Supplementary Information for "Quantum annealing versus classical machine learning applied to a simplified computational biology problem"

Richard Li,[1,2,3] Rosa Di Felice,[2,4,5] Remo Rohs,[1,2,4,6] and Daniel Lidar[1,3,4,7]

[1]*Department of Chemistry, University of Southern California, Los Angeles, California 90089, USA*

[2]*Computational Biology and Bioinformatics Program, Department of Biological Sciences, University of Southern California, Los Angeles, CA 90089, USA*

[3]*Center for Quantum Information Science & Technology, University of Southern California, Los Angeles, California 90089, USA*

[4]*Department of Physics and Astronomy, University of Southern California, Los Angeles, California 90089, USA*

[5]*Center S3, CNR Institute of Nanoscience, Via Campi 213/A, 41125 Modena, Italy*

[6]*Department of Computer Science, University of Southern California, Los Angeles, CA 90089, USA*

[7]*Department of Electrical Engineering, University of Southern California, Los Angeles, California 90089, USA*

Correspondence and requests for materials should be addressed to R.R. (rohs@usc.edu) or D.A.L. (lidar@usc.edu)

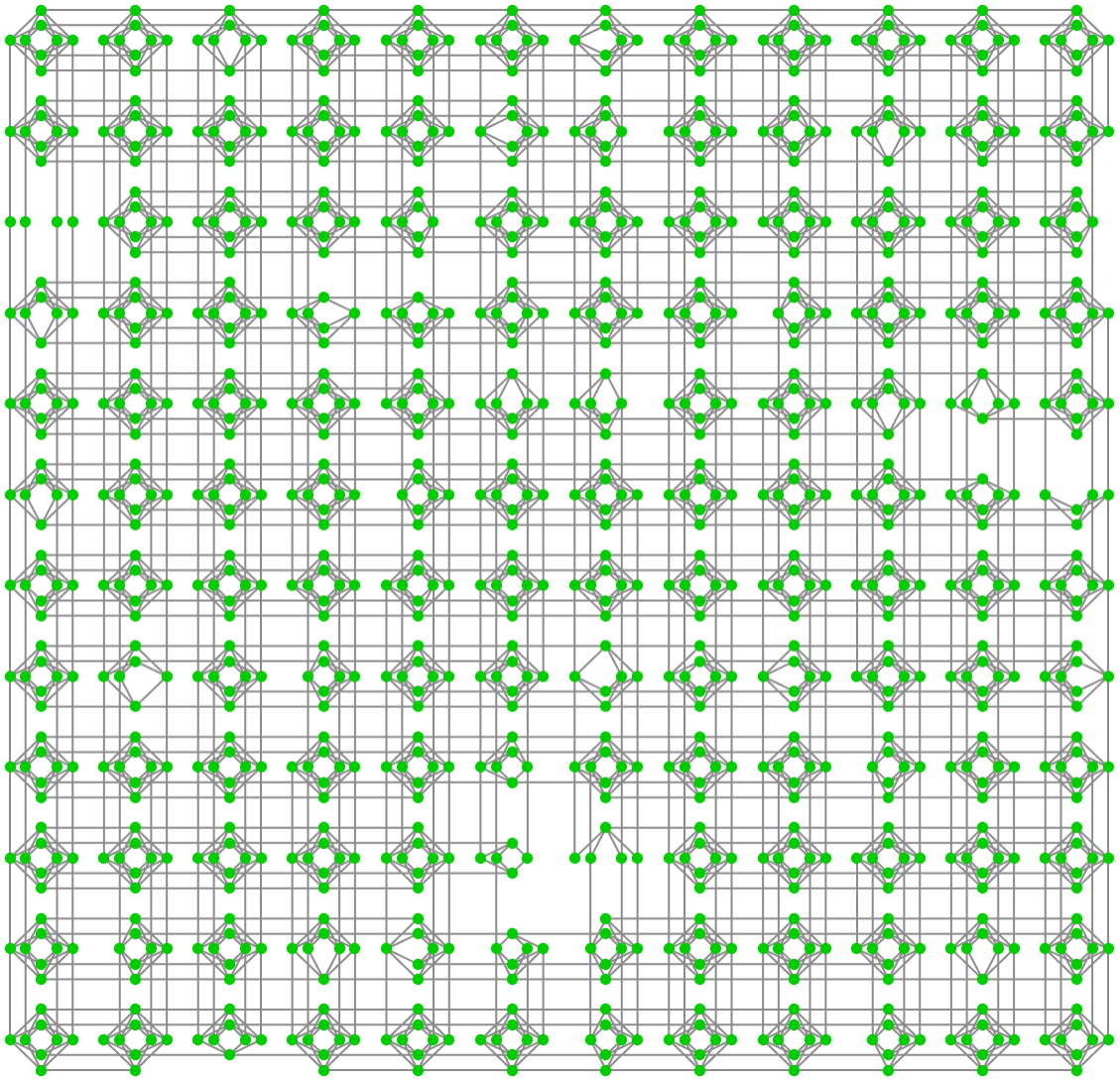

FIG. S1. Schematic representation of the "Chimera" hardware graph of the D-Wave Two X (DW2X) housed at the Information Sciences Institute at USC, used in our work. Green circles represent active qubits, inactive qubits are omitted, lines represent couplings between qubits. Each qubit can be coupled to a maximum of six other qubits.

S I. SUPPLEMENTARY METHODS

A. Additional Technical Details for Running DW

D-Wave processors currently employ a "Chimera" architecture with a limited graph connectivity (for a typical representation of a hardware graph, see Fig. S1). It is shown in the main text that the TF-DNA binding problem is reduced to finding a set of weights that minimize the contribu-

tion of a training loss and the regularization term. For ease of reference, relevant equations are reproduced here:

$$\begin{aligned} \vec{w}_{\text{opt}} &= \arg\min_{w} \sum_{n=1}^{N} \left(y_n - \vec{w}^{\intercal} \vec{\phi}_n \right)^2 + \lambda \sum_{m=1}^{4L} w_m \\ &= \arg\min_{w} \vec{w}^{\intercal} Q \vec{w} + \mathbf{w}^{\intercal} \mathbf{k}, \end{aligned} \tag{S1}$$

where

$$Q = \sum_{n} \vec{\phi}_n \vec{\phi}_n^{\intercal} , \quad \mathbf{k} = \lambda \mathbf{1} - 2 \sum_{n} y_n \vec{\phi}_n . \tag{S2}$$

Note that from the form of Eq. (S2), typical problem instances will require a complete graph with couplings between arbitrary vertices, as the summation for Q goes over all pairs of weights. In other words, we need to embed the original "logical problem" on the physical hardware, by finding a corresponding "physical problem" that can be input on the hardware. This introduces the need for a *minor embedding* of the hardware graph [1, 2]. A minor embedding maps a logical problem qubit to a set of physical qubits such that for every coupling between pairs of logical qubits in the logical problem there exists at least one physical coupling between the corresponding sets of physical qubits. A minor embedding is found by performing a series of edge contractions, which effectively join vertices together, thereby allowing for a graph with fewer vertices but a higher degree of connectivity to be obtained [3]. In order to ensure that physical qubits are aligned and act as a single logical qubit (or "chain"), a strong coupling bias is introduced between physical qubits that comprise a logical qubit. Then, for a fixed embedding, the way the values of the couplings and local fields for a logical qubit are distributed among the physical qubits is known as "parameter setting". A built-in function provided by D-Wave [2] has been used for minor embedding and parameter setting. By the embedding procedure and parameter setting, logical problems may be transformed into physical problems. Note that for one logical problem there may be many physical problems, depending both on the embedding and the parameter setting. For the DW processor used in this work, the largest complete graph that can be embedded onto the hardware graph has 42 vertices; i.e., only 42 features can be used. However, note that if there is some structure in the data, a complete graph may not be needed. In fact, HT-SELEX datasets had a greater number of features, but since they did not contain binding data for every possible sequence, the computational graph did not need to be fully connected, and thus we could find suitable embeddings with a greater number of features.

Ideally, once the strength of the coupling between logical qubits is set, all solutions returned by DW would correspond to valid logical solutions, i.e., all the physical qubits within a logical qubit would have the same spin (there would be no "broken chains"). However, due to the probabilistic nature of quantum annealing, as well as noise from different sources, there is often some percentage of solutions that have broken chains. To deal with broken chains DW offers three options for "decoding" the solutions. The first is to discard all solutions with broken chains and collect an additional set of solutions. Another option is to do a majority vote on the physical qubits, assigning the logical qubit the value with the majority of spins (in the event of a tie the spin can be randomly selected). The last option is to go through the broken chains one by one and select the value for the spin that minimizes the energy of the Hamiltonian of the logical problem. The likelihood of a solution having broken chains can be roughly adjusted by controlling a parameter J_c, the value of the strong coupling bias between physical qubits within a logical qubit; the larger the magnitude of J_c, the more likely will it be for the physical qubits within a logical qubit to have the same spin. The disadvantage of increasing the coupling bias too much is that it can wash out the details of the problem instances; thus, there is some tradeoff between getting solutions with broken chains and getting solutions which have lost the details of the problem.

Based on these considerations, our strategy for collecting solutions was the following. First, we generated 20 embeddings based on the procedure mentioned above. The one with the smallest average number of physical qubits per logical qubit was used to obtain weights for all the training instances. Then, for each training instance 10 sets of 1000 solutions were obtained from DW, giving the 10000 solutions mentioned in the main text. For each of the set of 1000 solutions we selected a different spin-reversal transformation (or, gauge [4]) to mitigate parameter misspecifications from the machine. To collect the solutions we started with a value of J_c initially set at -1 (recall that all h_i and J_{ij} are normalized to fall between -1 and 1) and successively increased the magnitude of J_c in steps by 0.1, until $1/4$ of the solutions did not have broken chains. Previous solutions were then discarded and with that value of J_c, a final 1000 reads were obtained and the solutions were decoded using energy minimization. Energy minimization decoding greedily decodes the broken chains; i.e., it fixes broken chains based which spin would result in the greatest decrease of the energy of the logical Hamiltonian. The strategy of using a stopping fraction of $1/4$ and energy minimization during the calibration phase for each dataset was selected based on some initial tests for which it seemed to work the best. It may be possible to slightly improve performance if we allowed for further tuning of these hyper-parameters.

B. Data Normalization

Due to the discrete nature of the weights returned by DW, SA and SQA, it was necessary to preprocess the data so that the range of the binding affinity values, y_n fell in the range achievable by the weights, $\vec{w}$, returned by DW, SA and SQA. Given a set of trained weights $\vec{w}$, the predicted binding affinity for transformed sequence $\vec{\phi}_n$ is $f_{\vec{w}}(\vec{\phi}_n) = \vec{w}^\intercal \vec{\phi}_n$. The weights returned by DW, SA and SQA are in the interval [0,1],[1] and since the datasets consisted of sequences of length L, there were at most L non-zero entries in each transformed feature vector, $\vec{\phi}_n$, because of the 1-mer encoding used. For DW, SA and SQA, $f_{\vec{w}}(\vec{\phi}_n) \in [0, L]$. Thus, after normalization, the binding affinity, y_n, should also be in the interval $[0, L]$. Let $\bar{y}_n$ and y_n refer to the unnormalized and normalized binding affinity, respectively. We tried two different approaches for normalizing the data: $y_n = \bar{y}_n \times \frac{L}{\bar{y}_{\max}}$ and $y_n = \bar{y}_n - (\bar{y}_{\max} - L)$, where $\bar{y}_{\max} = \max_n \bar{y}_n$. The second approach gave better results and was used for the datasets in this work. The minimum value of y_n after preprocessing was always greater than 0.

C. Using Excited State Solutions and Combining Weights (DW, SA and SQA)

We describe two heuristics used to take advantage of the distribution of solutions returned by DW, SA, SQA. The first is an iterative averaging to decrease the value of the objective function, and the other is a direct averaging of the best 20 solutions. In the training phase, all algorithms aim to to minimize an objective function $Obj(\vec{w}) = L(\vec{w}) + \Omega(\vec{w})$ [Eq. (2) of the main text].

Whether iterative averaging or direct averaging was used depended on which one gave higher metric (either AUPRC for classification tasks or Kendall's τ for ranking tasks) during the calibration phase. A list of the values of λ and the method used is indicated in Tables S1 and S2.

---

[1] Although the lowest-energy weights returned by DW, SA, and SQA are binary (i.e., $w_m \in \{0, 1\}$), as explained in Sec. S I C, several low-lying excited state solutions were averaged together, giving fractional weights that are in [0,1].

1. *Iterative averaging*

Unlike MLR, Lasso, and XGB, DW, SA and SQA are probabilistic algorithms that are run a number of times to return a distribution of binary weights with different energies; as such, it may be of interest to see whether higher energy (or, excited state) solutions that both DW, SA and SQA return may still be useful.[2] Fig. S2 shows the change in the objective value of the training instances for DW and SA upon inclusion of the first few excited (or higher-energy) state solutions ("weights" and "solutions" are used interchangeably) for a previous Mad dataset (see Sec. S III. To facilitate comparisons, both DW and SA have the same value of the regularizer λ so that values of the objective function are on the same scale (SQA was not included in Fig. S2 but performed similarly). We start with the lowest energy solution found. Higher energy solutions were then iteratively included if doing so yielded a lower value of the objective function. Namely, given a solution averaged over k solutions, $\vec{w}^{(k)} = \frac{1}{k}\sum_{i=1}^{k}\vec{w}_i$, the $k+1$th solution was included if $\vec{w}^{(k+1)} = \vec{w}^{(k)} + \frac{1}{k+1}(\vec{w}_{k+1} - \vec{w}^{(k)})$ yielded a lower value of the objective function, i.e., if $Obj(\vec{w}^{(k+1)}) < Obj(\vec{w}^{(k)})$; otherwise it was discarded and the next higher energy solution was examined in the same fashion. Up to 20 solutions were considered; beyond that, changes to the objective function value are not significant, and going further might blur the discrete nature of the weights. Although 20 solutions were examined, typically only about 4-7 excited state solutions were actually included. This way of including excited state solutions led to a monotonically decreasing objective function with increasing number of excited states. As can be seen from Fig. S2, including higher energy solutions generally improved optimization performance for both DW, SA and SQA. Performance improvement was not only in terms of the training data, but also on the held out testing data; hence, all the results in the main body of the text are for weights that have been iteratively averaged as described above.

---

[2] Note that by "energy" we refer to the energy of the Hamiltonian in Eq. (1), which is a measure of the optimization performance, and is different from the free energy of TF-DNA binding; see Methods in the main text for a mapping of the problem inputs to Ising Hamiltonian.

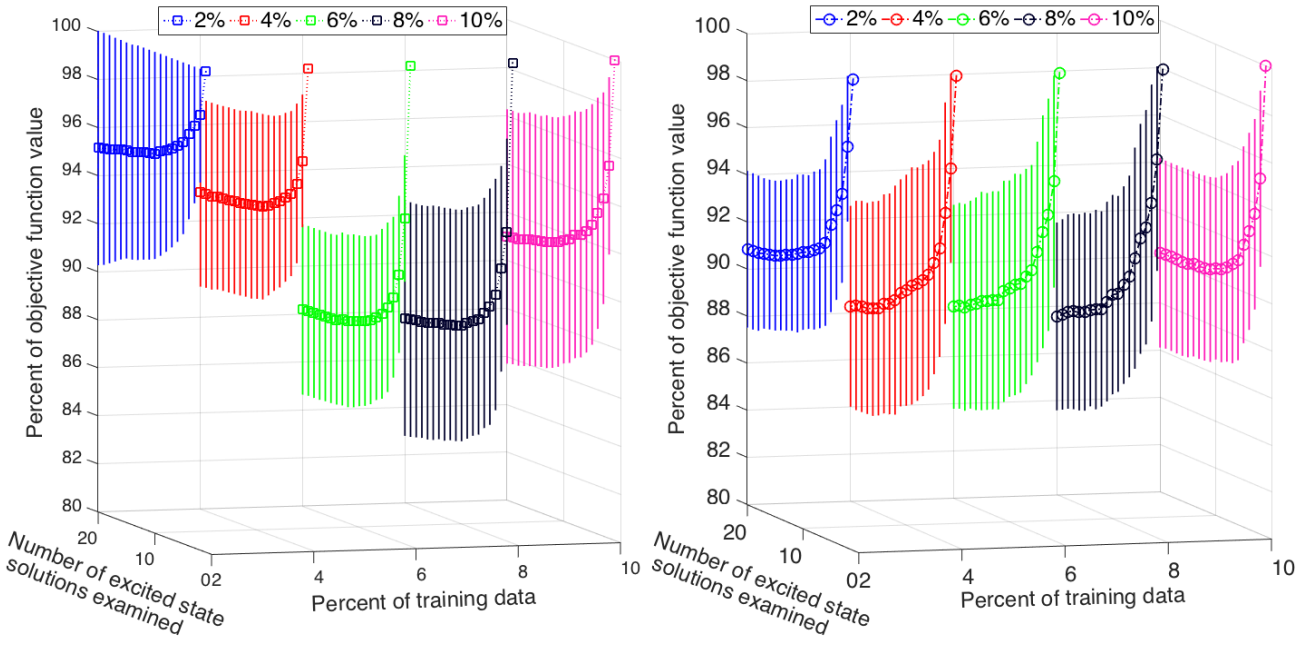

FIG. S2. Percent change in objective function versus training size and number of solutions included. Both DW (left) and SA's (right) performance improve with increasing the number of solutions included. SQA was not included but performed similarly. Error bars represent standard deviation.

2. *Direct averaging*

The second way of including excited state solutions is to simply average the lowest 20 energy solutions: $\vec{w}^* = \frac{1}{20}\sum_{i=1}^{20}\vec{w}_i$, where $\vec{w}_i$ is the solution corresponding to the ith lowest energy found, and $\vec{w}^*$ is the final, directly averaged solution used. For some of the training instances SA returned fewer than 20 unique solutions and hence we only averaged over the actual solutions returned. Whether iterative averaging or direct averaging was used depended on which gave higher metric (either AUPRC for classification tasks or Kendall's τ for ranking tasks) during the calibration phase. A list of the values of λ and the method used is indicated in Tables S1 and S2.

Dataset	2% Training					10% Training				
	DW	**SA**	**SQA**	**MLR**	**Lasso**	**DW**	**SA**	**SQA**	**MLR**	**Lasso**
Mad	64 (it)	64 (it)	64(it)	8	1	192 (dir)	256 (it)	256 (it)	32	1/32
Max*	96 (it)	96 (it)	96 (it)	12	0.5	384 (dir)	384 (it)	384 (it)	48	1/64
Myc	32 (dir)	48 (it)	24 (as)	12	1	256 (dir)	16 (dir)	96 (dir)	48	1/256
TCF4	192 (it)	192 (it)	192 (dir)	6	0.5	2 (it)	512 (it)	512(it)	32	0.25
Max†	256 (dir)	128 (it)	128(it)	6	1	192 (dir)	64 (it)	64 (dir)	0.5	1/256

TABLE S1. Value of hyper-parameter λ used for AUPRC. Parenthesis indicate method of combining solution used: (it) refers to iterative averaging and (dir) refers to direct averaging (see Sec. S I C). Since MLR has a closed-formed solution, there is no need to combine solutions. Max* refers to the gcPBM data, and Max† to the HT-SELEX dataset.

Dataset	2% Training					10% Training				
	DW	**SA**	**SQA**	**MLR**	**Lasso**	**DW**	**SA**	**SQA**	**MLR**	**Lasso**
Mad	64 (it)	64 (it)	64 (it)	4	1/32	256 (it)	3	6 (dir) (it)	12	1/64
Max*	96 (it)	96 (it)	96 (it)	12	1/32	2 (it)	4 (it)	4 (dir)	4	1/128
Myc	32 (as)	32 (it)	32 (dir)	48	1/64	256 (as)	4 (it)	16 (it)	8	1/64
TCF4	0.25 (it)	4 (it)	1 (it)	6	1/128	0.25 (it)	16 (it)	3 (it)	32	1/256
Max†	128 (it)	128 (it)	128(it)	6	1/64	6 (it)	8 (it)	512 (it)	0.5	1/128

TABLE S2. Value of hyper-parameter λ used for Kendall's τ. Otherwise the same as in Table S1.

D. Area under the precision-recall curve (AUPRC)

The data came from gcPBM and HT-SELEX experiments, both of which return a measure y of binding affinity (fluorescence intensity for gcPBM data and relative binding affinity for HT-SELEX). In order to treat the data as a classification problem we first need to classify the data based on the binding scores, y_n. To do so, we introduced a threshold θ and classify each $y_n \in \mathcal{D}^{\text{TEST}}$ such that

$$\hat{y}_n = \begin{cases} 0 & \text{if } y_n < \theta \\ 1 & \text{if } y_n \geq \theta \end{cases}, \text{ for } n = 1, \ldots, |\mathcal{D}^{\text{TEST}}|, \tag{S3}$$

was the class, $\mathcal{D}^{\text{TEST}}$ refers to the test dataset and $|\mathcal{D}^{\text{TEST}}|$ is the size of the test dataset. In other words, all y's above the threshold were given the actual label of positives ("strongly binding") and all other values were negatives ("weak binding"). Changing the threshold clearly changes the number of positive and negative labels. Since picking the threshold value is somewhat arbitrary, in order to evaluate performance at different proportions of class labels, we picked thresholds at fixed percentiles of the data (recall that the y-values were originally real-valued; hence the need to classify them). We picked the pth percentile of the data such that $p\%$ of the data was classified as negative and $(100 - p)\%$ of the data was classified as positive. In the main text we examined classification performance with relatively high thresholds at the 70th, 80th, 90th, and 99th percentile of the data. Thresholds at these percentiles can be thought of as properly selecting strong binding sites. As supporting material we also present performance at lower and medium thresholds. In total, 11 different thresholds were chosen corresponding to the percentiles at 1%,10%,20%,...,90%, and 99%.

In a binary classification problem, an algorithm must predict either positive or negative for a given sample of data based on some cut-off value of the prediction. A *confusion matrix* may then be defined in order to provide a more detailed analysis of the performance of a classifier by comparing the predicted labels to the actual labels (see Table S3). True positive (TP) are samples correctly predicted as being positive. False positives (FP) are negative samples incorrectly labeled as positive. False negatives (FN) are positive samples incorrectly labeled as negative. True negatives (TN) are negative labels correctly labeled as such.

Given the confusion matrix, an appropriate metric may then be constructed, based on the class

	predicted positive	predicted negative
actual positive	TP	FN
actual negative	FP	TN

TABLE S3. Confusion Matrix

imbalance. Commonly used terms are:

$$\text{Recall = True Positive Rate} = \frac{\#TP}{\#TP + \#FN} \tag{S4a}$$

$$\text{Precision} = \frac{\#TP}{\#TP + \#FP} \tag{S4b}$$

$$\text{False Positive Rate} = \frac{\#FP}{\#FP + \#TN} \tag{S4c}$$

In this work we used precision-recall (PR) curves, which plot precision versus recall and may give more realistic performance than commoly used receive operating characterstic (ROC) plots (ROC curves plot the false positive rate versus the true positive rate) for distributions that have a small number of actual positive labels (e.g., threshold at the higher percentiles) as it accounts for false positives in the denominator [5]. Curves may then be generated by varying a critical value, i.e., the value above which the algorithm predicts positive labels. More formally, we picked a critical value c and determined the predicted class $\tilde{y}$ such that

$$\tilde{y}_n = \begin{cases} 0 & \text{if } f_{\vec{w}}(\vec{\boldsymbol{\phi}}_n) < c \\ 1 & \text{if } f_{\vec{w}}(\vec{\boldsymbol{\phi}}_n) \geq c \end{cases}, \text{ for } n = 1, \ldots, |\mathcal{D}^{\text{TEST}}|, \tag{S5}$$

where $f_{\vec{w}}(\vec{\boldsymbol{\phi}}_n)$ was the raw predicted output given trained weights, $\vec{w}$. In our case, $f_{\vec{w}}(\vec{\boldsymbol{\phi}}_n) = \vec{w}^{\intercal}\vec{\boldsymbol{\phi}}_n$ for DW, SA, SQA, MLR, Lasso, and XGB. DW, SQA, SA, Lasso, and MLR directly assigned weights to individual features, whereas XGB made predictions based on some learned ensemble of trees. For each critical value, we could generate a point in ROC space or PR space according to the confusion matrix. The point was generated by comparing $\tilde{y}_n$ to $\hat{y}_n$ for a given percentile p and critical value c. If $\tilde{y}_n = 1$ and also $\hat{y}_n = 1$ the case n was counted as a TP. If $\tilde{y}_n = 0$ and also $\hat{y}_n = 0$ the case n was counted as a TN. If $\tilde{y}_n = 1$ while $\hat{y}_n = 0$ the case n was counted as a FP. And, if $\tilde{y}_n = 0$ while $\hat{y}_n = 1$ the case n was counted as a FN. The number of such cases, for fixed p, was tallied for $n = 1, \ldots, N$ and defined the number of TP, FP, TN, and FN used in Eq. (S4). i.e., $\#TP = (\text{number of cases such that } \hat{y}_n = \tilde{y}_n = 1)$, etc.

By sweeping through the critical value c, while holding the percentile p constant, a PR curve could be generated for each value of p. The area under the curve (AUC) is a quantitative measure of classification performance over a variety of critical values. To investigate performance at different levels of class imbalance, we used two different AUCs: the area under the ROC curve (AUROC) and the area under the precision recall curve (AUPRC). A perfect classifier will have an AUC of 1 for any of the metrics. A random classifier will have an AUROC of 0.5. This method of thresholding the data is general and can be applied to any of the datasets used.

S II. SUPPLEMENTARY DISCUSSION

A. Detailed comparisons of annealing methods for gcPBM data

Given the remarkable similarity in performance between the annealing algorithms (DW, SA, and SQA) for both the gcPBM and HT-SELEX datasets, it may be of interest to more closely examine the differences in performance between the three methods. In this section we focus our attention on the AUPRC for the gcPBM datasets by varying the number of sweeps for both SA and SQA on the physical problem. Recall that about 10% of the three gcPBM datasets (Mad, Max, and Myc) was held out as a test dataset. For training, we randomly sampled 2% of the data and 10% of the data 50 times to generate smaller training sub-datasets. For these supplementary results, we did not repeat the cross-validation procedure to choose the optimal value of λ, but rather ran SA and SQA on the same physical problem seen by D-Wave in an attempt to see if DW's performance could be reproduced by either of the methods. Comparisons for results on the physical problem may be thought of as a gauge of the quantumness, as well as an assessment of the effect on performance of using embedding and parameter-setting (see Sec. I A) in a noiseless setting.

The difference in mean AUPRC between DW, SA and SQA is shown in Fig. S3a and the difference between DW and SQA is shown in Fig. S3b versus the number of sweeps. After the anomalous behavior with a small number of sweeps, the mean AUPRC decreases with increasing number of sweeps. Note that with a large number of sweeps, SA and SQA give nearly the same value for the AUPRC as D-Wave. As before, this once again seems to indicate that the problems examined here are too easy classically, most likely due to the presence of a strong regularizer, λ. A smaller value of λ could allow us to better distinguish between SA and SQA, but would not yield

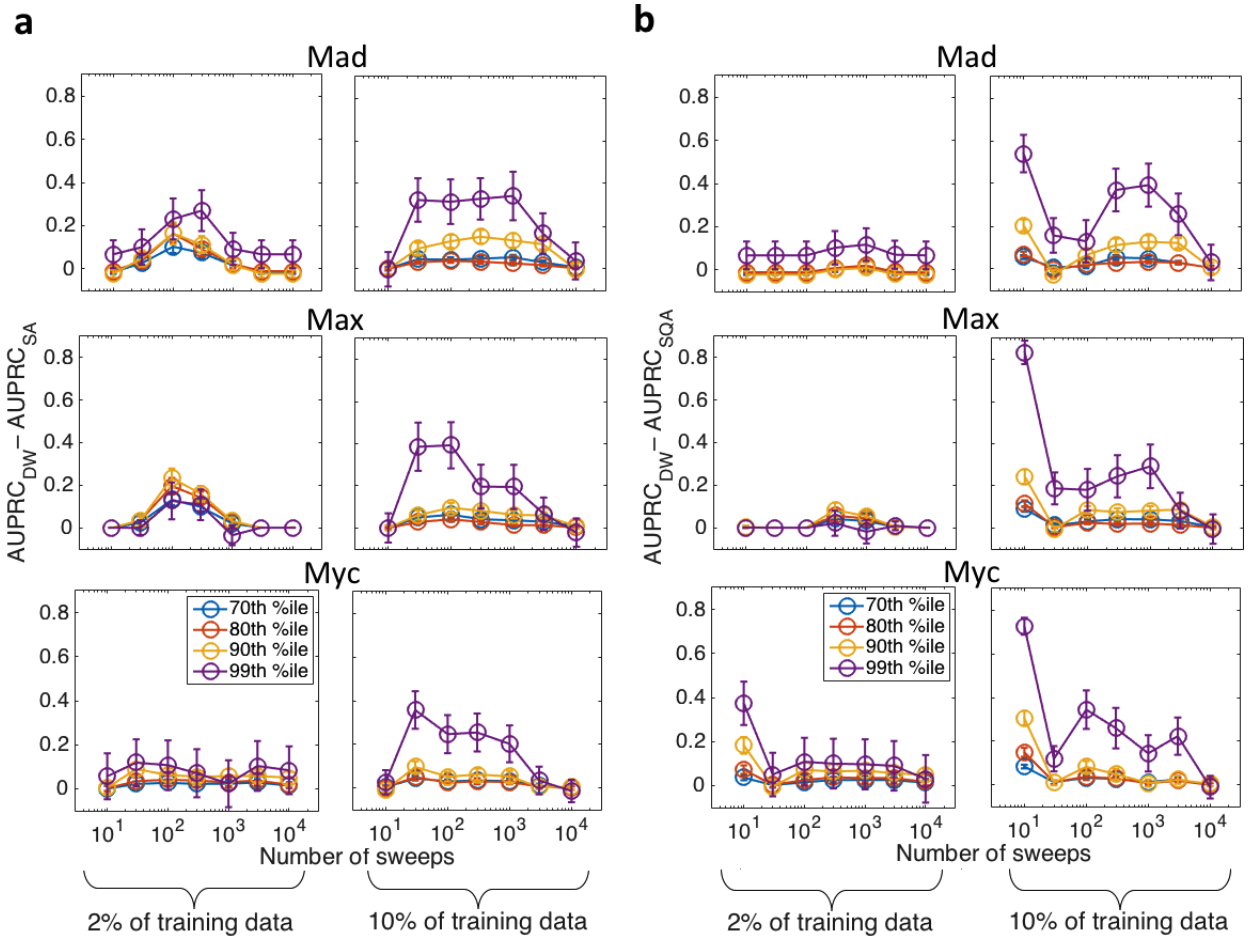

FIG. S3. Comparison of AUPRC on the gcPBM datasets for annealing methods on the same physical problem seen by DW. (**a**) Difference in AUPRC versus SA when training with 2% of the data (left) and 5% of the data (right). (**b**) Difference in AUPRC versus SQA when training with 2% of the data (left) and 5% of the data (right). Error bars represent two standard deviations. The legend indicates the percentile of the threshold used to classify the data.

optimal classification and ranking performance.

In order to understand the anomalous behavior with a small number of sweeps, we examined a few other quantities. First, we plot the mean of the lowest physical energy (i.e., we took the mean of lowest energies across the fifty training instances for each training sub-dataset) found by SA and SQA for each dataset as a function of sweeps in Fig. S4, as a sanity check. We find that increasing the number of sweeps indeed decreases the energy of the physical solutions that are found. A significant contribution to decrease in the physical energy appears to be the number of broken chains, pictured in the inset. This is perhaps more obvious for SQA, where with a large

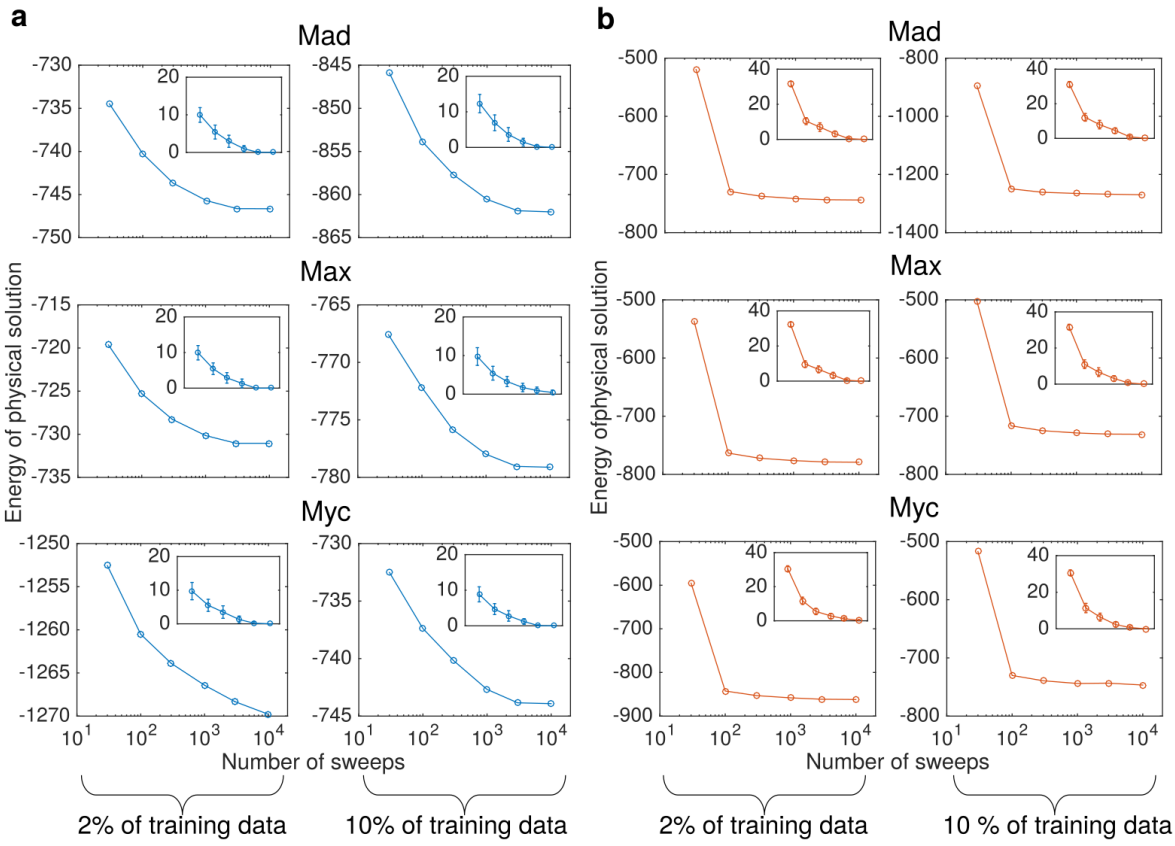

FIG. S4. **a** Shows the energies found by SA on the physical problem on the gcPBM datasets. Insets show the number of broken chains per number of sweeps. The x-scale of the inset is the same as the main plot, and **b** shows the energies found by SQA on the physical problem on the gcPBM datasets.

.

number of broken chains, the physical problem energy is much higher in comparison to SA. As the number of sweeps increases, there are fewer and fewer broken chains, and accordingly, the energy of the physical problem decreases. However, the solution with the lowest logical energy, found by fixing the broken chains using the greedy energy minimization technique (Fig. S5), does not monotonically decrease with increasing number of sweeps. Note that the lowest physical energy solution may not necessarily give the logical solution with the lowest logical energy; for the purposes of this work we selected solutions with the lowest logical energy. Given the above, there are two possible explanations for the unexpected maxima in the logical energy as a function of the number of peaks: first, the decoding procedure could result in solutions that have higher logical energy; second, the chains that are unbroken are "incorrect" in the sense that they differ from the true ground state solution. To test this, we defined a quantity ζ that represents the ratio

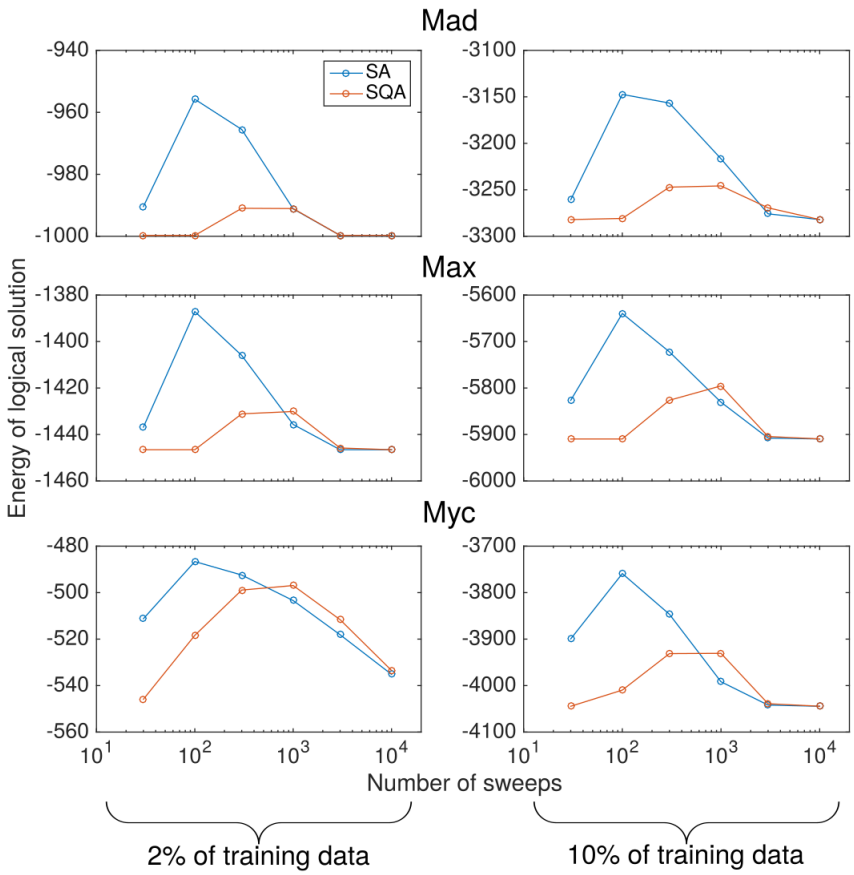

FIG. S5. The lowest energy solutions of the decoded logical problem.

of "incorrectly" fixed broken chains (i.e., after the energy minimization procedure the spin did not match the ground state solution) to the total number of broken chains for the solution that gave the lowest logical energy. We also defined a quantity γ which is the ratio of "incorrect" qubits to the number of unbroken chains (i.e., of the unbroken chains, γ is the fraction of spins that do not match the ground state solution). Plots of ζ and γ are shown in Fig. S6. For SA, there is no consistent trend for the correlation between ζ and the minimum logical energy. In some cases it is negative and in other cases it is quite positive. For SQA ζ is almost always 0, indicating that the energy minimization procedure is correctly fixing broken chains; the cases where it is nonzero, it gives a very high correlation with the minimum logical energy. However, γ has a very high Pearson correlation coefficient with the minimum logical energy for both SA and SQA in all cases. This indicates that the main reason for the anomalous behavior with an intermediate number of sweeps comes from chains that are constrained to a value; i.e., they are caught in local minima. To a much lesser extent, the logical energy is pushed up due to incorrectly fixed broken chains.

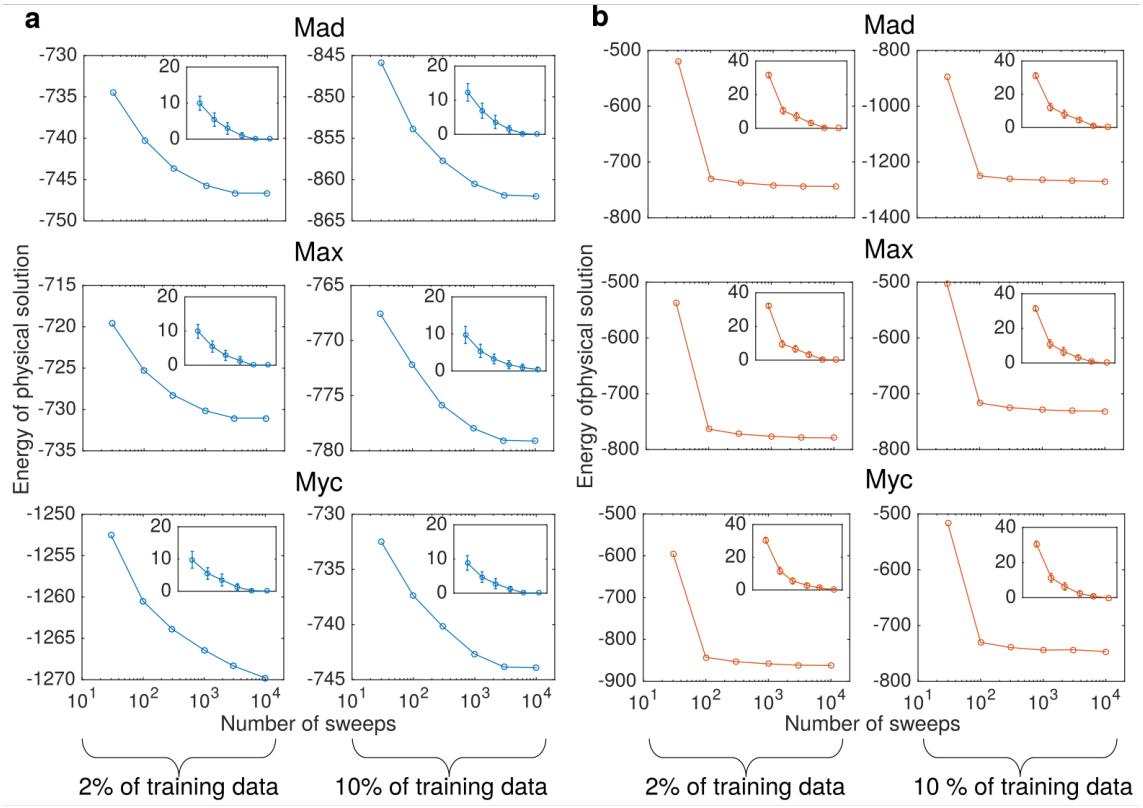

FIG. S6. Plot of ζ and γ versus the number of sweeps for both SA and SQA on the gcPBM datasets in **a** and **b**, respectively. The Pearson correlation coefficient between ζ or γ and the lowest energy solution found is indicated in the legend.

In summary, there are two contributions that may lead to an increase in the energy of the physical problem: one is an energy penalty from broken chains, and the other is from the original logical problem. With a smaller number of sweeps, both SA and SQA find solutions with a significant number of broken chains, pushing up the physical energy. However, because the classical problem is relatively easy, the energy minimization procedure works well and still finds solutions that work well, giving solutions with a lower logical energy. With an intermediate number of sweeps, a relatively small number of the chains are broken, but a few of the unbroken chains have gotten stuck in a local minima. For the ferromagnetic chain strengths chosen in the parameter-setting, the decrease in physical energy from having fewer broken chains is greater than the penalty from the actual logical problem. When further increasing the number of sweeps, there are virtually no solutions with broken chains, and in addition all the spins correspond to good solutions. This is somewhat similar to using gadgets, which preserve the ordering of lower-energy states, but not

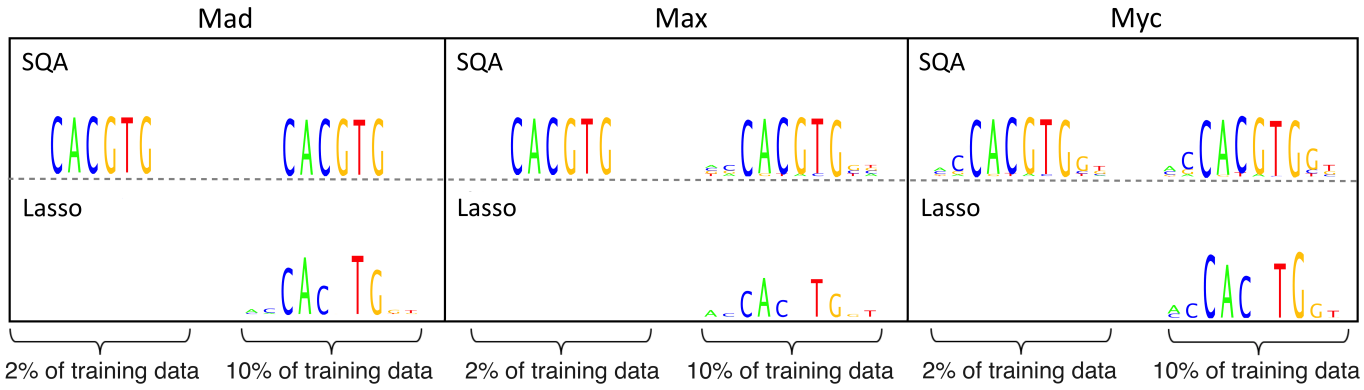

FIG. S7. Weight logos for SQA and Lasso weights on the gcPBM datasets

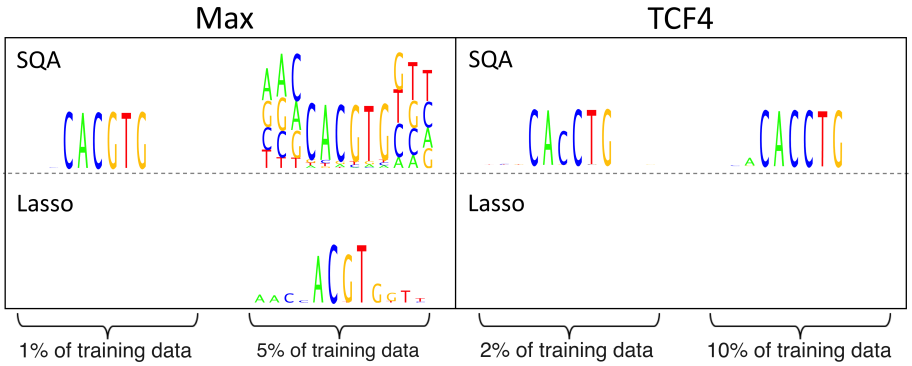

FIG. S8. Weight logos for SQA and Lasso weights on the HT-SELEX datasets

necessarily higher-energy ones [6].

B. Weight logos

In Figs. S7 and S8 we present the logos for SQA and Lasso for the gcPBM and HT-SELEX datasets, respectively. As with the main text, these logos were derived by averaging the weights found when using the AUPRC as the objective. Somewhat strikingly, the solution that works the best for LASSO in terms of the AUPRC is to give weights that are all zero; in other words, only the bias term survives. Comparing to the quantitative results in Main Fig 3, most of the other methods (besides XGB with small amounts of training data) perform better than Lasso when it returns weights that are all zero.

S III. SUPPLEMENTARY RESULTS FOR PREVIOUS DATASETS

In this section we present results for previous sets of gcPBM data from [7] (GEO accession number GSE47026) and HT-SELEX data from [8] (European Nucleotide Archive; ENA study identifier PRJEB3289)). The HT-SELEX datasets used in the main text had increased number of rounds of sequencing. The datasets were normalized in the same way as described in Sec. S I B. We first present a summary of the quantitative results (i.e., the AUPRC for classification tasks and Kendall's τ for ranking tasks) for the gcPBM data and HT-SELEX datasets in Sec. S III A. Then in Sec. S III B we plot the the difference in performance versus the Euclidean distance between DW, SA and SQA weights in attempt to differentiate between the solutions found by DW, SA and SQA. Next in Sec. S III D, we present plots of performance versus size of the training data. Finally, we present a table of the optimal values of λ found during the calibration phase in Sec. S III E.

A. Best performing results

The Figures in this section are analogous to Figs. 3 and 5 in the main text. The trends are reproducible.

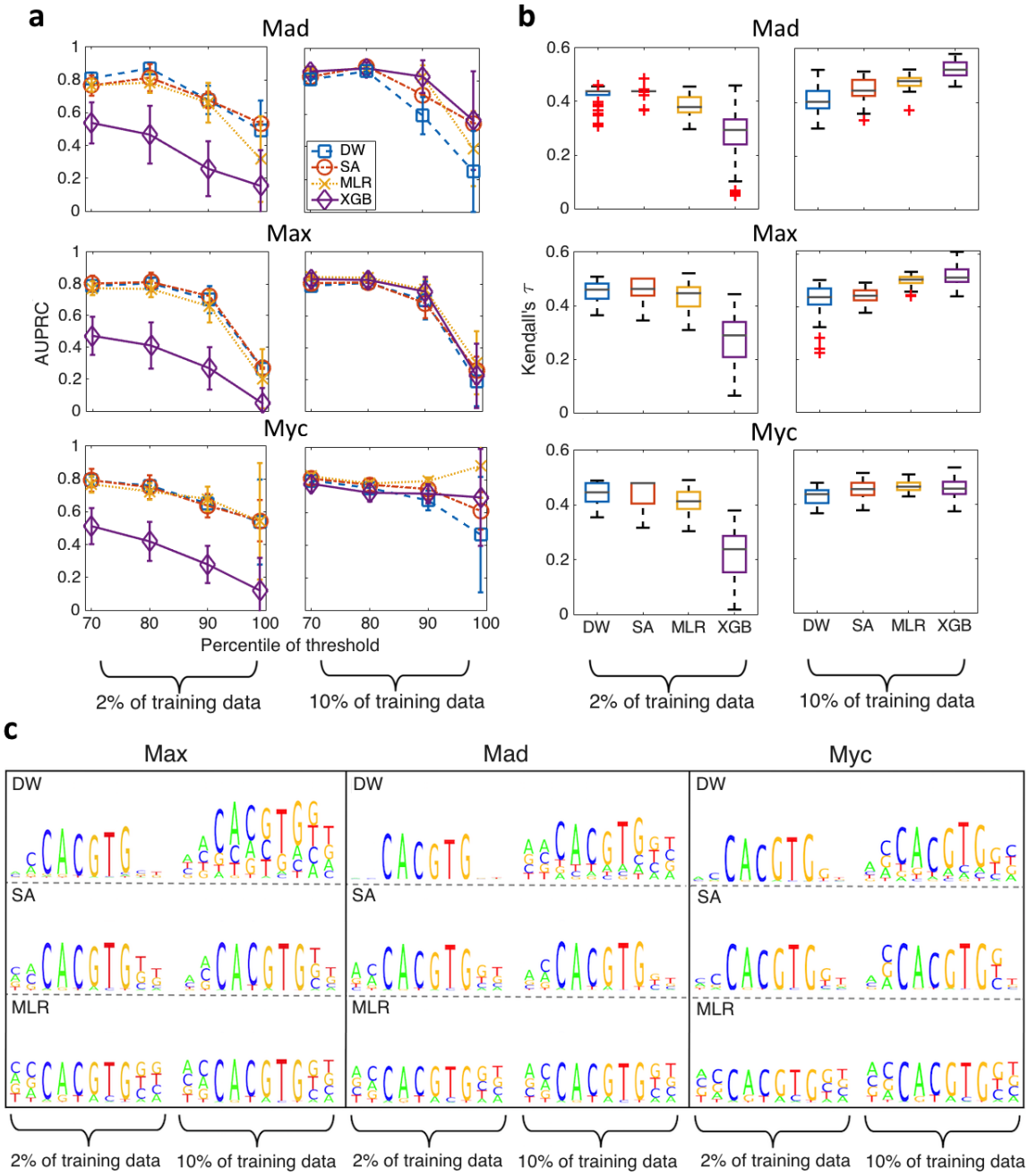
a
Mad
Max
Myc
AUPRC
DW
SA
MLR
XGB
Percentile of threshold
2% of training data
10% of training data
b
Kendall's τ
DW SA MLR XGB
c
Max
Mad
Myc
DW
SA
MLR

FIG. S9. Quantitative performance on held-out experimental test dataset of two different types of tasks for three high quality gcPBM datasets. (**a**) The mean AUPRC for Mad, Max, and Myc plotted versus threshold at certain threshold percentiles of the data, when training with about 30 sequences (left) and 150 sequences (right). In both cases 50 instances were randomly selected for training and performance of the 50 trained weights is evaluated on the same held-out test set. Error bars are standard deviations. (**b**) Boxplot of Kendall's τ on held-out test dataset. Red '+' indicate outliers, gray line represents the median. (**c**) Weight logos for DW, SA, and MLR.

For the gcPBM datasets, when training on instances with about 30 sequences (left column in Fig. S9a), DW, SA, and MLR all perform very similarly, though DW has a slight advantage at the 80th percentile for Mad (during the Calibration phase, we selected the λ that gave the best performance at the 80th percentile). MLR does fairly well and in some cases its mean AUPRC for different weights trained on the classifiers very slightly exceeds DW's. XGB does poorly with small training sizes. When training with about 150 sequences (right column in Fig. S9a), the trends of relative performance are modified. DW's performance is slightly worse than the other methods at 70th and 80th percentile and noticeably worse at higher thresholds. XGB usually performs the best, SA and and MLR tend to do almost as well as XGB. These trends are very similar for Kendall's τ (Fig. S9b).

As shown in Fig. S9c, the weights found by DW, SA, and MLR all give good agreement with the expected consensus sequence, indicating that the methods are learning some biologically relevant information. There is, however, more variation in the logos presented here than those in Fig. 4. This is due to the larger value of λ used for the results in the main text (Tables S1 and S2 are for the data in the main text, Tables S4 and S5 are for the data used in Sec. S II).

As with the gcPBM data, DW's performance is noticeably worse than the other methods on HT-SELEX data with increasing amounts of training data for classification and ranking tasks (Fig. S10**a** and **b**); when training with about 150 sequences, DW's performance is not competitive with the other methods, except at the 99th percentile, where it does slightly better than SA and MLR. When training with about 30 sequences, DW's performance is competitive with some of the other methods; DW performs better than SA, comparable albeit slightly worse than MLR, and worse than XGB. In contrast to the gcPBM data, DW does better when classifying data at the 90th and 99th percentile. We argue that DW's worse performance on the HT-SELEX dataset than on the gcPBM data is due the higher level of noise in the HT-SELEX data. Note that there is a

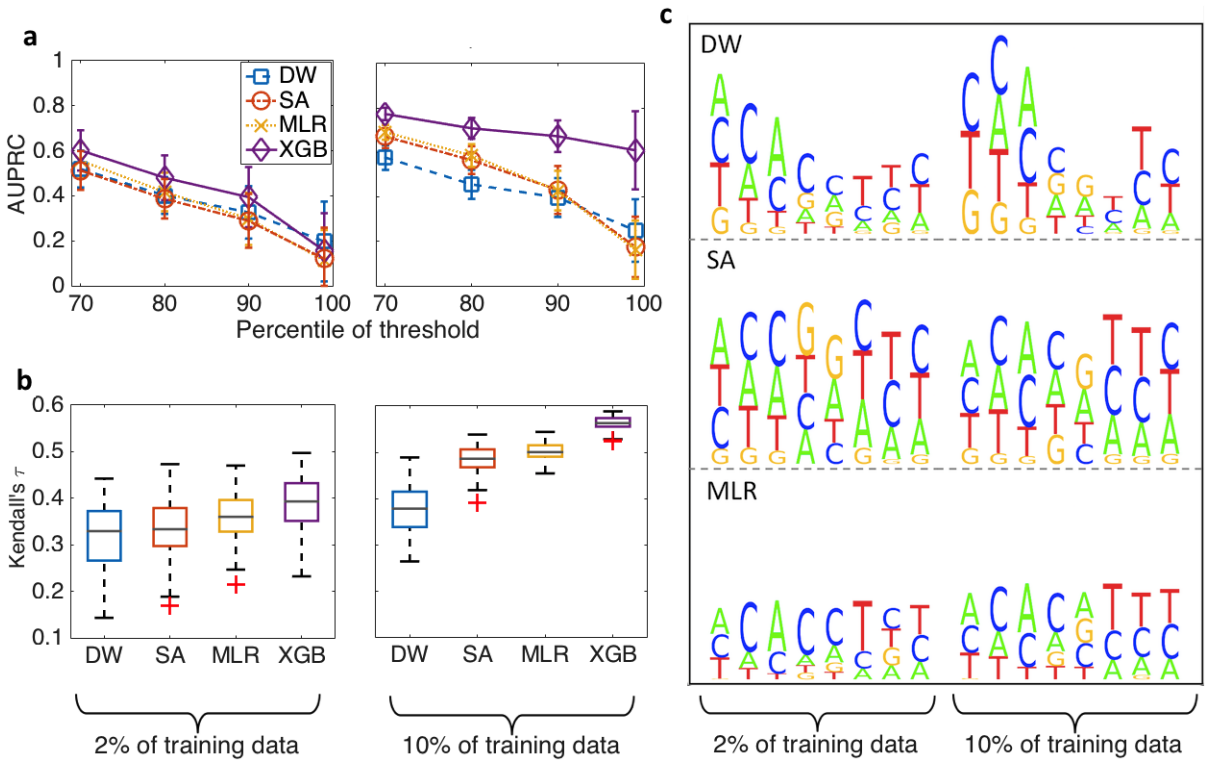

FIG. S10. Summary of results for TCF4. Panel a compares the AUPRC when training with about 30 sequence (left) and 150 sequences (right). Error bars are standard deviations over 50 instances. Panel b shows Kendall's τ when training with about 30 sequences (left) and about 150 sequences (right); panel c shows the weight logos when training with about 30 sequences and 150 sequences.

distinction to be made between noise and generalization. Small training data may be problematic due to some "local" patterns in the data; this is different from noise in the data, which affects not just small problem sizes. That this HT-SELEX data is noisier is supported by the poor agreement of the weight logos (Fig. S10c) with the expected consensus sequence, especially for SA. We also note that XGB does very well even with small datasets, suggesting that it may be able to handle noise better than the other methods.

B. Distinguishing DW and SA Weights on gcPBM Data

Given the similarity of DW, SA and SQA, which both share the energy functional of Eq. (1) and exhibit similar performance in Fig. S9 and S10, it is of interest to distinguish between the two methods. We should note, however, that hyper-parameter tuning was performed separately for each method and generally yielded different values of λ (see Table S4 and S5). Fig. S11 shows the

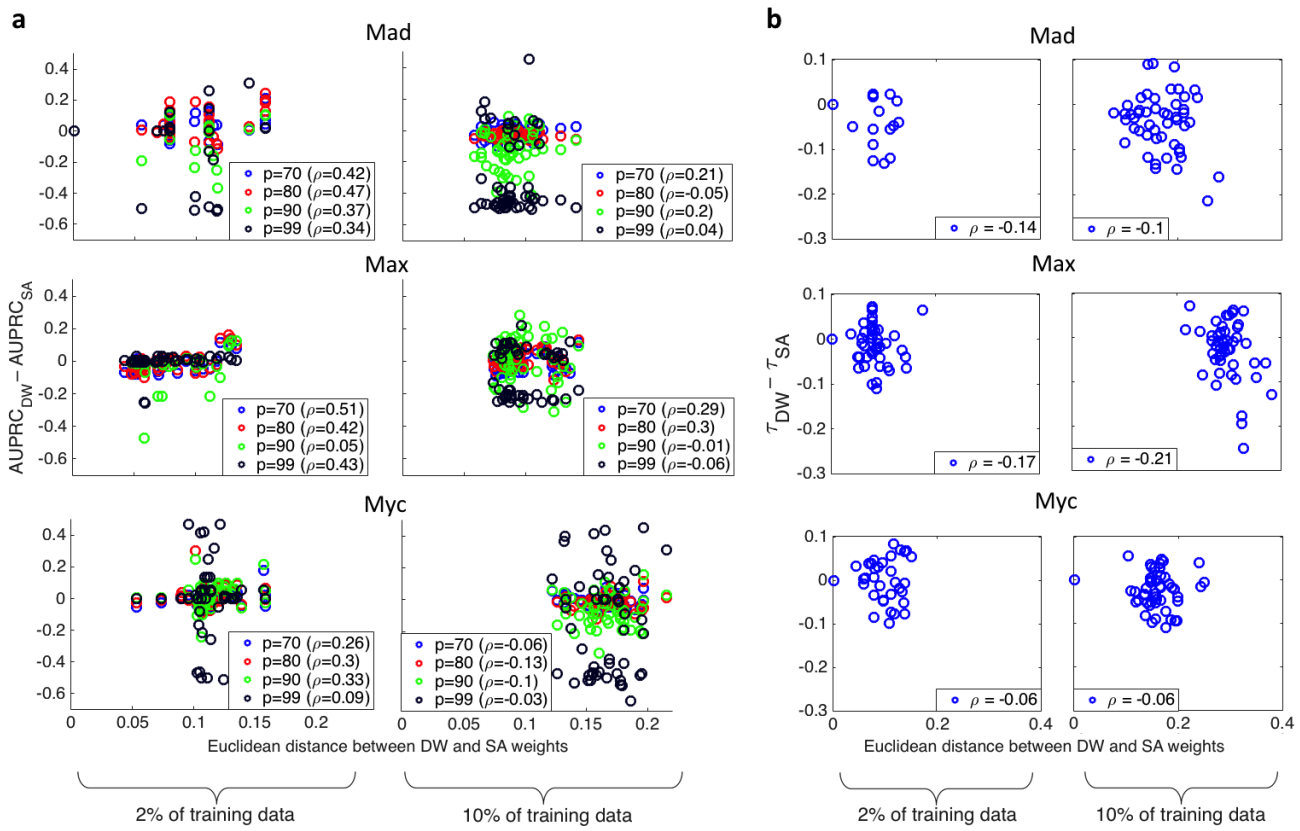

FIG. S11. Comparison between difference in performance metric and difference in Euclidean distance of DW, SA and SQA for Mad, Max, and Myc. Panel **a** shows the difference in AUPRC on the 50 randomly selected instances trained on about 30 sequences (left) and about 150 sequences (right). Thresholds are set at the pth percentile of the data and are shown in different colors, along with the value of the Spearman rank correlation coefficient ρ in parentheses. Panel **b** shows the difference in Kendall's τ.

difference in classification (a) and ranking performance (b) versus the Euclidean distance between the weights found by DW, SA and SQA, for the gcPBM data; a similar figure for the noisier HT-SELEX data is presented in Fig S12. Independently of the dataset and training size, the Euclidean distance is fairly small and the difference in AUPRC is close to 0. Nevertheless, we can identify a slight positive correlation, as quantified by the Spearman correlation coefficient between the Euclidean distance and difference in AUPRC when training with 2% of the training data, where DW does slightly better than or comparably to SA. The positive correlation indicates that DW performs better when its solutions differ from SA's. When training with 10% of the data, where DW's performance is mostly worse than SA's, there is a smaller positive correlation. This suggests that when the noise has less of an effect for DW (training with 2% of the data), DW performs better

the more it differs from SA. Conversely, when the effect of noise is greater (training with 10% of the data), it is disadvantageous for DW to find solutions that differ significantly from SA's. When the training instances are smaller, lack of precision on the inputs is less noticeable, suggesting that differences from the classical set of weights (SA) may be due to quantum effects. When the training instances are larger, the differences the weights found may be due to the mis-specification of the h_i and J_{ij}. We speculate that the advantage DW has with smaller training sizes, when the effect of precision limitations is smaller, may be due to quantum effects; with larger training sizes, any benefits due to quantum effects may be masked by precision limitations.

Fig. S11b shows that there is always a negative correlation between Euclidean distance and Kendall's τ, indicating DW performs worse when its solutions differ from SA's. When training with 2% of the training data, there are a good number of instances where DW, SA and SQA find the exact same solution and hence give the same value of τ. There also tends to be a greater Euclidean distance between SA and DW when training with 10% of the training data. Thus for ranking tasks on these datasets, currently SA is the preferred method.

C. Distinguishing DW, SA and SQA Weights on HT-SELEX TCF4 data

In Fig. S12 we present the AUPRC and Kendall's τ versus the Euclidean distance between the DW, SA and SQA solutions, when training with about 30 sequences and 150 sequences. In contrast to Fig. S11 of the main text, there is always (except when θ is set at the 99th percentile of the data and with 2% training data) a negative correlation between DW's performance and SA's performance, indicating that the more different the DW, SA and SQA solutions are, the better SA's solutions are.

D. Performance vs Size of Training Data

We present results, shown in Fig. S13, for the AUPRC at different thresholds when training with 2, 4, 6, 8, and 10% of the data for the gcPBM datasets for Max, Mad, and Myc, and for the HT-SELEX dataset for TCF4. When training with 2% of the data in the Training phase, we used 2% of the data during the Calibration phase; when using 4% and 6% of the data in the Training phase, we used 5% of the data during the Calibration phase; when using 8% and 10% of the training data in the Training phase, we used 10% of the data to tune hyper-parameter λ.

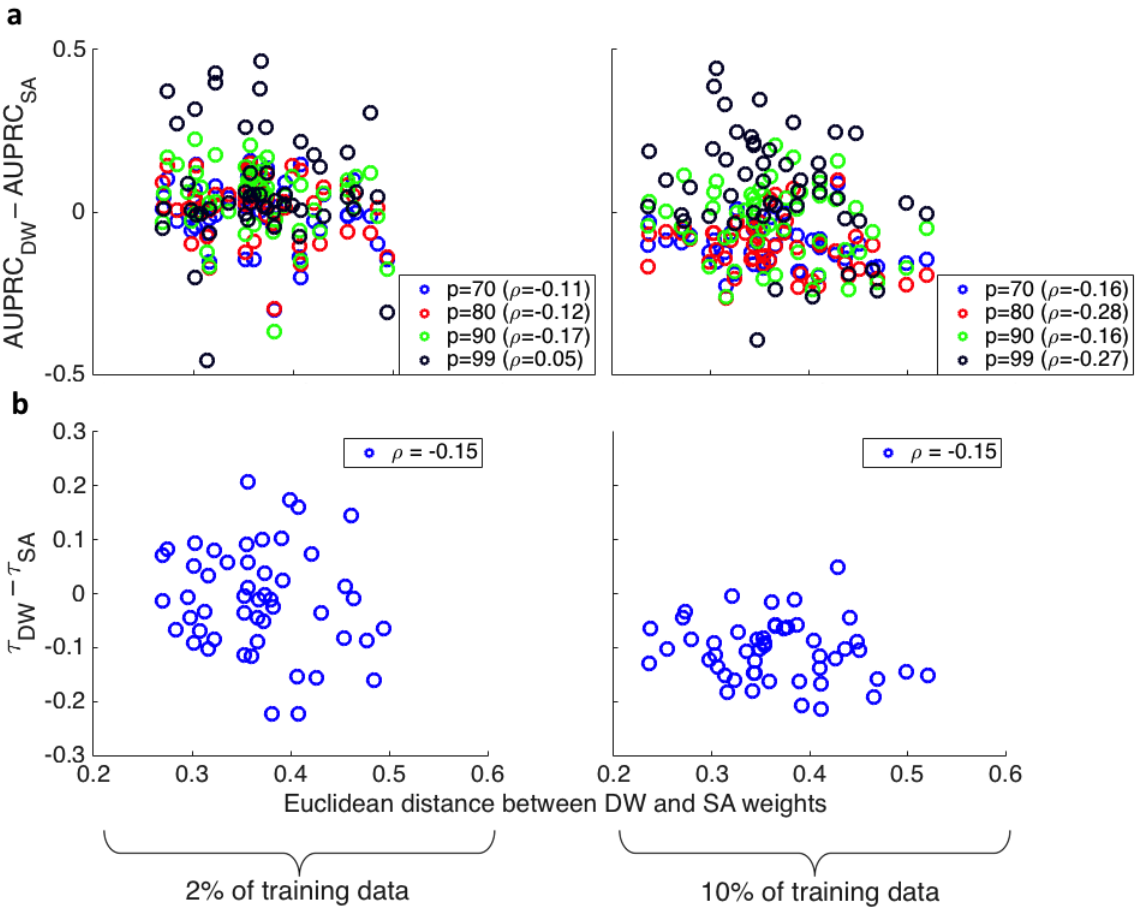

FIG. S12. (a) Plot of DW AUPRC - SA AUPRC vs Euclidean distance for TCF4 on 50 randomly selected instances trained on about 30 sequences (left) and on about 150 sequences (right). Thresholds at the pth percentile of the data are shown in different colors with the value of the Spearman correlation coefficient ρ. (b) Kendall's τ versus Euclidean distance when training with about 30 sequences (left) and about 150 sequences (right).

At 2% of the training data, DW performance is slightly better at the 70th and 80th percentile and fairly indistinguishable from SA and MLR at the 90th and 99th percentiles, but the difference between DW and other methods becomes more apparent with increasing training size. XGB shows a pronounced increase in performance with increasing training size, and SA and MLR both show slight increases. DW's performance remains relatively constant with increasing training size but tends to decrease when reaching training sizes of 10% of the training data. There are noticeable exceptions when introducing a threshold at the 99th percentile of the data, but larger variations in performance are expected given high class imbalance [9].

For a further inspection of the precision limitations of DW, the AUPRC versus training size

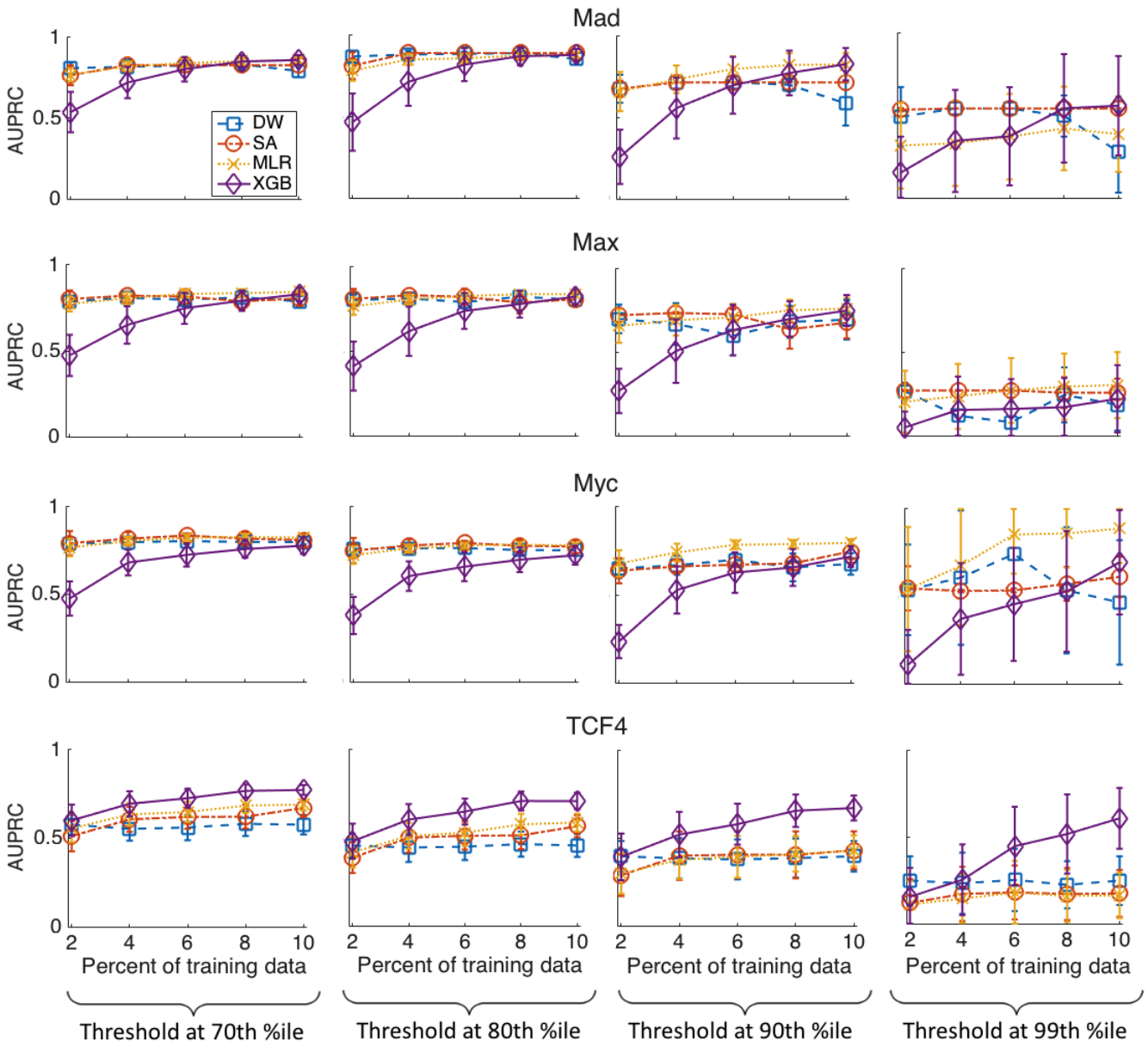

FIG. S13. AUPRC versus training size at a few different thresholds of the data for Mad, Max, Myc (gcPBM), and TCF4 (HT-SELEX).

is shown in Fig. S14. A suboptimal value of λ was used for all methods in Fig. S14 (i.e., λ was improperly tuned during the calibration phase). In contrast to the procedure for obtaining Fig. S13, λ was held constant across different training sizes for Fig. S14 (though it differs between DW, SA, and MLR). For DW, across thresholds and across datasets, there is a decrease in AUPRC when going from 2% to 4% of the data. This decrease is likely due to the fact that for these datasets, even with training on 4% of the data, precision-limiting noise limits DW's ability to perform

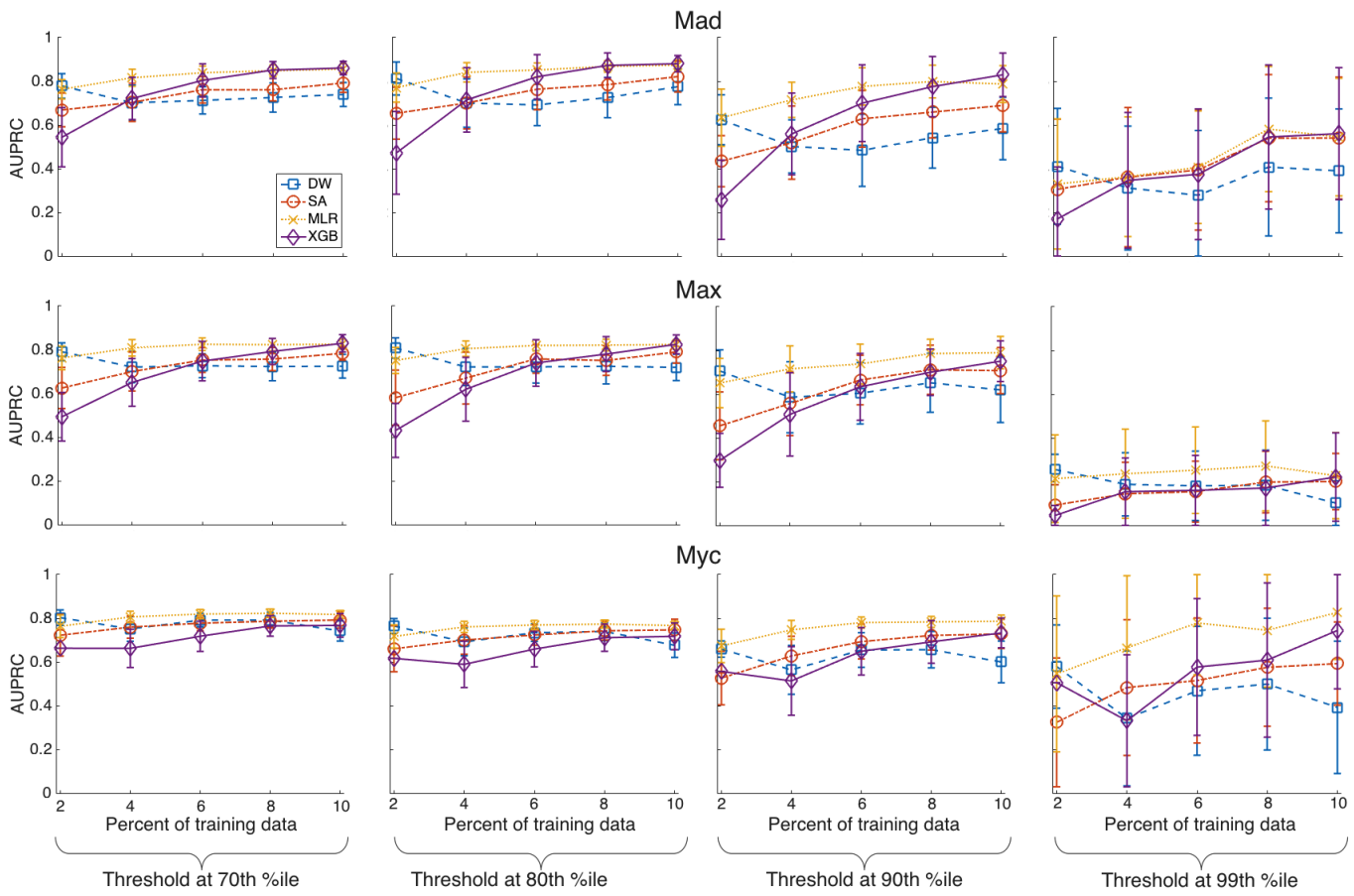

FIG. S14. AUPRC versus training size at a few different thresholds of the data for Mad, Max, Myc with a single λ for each dataset, rather than tuning separate λs for different training sizes.

well. The drop in performance between 2% and 4% of the data agrees with known estimates of the noise on the couplers and local fields in the D-Wave devices. The ranges of values taken by the h_i and J_{ij} scales roughly linearly with the number of sequences, S, but these problems inputs are scaled between -1 and 1 before being sent to the processor. If the noise fraction [i.e., $\max\{\Delta J/(\max J_{ij}), \Delta h/(\max h_i)\}$] is greater than about $1/S$, we would expect to see a decrease in performance. Estimates for the DW2X processor used in our experiments put the noise at about 3%, and the training sizes varied from about 1500 to 1800 sequences. In other words, 2% of the data corresponds to $S \approx 33$ sequences, so that $1/S$ corresponds to about 3%, which is just around the level of the noise. As we go above 2% of the training data, the effects of the precision-limiting noise are seen and DW's performance advantage disappears with the noise, as noticed by the dip in performance for all the higher quality gcPBM datasets. While these results came from improperly tuning λ, they do provide some insight about the effect of noise on DW's performance.

Dataset	2% Training			5% Training			10% Training		
	DW	**SA**	**MLR**	**DW**	**SA**	**MLR**	**DW**	**SA**	**MLR**
Mad	20 (it)	40 (it)	36	72 (it)	28 (it)	64	72 (dir)	40 (it)	128
Max	54 (dir)	24 (dir)	48	54 (dir)	60 (dir)	128	72 (dir)	24 (it)	128
Myc	54 (dir)	14 (it)	44	66 (dir)	36 (it)	128	66 (dir)	24 (dir)	128
TCF4	0 (it)	0 (it)	2	2 (it)	0 (it)	2	0 (it)	0 (it)	2

TABLE S4. Value of hyper-parameter λ used for AUPRC. Parenthesis indicate method of combining solution used: (it) refers to iterative averaging and (dir) refers to direct averaging (see Sec. S I C). Since MLR has a closed-formed solution, there is no need to combine solutions.

Dataset	2% Training			5% Training			10% Training		
	DW	**SA**	**MLR**	**DW**	**SA**	**MLR**	**DW**	**SA**	**MLR**
Mad	20 (it)	20 (it)	20	54 (it)	28 (it)	24	54 (it)	4 (it)	60
Max	16 (it)	24 (dir)	20	66 (it)	54 (it)	24	60 (it)	0 (dir)	60
Myc	20 (it)	14 (it)	28	72 (dir)	36 (it)	32	48 (it)	6 (it)	60
TCF4	0 (it)	0 (it)	2	2 (it)	0 (it)	0	0 (it)	0 (it)	0

TABLE S5. Value of hyper-parameter λ used for Kendall's τ. Otherwise the same as in Table S4.

E. Values of λ for datasets in Supplementary Results

The optimal values of λ found from the Calibration phase for the AUPRC and Kendall's τ are presented in Tables S4 and S5, respectively. These pertain to Figs. S7-S8.

[1] Choi, V. Minor-embedding in adiabatic quantum computation: I. The parameter setting problem. *Quant. Inf. Proc.*, **7,** 193–209, (2008).

[2] Cai, J., Macready, W. G., and Roy, A. A practical heuristic for finding graph minors. *arXiv:1406.2741*, (2014).

[3] Robertson, N. and Seymour, P. Graph minors. iii. planar tree-width. *Journal of Combinatorial Theory, Series B*, **36,** 49 – 64, (1984).

[4] Boixo, S., Albash, T., Spedalieri, F. M., Chancellor, N., and Lidar, D. A. Experimental signature of programmable quantum annealing. *Nat. Commun.*, **4,** 2067, (2013).

[5] Davis, J. and Goadrich, M. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, pages 233–240, New York, NY, USA, (2006). ACM.

[6] Cao, Y. and Nagaj, D. Perturbative gadgets without strong interactions. *Quantum Information & Computation*, **15,** 1197–1222, (2015).

[7] Yang, L., et al. TFBSshape: a motif database for DNA shape features of transcription factor binding sites. *Nucleic Acids Research*, **42,** D148–D155, (2014).

[8] Jolma, A., et al. Multiplexed massively parallel selex for characterization of human transcription factor binding specificities. *Genome Research*, **20,** 861–873, (2010).

[9] Cortes, C. and Mohri, M. Auc optimization vs. error rate minimization. *Neural Information Processing Systems*, (2004).