Computing the Partition Function of the Sherrington-Kirkpatrick Model is Hard on Average

Eren C. Kızıldağ, joint work with David Gamarnik

MIT

2020 IEEE International Symposium on Information Theory

June, 2020

Overview

Computing the partition function of the SK model

Computing the partition function of the SK model

- **Algorithmic Problem.** Computing *exactly* the partition function of the Sherrington-Kirkpatrick (SK) spin glass model with Gaussian couplings. The algorithmic hardness result.

Computing the partition function of the SK model

- **Algorithmic Problem.** Computing *exactly* the partition function of the Sherrington-Kirkpatrick (SK) spin glass model with Gaussian couplings. The algorithmic hardness result.
- **Model.** Let $n \in \mathbb{Z}^+$, and $\mathbf{J} = (J_{ij} : 1 \le i < j \le n) \in \mathbb{R}^{n(n-1)/2}$, called *couplings*.

Computing the partition function of the SK model

- **Algorithmic Problem.** Computing *exactly* the partition function of the Sherrington-Kirkpatrick (SK) spin glass model with Gaussian couplings. The algorithmic hardness result.
- **Model.** Let $n \in \mathbb{Z}^+$, and $\mathbf{J} = (J_{ij} : 1 \le i < j \le n) \in \mathbb{R}^{n(n-1)/2}$, called *couplings*.
- Consider n *sites* $[n] \triangleq \{1, 2, \ldots, n\}$, and assign a *spin* $\sigma_i \in \{\pm 1\}$ for each $i \in [n]$.

Computing the partition function of the SK model

- **Algorithmic Problem.** Computing *exactly* the partition function of the Sherrington-Kirkpatrick (SK) spin glass model with Gaussian couplings. The algorithmic hardness result.
- **Model.** Let $n \in \mathbb{Z}^+$, and $\mathbf{J} = (J_{ij} : 1 \le i < j \le n) \in \mathbb{R}^{n(n-1)/2}$, called *couplings*.
- Consider n *sites* $[n] \triangleq \{1, 2, \ldots, n\}$, and assign a *spin* $\sigma_i \in \{\pm 1\}$ for each $i \in [n]$.
- Energy of $\boldsymbol{\sigma} = (\sigma_i : i \in [n]) \in \{\pm 1\}^n$ at *inverse temperature* $\beta > 0$ given by Hamiltonian

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1 \le i < j \le n} J_{ij} \sigma_i \sigma_j.$$

Computing the partition function of the SK model

- **Algorithmic Problem.** Computing *exactly* the partition function of the Sherrington-Kirkpatrick (SK) spin glass model with Gaussian couplings. The algorithmic hardness result.
- **Model.** Let $n \in \mathbb{Z}^+$, and $\mathbf{J} = (J_{ij} : 1 \le i < j \le n) \in \mathbb{R}^{n(n-1)/2}$, called *couplings*.
- Consider n *sites* $[n] \triangleq \{1, 2, \ldots, n\}$, and assign a *spin* $\sigma_i \in \{\pm 1\}$ for each $i \in [n]$.
- Energy of $\boldsymbol{\sigma} = (\sigma_i : i \in [n]) \in \{\pm 1\}^n$ at *inverse temperature* $\beta > 0$ given by Hamiltonian

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1 \le i < j \le n} J_{ij} \sigma_i \sigma_j .$$

- An algorithm $\mathcal{A}$ to *exactly* compute the partition function

$$Z(\mathbf{J}, \beta) = \sum_{\boldsymbol{\sigma} \in \{\pm 1\}^n} \exp\left(-H(\boldsymbol{\sigma})\right).$$

Computing the partition function of the SK model

Computing the partition function of the SK model

- Problem of computing $Z(\mathbf{J})$ for arbitrary $\mathbf{J}$ is $\#P-$hard, Valiant [80s].

Computing the partition function of the SK model

- Problem of computing $Z(\mathbf{J})$ for arbitrary $\mathbf{J}$ is $\#P-$hard, Valiant [80s].
- Computing partition function for *arbitrary* input is hard for a broader class of statistical physics models: Barahona [82], Istrail [00], ...

Computing the partition function of the SK model

- Problem of computing $Z(\mathbf{J})$ for arbitrary $\mathbf{J}$ is $\#P-$hard, Valiant [80s].
- Computing partition function for *arbitrary* input is hard for a broader class of statistical physics models: Barahona [82], Istrail [00], ...
- **Requirement.** For *random* $\mathbf{J}$,

$$\mathbb{P}\left(Z_{\mathcal{A}}(\mathbf{J}) = Z(\mathbf{J})\right) \geq \delta,$$

probability with respect to draw of $\mathbf{J}$.

Computing the partition function of the SK model

- Problem of computing $Z(\mathbf{J})$ for arbitrary $\mathbf{J}$ is $\#P-$hard, Valiant [80s].
- Computing partition function for *arbitrary* input is hard for a broader class of statistical physics models: Barahona [82], Istrail [00], ...
- **Requirement.** For *random* $\mathbf{J}$,

$$\mathbb{P}\left(Z_{\mathcal{A}}(\mathbf{J}) = Z(\mathbf{J})\right) \geq \delta,$$

  probability with respect to draw of $\mathbf{J}$.
- Thus, our goal is *average-case* hardness. Classical reduction techniques for *worst-case* hardness do not transfer.

Computing the partition function of the SK model

- Problem of computing $Z(\mathbf{J})$ for arbitrary $\mathbf{J}$ is $\#P-$hard, Valiant [80s].
- Computing partition function for *arbitrary* input is hard for a broader class of statistical physics models: Barahona [82], Istrail [00], ...
- **Requirement.** For *random* $\mathbf{J}$,

$$\mathbb{P}\left(Z_{\mathcal{A}}(\mathbf{J}) = Z(\mathbf{J})\right) \geq \delta,$$

  probability with respect to draw of $\mathbf{J}$.
- Thus, our goal is *average-case* hardness. Classical reduction techniques for *worst-case* hardness do not transfer.
- Of interest in cryptography and TCS. Examples include shortest lattice vector problem (Ajtai [96]), and permanent (Lipton [89], Feige and Lund [92], Cai et al. [99]).

Overview

Part I. Hardness under Finite Precision Arithmetic. Modified Model

Part I. Hardness under Finite Precision Arithmetic. Modified Model

- A_i, $1 \le i \le n$, independent mean zero normal, called *external field*. Modified Hamiltonian:

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1 \le i < j \le n} J_{ij}\sigma_i\sigma_j + \sum_{1 \le i \le n} A_i\sigma_i.$$

Corresponding partition function $Z_1(\mathbf{J}, \mathbf{A})$, where $\mathbf{A} = (A_i : 1 \le i \le n)$.

Part I. Hardness under Finite Precision Arithmetic. Modified Model

- A_i, $1 \le i \le n$, independent mean zero normal, called *external field*. Modified Hamiltonian:

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1 \le i < j \le n} J_{ij}\sigma_i\sigma_j + \sum_{1 \le i \le n} A_i\sigma_i.$$

  Corresponding partition function $Z_1(\mathbf{J}, \mathbf{A})$, where $\mathbf{A} = (A_i : 1 \le i \le n)$.

- We study alternative Hamiltonian

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1 \le i < j \le n} J_{ij}\sigma_i\sigma_j + \sum_{1 \le i \le n} B_i\sigma_i - \sum_{1 \le i \le n} C_i\sigma_i.$$

  B_i, $1 \le i \le n$ and C_i, $1 \le i \le n$ independent, zero-mean; partition function $Z_2(\mathbf{J}, \mathbf{B}, \mathbf{C})$.

Part I. Hardness under Finite Precision Arithmetic. Modified Model

- A_i, $1 \le i \le n$, independent mean zero normal, called *external field*. Modified Hamiltonian:

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1\le i<j\le n} J_{ij}\sigma_i\sigma_j + \sum_{1\le i\le n} A_i\sigma_i.$$

  Corresponding partition function $Z_1(\mathbf{J},\mathbf{A})$, where $\mathbf{A} = (A_i : 1 \le i \le n)$.

- We study alternative Hamiltonian

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1\le i<j\le n} J_{ij}\sigma_i\sigma_j + \sum_{1\le i\le n} B_i\sigma_i - \sum_{1\le i\le n} C_i\sigma_i.$$

  B_i, $1 \le i \le n$ and C_i, $1 \le i \le n$ independent, zero-mean; partition function $Z_2(\mathbf{J},\mathbf{B},\mathbf{C})$.

- Equivalence: if $\mathcal{A}_1$ with input $(\mathbf{J},\mathbf{A})$ computes $Z_1(\mathbf{J},\mathbf{A})$ then $\mathcal{A}_1$ with input $(\mathbf{J},\mathbf{B}-\mathbf{C})$ computes $Z_2(\mathbf{J},\mathbf{B},\mathbf{C})$. If $\mathcal{A}_2$ with input $(\mathbf{Z},\mathbf{B},\mathbf{C})$ computes $Z_2(\mathbf{J},\mathbf{B},\mathbf{C})$ then $\mathcal{A}_2$ with input $(\mathbf{J},\frac{\mathbf{G}+\mathbf{A}}{2},\frac{\mathbf{G}-\mathbf{A}}{2})$ computes $Z_1(\mathbf{J},\mathbf{A})$, where $\mathbf{G} = (G_i : 1 \le i \le n)$ i.i.d. copy of $\mathbf{A}$.

Part I. Hardness under Finite Precision Arithmetic. Cuts/Polarities.

Part I. Hardness under Finite Precision Arithmetic. Cuts/Polarities.

- Thus our focus is on computing partition function $Z(\mathbf{J},\mathbf{B},\mathbf{C})$ for Hamiltonian

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1\le i<j\le n} J_{ij}\sigma_i\sigma_j + \sum_{1\le i\le n} B_i\sigma_i - \sum_{1\le i\le n} C_i\sigma_i.$$

Part I. Hardness under Finite Precision Arithmetic. Cuts/Polarities.

- Thus our focus is on computing partition function $Z(\mathbf{J},\mathbf{B},\mathbf{C})$ for Hamiltonian

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1\le i<j\le n} J_{ij}\sigma_i\sigma_j + \sum_{1\le i\le n} B_i\sigma_i - \sum_{1\le i\le n} C_i\sigma_i.$$

- Incorporate *cuts and polarities* induced by $\boldsymbol{\sigma} \in \{\pm 1\}^n$: set

$$\Sigma_{\boldsymbol{\sigma}}^{+} \triangleq \frac{\beta}{\sqrt{n}} \sum_{\sigma_i=\sigma_j} J_{ij} + \sum_{\sigma_i=+1} B_i + \sum_{\sigma_i=-1} C_i \quad \text{and} \quad \Sigma_{\boldsymbol{\sigma}}^{-} \triangleq \frac{\beta}{\sqrt{n}} \sum_{\sigma_i\ne\sigma_j} J_{ij} + \sum_{\sigma_i=-1} B_i + \sum_{\sigma_i=+1} C_i.$$

Note that $H(\boldsymbol{\sigma}) = \Sigma_{\boldsymbol{\sigma}}^{+} - \Sigma_{\boldsymbol{\sigma}}^{-}$. Furthermore, $\Sigma \triangleq \Sigma_{\boldsymbol{\sigma}}^{+} + \Sigma_{\boldsymbol{\sigma}}^{-} = \sum_{i<j} J_{ij} + \sum_i (B_i + C_i)$ independent of $\boldsymbol{\sigma}$ and polynomial-time computable.

Part I. Hardness under Finite Precision Arithmetic. Cuts/Polarities.

- Thus our focus is on computing partition function $Z(\mathbf{J},\mathbf{B},\mathbf{C})$ for Hamiltonian

$$H(\boldsymbol{\sigma}) = \frac{\beta}{\sqrt{n}} \sum_{1\le i<j\le n} J_{ij}\sigma_i\sigma_j + \sum_{1\le i\le n} B_i\sigma_i - \sum_{1\le i\le n} C_i\sigma_i.$$

- Incorporate *cuts and polarities* induced by $\boldsymbol{\sigma} \in \{\pm 1\}^n$: set

$$\Sigma_{\boldsymbol{\sigma}}^{+} \triangleq \frac{\beta}{\sqrt{n}} \sum_{\sigma_i=\sigma_j} J_{ij} + \sum_{\sigma_i=+1} B_i + \sum_{\sigma_i=-1} C_i \quad \text{and} \quad \Sigma_{\boldsymbol{\sigma}}^{-} \triangleq \frac{\beta}{\sqrt{n}} \sum_{\sigma_i\ne\sigma_j} J_{ij} + \sum_{\sigma_i=-1} B_i + \sum_{\sigma_i=+1} C_i.$$

  Note that $H(\boldsymbol{\sigma}) = \Sigma_{\boldsymbol{\sigma}}^{+} - \Sigma_{\boldsymbol{\sigma}}^{-}$. Furthermore, $\Sigma \triangleq \Sigma_{\boldsymbol{\sigma}}^{+} + \Sigma_{\boldsymbol{\sigma}}^{-} = \sum_{i<j} J_{ij} + \sum_i (B_i + C_i)$ independent of $\boldsymbol{\sigma}$ and polynomial-time computable.

- Thus $Z(\mathbf{J},\mathbf{B},\mathbf{C}) = \sum_{\boldsymbol{\sigma}\in\{\pm1\}^n} \exp(-H(\boldsymbol{\sigma})) = \sum_{\boldsymbol{\sigma}\in\{\pm1\}^n} \exp(-\Sigma)\exp(2\Sigma_{\sigma}^{-})$ is computable iff $\sum_{\boldsymbol{\sigma}\in\{\pm1\}^n} \exp(2\Sigma_{\boldsymbol{\sigma}}^{-})$ is computable. Ignore 2.

Part I. Hardness under Finite Precision Arithmetic. Truncation.

Part I. Hardness under Finite Precision Arithmetic. Truncation.

- Let $\widehat{J_{ij}} = \exp(\beta J_{ij}/\sqrt{n})$, $\widehat{B_i} = \exp(B_i)$, and $\widehat{C_i} = \exp(C_i)$.

Part I. Hardness under Finite Precision Arithmetic. Truncation.

- Let $\widehat{J}_{ij} = \exp(\beta J_{ij}/\sqrt{n})$, $\widehat{B}_i = \exp(B_i)$, and $\widehat{C}_i = \exp(C_i)$.
- **Truncation:** Fix $N \in \mathbb{Z}_+$, let $x^{[N]} \triangleq 2^{-N}\lfloor 2^N x \rfloor$. Truncate inputs: $\widehat{J}_{ij}^{[N]}$, $\widehat{B}_i^{[N]}$, and $\widehat{C}_i^{[N]}$. Goal is to compute

$$Z(\widehat{\mathbf{J}}^{[\mathbf{N}]}, \widehat{\mathbf{B}}^{[\mathbf{N}]}, \widehat{\mathbf{C}}^{[\mathbf{N}]}) = \sum_{\boldsymbol{\sigma} \in \{-1,1\}^n} \left(\prod_{\sigma_i \neq \sigma_j} \widehat{J}_{ij}^{[N]} \right) \left(\prod_{\sigma_i = -1} \widehat{B}_i^{[N]} \right) \left(\prod_{\sigma_i = +1} \widehat{C}_i^{[N]} \right).$$

Part I. Hardness under Finite Precision Arithmetic. Truncation.

- Let $\widehat{J}_{ij} = \exp(\beta J_{ij}/\sqrt{n})$, $\widehat{B}_i = \exp(B_i)$, and $\widehat{C}_i = \exp(C_i)$.
- **Truncation:** Fix $N \in \mathbb{Z}_+$, let $x^{[N]} \triangleq 2^{-N}\lfloor 2^N x \rfloor$. Truncate inputs: $\widehat{J}_{ij}^{[N]}, \widehat{B}_i^{[N]}$, and $\widehat{C}_i^{[N]}$. Goal is to compute

$$Z(\widehat{\mathbf{J}}^{[\mathbf{N}]}, \widehat{\mathbf{B}}^{[\mathbf{N}]}, \widehat{\mathbf{C}}^{[\mathbf{N}]}) = \sum_{\boldsymbol{\sigma} \in \{-1,1\}^n} \left(\prod_{\sigma_i \neq \sigma_j} \widehat{J}_{ij}^{[N]} \right) \left(\prod_{\sigma_i = -1} \widehat{B}_i^{[N]} \right) \left(\prod_{\sigma_i = +1} \widehat{C}_i^{[N]} \right).$$

- **Switching to Integer Inputs:** Define $\widetilde{J}_{ij} \triangleq 2^N \widehat{J}_{ij}^{[N]} \in \mathbb{Z}$, and $\widetilde{B}_i, \widetilde{C}_i$ similarly. Focus:

$$Z_n(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}) = \sum_{\boldsymbol{\sigma} \in \{-1,1\}^n} 2^{Nf(n,\boldsymbol{\sigma})} \left(\prod_{\sigma_i \neq \sigma_j} \widetilde{J}_{ij} \right) \left(\prod_{\sigma_i = -1} \widetilde{B}_i \right) \left(\prod_{\sigma_i = +1} \widetilde{C}_i \right),$$

where $f(n, \boldsymbol{\sigma}) = n(n-1)/2 - n - |\{(i,j) : 1 \leq i < j \leq n, \sigma_i \neq \sigma_j\}|$.

Part I. Hardness under Finite Precision Arithmetic. Truncation.

- Let $\widehat{J}_{ij} = \exp(\beta J_{ij}/\sqrt{n})$, $\widehat{B}_i = \exp(B_i)$, and $\widehat{C}_i = \exp(C_i)$.
- **Truncation:** Fix $N \in \mathbb{Z}_+$, let $x^{[N]} \triangleq 2^{-N}\lfloor 2^N x \rfloor$. Truncate inputs: $\widehat{J}_{ij}^{[N]}, \widehat{B}_i^{[N]}$, and $\widehat{C}_i^{[N]}$. Goal is to compute

$$Z(\widehat{\mathbf{J}}^{[\mathbf{N}]}, \widehat{\mathbf{B}}^{[\mathbf{N}]}, \widehat{\mathbf{C}}^{[\mathbf{N}]}) = \sum_{\boldsymbol{\sigma} \in \{-1,1\}^n} \left(\prod_{\sigma_i \neq \sigma_j} \widehat{J}_{ij}^{[N]} \right) \left(\prod_{\sigma_i = -1} \widehat{B}_i^{[N]} \right) \left(\prod_{\sigma_i = +1} \widehat{C}_i^{[N]} \right).$$

- **Switching to Integer Inputs:** Define $\widetilde{J}_{ij} \triangleq 2^N \widehat{J}_{ij}^{[N]} \in \mathbb{Z}$, and $\widetilde{B}_i, \widetilde{C}_i$ similarly. Focus:

$$Z_n(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}) = \sum_{\boldsymbol{\sigma} \in \{-1,1\}^n} 2^{Nf(n,\boldsymbol{\sigma})} \left(\prod_{\sigma_i \neq \sigma_j} \widetilde{J}_{ij} \right) \left(\prod_{\sigma_i = -1} \widetilde{B}_i \right) \left(\prod_{\sigma_i = +1} \widetilde{C}_i \right),$$

where $f(n, \boldsymbol{\sigma}) = n(n-1)/2 - n - |\{(i,j) : 1 \leq i < j \leq n, \sigma_i \neq \sigma_j\}|$.
- Observe that $Z_n(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}) = 2^{Nn(n-1)/2} Z(\widehat{\mathbf{J}}^{[\mathbf{N}]}, \widehat{\mathbf{B}}^{[\mathbf{N}]}, \widehat{\mathbf{C}}^{[\mathbf{N}]}) \in \mathbb{Z}$.

Part I. Hardness under Finite Precision Arithmetic. Main Result

Part I. Hardness under Finite Precision Arithmetic. Main Result

Theorem (Gamarnik & K., 2019)

Let $k, \alpha, \epsilon > 0$ *be arbitrary constants. Suppose that the precision value* N *satisfies* $(3\alpha + 21k/2 + 10 + \epsilon)\log n \le N \le n^{\alpha}$, *and that there exists a polynomial-in-n time algorithm* $\mathcal{A}$, *which, on input* $(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ *produces a value* $Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ *such that* $\mathbb{P}\left(Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}) = Z_n(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})\right) \ge 1/n^k$ *for all sufficiently large* n*. Then,* $P = \#P$.

Part I. Hardness under Finite Precision Arithmetic. Main Result

Theorem (Gamarnik & K., 2019)

Let $k, \alpha, \epsilon > 0$ *be arbitrary constants. Suppose that the precision value* N *satisfies* $(3\alpha + 21k/2 + 10 + \epsilon)\log n \le N \le n^{\alpha}$*, and that there exists a polynomial-in-n time algorithm* $\mathcal{A}$*, which, on input* $(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ *produces a value* $Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ *such that* $\mathbb{P}\left(Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}) = Z_n(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})\right) \ge 1/n^k$ *for all sufficiently large* n*. Then,* $P = \#P$*.*

Comments.

Part I. Hardness under Finite Precision Arithmetic. Main Result

Theorem (Gamarnik & K., 2019)

Let $k, \alpha, \epsilon > 0$ be arbitrary constants. Suppose that the precision value N satisfies $(3\alpha + 21k/2 + 10 + \epsilon)\log n \le N \le n^{\alpha}$, and that there exists a polynomial-in-n time algorithm $\mathcal{A}$, which, on input $(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ produces a value $Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ such that $\mathbb{P}\left(Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}) = Z_n(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})\right) \ge 1/n^k$ for all sufficiently large n. Then, $P = \#P$.

Comments.

- Probability taken with respect to randomness in $(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$, which originates from randomness in input $(\mathbf{J}, \mathbf{B}, \mathbf{C})$.

Part I. Hardness under Finite Precision Arithmetic. Main Result

Theorem (Gamarnik & K., 2019)

Let $k, \alpha, \epsilon > 0$ be arbitrary constants. Suppose that the precision value N satisfies $(3\alpha + 21k/2 + 10 + \epsilon)\log n \le N \le n^{\alpha}$, and that there exists a polynomial-in-n time algorithm $\mathcal{A}$, which, on input $(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ produces a value $Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ such that $\mathbb{P}\left(Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}) = Z_n(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})\right) \ge 1/n^k$ for all sufficiently large n. Then, $P = \#P$.

Comments.

- Probability taken with respect to randomness in $(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$, which originates from randomness in input $(\mathbf{J}, \mathbf{B}, \mathbf{C})$.
- Number N of bits in precision is at least logarithmic and at most polynomial in n.

Part I. Hardness under Finite Precision Arithmetic. Main Result

Theorem (Gamarnik & K., 2019)

Let $k,\alpha,\epsilon>0$ be arbitrary constants. Suppose that the precision value N satisfies $(3\alpha+21k/2+10+\epsilon)\log n\le N\le n^\alpha$, and that there exists a polynomial-in-n time algorithm $\mathcal{A}$, which, on input $(\widetilde{\mathbf{J}},\widetilde{\mathbf{B}},\widetilde{\mathbf{C}})$ produces a value $Z_{\mathcal{A}}(\widetilde{\mathbf{J}},\widetilde{\mathbf{B}},\widetilde{\mathbf{C}})$ such that $\mathbb{P}\left(Z_{\mathcal{A}}(\widetilde{\mathbf{J}},\widetilde{\mathbf{B}},\widetilde{\mathbf{C}})=Z_n(\widetilde{\mathbf{J}},\widetilde{\mathbf{B}},\widetilde{\mathbf{C}})\right)\ge 1/n^k$ for all sufficiently large n. Then, $P=\#P$.

Comments.

- Probability taken with respect to randomness in $(\widetilde{\mathbf{J}},\widetilde{\mathbf{B}},\widetilde{\mathbf{C}})$, which originates from randomness in input $(\mathbf{J},\mathbf{B},\mathbf{C})$.
- Number N of bits in precision is at least logarithmic and at most polynomial in n.
- Upper bound ensures bit stream supplied to algorithm is of polynomial length.

Part I. Hardness under Finite Precision Arithmetic. Main Result

Theorem (Gamarnik & K., 2019)

Let $k, \alpha, \epsilon > 0$ be arbitrary constants. Suppose that the precision value N satisfies $(3\alpha + 21k/2 + 10 + \epsilon)\log n \le N \le n^{\alpha}$, and that there exists a polynomial-in-n time algorithm $\mathcal{A}$, which, on input $(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ produces a value $Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$ such that $\mathbb{P}\left(Z_{\mathcal{A}}(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}}) = Z_n(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})\right) \ge 1/n^k$ for all sufficiently large n. Then, $P = \#P$.

Comments.

- Probability taken with respect to randomness in $(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$, which originates from randomness in input $(\mathbf{J}, \mathbf{B}, \mathbf{C})$.
- Number N of bits in precision is at least logarithmic and at most polynomial in n.
- Upper bound ensures bit stream supplied to algorithm is of polynomial length.
- Lower bound required for technical reasons when establishing near-uniformity of $(\widetilde{\mathbf{J}}, \widetilde{\mathbf{B}}, \widetilde{\mathbf{C}})$.

Idea of Proof.

Idea of Proof.

- Inspired from average-case hardness proof by Cai et al. [99] for computing permanent over a finite field. Recall that for an $A\in\mathbb{R}^{m\times m}$,

$$\mathrm{permanent}(A)=\sum_{\sigma\in S_n}\prod_{1\le i\le n}a_{i,\sigma(i)},$$

where S_n is the set of all permutations of $\{1,2,\ldots,n\}$. $\#P-$hard to compute for *arbitrary inputs*.

Idea of Proof.

- Inspired from average-case hardness proof by Cai et al. [99] for computing permanent over a finite field. Recall that for an $A \in \mathbb{R}^{m\times m}$,

$$\text{permanent}(A) = \sum_{\sigma\in S_n} \prod_{1\le i\le n} a_{i,\sigma(i)},$$

where S_n is the set of all permutations of $\{1,2,\dots,n\}$. $\#P-$hard to compute for *arbitrary inputs*.

- Let $\mathbb{Z}_p$ be a finite field. Permanent of a $M \in \mathbb{Z}_p^{n\times n}$ equals to a weighted sum of permanents of n minors $M_{11},\dots,M_{n1} \in \mathbb{Z}_p^{(n-1)\times(n-1)}$.

Idea of Proof.

- Inspired from average-case hardness proof by Cai et al. [99] for computing permanent over a finite field. Recall that for an $A \in \mathbb{R}^{m\times m}$,

$$\text{permanent}(A) = \sum_{\sigma \in S_n} \prod_{1\le i\le n} a_{i,\sigma(i)},$$

  where S_n is the set of all permutations of $\{1,2,\dots,n\}$. $\#P-$hard to compute for *arbitrary inputs*.
- Let $\mathbb{Z}_p$ be a finite field. Permanent of a $M \in \mathbb{Z}_p^{n\times n}$ equals to a weighted sum of permanents of n minors $M_{11},\dots,M_{n1} \in \mathbb{Z}_p^{(n-1)\times(n-1)}$.
- Construct a *matrix polynomial* whose value at $k \in \{1,2,\dots,n\}$ is minor M_{k1}. The permanent of this matrix polynomial is a low-degree univariate polynomial. Call it φ.

Idea of Proof (Cont'd).

Idea of Proof (Cont'd).

- Assume there exists a polynomial-time algorithm $\mathcal{A}$ to exactly compute permanent on a fraction of all inputs. Use $\mathcal{A}$ to generate a *list* of noisy samples of φ.

Idea of Proof (Cont'd).

- Assume there exists a polynomial-time algorithm $\mathcal{A}$ to exactly compute permanent on a fraction of all inputs. Use $\mathcal{A}$ to generate a *list* of noisy samples of φ.
- Reconstruct φ from its noisy samples: list decoding (Berlekamp-Welch [86], Sudan [96]).

Idea of Proof (Cont'd).

- Assume there exists a polynomial-time algorithm $\mathcal{A}$ to exactly compute permanent on a fraction of all inputs. Use $\mathcal{A}$ to generate a *list* of noisy samples of φ.
- Reconstruct φ from its noisy samples: list decoding (Berlekamp-Welch [86], Sudan [96]).
- Thus, if $\mathcal{A}$ exists, permanent of an *arbitrary* A can be computed, implying $P = \#P$.

Idea of Proof (Cont'd).

- Assume there exists a polynomial-time algorithm $\mathcal{A}$ to exactly compute permanent on a fraction of all inputs. Use $\mathcal{A}$ to generate a *list* of noisy samples of φ.
- Reconstruct φ from its noisy samples: list decoding (Berlekamp-Welch [86], Sudan [96]).
- Thus, if $\mathcal{A}$ exists, permanent of an *arbitrary* A can be computed, implying $P = \#P$.

Technical Challenges for the SK Model.

Idea of Proof (Cont'd).

- Assume there exists a polynomial-time algorithm $\mathcal{A}$ to exactly compute permanent on a fraction of all inputs. Use $\mathcal{A}$ to generate a *list* of noisy samples of φ.
- Reconstruct φ from its noisy samples: list decoding (Berlekamp-Welch [86], Sudan [96]).
- Thus, if $\mathcal{A}$ exists, permanent of an *arbitrary* A can be computed, implying $P = \#P$.

Technical Challenges for the SK Model.

- Not clear if a Laplace-like self-recursion takes place for partition function.

Idea of Proof (Cont'd).

- Assume there exists a polynomial-time algorithm $\mathcal{A}$ to exactly compute permanent on a fraction of all inputs. Use $\mathcal{A}$ to generate a *list* of noisy samples of φ.
- Reconstruct φ from its noisy samples: list decoding (Berlekamp-Welch [86], Sudan [96]).
- Thus, if $\mathcal{A}$ exists, permanent of an *arbitrary* A can be computed, implying $P = \#P$.

Technical Challenges for the SK Model.

- Not clear if a Laplace-like self-recursion takes place for partition function.
- Hardness results above address uniform input over $\mathbb{Z}_p$. We have truncated log-normals.

Proof Sketch.

Proof Sketch.

For an $n-$spin system, $Z_n(\cdot)$ requires (integer) input, of size $n(n-1)/2+2n$. We follow an outline similar to Cai et al. [99] for permanent.

Proof Sketch.

For an $n-$spin system, $Z_n(\cdot)$ requires (integer) input, of size $n(n-1)/2+2n$. We follow an outline similar to Cai et al. [99] for permanent.

- Let $p_n > 9n^{2k+2}$ be a prime. For any $\Xi \in \mathbb{Z}^{n(n-1)/2+2n}$, let $Z_n(\Xi;p_n) \triangleq Z_n(\Xi) \pmod{p_n}$.

Proof Sketch.

For an $n-$spin system, $Z_n(\cdot)$ requires (integer) input, of size $n(n-1)/2+2n$. We follow an outline similar to Cai et al. [99] for permanent.

- Let $p_n > 9n^{2k+2}$ be a prime. For any $\Xi \in \mathbb{Z}^{n(n-1)/2+2n}$, let $Z_n(\Xi;p_n) \triangleq Z_n(\Xi) \pmod{p_n}$.
- Suppose $\mathbf{U} \in \mathbb{Z}_{p_n}^{n(n-1)/2+2n}$ generated uniformly at random.

Proof Sketch.

For an $n-$spin system, $Z_n(\cdot)$ requires (integer) input, of size $n(n-1)/2+2n$. We follow an outline similar to Cai et al. [99] for permanent.

- Let $p_n > 9n^{2k+2}$ be a prime. For any $\Xi \in \mathbb{Z}^{n(n-1)/2+2n}$, let $Z_n(\Xi;p_n) \triangleq Z_n(\Xi) \pmod{p_n}$.
- Suppose $\mathbf{U} \in \mathbb{Z}_{p_n}^{n(n-1)/2+2n}$ generated uniformly at random.
- **Claim.** Computing $Z_n(\mathbf{U};p_n)$ is hard on average by worst-case to average reduction: if there exists an algorithm $\mathcal{A}$ enjoying

$$\mathbb{P}(Z_{\mathcal{A}}(\mathbf{U};p_n) = Z_n(\mathbf{U};p_n)) \geq n^{-k},$$

then $P = \#P$. Based on worst-case hardness for arbitrary inputs.

Proof Sketch.

For an $n-$spin system, $Z_n(\cdot)$ requires (integer) input, of size $n(n-1)/2+2n$. We follow an outline similar to Cai et al. [99] for permanent.

- Let $p_n > 9n^{2k+2}$ be a prime. For any $\Xi \in \mathbb{Z}^{n(n-1)/2+2n}$, let $Z_n(\Xi;p_n) \triangleq Z_n(\Xi) \pmod{p_n}$.
- Suppose $\mathbf{U} \in \mathbb{Z}_{p_n}^{n(n-1)/2+2n}$ generated uniformly at random.
- **Claim.** Computing $Z_n(\mathbf{U};p_n)$ is hard on average by worst-case to average reduction: if there exists an algorithm $\mathcal{A}$ enjoying

$$\mathbb{P}(Z_{\mathcal{A}}(\mathbf{U};p_n) = Z_n(\mathbf{U};p_n)) \geq n^{-k},$$

  then $P = \#P$. Based on worst-case hardness for arbitrary inputs.
- Downward self-reduction from $n-$spin system to $(n-1)-$spin system: for some parameters $B'_n, C'_n \in \mathbb{Z}_{p_n}$ and $\mathbf{B}^+, \mathbf{B}^-, \mathbf{C}^+, \mathbf{C}^- \in \mathbb{Z}_{p_n}^{n-1}$, it holds:

$$Z_n(\mathbf{J},\mathbf{B},\mathbf{C};p_n) = C'_n Z_{n-1}(\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+;p_n) + B'_n Z_{n-1}(\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-;p_n).$$

  Analogous to Laplace expansion for permanent.

Proof Sketch.

Proof Sketch.

- Recall. The object of interest satisfies

$$Z_n(\mathbf{J},\mathbf{B},\mathbf{C};p_n) = C_n' Z_{n-1}(\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+;p_n)_+ B_n' Z_{n-1}(\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-;p_n).$$

Proof Sketch.

- Recall. The object of interest satisfies

$$Z_n(\mathbf{J},\mathbf{B},\mathbf{C};p_n) = C_n' Z_{n-1}(\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+;p_n)_+ B_n' Z_{n-1}(\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-;p_n).$$

- Construct a vector polynomial $D(x)$ such that $D(1)=(\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+)$ and $D(2)=(\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-)$. $D(x)$ thought of as a vector carrying parameters required for an $(n-1)-$spin system.

Proof Sketch.

- Recall. The object of interest satisfies

$$Z_n(\mathbf{J},\mathbf{B},\mathbf{C};p_n) = C_n' Z_{n-1}(\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+;p_n)_+ B_n' Z_{n-1}(\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-;p_n).$$

- Construct a vector polynomial $D(x)$ such that $D(1) = (\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+)$ and $D(2) = (\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-)$. $D(x)$ thought of as a vector carrying parameters required for an $(n-1)-$spin system.
- Let $\phi(x) = Z_n(D(x);p_n)$, associated partition function. $\phi(\cdot)$ is univariate polynomial, of degree at most n^2.

Proof Sketch.

- Recall. The object of interest satisfies

$$Z_n(\mathbf{J},\mathbf{B},\mathbf{C};p_n)=C_n'Z_{n-1}(\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+;p_n)_+B_n'Z_{n-1}(\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-;p_n).$$

- Construct a vector polynomial $D(x)$ such that $D(1)=(\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+)$ and $D(2)=(\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-)$. $D(x)$ thought of as a vector carrying parameters required for an $(n-1)-$spin system.
- Let $\phi(x)=Z_n(D(x);p_n)$, associated partition function. $\phi(\cdot)$ is univariate polynomial, of degree at most n^2.
- Note that

$$Z_n(\mathbf{J},\mathbf{B},\mathbf{C};p_n)=C_n'\phi(1)+B_n'\phi(2).$$

Proof Sketch.

- Recall. The object of interest satisfies

$$Z_n(\mathbf{J},\mathbf{B},\mathbf{C};p_n) = C_n' Z_{n-1}(\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+;p_n) + B_n' Z_{n-1}(\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-;p_n).$$

- Construct a vector polynomial $D(x)$ such that $D(1)=(\mathbf{J}',\mathbf{B}^+,\mathbf{C}^+)$ and $D(2)=(\mathbf{J}',\mathbf{B}^-,\mathbf{C}^-)$. $D(x)$ thought of as a vector carrying parameters required for an $(n-1)-$spin system.
- Let $\phi(x)=Z_n(D(x);p_n)$, associated partition function. $\phi(\cdot)$ is univariate polynomial, of degree at most n^2.
- Note that

$$Z_n(\mathbf{J},\mathbf{B},\mathbf{C};p_n) = C_n'\phi(1) + B_n'\phi(2).$$

- Thus Z_n can be computed provided $\phi(\cdot)$ can be reconstructed.

Proof Sketch.

Proof Sketch.

- Use $\mathcal{A}$ to generate a list of noisy samples of $\phi(\cdot)$. Reconstruct ϕ using a list-decoder by Sudan [96].

Proof Sketch.

- Use $\mathcal{A}$ to generate a list of noisy samples of $\phi(\cdot)$. Reconstruct ϕ using a list-decoder by Sudan [96].
- Thus, if $\mathcal{A}$ (exactly) computes Z_n correctly for n^{-k} fraction of all inputs from $\mathbb{Z}_{p_n}^{n(n-1)/2}$, then it computes $Z_n(\mathbf{a}; p_n)$ for **any a**, with probability $1 - o(1)$.

Proof Sketch.

- Use $\mathcal{A}$ to generate a list of noisy samples of $\phi(\cdot)$. Reconstruct ϕ using a list-decoder by Sudan [96].
- Thus, if $\mathcal{A}$ (exactly) computes Z_n correctly for n^{-k} fraction of all inputs from $\mathbb{Z}_{p_n}^{n(n-1)/2}$, then it computes $Z_n(\mathbf{a}; p_n)$ for **any a**, with probability $1 - o(1)$.
- Use tail bound to control value of partition function.

Proof Sketch.

- Use $\mathcal{A}$ to generate a list of noisy samples of $\phi(\cdot)$. Reconstruct ϕ using a list-decoder by Sudan [96].
- Thus, if $\mathcal{A}$ (exactly) computes Z_n correctly for n^{-k} fraction of all inputs from $\mathbb{Z}_{p_n}^{n(n-1)/2}$, then it computes $Z_n(\mathbf{a}; p_n)$ for **any a**, with probability $1 - o(1)$.
- Use tail bound to control value of partition function.
- Use prime density to take sufficiently many primes, product larger than partition function. Apply Chinese remaindering.

Proof Sketch.

Proof Sketch.

- Rest is a probabilistic coupling argument.

Proof Sketch.

- Rest is a probabilistic coupling argument.
- Recall $\widetilde{J}_{ij} = 2^N \widehat{J}_{ij}^{[N]}$, where $\widehat{J}_{ij}^{[N]} = 2^{-N}\lfloor 2^N \widehat{J}_{ij} \rfloor$, and $\widehat{J}_{ij} = \exp(\beta J_{ij} n^{-1/2})$. Recall also $\widetilde{B}_i, \widetilde{C}_i$.

Proof Sketch.

- Rest is a probabilistic coupling argument.
- Recall $\widetilde{J}_{ij} = 2^N \widehat{J}_{ij}^{[N]}$, where $\widehat{J}_{ij}^{[N]} = 2^{-N}\lfloor 2^N \widehat{J}_{ij} \rfloor$, and $\widehat{J}_{ij} = \exp(\beta J_{ij} n^{-1/2})$. Recall also $\widetilde{B}_i, \widetilde{C}_i$.
- Show $\widetilde{J}_{ij}, \widetilde{B}_i, \widetilde{C}_i$ modulo p_n are close to uniform distribution.

Proof Sketch.

- Rest is a probabilistic coupling argument.
- Recall $\widetilde{J}_{ij} = 2^N \widehat{J}_{ij}^{[N]}$, where $\widehat{J}_{ij}^{[N]} = 2^{-N}\lfloor 2^N \widehat{J}_{ij} \rfloor$, and $\widehat{J}_{ij} = \exp(\beta J_{ij} n^{-1/2})$. Recall also $\widetilde{B}_i, \widetilde{C}_i$.
- Show $\widetilde{J}_{ij}, \widetilde{B}_i, \widetilde{C}_i$ modulo p_n are close to uniform distribution.
- Use coupling idea to conclude.

Overview

Part II. Hardness under Real-Valued Model. Setup and Model

Part II. Hardness under Real-Valued Model. Setup and Model

- Hardness when computational engine (e.g. Blum-Shub-Smale machine) operates over real-valued inputs. Each arithmetic operation has unit cost.

Part II. Hardness under Real-Valued Model. Setup and Model

- Hardness when computational engine (e.g. Blum-Shub-Smale machine) operates over real-valued inputs. Each arithmetic operation has unit cost.
- We consider Hamiltonian without external field: $H(\boldsymbol{\sigma}) = \sum_{i<j} J_{ij}\sigma_i\sigma_j$.

Part II. Hardness under Real-Valued Model. Setup and Model

- Hardness when computational engine (e.g. Blum-Shub-Smale machine) operates over real-valued inputs. Each arithmetic operation has unit cost.
- We consider Hamiltonian without external field: $H(\boldsymbol{\sigma}) = \sum_{i<j} J_{ij}\sigma_i\sigma_j$.
- Scaling $\sqrt{n}$ and inverse temperature β suppressed for simplicity.

Part II. Hardness under Real-Valued Model. Setup and Model

- Hardness when computational engine (e.g. Blum-Shub-Smale machine) operates over real-valued inputs. Each arithmetic operation has unit cost.
- We consider Hamiltonian without external field: $H(\boldsymbol{\sigma}) = \sum_{i<j} J_{ij}\sigma_i\sigma_j$.
- Scaling $\sqrt{n}$ and inverse temperature β suppressed for simplicity.
- After reducing to cuts analogously, boils down computing

$$\widehat{Z}(\mathbf{J}) = \sum_{\boldsymbol{\sigma}\in\{\pm 1\}^n} \exp\left(\sum_{\sigma_i\neq\sigma_j} 2J_{ij}\right).$$

Part II. Hardness under Real-Valued Model. Setup and Model

- Hardness when computational engine (e.g. Blum-Shub-Smale machine) operates over real-valued inputs. Each arithmetic operation has unit cost.
- We consider Hamiltonian without external field: $H(\boldsymbol{\sigma}) = \sum_{i<j} J_{ij}\sigma_i\sigma_j$.
- Scaling $\sqrt{n}$ and inverse temperature β suppressed for simplicity.
- After reducing to cuts analogously, boils down computing

$$\widehat{Z}(\mathbf{J}) = \sum_{\boldsymbol{\sigma}\in\{\pm 1\}^n} \exp\left(\sum_{\sigma_i\neq\sigma_j} 2J_{ij}\right).$$

- Techniques of previous setting tailored to finite precision model: finite field structure $\mathbb{Z}_p$ is lost upon passing real-valued model. By pass through an argument by Aaronson and Arkhipov [2011].

Part II. Hardness under Real-Valued Model: Main result

Part II. Hardness under Real-Valued Model: Main result

Theorem (Gamarnik & K., 2019)

Let $\mathbf{J} = (J_{ij} : 1 \le i < j \le n) \in \mathbb{R}^{n(n-1)/2}$ *consists of iid standard normal entries, and* $\mathcal{A}$ *be a polynomial-in-n time algorithm such that* $\mathbb{P}(\mathcal{A}(\mathbf{J}) = \widehat{Z}(\mathbf{J})) \ge \frac{3}{4} + \delta$, *where* $\delta \ge 1/\mathrm{poly}(n) > 0$ *is arbitrary. Then,* $P = \#P$.

Part II. Hardness under Real-Valued Model: Main result

Theorem (Gamarnik & K., 2019)

Let $\mathbf{J} = (J_{ij} : 1 \le i < j \le n) \in \mathbb{R}^{n(n-1)/2}$ *consists of iid standard normal entries, and* $\mathcal{A}$ *be a polynomial-in-n time algorithm such that* $\mathbb{P}(\mathcal{A}(\mathbf{J}) = \widehat{Z}(\mathbf{J})) \ge \frac{3}{4} + \delta$*, where* $\delta \ge 1/\mathrm{poly}(n) > 0$ *is arbitrary. Then,* $P = \#P$*.*

Remarks.

Part II. Hardness under Real-Valued Model: Main result

Theorem (Gamarnik & K., 2019)

Let $\mathbf{J} = (J_{ij} : 1 \le i < j \le n) \in \mathbb{R}^{n(n-1)/2}$ *consists of iid standard normal entries, and* $\mathcal{A}$ *be a polynomial-in-n time algorithm such that* $\mathbb{P}(\mathcal{A}(\mathbf{J}) = \widehat{Z}(\mathbf{J})) \ge \frac{3}{4} + \delta$, *where* $\delta \ge 1/\mathrm{poly}(n) > 0$ *is arbitrary. Then,* $P = \#P$.

Remarks.

- Again, based on hardness of computing partition function for arbitrary inputs.

Part II. Hardness under Real-Valued Model: Main result

Theorem (Gamarnik & K., 2019)

Let $\mathbf{J} = (J_{ij} : 1 \le i < j \le n) \in \mathbb{R}^{n(n-1)/2}$ *consists of iid standard normal entries, and* $\mathcal{A}$ *be a polynomial-in-n time algorithm such that* $\mathbb{P}(\mathcal{A}(\mathbf{J}) = \widehat{Z}(\mathbf{J})) \ge \frac{3}{4} + \delta$*, where* $\delta \ge 1/\mathrm{poly}(n) > 0$ *is arbitrary. Then,* $P = \#P$.

Remarks.

- Again, based on hardness of computing partition function for arbitrary inputs.
- A similar program: boils down reconstructing a certain low-degree polynomial from its noisy samples. This time, Berklekamp-Welch decoder is used instead.

Part II. Hardness under Real-Valued Model: Main result

Theorem (Gamarnik & K., 2019)

Let $\mathbf{J} = (J_{ij} : 1 \le i < j \le n) \in \mathbb{R}^{n(n-1)/2}$ *consists of iid standard normal entries, and* $\mathcal{A}$ *be a polynomial-in-n time algorithm such that* $\mathbb{P}(\mathcal{A}(\mathbf{J}) = \widehat{Z}(\mathbf{J})) \ge \frac{3}{4} + \delta$*, where* $\delta \ge 1/\mathrm{poly}(n) > 0$ *is arbitrary. Then,* $P = \#P$.

Remarks.

- Again, based on hardness of computing partition function for arbitrary inputs.
- A similar program: boils down reconstructing a certain low-degree polynomial from its noisy samples. This time, Berklekamp-Welch decoder is used instead.
- Uses a control for total variation distance for log-normal random variables, in presence of a convex perturbation.

Overview

Concluding Remarks : Extensions

Concluding Remarks : Extensions

- Average-case hardness of algorithmic problem of exactly computing partition function of SK spin glass model. Under both finite precision arithmetic and real-valued computational models.

Concluding Remarks : Extensions

- Average-case hardness of algorithmic problem of exactly computing partition function of SK spin glass model. Under both finite precision arithmetic and real-valued computational models.
- To the best of our knowledge, first such average-case hardness result for a statistical physics model.

Concluding Remarks : Extensions

- Average-case hardness of algorithmic problem of exactly computing partition function of SK spin glass model. Under both finite precision arithmetic and real-valued computational models.
- To the best of our knowledge, first such average-case hardness result for a statistical physics model.

Extensions.

Concluding Remarks : Extensions

- Average-case hardness of algorithmic problem of exactly computing partition function of SK spin glass model. Under both finite precision arithmetic and real-valued computational models.
- To the best of our knowledge, first such average-case hardness result for a statistical physics model.

Extensions.

- 2$-$spin assumption is non-essential: extends to the $p-$spin models.

Concluding Remarks : Extensions

- Average-case hardness of algorithmic problem of exactly computing partition function of SK spin glass model. Under both finite precision arithmetic and real-valued computational models.
- To the best of our knowledge, first such average-case hardness result for a statistical physics model.

Extensions.

- $2-$spin assumption is non-essential: extends to the $p-$spin models.
- Gaussianity of the couplings is non-essential. Well behaved distributions with sufficiently smooth density should be enough.

Concluding Remarks : Extensions

- Average-case hardness of algorithmic problem of exactly computing partition function of SK spin glass model. Under both finite precision arithmetic and real-valued computational models.
- To the best of our knowledge, first such average-case hardness result for a statistical physics model.

Extensions.

- $2-$spin assumption is non-essential: extends to the $p-$spin models.
- Gaussianity of the couplings is non-essential. Well behaved distributions with sufficiently smooth density should be enough.
- The scaling $n^{-\frac{1}{2}}$ is non-essential: any constant power of n is ok.

Concluding Remarks : Limitations and Open Problems

Concluding Remarks : Limitations and Open Problems

- Our approach does not treat the same problem when couplings are i.i.d. Rademacher. Not surprising though in light of the fact that average-case hardness of computing permanent of a binary matrix is open as well.

Concluding Remarks : Limitations and Open Problems

- Our approach does not treat the same problem when couplings are i.i.d. Rademacher. Not surprising though in light of the fact that average-case hardness of computing permanent of a binary matrix is open as well.
- The trick of $\pmod{p_n}$ computation is too "fragile" to survive the approximate computation: average-case hardness of computing $Z(\mathbf{J},\beta)$ to within a multiplicative factor of $1\pm\epsilon$ remains open.

Concluding Remarks : Limitations and Open Problems

- Our approach does not treat the same problem when couplings are i.i.d. Rademacher. Not surprising though in light of the fact that average-case hardness of computing permanent of a binary matrix is open as well.
- The trick of $\pmod{p_n}$ computation is too "fragile" to survive the approximate computation: average-case hardness of computing $Z(\mathbf{J},\beta)$ to within a multiplicative factor of $1\pm\epsilon$ remains open.

A related problem: Ground-state computation. $\boldsymbol{\sigma}^* \in \{\pm 1\}^n$ is called a *ground-state* if $H(\boldsymbol{\sigma}^*) = \max_{\boldsymbol{\sigma}\in\{\pm 1\}^n} H(\boldsymbol{\sigma})$.

Concluding Remarks : Limitations and Open Problems

- Our approach does not treat the same problem when couplings are i.i.d. Rademacher. Not surprising though in light of the fact that average-case hardness of computing permanent of a binary matrix is open as well.
- The trick of $\pmod{p_n}$ computation is too "fragile" to survive the approximate computation: average-case hardness of computing $Z(\mathbf{J},\beta)$ to within a multiplicative factor of $1\pm\epsilon$ remains open.

A related problem: Ground-state computation. $\boldsymbol{\sigma}^*\in\{\pm 1\}^n$ is called a *ground-state* if $H(\boldsymbol{\sigma}^*)=\max_{\boldsymbol{\sigma}\in\{\pm 1\}^n} H(\boldsymbol{\sigma})$.

- Arora et al. [05]: problem of computing ground state is NP-hard (in worst-case sense).

Concluding Remarks : Limitations and Open Problems

- Our approach does not treat the same problem when couplings are i.i.d. Rademacher. Not surprising though in light of the fact that average-case hardness of computing permanent of a binary matrix is open as well.
- The trick of $\pmod{p_n}$ computation is too "fragile" to survive the approximate computation: average-case hardness of computing $Z(\mathbf{J},\beta)$ to within a multiplicative factor of $1\pm\epsilon$ remains open.

A related problem: Ground-state computation. $\boldsymbol{\sigma}^* \in \{\pm 1\}^n$ is called a *ground-state* if $H(\boldsymbol{\sigma}^*) = \max_{\boldsymbol{\sigma}\in\{\pm 1\}^n} H(\boldsymbol{\sigma})$.

- Arora et al. [05]: problem of computing ground state is NP-hard (in worst-case sense).
- Montanari [19]: a message-passing algorithm, which for any $\epsilon > 0$, finds (in time $O(n^2)$) a state $\boldsymbol{\sigma}_* \in \{\pm 1\}^n$ such that $H(\boldsymbol{\sigma}_*) \ge (1-\epsilon)H(\boldsymbol{\sigma}^*)$ whp.

Concluding Remarks : Limitations and Open Problems

- Our approach does not treat the same problem when couplings are i.i.d. Rademacher. Not surprising though in light of the fact that average-case hardness of computing permanent of a binary matrix is open as well.
- The trick of $\pmod{p_n}$ computation is too "fragile" to survive the approximate computation: average-case hardness of computing $Z(\mathbf{J},\beta)$ to within a multiplicative factor of $1 \pm \epsilon$ remains open.

A related problem: Ground-state computation. $\boldsymbol{\sigma}^* \in \{\pm 1\}^n$ is called a *ground-state* if $H(\boldsymbol{\sigma}^*) = \max_{\boldsymbol{\sigma} \in \{\pm 1\}^n} H(\boldsymbol{\sigma})$.

- Arora et al. [05]: problem of computing ground state is NP-hard (in worst-case sense).
- Montanari [19]: a message-passing algorithm, which for any $\epsilon > 0$, finds (in time $O(n^2)$) a state $\boldsymbol{\sigma}_* \in \{\pm 1\}^n$ such that $H(\boldsymbol{\sigma}_*) \ge (1-\epsilon)H(\boldsymbol{\sigma}^*)$ whp.
- Average-case hardness of problem of **exactly** computing $\boldsymbol{\sigma}^*$ remains open: algebraic structure is lost upon passing to maximization.

Thank you!