Learning Graphical Models Using Multiplicative Weights

Chongyang Bai, Chao Chen, Rui Liu

November 12, 2018

1 Background

Probabilistic graphical model have numerous real-world applications. It's used in Image Generation, Image Denoising, Language Translation, Speech Recognition etc.

A Markov Random Field (MRF) is a probability distribution p over variables $x_1, ..., x_n$ defined by an undirected graph G in which nodes correspond to variables x_i. The probability p has the form

$$p(x_1, ..., x_n) = \frac{1}{Z} \prod_{c \in C} \phi_c(x_c)$$

, where C denotes the set of cliques (i.e. fully connected subgraphs) of G.
The value

$$Z = \sum_{x_1, ..., x_n} \prod_{c \in C} \phi_c(x_c)$$

is a normalizing constant that ensures that the distribution sums to one.

t-wise Markov random field denotes that all cliques of size is at most t. Ising model is a special case of MRF where t=2. For a graph G = (V,E), define a binary Ising model on to be a distribution D on $\{1, -1\}^n$. We have

$$\mathop{\mathbf{Pr}}_{Z \sim D}[Z = z] \propto \exp(\sum_{i \neq j \in [n]} A_{ij} z_i z_j + \sum_i \theta_i z_i) \tag{1}$$

Interesting problems related to probabilistic graphical model:

- Inference Problem: Given a probabilistic model (such as an MRF), we wanna used it to do prediction on particular problems.
- Learning Problem: A graphical model has two components: the graph structure, and the parameters of the factors induced by this graph. These components lead to two different learning tasks:
  - Parameter learning, where the graph structure is known and we want to estimate the factors.
  - Structure learning, where we want to estimate the graph, i.e. determine from data how the variables depend on each other.

This paper focuses on structure learning and parameter learning. It gives a simple multiplicative-weight update algorithm for learning undirected graphical models or Markov random fields (MRFs) with nearly optimal sample complexity and running time.
(Attribution: all things above are referenced from the notes [1])

2 Main Problem

This paper mainly talks about developing efficient algorithms for inferring the structure of the underlying graph from random samples from a distribution $\mathbb{D}$. First, it describes Sparsitron algorithm for learning Sparse Generalized Linear Models. Second, it applies Sparsitron algorithm to recover the parameters of an binary Ising model. And then, it generalizes the Sparsitron algorithm for learning structure as well as parameters of t-wise binary Markov Random Fields. Last, it extends the results for the Ising model from binary to work over general alphabet.

Learning Sparse Generalized Linear Models (GLM) Let $\mathbb{D}$ be a distribution on $[-1,1]^n \times \{0,1\}$ where for $(X,Y) \sim \mathbb{D}$, $\mathbb{E}[Y|X=x] = u(w \cdot x)$ for a non-decreasing 1-Lipschitz function u: $R \longrightarrow$[0,1]. Suppose that $\|w\|_1 \leq \lambda$ for a known $\lambda \geq 0$. Given enough independent examples from $\mathbb{D}$, we want to learn the parameters w. In other words, we want an algorithm that could learn from the examples and return a v which is a close approximation of w.

This paper gives an algorithm that for all $\epsilon, \delta \in [0,1]$ given $T = \mathcal{O}(\lambda^2 \frac{ln(\frac{n}{\delta\epsilon})}{\epsilon^2})$ independent examples from $\mathbb{D}$, produces a vector $v \in R^n$ such that with probability at least $1-\delta$, $\mathbb{E}_{(X,Y)\leftarrow D}[(u(w \cdot x) - u(v \cdot x))^2] \leq \epsilon$. The run-time of the algorithm is $\mathcal{O}(nT)$. Moreover, the algorithm can be run in an online manner.

Learning binary Ising Models Let $A \in R^{n \times n}$ be a weight matrix and θ be a mean-field vector. The associated n-variable Ising model is a distribution $\mathbb{D}(A,\theta)$ on $\{1,-1\}^n$ given by the equation 1. The dependency graph of $\mathbb{D}(A,\theta)$ is the graph G of all pairs $\{i,j\}$ with $|A_{ij}| \neq 0$. Given enough independent examples from $\mathbb{D}$, we want to learn the parameters and structure of the Ising model. In other words, we want an algorithm that is able to return a close approximation of A and θ.

This paper gives a simple, sample-efficient, and online algorithm for recovering the parameters of Ising models. The idea is: run Sparsitron algorithm to recover weights $\{A_{ij} : j \neq i\}$ for every $i \in \{1,2,...,n\}$. Given $\lambda, \epsilon, \rho \in (0,1)$, and $N = \mathcal{O}(\lambda^2 \frac{\exp \mathcal{O}(\lambda)}{\epsilon^4}) \cdot (\log(\frac{n}{\rho\epsilon}))$ independent samples from $\mathbb{D}$, the algorithm produces $\hat{A}$ such that with probability at least $1-\rho$, $\|A - \hat{A}\|_\infty \leq \epsilon$. The run-time of the algorithm is $\mathcal{O}(n^2 N)$. Moreover, the algorithm can be run in an online manner.

Learning t-wise binary Markov Random Fields Given a graph $G = (V,E)$ on n vertices, let $C_t(G)$ denotes all cliques of size at most t in G. A binary t-wise MRF with dependency graph G is a distribution $\mathbb{D}$ on $\{1,-1\}^n$ where the probability density function of $\mathbb{D}$ can be writen as $\mathbf{Pr}_{Z\sim D}[Z=x] \propto \exp(\sum_{\mathbb{I}\in S} \psi_I(x))$, where $S \subseteq C_t(G)$ and each $\psi_I : R^n \to R$ is a function that only depends on variables in I. Given enough independent examples from $\mathbb{D}$, we want to learn the parameters and structure of the MRF. In other words, we want an algorithm that is able to return a close approximation of dependency graph G and factorization polynomial $\varphi(x) = \sum_{\mathbb{I}\in C_t(G)} \varphi_I(x)$.

This paper gives an algorithm that given λ and $\epsilon, \rho \in (0, \frac{1}{2})$, and $N = \frac{(2t)^{\mathcal{O}(t)} e^{\mathcal{O}(\lambda t)}}{\epsilon^4} \cdot n^{4t} \cdot \log(\frac{n}{\rho\epsilon})$ independent samples $Z^1, ..., Z^N$ from $\mathbb{D}$, produces a t-wise MRF D' with dependency graph H and a factorization polynomial $\varphi(x) = \sum_{\mathbb{I}\in C_t(H)} \varphi_I(x)$ such that with probability $1-\rho$: $\forall x, \mathbf{Pr}_{Z\sim D}[Z=x] = (1\pm\epsilon)\mathbf{Pr}_{Z\sim D'}[Z=x]$. The algorithm run in time $\mathcal{O}(Nn^t)$.

Learning non-binary Ising model Let $\mathcal{W} = (W_{ij} \in R^{k\times k} : i \neq j \in [n])$ be a collection of matrices and $\theta \in R^{[n]\times[k]}$. Then the non-binary Ising model $\mathcal{D} \equiv \mathcal{D}(\mathcal{W},\theta)$ is the distribution on $[k]^n$ where $\mathbf{Pr}_{X\sim\mathcal{D}}[X = x] \propto \exp(\sum_{i\neq j} W_{ij}(x_i,x_j) + \sum_i \theta_i(x_i))$. The dependency graph G of $\mathcal{D}$ is formed by pairs such that $W_{ij} \neq 0$. The goal here is to learn the structure and parameters of a general alphabet Ising model. The paper give a sample-efficient, online algorithm for learning the structure and parameters.

3 Problem Importance

Probabilistic graphical modeling can be used to solve problems in fields as diverse as medicine, language processing, vision, and many others. Developing efficient algorithms for inferring the structure of the underlying graph from random samples from a distribution $\mathbb{D}$ is one of the critical problem in probabilistic graphical modeling. The algorithm proposed by the authors starts from learning GLMs, extends to learning Ising models, and further contributes to learning general RMF models. The core idea of the algorithm can be extend to a set of probabilistic graphical models.

4 Previous work

The current frontier of MRF learning has focused on the Ising model on bounded-degree graphs, a special class of graphical models with only pairwise interactions and vertices having degreee at most d in the underlying dependency graph [5].

Previous work on learning $t - wise$ MRFs runs in time of $n^{\Omega(d)}$ and does not output a function f that could generate an approximation to the distribution in statistical distance, even for the special case where $t = 3$. The authors of this paper give a nearly optimal by applying a simple reduction from the problem of learning sparse parties with noise on t variables to learning $t - wise$ MRFs.

Bresler observes that even for the simplest possible Ising model where the graph has a single edge, beating $O(n^2)$ run-time corresponds to fast algorithms for the well-studied *light bulb* problem, and thus the run-time is bounded by $O(n^{1.62})$ [2].

5 Results

The paper talks about a multiplicative-weight update algorithm for learning GLMs, Ising models, and MRFs. The authors claim that they obtain the following new results (contributions):

- The proposed algorithm is an efficient online algorithm for learning Ising models on arbitrary graphs with nearly optimal sample complexity and running time $\tilde{O}(n^2)$. Particularly, they achieve a run-time of $\tilde{O}(n^2)$ with nearly optimal sample complexity for bounded degree graphs. This improves all prior works.
- The algorithm is the first that works for *unbounded-degree* graphs as long as the l_1 norm of the weight vector of each neighborhood is bounded.
- It is an algorithm for learning the dependency graph of binary $t - wise$ Markove random fields with nearly optimal sample complexity and run-time $n^{O(t)}$. Given access to $n^{O(t)}$ samples, the algorithm could reconstruct the parameters of the models and output a $t - wise$ MRF that gives a point-wise approximation to the original distribution.
- The algorithm is easy to implement, which has only one tunable parameter, and works in an online fashion. The *Sparsitron* algorithm solves the problem of learning a sparse Generalized Linear Model. Given $(X, Y) \in [-1, 1]^n \times [0, 1]$ drawn from a distribution $\mathcal{D}$ with the property that $\mathbb{E}[Y|X = x] = \sigma(w \cdot x)$ for some monotonic, where Lipschitz σ and unknown w with $||w||_1 \leq \lambda$, the *Sparsitron* efficiently outputs a w' such that $\sigma(w' \cdot x)$ is close to $\sigma(w \cdot x)$ in *squared-loss* and has sample complexity $O(\lambda^2 \log n)$.

6 Interesting Parts

There are several interesting points in the paper which impressed us, or became more clear and intuitive after our discussion.

- For a graph $G((v_1, ..., v_n), E)$ with n and a binary t-wise MRF on G, how to turn the unsupervised learning problem into a supervised manner?

  Assume
  $$Pr_{Z\sim D}(Z = z) \propto \exp(\sum_{I\in C_t(G)} \varphi_I(z)) \tag{2}$$
  where $\varphi_I : \mathbb{R}^n \to \mathbb{R}$ is a multilinear polynomial depends only on the variables of clique I, the unsupervised task is to learn $\varphi_I(z)$ given m observed samples $Z^1, Z^2, ..., Z^m$, $(\forall i, Z^i = (Z^i_1, Z^i_2, ..., Z^i_n))$. Let's see how to learn $\sum \varphi_I(z)$ for any I that contains node Z_n.

  We know that multilinear polynomial $\sum_{I\in C_t(G)} \varphi_I(z) := p(z) = z_n\partial_n p + p'$ where $\partial_n p, p'$ don't depend on z_n, so
  $$Pr(Z_n = -1|Z_{-n} = x) = \frac{Pr(Z_n = -1, Z_{-n} = x)}{Pr(Z_{-n} = x)} := K\exp(-\partial_n p(x) + p'(x)) \tag{3}$$
  and for the same reason $Pr(Z_n = 1|Z_{-n} = x) = K\exp(\partial_n p(x) + p'(x))$. So
  $$Pr(Z_n = -1|Z_{-n} = x) = \frac{Pr(Z_n = -1|Z_{-n} = x)}{Pr(Z_n = -1|Z_{-n} = x) + Pr(Z_n = 1|Z_{-n} = x)} = \sigma(-2\partial_n p(x)) \tag{4}$$
  where $\sigma(s) = 1/(1 + e^{-s})$ is the sigmoid function. Let $Y_n = \frac{Z_n-1}{2}$, we get
  $$E(Y|Z_{-n} = x) = 1 \cdot Pr(Z_n = 1|Z_{-n} = x) = \sigma(-2\partial_n p(x)) \tag{5}$$
  Now the problem becomes supervised: given observed samples Z_{-n}, labels Y and $E(Y|Z_{-n})$, learn $\partial_n p$. This is the well-known significant *Generalized Linear Model* (GLM) [4] problem in machine learning. As such, ***Sparsitron* provides a way to learn GLM using fewer data with faster convergence**, which might be inspiring to our research.

  The same thing can be done for each $Z_i, i \in [n]$, finally we can combine learned parameters to get the final $p(z)$. In particular, when $t = 2$, the model becomes Ising model and $\partial_i p$ is exactly the weights and bias between connected nodes.

- What is the intuition of *Sparsitron* for learning GLM?

  The algorithm is in Figure 1. We want to estimate the vector w in the function $u(w \cdot x)$ given training samples and labels $(x^1, y^1), (x^2, y^2), ..., (x^n, y^n)$. From an online expert voting perspective, the coordinates of a data point can be seen as actions of experts, and *Sparsitron* learns p as weights to combine the votes of experts.

  The first key is to define *penalty* l^t_i for each expert i at time t. Recall that $alg_t = \sum_i \lambda p^t_i l^t_i$ and we want to update p^t according to l^t. Let's check $s := (u(\lambda p^t \cdot x^t) - y^t)\lambda p^t \cdot x^t$, taking u as the sigmoid function (i.e., Logistic Regression). If the model misclassifies a data point x_*, for the case of $y = 1$, we must get $p \cdot x_* \le 0$ and $0 \le u(p \cdot x_*) \le \frac{1}{2}$, so $\frac{1}{2} \le s \le 1$. In the case of $y = 0$, we must get $p \cdot x_* \ge 0$ and $\frac{1}{2} \le u(p \cdot x_*) \le 1$, we still get $\frac{1}{2} \le s \le 1$. On the other hand, we get $-\frac{1}{2} \le s \le 0$ whenever a data point is classified correctly. As such, we can define $(u(\lambda p^t \cdot x^t) - y^t)x^t$ as the penalty and further shift and scale it to $[0, 1]^n$.

  The second key is to update p, i.e., when a data point is misclassified, $l_i < 0$, we want to decrease (penalize) p_i, so $w^t_i = w^t_i\beta^{l^t_i}$ satisfies it with the learning rate $\beta < 1$. Note that the exponential update scheme is called Hedge algorithm [3], which is different from $w^t_i = w^{t-1}_i(1 - \beta l^t_i)$.

  Intuitively, the penalty l_i for p_i is determined by two terms: the closeness of prediction $(u(\lambda p^t \cdot x^t) - y^t$ and the scale of the i^{th} entry of x. If the expert i votes a high score x_i and vote far from the true value y, he gets penalized heavily.

- How does the paper prove that the underlying graph structure of t-wise MRF can be uncovered v.s. the distribution parameters can be learned?

  Basically, learning the distribution parameters implies uncovering the graph structures. Specifically, for a t-wise MRF, the paper proves that the sample complexity for the former is $\frac{(2t)^{\mathcal{O}(t)}e^{\mathcal{O}(\lambda t)}}{\epsilon^4} \cdot \log(\frac{n}{\rho\epsilon})$ while the one for the latter is smaller, $\frac{(e)^{\mathcal{O}(t)}e^{\mathcal{O}(\lambda t)}}{\epsilon^4} \cdot \log(\frac{n}{\rho\epsilon})$.

  To estimate the distribution, *Sparsitron* actually constructs a mutilinear polynomial function which approximates the real multilinear polynomial by l_1 form, and this implies that the constructed function *uniformly* approximates the real one given that all variables are within $[-1, 1]$. In contrast, the graph structure can be fully determined by all maximal monomials of the distribution polynomial. This is because **in order to know if any node v_1, v_2 are connected, we only need to check the max clique containing v_1, v_2**, which corresponds to the maximal monomial of x_1, x_2. In other words, suppose the real distribution function of the MRF is $3x_1x_2x_3x_4 + x_1x_2$, once we construct a polynomial $ax_1x_2x_3x_4, (a \neq 0)$, we uncover the graph structure. **Note that for a 2-wise MRF, i.e., Ising model, running *Sparsitron* for the latter already infers all monomials of the real distribution polynomial.**

Algorithm 2 SPARSITRON

Initialize $w^0 = \mathbf{1}/n$.
for $t = 1, \ldots, T$ **do**
  Let $p^t = w^{t-1}/\|w^{t-1}\|_1$.
  Define $\ell^t \in \mathbb{R}^n$ by $\ell^t = (1/2)(\mathbf{1} + (u(\lambda p^t \cdot x^t) - y^t)x^t)$.
  Update the weight vectors w^t: for each $i \in [n]$, set $w_i^t = w_i^{t-1} \cdot \beta^{\ell_i^t}$.

Figure 1: *Sparsitron* for learning Generalized Linear Models (GLM)

7 New Questions

We come up with several questions to this paper:

- Since it works for Ising models, we are wondering how it works in the context of deep learning, for example Deep Belief Network (stacked Restricted Boltzman Machines). Since *Sparsitron* is a multiplicative update method which requires less training samples and provides faster convergence, it will be powerful if we could find a way to apply it to current scheme of training a deep neural network. For example, current deep learning methods require huge training data (ImageNet as an example) to learn, and the training tasks usually cost several weeks without GPU. To apply *Sparsitron* as a replacement of stochastic gradient descent strategy (additive update method), we might address two challenges:
  - Find a way to combine losses for all layers of a neural network. In particular, overcome the vanish or explosion of losses while propogating between layers.
  - 'stochastic' weights update of *Sparsitron* with some theoretical promise of the convergence, to make the training faster.

8 Experiment

In this section, we validate the claims made by the authors of this paper. Figure 2 shows the structure of the Ising model used to evaluate the algorithm. The weight of each edge is a random number from $[0, 1]$. For

simplicity, we only inspect the structure of Node n, since the structure of the whole graph is based on the neighbors of each node in the graph.

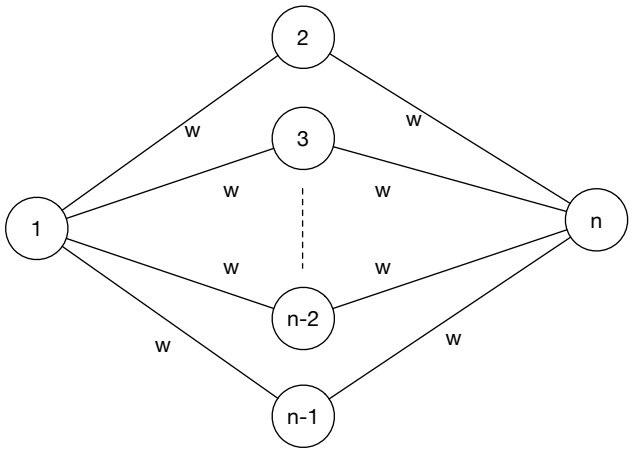

Figure 2: The diamond structure of Ising model. The Ising model include n nodes.

8.1 Number of samples

The authors claim that 'the algorithm works with nearly optimal sample complexity'. To validate this claim, we fix the structure and the size of the Ising model, then evaluate how different sample sizes affect the error by the algorithm. Specifically, we use the exact structure of the Ising model in Figure 2, which include 6 nodes. Thus, the ground truth of the weights is $w_{true} = [0, w_{2,n}, ..., w_{n-1,n}]$, because, Node 1 doesn't connect to Node n. Suppose the weights learned by the algorithm is $w_{pred} = [w_1, ..., w_{n-1}]$, we use the l_2-norm to measure the error, which is $error = ||w_{true} - w_{pred}||_2$. For each sample size, we run the algorithm 20 times and report the average l_2-norm error of each run. Figure 3 shows how the sample size affect the error. We learn from the figure that the algorithm works even for small sample size, which validates their claim that 'the algorithm works with nearly optimal sample complexity'.

8.2 Size of network

We also evaluate how the size of Ising model affects the performance of the algorithm. In this experiment, we vary the number of nodes in the Ising models. Each network is trained with $10 * n^2$ samples, because we see from the previous experiment small number of samples still achieve good performance. As with the previous experiment, we also run the algorithm 20 times and report the average $l2$-norm for each network size. Figure 4 plots the l_2 prediction error against network sizes. We found for network size larger than 7, the prediction error is significantly higher than the error when network size is 5. This may be caused by insufficient samples fed into the algorithm.

8.3 Importance of the parameter

The authors claim that the algorithm is easy to implement, because the algorithm has only one parameter. To show how the parameter affect the performance of the algorithm, we fix the network size and number of samples, but vary the parameter and evaluate the prediction error. Specifically, we use the network size of 6 and sample size of 200. Figure 5 shows the performance of the algorithm under different parameter factors. We found the parameter factor is critical for a good performance of the algorithm. It is almost the factor of the error.

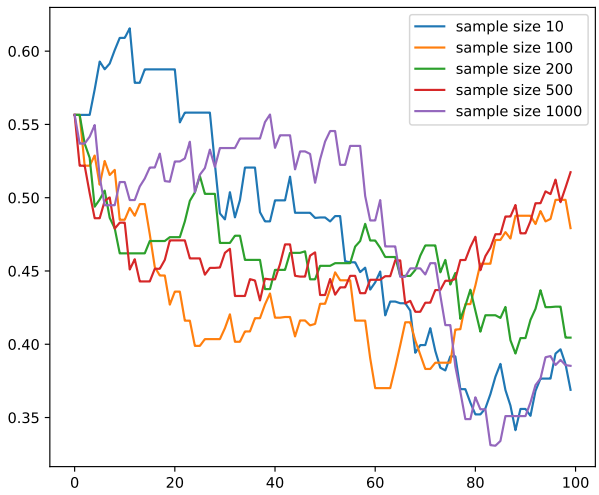

Figure 3: Sample size vs. prediction error. The x-axis represents the iteration numbers. The y-axis represents the average l_2-norm error.

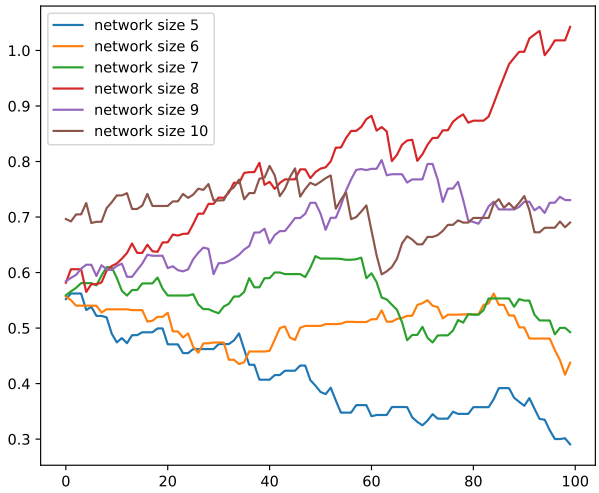

Figure 4: Network size vs. prediction error. The x-axis represents the iteration numbers. The y-axis represents the average l_2-norm error.

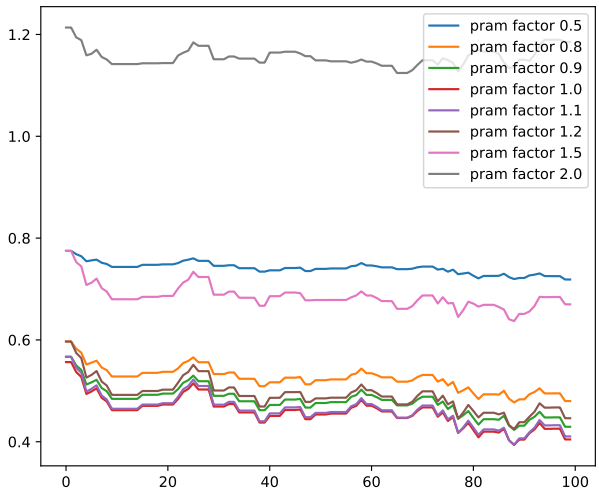

Figure 5: Prameter factor vs. prediction error. The x-axis represents the iteration numbers. The y-axis represents the average l_2-norm error.

References

[1] Probabilistic graphical models notes. https://ermongroup.github.io/cs228-notes/. 1

[2] Guy Bresler. Efficiently learning ising models on arbitrary graphs. In *Proceedings of the annual ACM Symposium on Theory of Computing (SOTC'15)*, pages 771–782, 2015. 3

[3] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. 4

[4] John Ashworth Nelder and R Jacob Baker. Generalized linear models. *Encyclopedia of statistical sciences*, 4, 2004. 4

[5] Marc Vuffray, Sidhant Misra, Andrey Lokhov, and Michael Chertkov. Interaction screening: Efficient and sample-optimal learning of ising models. In *Advances in Neural Information Processing Systems (NIPS'16)*, pages 2595–2603, 2016. 3