times obtained by qbsolv, but using a large set of heuristics enables it to solve more problems relatively quickly.

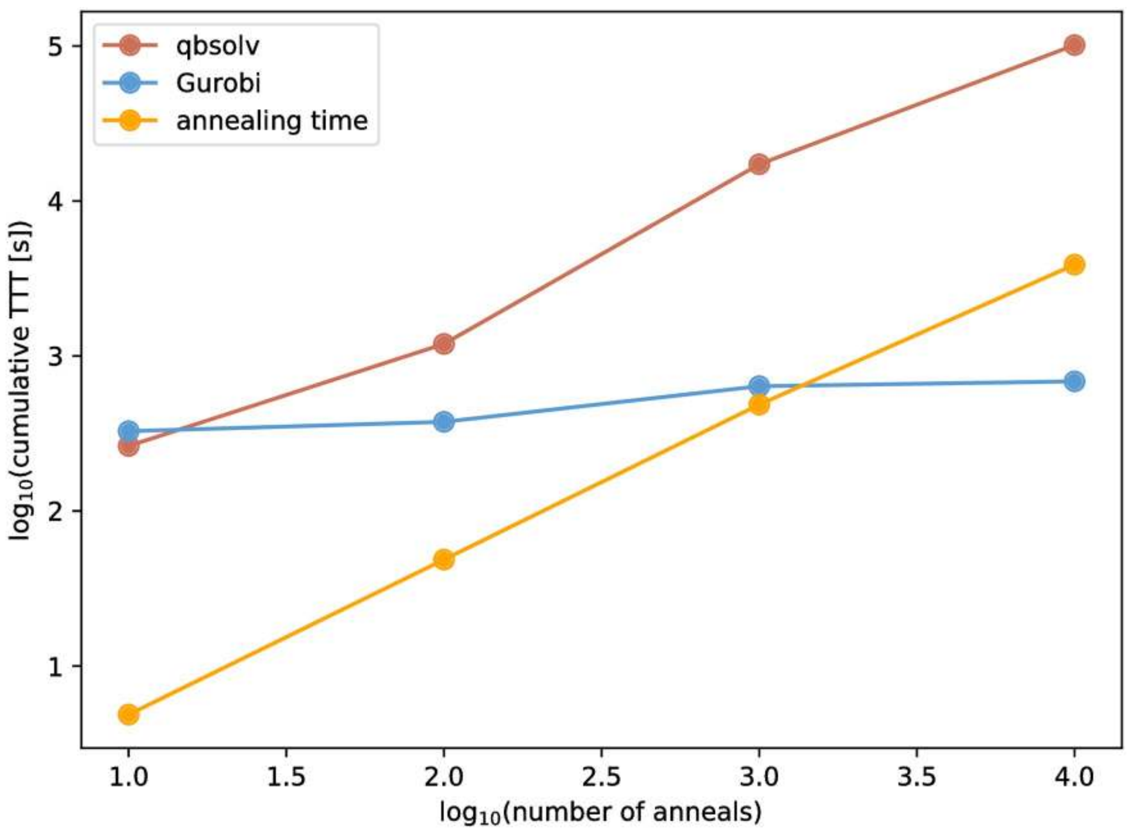

Fig 3. Cumulative time-to-targets for qbsolv (red) and Gurobi (blue) as a function of the number of annealing cycles executed by the D-Wave. When the number of anneals is 10, 100, or 1,000, the cumulative time-to-targets for both qbsolv and Gurobi exceeds the cumulative annealing time (orange). When the number of anneals is 10,000, the cumulative time-to-targets for qbsolv exceeds the cumulative annealing time, but Gurobi's cumulative time-to-target is less than the cumulative annealing time. Note that in each of the 10, 100, and 1,000 anneal cases, 24,290 QUBOs were solved whereas only 19,432 QUBOs were solved in the 10,000 anneal cases. This was caused by earlier termination of the NBMF (algorithm 1).

https://doi.org/10.1371/journal.pone.0206653.g003

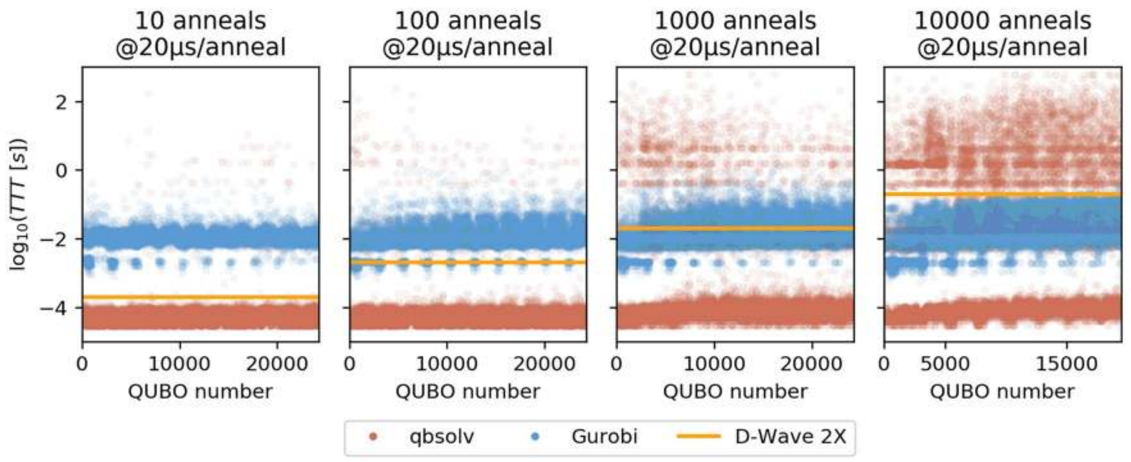

Fig 4. The time-to-target for each instance of Eq 3 is shown for qbsolv (red dots), Gurobi (blue dots) in comparison to the annealing time used by the D-Wave (orange line). A different number of anneals was used in each execution of the NBMF (algorithm 1) ranging from 10 to 10,000.

https://doi.org/10.1371/journal.pone.0206653.g004

Discussion

We have identified a performance regime (fast solutions that are good, but not necessarily optimal) where the D-Wave, at least by the benchmark used here, outperforms two classical codes in a problem that is not custom-tailored to the D-Wave as other problems have been [47, 48]. This was demonstrated by the cases we studied with 10, 100, and 1,000 anneals where the cumulative time-to-targets for both Gurobi and qbsolv exceeded the annealing time used by the D-Wave. We emphasize that these results do not demonstrate any sort of quantum supremacy. In fact, they suggest a classical heuristic which would likely outperform the D-Wave: run qbsolv for a short period and if it fails to match the target, then switch to Gurobi. This would leverage qbsolv's ability to frequently outperform the D-Wave, and Gurobi's ability to never lose too badly to the D-Wave. Other benchmarks could also be used that might cast the D-Wave in a more negative light. For example, allowing the D-Wave to set the targets in the cumulative time-to-targets benchmark might be an advantage, and reversing the roles (i.e., allowing Gurobi or qbsolv to set the targets in a fixed amount of computational time) might produce different results.

Given the remarkable performance improvements over many generations of classical microprocessors [1] and the impressive algorithmic improvements in mixed-integer programming tools like Gurobi [49] over the past several decades, it is surprising that D-Wave's third generation hardware and our straight-forward algorithm can be competitive at all. In the series of four chips that D-Wave has released, the number of qubits has approximately doubled from one generation to the next while the number of couplers per qubit has remained essentially unchanged. D-Wave's fifth generation chip is expected to at least double the number of couplers per qubit [7, 50]. While it is not known what form this chip will take, a design that would achieve this could be obtained by replacing the $\mathcal{C}_{16,16,4}$ design (which is used by the D-Wave 2000Q) with a $\mathcal{C}_{16,16,12}$ design. We hypothesize that adding more couplers per qubit would have a significant, positive impact on the performance of the D-Wave for the problems we consider here. If the number of binary variables were fixed at 35, it would result in fewer physical qubits being used to represent each binary variable. For example, a complete graph with 35 nodes could be embedded in $\mathcal{C}_{3,3,12}$ (which is a subgraph of $\mathcal{C}_{16,16,12}$) using 4 physical bits per variable. This is much less than the 10 physical bits per variable needed to embed in the $\mathcal{C}_{12,12,4}$ graph that forms the basis for the D-Wave 2X used here. Because of this, if the size of the problem is fixed, we hypothesize that D-Wave's fifth generation chip would provide much better performance than the D-Wave 2X. Increasing the number of couplers per qubit would also enable the possibility of going well beyond the 35 binary variables considered here without using an excessive number of physical qubits to represent each binary variable. That is, we would expect D-Wave's fifth generation chip to be capable of solving NBMF problems with more features than can be solved with the third generation chip that we have used here. For example, a complete graph with 192 nodes can be embedded in $\mathcal{C}_{16,16,12}$ whereas it is neither possible to embed a 66 node complete graph in $\mathcal{C}_{16,16,4}$ (D-Wave 2000Q) nor a 50 node complete graph in $\mathcal{C}_{12,12,4}$ (D-Wave 2X) [44]. In addition to the improvements that are anticipated in future D-Wave hardware, there are techniques that could be utilized to potentially improve our algorithm. Symmetries could be exploited (e.g., via spin reversal transformations or symmetry in the complete graph that we embed in D-Wave's hardware graph), the strength of chains arising from the embedding process could be optimized, hardware biases could be learned [51] and exploited, and the embeddings could be improved by setting quadratic coefficients in the QUBO that are approximately zero to exactly zero. Exploring and exploiting these techniques is beyond the scope of this manuscript, but we expect that some or all of them could provide significant algorithmic improvements.

The relatively short computational time (from $200\mu s$ up to $20,000\mu s$) regime where the D-Wave outperforms qbsolv and Gurobi in our benchmark could be important for big data problems. For example, when learning the features of a large set of images (much larger than the 2,429 images considered here), only a small amount of computational time may be available to solve each problem given in Eq 3. However, in order for the D-Wave's performance advantages in this regime to be beneficially leveraged, there must be significant performance improvements in the time it takes to get problems into and solutions out of the D-Wave. At present, input and output is performed via the HTTPS internet protocol. The performance of this bottleneck can clearly be improved. Beyond this, there are other potential bottlenecks that could prevent the D-Wave from being more performant for these types of problems, such as the D-Wave's programming time ("qpu_programming_time", as reported by D-Wave's software) which was typically about 15 milliseconds in the problems analyzed here. If this programming time remains constant as new D-Wave chips become available, then there would not be much advantage to solving problems where the total annealing time is much less than 15 milliseconds. This could hinder the D-Wave's performance in the short computational time regime we have identified where it outperforms Gurobi and qbsolv in the cumulative time-to-targets benchmark.

In summary, we have demonstrated that this NBMF algorithm can leverage the D-Wave 2X as a key component in an unsupervised machine learning analysis. Getting performance from the D-Wave 2X that is competitive with advanced classical tools on a real-world problem is a significant step forward on the journey towards practical quantum annealing. While there is still much work to be done to make these quantum annealers of practical use for this type of problem, our performance results give a glimmer of hope that this may someday be the case.

Supporting information

S1 Appendix.
(PDF)

Author Contributions

Conceptualization: Daniel O'Malley, Velimir V. Vesselinov, Boian S. Alexandrov, Ludmil B. Alexandrov.

Funding acquisition: Daniel O'Malley.

Investigation: Daniel O'Malley.

Methodology: Daniel O'Malley.

Software: Daniel O'Malley.

Visualization: Daniel O'Malley.

Writing – original draft: Daniel O'Malley.

Writing – review & editing: Daniel O'Malley, Velimir V. Vesselinov, Boian S. Alexandrov, Ludmil B. Alexandrov.

References

1. Danowitz A, Kelley K, Mao J, Stevenson JP, Horowitz M. CPU DB: recording microprocessor history. Communications of the ACM. 2012; 55(4):55–63. https://doi.org/10.1145/2133806.2133822
2. Geer D. Chip makers turn to multicore processors. Computer. 2005; 38(5):11–13. https://doi.org/10.1109/MC.2005.160

3. Owens JD, Houston M, Luebke D, Green S, Stone JE, Phillips JC. GPU computing. Proceedings of the IEEE. 2008; 96(5):879–899. https://doi.org/10.1109/JPROC.2008.917757
4. Monroe D. Neuromorphic computing gets ready for the (really) big time. Communications of the ACM. 2014; 57(6):13–15.
5. Kadowaki T, Nishimori H. Quantum annealing in the transverse Ising model. Physical Review E. 1998; 58(5):5355. https://doi.org/10.1103/PhysRevE.58.5355
6. Johnson MW, Amin MH, Gildert S, Lanting T, Hamze F, Dickson N, et al. Quantum annealing with manufactured spins. Nature. 2011; 473(7346):194–198. https://doi.org/10.1038/nature10012 PMID: 21562559
7. Gibney E. D-wave upgrade: How scientists are using the world's most controversial quantum computer. Nature. 2017; 541(7638):447–448. https://doi.org/10.1038/541447b PMID: 28128267
8. Neven H, Denchev VS, Rose G, Macready WG. Training a binary classifier with the quantum adiabatic algorithm. arXiv preprint arXiv:08110416. 2008;.
9. Neven H, Denchev VS, Rose G, Macready WG. Training a large scale classifier with the quantum adiabatic algorithm. arXiv preprint arXiv:09120779. 2009;.
10. Pudenz KL, Lidar DA. Quantum adiabatic machine learning. Quantum information processing. 2013; 12(5):2027–2070. https://doi.org/10.1007/s11128-012-0506-4
11. Golub GH, Van Loan CF. Matrix computations. vol. 3. JHU Press; 2012.
12. Strang G. Computational science and engineering. vol. 791. Wellesley-Cambridge Press Wellesley; 2007.
13. Turk MA, Pentland AP. Face recognition using eigenfaces. In: Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on. IEEE; 1991. p. 586–591.
14. Lee DD, Seung HS. Learning the parts of objects by non-negative matrix factorization. Nature. 1999; 401(6755):788–791. https://doi.org/10.1038/44565 PMID: 10548103
15. Zhu P, Zhang L, Zuo W, Zhang D. From point to set: Extend the learning of distance metrics. In: Computer Vision (ICCV), 2013 IEEE International Conference on. IEEE; 2013. p. 2664–2671.
16. Zhu P, Zuo W, Zhang L, Shiu SCK, Zhang D. Image set-based collaborative representation for face recognition. IEEE transactions on information forensics and security. 2014; 9(7):1120–1132. https://doi.org/10.1109/TIFS.2014.2324277
17. Tosic I, Frossard P. Dictionary learning. IEEE Signal Processing Magazine. 2011; 28(2):27–38. https://doi.org/10.1109/MSP.2010.939537
18. Gupta MD, Xiao J. Non-negative matrix factorization as a feature selection tool for maximum margin classifiers. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. IEEE; 2011. p. 2841–2848.
19. Zhu P, Zuo W, Zhang L, Hu Q, Shiu SC. Unsupervised feature selection by regularized self-representation. Pattern Recognition. 2015; 48(2):438–446. https://doi.org/10.1016/j.patcog.2014.08.006
20. Zhu P, Hu Q, Han Y, Zhang C, Du Y. Combining neighborhood separable subspaces for classification via sparsity regularized optimization. Information Sciences. 2016; 370:270–287. https://doi.org/10.1016/j.ins.2016.08.004
21. Zhu P, Zhu W, Wang W, Zuo W, Hu Q. Non-convex regularized self-representation for unsupervised feature selection. Image and Vision Computing. 2017; 60:22–29. https://doi.org/10.1016/j.imavis.2016.11.014
22. Zhu P, Zhu W, Hu Q, Zhang C, Zuo W. Subspace clustering guided unsupervised feature selection. Pattern Recognition. 2017; 66:364–374. https://doi.org/10.1016/j.patcog.2017.01.016
23. Zhu P, Xu Q, Hu Q, Zhang C, Zhao H. Multi-label feature selection with missing labels. Pattern Recognition. 2018; 74:488–502. https://doi.org/10.1016/j.patcog.2017.09.036
24. Nasrabadi NM. Pattern recognition and machine learning. Journal of electronic imaging. 2007; 16(4):049901. https://doi.org/10.1117/1.2819119
25. Viola P, Jones M. Rapid object detection using a boosted cascade of simple features. In: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. vol. 1. IEEE; 2001. p. I–I.
26. Zuo W, Ren D, Zhang D, Gu S, Zhang L. Learning iteration-wise generalized shrinkage–thresholding operators for blind deconvolution. IEEE Transactions on Image Processing. 2016; 25(4):1751–1764.
27. Ren D, Zuo W, Zhang D, Xu J, Zhang L. Partial Deconvolution With Inaccurate Blur Kernel. IEEE Transactions on Image Processing. 2018; 27(1):511–524. https://doi.org/10.1109/TIP.2017.2764261

28. McGeoch CC, Wang C. Experimental evaluation of an adiabatic quantum system for combinatorial optimization. In: Proceedings of the ACM International Conference on Computing Frontiers. ACM; 2013. p. 23.
29. Isakov SV, Zintchenko IN, Rønnow TF, Troyer M. Optimised simulated annealing for Ising spin glasses. Computer Physics Communications. 2015; 192:265–271. https://doi.org/10.1016/j.cpc.2015.02.015
30. King J, Yarkoni S, Raymond J, Ozfidan I, King AD, Nevisi MM, et al. Quantum annealing amid local ruggedness and global frustration. arXiv preprint arXiv:170104579. 2017;.
31. Mandrà S, Katzgraber HG, Thomas C. The pitfalls of planar spin-glass benchmarks: Raising the bar for quantum annealers (again). arXiv preprint arXiv:170300622. 2017;.
32. Lin CJ. Projected gradient methods for nonnegative matrix factorization. Neural computation. 2007; 19(10):2756–2779. https://doi.org/10.1162/neco.2007.19.10.2756 PMID: 17716011
33. O'Malley D. Origami.jl; 2018. https://github.com/lanl/Origami.jl.
34. O'Malley D, Vesselinov VV. ToQ. jl: A high-level programming language for D-Wave machines based on Julia. In: High Performance Extreme Computing Conference (HPEC), 2016 IEEE. IEEE; 2016. p. 1–7.
35. Dunning I, Huchette J, Lubin M. JuMP: A modeling language for mathematical optimization. arXiv:150801982 [mathOC]. 2015;.
36. Optimization G. Gurobi Optimizer version 7.0.2; 2017. http://www.gurobi.com.
37. Systems DW. qbsolv; 2017. https://github.com/dwavesystems/qbsolv.
38. Glover F. Tabu search—part I. ORSA Journal on computing. 1989; 1(3):190–206. https://doi.org/10.1287/ijoc.1.3.190
39. Glover F. Tabu search—part II. ORSA Journal on computing. 1990; 2(1):4–32. https://doi.org/10.1287/ijoc.2.1.4
40. King J, Yarkoni S, Nevisi MM, Hilton JP, McGeoch CC. Benchmarking a quantum annealing processor with the time-to-target metric. arXiv preprint arXiv:150805087. 2015;.
41. McGeoch CC. Adiabatic quantum computation and quantum annealing: Theory and practice. Synthesis Lectures on Quantum Computing. 2014; 5(2):1–93. https://doi.org/10.2200/S00585ED1V01Y201407QMC008
42. Choi V. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. Quantum Information Processing. 2008; 7(5):193–209. https://doi.org/10.1007/s11128-008-0082-9
43. Choi V. Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design. Quantum Information Processing. 2011; 10(3):343–353. https://doi.org/10.1007/s11128-010-0200-3
44. Boothby T, King AD, Roy A. Fast clique minor generation in Chimera qubit connectivity graphs. Quantum Information Processing. 2016; 15(1):495–508. https://doi.org/10.1007/s11128-015-1150-6
45. Slawski M, Hein M, Lutsik P. Matrix factorization with binary components. In: Advances in Neural Information Processing Systems; 2013. p. 3210–3218.
46. MIT-CBCL. CBCL Face Database #1; 1999. http://cbcl.mit.edu/projects/cbcl/software-datasets/faces.tar.gz.
47. Rønnow TF, Wang Z, Job J, Boixo S, Isakov SV, Wecker D, et al. Defining and detecting quantum speedup. Science. 2014; 345(6195):420–424. https://doi.org/10.1126/science.1252319 PMID: 25061205
48. Denchev VS, Boixo S, Isakov SV, Ding N, Babbush R, Smelyanskiy V, et al. What is the Computational Value of Finite-Range Tunneling? Physical Review X. 2016; 6(3):031015. https://doi.org/10.1103/PhysRevX.6.031015
49. Bixby RE. A brief history of linear and mixed-integer programming computation. Documenta Mathematica. 2012; p. 107–121.
50. Hilton J. Systems Progress; 2016.
51. Lokhov AY, Vuffray M, Misra S, Chertkov M. Optimal structure and parameter learning of Ising models. arXiv preprint arXiv:161205024. 2016;.