Smooth Structured Prediction Using Quantum and Classical Gibbs Samplers

Behrooz Sepehry,[1] Ehsan Iranmanesh,[1] Michael P. Friedlander,[1,2] and Pooya Ronagh[1,3,4,✉]
[1] *1QB Information Technologies (1QBit), Vancouver, BC*
[2] *University of British Columbia, Vancouver, BC*
[3] *Institute for Quantum Computing, Waterloo, ON*
[4] *University of Waterloo, Waterloo, ON*
(Dated: January 10, 2019)

We introduce a quantum algorithm for solving structured prediction problems with a runtime that scales with the square root of the size of the label space, but scales in $\widetilde{O}\left(\epsilon^{-3.5}\right)$ with respect to the precision, ϵ, of the solution. In doing so, we analyze a stochastic gradient algorithm for convex optimization in the presence of an additive error in the calculation of the gradients, and show that its convergence rate does not deteriorate if the additive errors are of the order $O(\sqrt{\epsilon})$. Our algorithm uses quantum Gibbs sampling at temperature $\Omega(\epsilon)$ as a subroutine. Based on these theoretical observations, we propose a method for using quantum Gibbs samplers to combine feedforward neural networks with probabilistic graphical models for quantum machine learning. Numerical results using Monte Carlo simulations on an image tagging task demonstrate the benefit of the approach.

I. INTRODUCTION

Classification is a central task in machine learning, where the aim is to assign categories to observations. This is an inherently combinatorial task that often gives rise to piecewise smooth models, such as support vector machines (SVM). This combinatorial aspect is especially egregious in structured prediction, where the task involves the prediction of vectors, rather than simply scalar value assignments. For example, in structured SVMs (SSVM), the number of pieces in the piecewise smooth model is often exponentially large in the dimension of prediction vectors. A common technique to deal with nonsmooth models is to optimize smooth approximations, for example using softmax operators. Although these techniques are effective at hiding the nonsmooth aspects of the problem by replacing a piecewise nonsmooth problem with a single smooth approximation, computing that approximation can be intractable when the number of pieces is large. In this paper, we consider a smoothing that combines ideas from softmax approximations and quantum Gibbs sampling in order to obtain a quantum speedup for structured prediction tasks.

It has been speculated for the past 20 years that quantum computers can be used to generate samples from Gibbs states [TD00]. Since then, many algorithms for Gibbs sampling based on a quantum-circuit model have been introduced [PW09, TOV$^+$11, KB16, CS16, AGGW17]. The most recently proposed Gibbs sampler, due to van Apeldoorn et al. [AGGW17], has a logarithmic dependence on the error of the simulated distribution. The sampler of Chowdhury and Somma [CS16] similarly has a logarithmic error dependence, but must assume a query access to the entries of the square root of the problem Hamiltonian. These quantum-circuit algorithms use phase estimation and amplitude amplification techniques to create a quadratic speedup in Gibbs sampling. In practice, this would still result in an exponentially long runtime. Separately, the Gibbs sampler

✉ Corresponding author: pooya.ronagh@1qbit.com

of Temme et al. [TOV+11] has an unknown runtime, but has the potential to provide efficient heuristics since it relies on a quantum Metropolis algorithm.

On the other hand, other quantum and semi-classical evolutions can be used as physical realizations of improved Gibbs samplers. For example, contemporary investigation in quantum adiabatic theory focuses on adiabaticity in open quantum systems [SL05, AFGG12, ABLZ12, BDRF16, VALZ16]. These authors prove adiabatic theorems to various degrees of generality and assumptions. These adiabatic theorems suggest the possibility of using controlled adiabatic evolutions of quantum many-body systems as samplers of the instantaneous steady states of quantum systems. Takeda et al. [TTY+17] show that a network of non-degenerate optic parametric pulses can produce good estimations of Boltzmann distributions. Another possible approach to improved Gibbs samplers is to design customized Gibbs-sampling algorithms that rely on Monte Carlo and quantum Monte Carlo methods implemented on digital high-performance computing hardware [MTT+17, OHY17].

The idea of using Gibbs sampling as a subroutine in machine learning tasks has already been considered. Wiebe et al. [WKS14] use Gibbs-state preparation to propose an improved framework for quantum deep learning. Crawford et al. [CLG+16] and Levit et al. [LCG+17] introduce a framework for reinforcement learning that uses Gibbs states as function approximators in Q-learning. Quantum Gibbs sampling has recently been shown to provide a quadratic speedup in solving linear programs (LP) and semi-definite programs (SDP) [BS17, BKL+17, AGGW17]. The speedup in these quantum algorithms with respect to the problem size often comes at the expense of much worse scaling in terms of solution precision. For example, van Apeldoorn et al. [AGGW17] propose a quantum algorithm for LP that requires $\widetilde{O}(\epsilon^{-5})$ quantum gates, and an algorithm for SDPs that requires $\widetilde{O}(\epsilon^{-8})$ quantum gates, where ϵ is an additive error on the accuracy of the final solution.

Our main contribution in this paper is the introduction of a quantum algorithm for solving a min-max optimization problem of the form

$$\min_{w} \; r(w) + \frac{1}{n}\sum_{i=1}^{n} g_i(w), \quad \text{where} \quad g_i(w) = \max_{y\in\mathcal{Y}} f_i(y,w)\,, \tag{1}$$

where the functions r and f_i are convex with Lipschitz continuous gradients, r is strongly convex, and $\mathcal{Y}$ is a finite set. This can be easily extended to the case in which each function f_i is defined on a distinct domain $\mathcal{Y}_i$. The size of $\mathcal{Y}$ can cause the evaluation of the max operator to be computationally intractable. These problems arise frequently in applications of machine learning, and include SVMs and SSVMs as special cases. Various algorithms have been applied to this class of problems, including stochastic subgradient methods [SZ13] and optimal first-order methods for nonsmooth problems [Nes05]. Other algorithms for smooth problems, such as SAGA [DBLJ14], can be applied by replacing each function g_i with an approximation that is strongly convex with a Lipschitz continuous gradient. However, these smooth approximations typically rely on replacing the max operation with the differentiable softmax operator [GP17, BT12], that is, each function g_i is replaced by the smooth approximation

$$g_i^{\beta}(w) := \frac{1}{\beta}\log\sum_{y\in\mathcal{Y}} e^{\beta f_i(y,w)}\,,$$

which is at least as computationally difficult as evaluating the original max operation. This approximation can be interpreted from a thermodynamic perspective: each g_i^{β} is the free energy of a system with an energy spectrum described by f_i. Our quantum algorithm relies on quantum Gibbs sampling to estimate derivatives of the softmax operator.

Quantum Gibbs sampling achieves quadratic speedup in the size of the sample space, but can only be used to produce an approximate gradient of the smooth functions g_i^β. Thus, any first-order method applied to the smooth approximation of the objective function (1) must be modified to take into account the error in the computed gradient. In our analysis, we show how the SAGA algorithm can be modified so that it continues to enjoy its original $O\left(\log(\frac{1}{\epsilon})\right)$ number of iterations even in the presence of additive error in the approximate gradients, provided the errors in gradient estimates are accurate to within $O(\epsilon)$. We then consider a quantum version of SAGA that uses Gibbs sampling as a computational kernel. For a fixed parameter β, this algorithm obtains an ϵ-accurate minimizer of the smooth approximation within $\widetilde{O}(\frac{1}{\epsilon})$ quantum gates. To solve the original min-max problem with accuracy ϵ, the temperature has to be assigned proportional to ϵ. In total, this results in $\widetilde{O}(\epsilon^{-3.5})$ quantum gates.

We also provide several numerical results. We use single-spin flip Monte Carlo simulation of Ising models to perform image tagging as an example of a structured-prediction task. We compare several contemporary structured-prediction objective functions and demonstrate a working framework of application of classical and quantum Gibbs samplers in machine learning. In deep learning, softmax operators are often used in the last layers of a feedforward neural network. Our approach proposes the use of a quantum Gibbs sampler to *thicken* the softmax layer of a neural network with internal connections. The resultant architecture consists of a leading directed neural network serving as a *feature extractor*, and a trailing undirected neural network responsible for smooth prediction based on the feature vectors.

The paper is organized as follows. In Section II, we give an overview of SVMs, SSVMs, and more-general structured-prediction problems. In Section III, we introduce the mathematical model of the min-max optimization problem frequently used in structured prediction. We explain how quantum Gibbs sampling can provide gradients for optimizing smooth approximation of the min-max objective function. We then analyze the effect of approximation errors in gradient calculations for SAGA. The main result is Theorem IV.6, which shows that the convergence of SAGA does not deteriorate in the presence of sufficiently small gradient approximation errors. We also give corollaries that analyze the complexity of solving the smooth approximation problem and the original min-max optimization problem using the quantum version of SAGA. In Section V, we give the results of numerical experiments on small problem instances and study the effect of smoothing, various temperature schedules, and the role of smoothing in SAGA versus standard stochastic gradient descent. We also give the results of an image tagging experiment on the MIRFLICKR25000 dataset [HL08]. In Section VI, we introduce several structured-prediction objective functions. Finally, in Section VII, we report the results of using Monte Carlo simulation of a Gibbs sampler in performing the image tagging task.

II. BACKGROUND

We first present a brief account of SVMs and SSVMs. We refer the reader to [Ng10] for the basics of SVMs and to [Yu11] for SSVMs. We then introduce the more general framework of structured prediction tasks in machine learning.

A. SVMs and SSVMs

Let $\mathcal{X}$ be a feature set and $\mathcal{Y} = \{-1, 1\}$ be the label set. We are also given a training dataset $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y}$. A linear classifier is then given by two (tunable) parameters w and b defining a separating hyperplane $w^T x + b$. For a point $(x, y) \in \mathcal{S}$, the positivity of $y(w^T x + b)$ indicates the correct classification of x. The SVM optimization problem can be expressed as

$$\begin{aligned} \min_{w} \quad & \frac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y\left(w^T x + b\right) \geq 1, \quad \forall (x, y) \in \mathcal{S}. \end{aligned}$$

The constraints ensure not only that every $(x, y) \in \mathcal{S}$ is classified correctly, but also with a confidence margin. If $y(w^T x + b)$ is positive, one could superficially satisfy $y\left(w^T x + b\right) \geq 1$ by scaling up w and b. To avoid this we minimize $\frac{1}{2}\|w\|^2$. In other words, the constraints ensure that the distance of $\mathcal{S}$ to the classifying hyperplane is at least $1/\|w\|$, and the objective function asks for this margin to be maximized.

Often, the above optimization problem is infeasible, so we would rather solve a relaxation of it by introducing slack variables for every data point in $\mathcal{S}$:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\|w\|^2 + C \sum_{(x,y)\in\mathcal{S}} \xi_{(x,y)} \\ \text{s.t.} \quad & y\left(w^T x + b\right) \geq 1 - \xi_{(x,y)}, \quad \forall (x, y) \in \mathcal{S} \\ & \xi_{(x,y)} \geq 0 \quad \forall (x, y) \in \mathcal{S}. \end{aligned} \tag{2}$$

For simplicity, we will remove the bias from the rest of the analysis and consider it a trainable feature of x. Let $\mathcal{Y}$ now contain more than just two classes. The score of class y is then represented by the dot product $w_y^T x$. The Crammer–Singer formulation of the multi-class SVM problem is the following:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2}\sum_{y\in\mathcal{Y}} \|w_y\|^2 + C \sum \xi_{(x,y)} \\ \text{s.t.} \quad & w_y^T x - w_{y'}^T x \geq 1 - \xi_{(x,y)}, \forall (x, y) \in \mathcal{S}, \forall y' \in \mathcal{Y} \setminus \{y\} \\ & \xi_{(x,y)} \geq 0 \quad \forall (x, y) \in \mathcal{S}. \end{aligned}$$

We can rewrite this in a notation more suitable for introducing SSVMs as a generalization of SVMs. We first concatenate the weight vectors w_y into a single vector:

$$w^T = (w_1^T, \ldots, w_k^T)\,.$$

We then introduce the notation

$$\Phi(x, y) = (0, \ldots, x, \ldots, 0)\,,$$

with x in the y-th position represents a *joint feature map*, and all other elements are 0. Lastly, we introduce a notion of *distance* or *loss function* on $\mathcal{Y}$:

$$\Delta(y', y) = \begin{cases} 1 & y = y'\,, \\ 0 & \text{otherwise.} \end{cases}$$

Then, the model can be rewritten as

$$
\begin{aligned}
\min_{w,\xi} \quad & \frac{1}{2}\|w\|^2 + C\sum \xi_{(x,y)} \\
\text{s.t.} \quad & \xi_{(x,y)} \geq \Delta(y',y) - w^T\Phi(x,y) + w^T\Phi(x,y') \\
& \qquad\qquad\qquad\qquad \forall (x,y) \in \mathcal{S}, \forall y' \in \mathcal{Y} \\
& \xi_{(x,y)} \geq 0 \quad \forall (x,y) \in \mathcal{S}.
\end{aligned} \tag{3}
$$

The above model is that of an SSVM in general, with possibly more-complicated joint feature maps and loss functions.

The problem (3) can be rewritten as a min-max problem of the form

$$
\min_{w} \left(f(w) = \left\{ \sum_{x,y} \max_{y'} f_{(x,y)}(y';w) \right\} \right), \tag{4}
$$

where the summands $f_{(x,y)}(y';w)$ are of the form

$$
f_{(x,y)}(y';w) = \Delta(y',y) - w^T\Phi(x,y) + w^T\Phi(x,y') \tag{5}
$$

by omitting the regularizer term $\frac{1}{2}\|w\|^2$. We will continue with this omission of regularizers for the rest of this paper, noting that in all experiments regularizer terms will be included.

Without the regularizer term, problem (3) is therefore readily of the mathematical form of the Lagrangian dual problems studied in [RWI16, KR17], and cutting plane or subgradient descent approaches could be used to solve them efficiently under the assumption of the existence of *noise-free* discrete optimization oracles. It is also a linear problem, and the quantum linear programming technique of [AGGW17] could be used to provide quadratic speedup in the number of constraints and variables of the problem. In most practical cases, however (see below), the instances are very large, and it would not be realistic to assume the entire problem is available via an efficient circuit for oracle construction. Stochastic gradient descent methods overcome this difficulty (for classical training data) by randomly choosing training samples or mini-batches. This is also our approach in what follows.

B. Structured Prediction

We now introduce the general framework of *structured prediction* as a supervised learning task in machine learning. SSVMs are only one of the mathematical models used to solve structured prediction problems. As we will see, the distinguishing factor between techniques for solving structured prediction problems is the choice of an objective function similar to (5).

We will assume that structured prediction problems are equipped with the following.

(a) A training dataset $\mathcal{S} \subseteq \mathcal{X} \times \mathcal{Y}$.

$\mathcal{X}$ and $\mathcal{Y}$ are, respectively, the set of all possible inputs and outputs. The elements of $\mathcal{Y}$ encode a certain *structure* (e.g., the syntactic representation of an English sentence). In structured prediction, the outputs are therefore vectors instead of scalar discrete or real values. In particular, the set $\mathcal{Y}$ may be exponentially large in the size of the input. This distinguishes structured prediction from multi-class classification.

(b) A real-valued loss function $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$.

We assume that the minimum of Δ over its first component is uniquely attained along its diagonal, that is,

$$y = \arg\min_{y'} \Delta(y', y). \tag{6}$$

The goal is to find a prediction rule $h : \mathcal{X} \to \mathcal{Y}$ that minimizes the empirical risk

$$R(h) = \frac{1}{|S|} \sum_{(x,y) \in \mathcal{S}} \Delta(h(x), y)\,. \tag{7}$$

Without loss of generality, we may assume that Δ vanishes on its diagonal

$$\Delta(y, y) = 0, \quad \forall y \in \mathcal{Y}\,, \tag{8}$$

since we can always shift it to $\Delta'(y', y) = \Delta(y', y) - \Delta(y, y)$. This decreases the empirical risk by the constant $\frac{1}{|S|} \sum_{(x,y) \in \mathcal{S}} \Delta(y, y)$, which is an invariant of $\mathcal{S}$.

(c) A scoring function $s_w : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$.

The scoring function $s_w(x, y) = s(x, y, w)$ is indicative of suitability of a label y for a given input x. Here w is a vector of tunable parameters, often trained via a machine learning procedure on the given training dataset $\mathcal{S}$.

Example. In the SSVM framework of Section II A, the loss function is simply the Kronecker delta function $\Delta(y, y') = \delta_{y,y'}$. In the model (3), the scoring function is linear in the training parameters

$$s(x, y, w) = w^T \Phi(x, y)\,.$$

In terms of a scoring function s and a loss function Δ, the objective function of (4) can be rewritten as

$$f_{\mathrm{MM}}(w) = \sum_{x,y} \max_{y'} \left\{ \Delta(y', y) + s(x, y', w) - s(x, y, w) \right\}, \tag{9}$$

which is also called the *max-margin* objective function [YJ09]. One can show that solving (4) with this objective function is a step towards solving the risk minimization problem (7), since (9) is an upper bound on the risk function [YJ09],

$$R_{\mathrm{MS}}(w) = \sum_{x,y} \Delta\left(h_{\mathrm{MS}}(x), y\right)\,, \tag{10}$$

where the prediction rule is

$$h_{\mathrm{MS}}(x) = \arg\max_{y'} s(x, y', w)\,, \tag{11}$$

which we call here the *maximum score* prediction rule. This is easy to see given

$$\begin{aligned}\Delta\left(\arg\max_{y'} s(x,y',w),y\right)+s(x,y,w) &\le \Delta\left(\arg\max_{y'} s(x,y',w),y\right)+\max_{y'} s(x,y',w)\\ &= \Delta\left(\arg\max_{y'} s(x,y',w),y\right)+s(x,\arg\max_{y'} s(x,y',w),w)\\ &= \Delta\left(y_*,y\right)+s(x,y_*,w) \le \max_{y'}\{\Delta(y',y)+s(x,y',w)\},\end{aligned}$$

where $y_* = \arg\max_{y'} s(x,y',w)$. By subtracting $s(x,y,w)$ from both sides, we get

$$\Delta\left(\arg\max_{y'} s(x,y',w),y\right) \le \max_{y'}\{\Delta(y',y)+s(x,y',w)-s(x,y,w)\}.$$

SVMs solve what is called the maximum-margin problem [Vap63]. Aside from machine learning applications, this model is very well-motivated from the perspective of constrained integer programming using quantum algorithms [RWI16, KR17]. Many generalizations of SVMs have been proposed and used to solve multi-class prediction problems [WW$^+$99, SFB$^+$98, FISS03, YJ09]. In a survey on SSVMs [Sch09], the author reviews the optimization methods for SSVMs, including subgradient methods [Col02, ATH03, Zha04, SSSSC11], cutting plane and bundle methods [THJA04, Joa06, TSVL07, LSV08, JFY09], polynomial-sized reformulations [TGK04, BCTM05, CGK$^+$08], and min-max formulations [TCK04, TLJJ06a, TLJJ06b]. Subsequently, in [LJJSP12], the authors propose a coordinate descent approach.

III. SMOOTH APPROXIMATION

Section II motivates solving a particular set of min-max optimization problems in machine learning applications. In this section, we present these mathematical programming models and consider quantum algorithms for solving them.

A. A Min-Max Optimization Problem

We define the objective function $f(w)$ as

$$f(w) = r(w) + \frac{1}{n}\sum_{i=1}^{n}\max_{y\in\mathcal{Y}} f_i(y,w)\,, \tag{12}$$

where w is a vector of tunable real-valued parameters, n is a positive integer, and r and all f_i are convex real-valued functions of w with Lipschitz continuous gradients. Furthermore, r is strongly convex, and each f_i is defined in its first argument y over a finite set $\mathcal{Y}$. In practical examples, r could represent a regularizer for a machine learning model. We are interested in solving the optimization problem

$$w_* = \arg\min_{w} f(w)\,. \tag{13}$$

Although the functions f_i are differentiable, f is not generally differentiable because of the max operator involved. However, since the max operator preserves convexity, f is a convex function.

As discussed in Section II, if f_i are linear in w, this problem is readily of the mathematical form of the Lagrangian dual problems studied in [RWI16, KR17], and cutting plane or subgradient descent approaches could be used to solve them efficiently under the assumption of the existence of *noise-free* discrete optimization oracles. The role of the discrete optimization subroutine is to minimize $f_i(y, w)$ over its discrete variable y with fixed choices of w. Then the cutting plane and subgradient descent meta-algorithms would converge to the optimal dual variable w_* by iterative calls to the optimizer.

On the other hand, in absence of a regularizer (13) is a linear problem of the form

$$\begin{aligned} \min_{w,\mu} \quad & \sum_i \mu_i \\ \text{s.t.} \quad & \mu_i \geq f_i(y, w) \quad \forall y \in \mathcal{Y}\,, \end{aligned} \tag{14}$$

and the quantum linear programming technique of [AGGW17] could readily be used to provide quadratic speedup in the number of constraints and variables of the problem. In practice, however, nonlinear regularizers play important roles. Our technique will allow for the solving of such nonlinear problems with the same quantum speedup as in [AGGW17], but with better scaling in terms of precision.

At its core, the quantum linear programming algorithm of [AGGW17] in particular, and more generally the quantum SDP solvers of [BS17] and [AGGW17], rely on amplitude amplification procedures that prepare Gibbs states up to the needed precision. It is therefore tempting to use Gibbs-state preparation directly to solve (13), given that, in classical algorithms, smooth approximation of piece-wise linear objective functions is a common method for designing improved gradient-based solvers [BT12]. We construct such a smooth approximation of the function f, and find the minimum of the approximation. This is a promising approach, since convex optimization for smooth functions can achieve faster convergence rates.

One approach to smoothing the max of a set of functions is *softmax smoothing* [BT12]. For a finite set $\mathcal{Y}$ and $\beta > 0$, the softmax approximation of the max operator over a set of values $\mathcal{Y}$ is defined as

$$\max_{y\in\mathcal{Y}}{}^{\beta} y = \frac{1}{\beta} \log \sum_{y\in\mathcal{Y}} \exp(\beta y)\,. \tag{15}$$

This is the negative free energy of a physical system with an energy spectrum $\{-y : y \in Y\}$. We now apply smoothing to the range of every summand f_i in (12) and the resultant summation is called the smooth approximation of f at inverse temperature β, denoted by $f^\beta(w)$:

$$f^\beta(w) = r(w) + \frac{1}{n} \sum_i \max_{y\in\mathcal{Y}}{}^{\beta} f_i(y, w)\,. \tag{16}$$

We note that $f^\beta(w)$ converges uniformly to $f(w)$ in the limit of $\beta \to \infty$ (refer to (28) below). So, on one hand, β can be interpreted as the thermodynamic inverse temperature at equilibrium for each energy function $-f_i$ and, on the other hand, as a parameter controlling the amount of smoothing imposed on f. That is, when β is large, a better approximation of f is obtained, but with a larger Lipschitz constant for the gradient of f (i.e., less smoothness). Consequently, we approximate w_* in (13) with

$$w_*^\beta = \arg\min_w f^\beta(w)\,. \tag{17}$$

To perform gradient-based convex optimization on f^β, we calculate its gradient via

$$\nabla_w f^\beta(w) = \nabla_w r(w) + \frac{1}{n}\sum_i \mathbb{E}_{Y_i}\left(\nabla_w f_i(Y_i, w)\right), \tag{18}$$

where Y_i is a random variable on $\mathcal{Y}$ with its probability distribution function being the Boltzmann distribution of a system with the configuration set $\mathcal{Y}$, energy function $-f_i(y, w)$, and inverse temperature β:

$$\mathbb{P}(Y = y) = \frac{\exp(\beta f_i(y, w))}{\sum_{y\in\mathcal{Y}} \exp(\beta f_i(y, w))}, \quad y \in \mathcal{Y}. \tag{19}$$

B. Quantum Gibbs Sampling

We now describe the above problem in terms of Hermitian matrices we intend to simulate on a quantum computer. For each i, we assume that the range of $f_i(-, w) : \mathcal{Y} \to \mathbb{R}$ corresponds (up to a sign) to the spectrum of a diagonal Hermitian matrix H_i^w. We assume there is oracle access to H_i^w and its partial derivatives of the following form:

$$|k\rangle\,|z\rangle \mapsto |k\rangle\,|z \oplus (H_i^w)_{kk}\rangle\,, \qquad \text{and} \qquad |k\rangle\,|z\rangle \mapsto |k\rangle\,|z \oplus (\partial_j H_i^w)_{kk}\rangle \quad (\forall j).$$

Here and it what follows the notation ∂_j is used to abbreviate partial derivatives with respect to the vector w, i.e., $\partial_j = \partial/\partial w_j$. The assumption is that access to such an oracle would require logarithmically many qubits in the size of the Hermitian. For instance, if $f_i(-, w)$ is a quadratic polynomial in binary variables with quadratic and linear coefficients dependent on w, we may associate an Ising model with logarithmically many spins present in the system. For more-general remarks on the oracle construction, we refer the reader to [AGGW17, Section 2].

The operator $\max^\beta$ would then simply be the negative free energy of H_i^w:

$$\max_{y\in\mathcal{Y}}{}^{\beta} f_i(y, w) = \frac{1}{\beta}\log \mathrm{Tr}(\exp(-\beta H_i^w))\,. \tag{20}$$

Applying stochastic gradient descent for solving (13) would require calculation of the gradients of $f_i(y, w)$ with respect to w, which is $\mathrm{Tr}(A\rho)$ when $\rho = \frac{\exp(-\beta H)}{\mathrm{Tr}(\exp(-\beta H))}$ is the Gibbs state and every partial derivative is given by

$$\partial_k \max_y{}^{\beta} f_i(y, w) = \mathrm{Tr}\left[(-\partial_k H_i^w)\,\rho\right]\,.$$

This is exactly the type of quantity studied in [AGGW17]. They show that for $N \times N$ diagonal matrices H and A, such that $\|A\| \leq 1$ and given an inverse temperature β, the quantity $\mathrm{Tr}(A\rho)$ can be approximated up to an additive error of at most θ with high probability. We need to slightly modify the result of [AGGW17] for our application and for reference we first state their result.

Proposition III.1 (Corollary 12 in [AGGW17])**.** *Let $A, H \in \mathbb{R}^{n\times n}$ be diagonal matrices with $\|A\| \leq 1$. An additive θ-approximation of* $\mathrm{Tr}(A\rho)$ *can be computed using $O(\sqrt{n}/\theta)$ queries to A and H, and $\tilde{O}(\sqrt{n}/\theta)$ other gates.*

Firstly, the same result holds with the boundedness assumption on the norm of A being with respect to the infinity norm, i.e., $\|A\|_\infty \leq 1$. This allows for the arguments of [AGGW17, Sections 2.2.1 and 2.2.2] to similarly hold true. We also need a control over the success probability of the approximation achieved in the above statement.

Lemma III.2. *Suppose we have a unitary U acting on q qubits such that $U|0\ldots0\rangle = |0\rangle|\psi\rangle + |\Phi\rangle$, with $\langle 0| \otimes I |\Phi\rangle = 0$ and $\|\psi\|^2 = p \leq p_{\min}$ for some known bound $p_{\min}$. Let $\mu \in (0,1]$ be the allowed multiplicative error in our estimation of p. Then, with $O\left(\frac{\zeta}{\mu\sqrt{p_{\min}}}\right)$ uses of U and U^{-1} and using $O\left(\frac{\zeta q}{\mu\sqrt{p_{\min}}}\right)$ gates on the q qubits, we obtain a $\tilde{p}$ such that $|p-\tilde{p}| \leq \mu p$ with probability $O(1-\frac{1}{\zeta})$.*

Proof. The proof is similar to [AGGW17, Lemma 9] with M applications of U and U^{-1} in the amplitude estimation algorithm of [BHMT02, Theorem 12] except that we allow for $k \geq 2$. Then

$$|p-\tilde{p}| \leq 2\pi k\frac{\sqrt{p(1-p)}}{M} + k^2\frac{\pi^2}{M^2} \leq \frac{k\pi}{M}\left(2\sqrt{p} + \frac{k\pi}{M}\right).$$

So for $M \geq \frac{3\pi k}{\mu\sqrt{p_{\min}}}$ with probability $1-\frac{1}{2(k-1)}$, we get $|p-\tilde{p}| \leq \mu p$ and the result follows. □

We now have the following corollary.

Corollary. *Let $A, H \in \mathbb{R}^{n\times n}$ be diagonal matrices with $\|A\|_\infty \leq \Delta$ and $\|H\| \leq K$, and ρ be the Gibbs state of H at inverse temperature β. An additive θ-approximation of $\mathrm{Tr}(A\rho)$ can be computed with success probability $1-\zeta$ using $O(\frac{\sqrt{n}\Delta\beta K}{\zeta\theta})$ queries to A and H, and $\widetilde{O}(\frac{\sqrt{n}\Delta\beta K}{\zeta\theta})$ other gates*.*

Proof. This follows from the lemma above and its usage to generalize Corollaries 12 and 14 in [AGGW17]. □

Conditions 1. Let $\mathcal{Y}$ be a finite set and $f : \mathcal{Y} \times \mathbb{R}^D \to \mathbb{R}$ be a real-valued function. We assume that (1) there exist $\Delta > 0$ such that $\|\partial_k f\| \leq \Delta$ for all $w \in \mathbb{R}^D$, $y \in \mathcal{Y}$, and $k = 1,\ldots,D$; and, (2) there exist quantum oracles acting on $O(\mathrm{polylog}(\frac{1}{\delta}, |\mathcal{Y}|))$ qubits to compute f and $\partial_k f$ with additive error δ.

Theorem III.3. *Let $f_i : \mathcal{Y} \times \mathbb{R}^D \to \mathbb{R}$ be a real-valued function satisfying Conditions 1. Then the gradients of* (20) *with respect to the parameter vector w can be calculated in*

$$O\left(\frac{D^2\sqrt{|\mathcal{Y}|}\Delta\beta K}{\zeta\theta}\right)$$

queries to the oracles of f_i while using almost the same order of other gates. Here Δ is a bound on the partial derivatives $\|\partial_j f_i(y,w)\|$ at all w, K is a bound on values of all f_i, and $1-\zeta$ is the probability that all dimensions of the gradient estimate have an additive error of at most θ.

Proof. With H_w^i diagonal real-valued matrices realizing $f_i(-,w)$ and $A = \partial H_w^i$, the boundedness of derivatives, $\|f_i'(w)\|$ for all w, is equivalent to $\|A\| \leq \Delta$. In order to estimate all partial derivatives in the gradient with an additive error of at most θ successfully with probability at least $1-\zeta$, we may calculate each of the partial derivatives with success probability $1-\zeta/D$, because $(1-\frac{\zeta}{D})^D \geq 1-\frac{\zeta}{D}D = 1-\zeta$. By the previous corollary, each partial derivative is therefore calculated in $O\left(D\sqrt{|\mathcal{Y}|}\Delta\beta K/\zeta\theta\right)$ and, since there are D such partial derivatives, the result follows. □

*In the rest of this paper a characterization as such for the number of quantum gates will be written as *almost the same order of other gates.*

IV. COMPUTATIONAL COMPLEXITY

Stochastic average gradient (SAG) [SLRB17] and its variant SAGA [DBLJ14] are two optimization methods that are specifically designed for minimizing sums of smooth functions. SAG and SAGA usually perform better than the standard stochastic gradient descent. The general idea behind SAG and SAGA is to store the gradients for each of the n functions in a cache, and use their summation to obtain an estimation of the full gradient. Whenever we evaluate the gradient for one (or some) of the functions, we update the cache with the new gradients. Although the gradients in the cache are for older points, if the step size is small enough, the old points will be close to the current point and, because the functions are smooth, the gradients in the cache will not be far from the gradients for the current point; thus, using them will reduce the error of estimation of the full gradient, leading to an improved convergence rate.

Here we provide a time complexity analysis on the optimization of problem (12). Our approach is to use SAGA to optimize the smooth strongly convex objective function $f^{\beta}(w)$. A quantum Gibbs sampler will provide approximations of the derivatives of the functions $\max_y^{\beta} f_i(y, w)$ (but not exactly), as stated in the corollary in Theorem III.3. Consequently, we need to revisit the convergence of SAGA in the presence of approximation errors in calculating the gradients. This is done in Section IV A. Finally, we have to account for the error introduced by optimizing the smooth function $f^{\beta}(w)$ instead of f itself. We do this in Section IV B. In this section the notation $\langle -, - \rangle$ is used for inner products of real numbers.

In all subsequent sections we make the following assumption.

Conditions 2. Each function f_i is convex and the function r is strongly convex, resulting in each $g_i(w) = r(w) + \max_y f_i(y, w)$ being μ-strongly convex. The vector w is restricted to a convex compact set $\mathcal{W}$. Furthermore, the gradients of $r(w) + f_i(y, w)$ are ℓ-Lipschitz smooth, and the partial derivatives are bounded by

$$\Delta = \max_{w,i,j,y} \|\partial_j [r(w) + f_i(y, w)]\| \tag{21}$$

for every index i, every $y \in \mathcal{Y}$, every $w \in \mathcal{W}$, and every j-th component of w. Finally, by shifting each g_i via a constant if needed, we may assume that there is some $w_0 \in \mathcal{W}$ such that $g_i(w_0) = 0$ for all i.

Lemma IV.1. *Under Conditions 2, for any $w_1, w_2 \in \mathcal{W}$ we have*

$$\|w_1 - w_2\| \leq \frac{2\sqrt{D}\Delta}{\mu}.$$

Proof. Using strong convexity, for any i we have

$$\begin{aligned} g_i(w_1) &\geq g_i(w_2) + \langle w_1 - w_2, \nabla g_i(w_2) \rangle + \frac{\mu \|w_1 - w_2\|^2}{2}, \text{ and} \\ g_i(w_2) &\geq g_i(w_1) + \langle w_2 - w_1, \nabla g_i(w_1) \rangle + \frac{\mu \|w_1 - w_2\|^2}{2}. \end{aligned}$$

By adding these two inequalities,

$$\langle w_1 - w_2, \nabla g_i(w_1) \rangle + \langle w_2 - w_1, \nabla g_i(w_2) \rangle \geq \mu \|w_1 - w_2\|^2.$$

Using the Cauchy–Schwarz inequality,

$$\sqrt{\|w_1 - w_2\|^2}\sqrt{\|\nabla g_i(w_1)\|^2} + \sqrt{\|w_2 - w_1\|^2}\sqrt{\|\nabla g_i(w_2)\|^2} \geq \mu\|w_1 - w_2\|^2.$$

Finally, since $D\Delta^2 \geq \|\nabla g_i(w_1)\|^2$, we conclude that $2\sqrt{D}\Delta \geq \mu\|w_1 - w_2\|$. ☐

Lemma IV.2. *For any index i and point $w \in \mathcal{W}$, we have $|g_i(w)| \leq \frac{2D\Delta^2}{\mu}$.*

Proof. As in Conditions 2, there exists a $w_0 \in \mathcal{W}$ such that $g_i(w_0) = 0$. By convexity, we have

$$g_i(w_0) \geq g_i(w) + \langle w_0 - w, \nabla g_i(w) \rangle.$$

Using the Cauchy–Schwarz inequality and $g_i(w_0) = 0$, we get

$$\sqrt{\|w_0 - w\|^2}\sqrt{\|\nabla g_i(w)\|^2} \geq g_i(w)\,.$$

Using Lemma IV.1 and $\|\nabla g_i(w)\|^2 \leq D\Delta^2$, we get $\frac{2D\Delta^2}{\mu} \geq g_i(w)$. By a similar argument starting with

$$g_i(w) \geq g_i(w_0) + \langle w - w_0, \nabla g_i(w_0) \rangle,$$

we have $g_i(w) \geq -\frac{2D\Delta^2}{\mu}$, which completes the proof. ☐

A. A-SAGA: Approximate SAGA

We first analyze SAGA under an additive error in calculation of the gradients. We recall the SAGA algorithm from [DBLJ14]. In the approximate SAGA algorithm (A-SAGA), we have an estimate of the gradient with an additive error of at most $\theta/3$ † in each partial derivative appearing in the gradient. We let

$$g(w) = \frac{1}{n}\sum_i g_i(w)\,.$$

A-SAGA: Given the value of w^t and of each $g_i'(\phi_i^t)$ at the end of iteration t, the updates for iteration $t+1$ are as follows:

1. For a random choice of index j, set $\phi_j^{t+1} = w^t$, and $\phi_i^{t+1} = \phi_i^t$ for all i not equal j, and store $g_j'(\phi_j^{t+1}) + \Upsilon_j^{t+1}$ in a table, where the vector Υ_j^{t+1} is the additive error in the gradient estimation of $g_j'(\phi_j^{t+1})$.

2. Using $g_j'(\phi_j^{t+1}) + \Upsilon_j^{t+1}$, $g_j'(\phi_j^t) + \Upsilon_j^t$, and the table average, update w according to

$$\begin{aligned} v^{t+1} &= w^t - \gamma\left[g_j'(\phi_j^{t+1}) - g_j'(\phi_j^t) + \frac{1}{n}\sum_{i=1}^n g_i'(\phi_i^t)\right] + \gamma\Theta^{t+1} \text{ and} \\ w^{t+1} &= \prod_{\mathcal{W}} v^{t+1}, \end{aligned} \tag{22}$$

   where Θ^{t+1} contains the contributions of all additive errors, and $\prod_{\mathcal{W}}$ denotes projection onto the set $\mathcal{W}$.

†The division by 3 was chosen to simplify the formulae.

Here the update rule for SAGA from [DBLJ14, Equation (1)] has been modified to take into account an approximation error Θ^{t+1} in step $t+1$, where the vector Θ^{t+1} comprises all the additive errors, (that arise from the Gibbs sampler in the following section[‡]), i.e.,

$$\Theta^{t+1} = \Upsilon_j^{t+1} - \Upsilon_j^t + \frac{1}{n}\sum_{i=1}^{n} \Upsilon_i^t. \tag{23}$$

Note that for all vectors Υ_i^t, every element has an absolute value of at most $\theta/3$. Based on the definition of Θ^{t+1} from Equation (23), we can conclude that every element of the vector Θ^{t+1} is at most θ.

Defazio et al. prove the three lemmas below [DBLJ14]. Following their convention, all expectations are taken with respect to the choice of j at iteration $t+1$ and conditioned on w^t and each $g_i'(\phi_i^t)$ and additive errors Υ_j^t, unless otherwise stated.

Lemma IV.3. *Let $g(w) = \frac{1}{n}\sum_{i=1}^n g_i(w)$. Suppose each g_i is μ-strongly convex and has Lipschitz continuous gradients with constant L. Then for all w and w_*:*

$$\begin{aligned}\left\langle g'(w), w_* - w\right\rangle \le \frac{L-\mu}{L}\left[g(w_*) - g(w)\right] - \frac{\mu}{2}\left\|w_* - w\right\|^2 \\ - \frac{1}{2Ln}\sum_i \left\|g_i'(w_*) - g_i'(w)\right\|^2 - \frac{\mu}{L}\left\langle g'(w_*), w - w_*\right\rangle .\end{aligned}$$

Lemma IV.4. *For all ϕ_i and w_*:*

$$\frac{1}{n}\sum_i \left\|g_i'(\phi_i) - g_i'(w_*)\right\|^2 \le 2L\left[\frac{1}{n}\sum_i g_i(\phi_i) - g(w_*) - \frac{1}{n}\sum_i \left\langle g_i'(w_*), \phi_i - w_*\right\rangle\right].$$

The last lemma in [DBLJ14] is only true if the error in the A-SAGA update rule is disregarded. We therefore restate this lemma as follows.

Lemma IV.5. *For any ϕ_i^t, w_*, w^t and $\alpha > 0$, with v^{t+1} as defined in SAGA, if*

$$X = g_j'(\phi_j^t) - g_j'(w^t) + g'(w_*) - \frac{1}{n}\sum_i g_i'(\phi_i^t),$$

it holds that

$$\mathbb{E}[X] = g'(w^t) - g'(w_*) \tag{24}$$

$$\mathbb{E}\|X\|^2 \le (1+\alpha^{-1})\mathbb{E}\left\|g_j'(\phi_j^t) - g_j'(w_*)\right\|^2 + (1+\alpha)\mathbb{E}\left\|g_j'(w^t) - g_j'(w_*)\right\|^2 - \alpha\left\|g'(w^t) - g'(w_*)\right\|^2. \tag{25}$$

The main result of [DBLJ14] creates a bound for $\|w^t - w_*\|$ using the Lyapunov function T defined as

$$T^t := T(w^t, \{\phi_i^t\}_{i=1}^n) := \frac{1}{n}\sum_i g_i(\phi_i^t) - g(w_*) - \frac{1}{n}\sum_i \left\langle g_i'(w_*), \phi_i^t - w_*\right\rangle + c\left\|w^t - w_*\right\|^2, \tag{26}$$

by proving the inequality $\mathbb{E}[T^{t+1}] \le (1 - \frac{1}{\tau})T^t$. We will follow the logic of the same proof to obtain a similar result in the case that an additive error on the gradients exists.

[‡] In fact, the Gibbs sampler is used to calculate each directional derivative up to an additive error. Therefore, the approximation errors in all the terms in the square brackets in Equation (22) contribute to the bound on Θ. More precisely, if the Gibbs sampler calculates the derivatives with error $\frac{\theta}{3}$, then $\|\Theta^{t+1}\|_\infty \le \theta$.

Theorem IV.6. *Let w_* be the optimal solution, γ be the step size in Equation (22), c be the constant in Equation (26), α be the constant in Equation (25), and θ be a bound on the precision of a subroutine calculating the gradients of g_i at every point. There exists a choice of γ, c, τ, and θ such that for all t,*

$$\mathbb{E}[T^{t+1}] \leq (1 - \frac{1}{\tau})T^t \,.$$

Proof. The first three terms in T^{t+1} can be bounded in a way similar to the proof of [DBLJ14, Theorem 1]:

$$\mathbb{E}\left[\frac{1}{n}\sum_i g_i(\phi_i^{t+1})\right] = \frac{1}{n}g(w^t) + \left(1-\frac{1}{n}\right)\frac{1}{n}\sum_i g_i(\phi_i^t)$$

$$\mathbb{E}\left[-\frac{1}{n}\sum_i \left\langle g_i'(w_*), \phi_i^{t+1} - w_*\right\rangle\right] = -\frac{1}{n}\left\langle g'(w_*), w^t - w_*\right\rangle - \left(1-\frac{1}{n}\right)\frac{1}{n}\sum_i \left\langle g_i'(w_*), \phi_i^t - w_*\right\rangle .$$

The last term is bounded by the inequality $c\left\|w^{t+1}-w_*\right\|^2 = c\left\|\prod_{\mathcal{W}} v^{t+1} - \prod_{\mathcal{W}}[w_* - \gamma g'(w_*)]\right\|^2 \leq c\left\|v^{t+1} - w_* + \gamma g'(w_*)\right\|^2$, by the optimality of w_* and non-expansiveness of the projection operator $\prod_{\mathcal{W}}$. We can now bound the expected value of the right-hand side of this inequality in terms of X and $\|w^t - w_*\|$ by expanding the quadratics.

$$\begin{aligned}
c\mathbb{E}\left\|v^{t+1} - w_* + \gamma g'(w_*)\right\|^2 &= c\mathbb{E}\left\|w^t - w_* + \gamma X + \gamma\Theta^{t+1}\right\|^2 \\
&= c\left\|w^t - w_*\right\|^2 + \left\{2c\mathbb{E}\left[\left\langle \gamma X + \gamma\Theta^{t+1}, w^t - w_*\right\rangle\right] + c\mathbb{E}\left\|\gamma X + \gamma\Theta^{t+1}\right\|^2\right\} \\
&= c\left\|w^t - w_*\right\|^2 + \Big\{ -2c\gamma\left\langle g'(w^t) - g'(w_*), w^t - w_*\right\rangle + 2c\gamma\mathbb{E}\left[\left\langle\Theta^{t+1}, w^t - w_*\right\rangle\right] \\
&\qquad + c\gamma^2\mathbb{E}\left\|X\right\|^2 + 2c\gamma^2\mathbb{E}\left[\left\langle\Theta^{t+1}, X\right\rangle\right] + c\gamma^2\mathbb{E}\left\|\Theta^{t+1}\right\|^2\Big\}
\end{aligned}$$

Using Jensen's inequality applied to the square root function, in the second inequality below, and then using $\sqrt{x} \leq \frac{1}{2} + \frac{x}{2}$, we have

$$\mathbb{E}\left[\left\langle\Theta^{t+1}, X\right\rangle\right] \leq \theta\sqrt{D}\mathbb{E}\left[\|X\|\right] \leq \theta\sqrt{D}\sqrt{\mathbb{E}\left[\|X\|^2\right]} \leq \frac{\theta\sqrt{D}}{2} + \frac{\theta\sqrt{D}\mathbb{E}\|X\|^2}{2} .$$

We now apply Lemma IV.5 and the assumption that $\|\Theta^{t+1}\| \leq \theta\sqrt{D}$.

$$\begin{aligned}
&c\mathbb{E}\left\|v^{t+1} - w_* + \gamma g'(w_*)\right\|^2 \\
&\leq c\left\|w^t - w_*\right\|^2 + \Big\{ -2c\gamma\left\langle g'(w^t) - g'(w_*), w^t - w_*\right\rangle + 2c\gamma\mathbb{E}\left[\left\langle\Theta^{t+1}, w^t - w_*\right\rangle\right] \\
&\qquad + \left(c\gamma^2(1+\theta\sqrt{D})\right)\mathbb{E}\left\|X\right\|^2 + c\gamma^2\theta\sqrt{D} + c\gamma^2\mathbb{E}\left\|\Theta^{t+1}\right\|^2 \Big\} \\
&\leq c\left\|w^t - w_*\right\|^2 + \Big\{-2c\gamma\left\langle g'(w^t), w^t - w_*\right\rangle + 2c\gamma\left\langle g'(w_*), w^t - w_*\right\rangle + 2c\gamma\theta\sqrt{D}\|w^t - w_*\| \\
&\qquad - \left(c\gamma^2(1+\theta\sqrt{D})\right)\alpha\left\|g'(w^t) - g'(w_*)\right\|^2 \\
&\qquad + \left(1+\alpha^{-1}\right)\left(c\gamma^2(1+\theta\sqrt{D})\right)\mathbb{E}\left\|g_j'(\phi_j^t) - g_j'(w_*)\right\|^2 \\
&\qquad + (1+\alpha)\left(c\gamma^2(1+\theta\sqrt{D})\right)\mathbb{E}\left\|g_j'(w^t) - g_j'(w_*)\right\|^2 \\
&\qquad + c\gamma^2\theta\sqrt{D} + c\gamma^2\theta^2 D\Big\} .
\end{aligned}$$

We now apply Lemma IV.3 and Lemma IV.4 to respectively bound $-2c\gamma \left\langle g'(w^t), w^t - w_* \right\rangle$ and $\mathbb{E}\left\|g'_j(\phi^t_j) - g'_j(w_*)\right\|^2$:

$$
\begin{aligned}
c\mathbb{E}\left\|w^{t+1} - w_*\right\|^2 \le (c - c\gamma\mu)\left\|w^t - w_*\right\|^2 + \Bigg\{ & \left((1+\theta\sqrt{D})(1+\alpha)c\gamma^2 - \frac{c\gamma}{L}\right)\mathbb{E}\left\|g'_j(w^t) - g'_j(w_*)\right\|^2 \\
& - \frac{2c\gamma(L-\mu)}{L}\left[g(w^t) - g(w_*) - \left\langle g'(w_*), w^t - w_*\right\rangle\right] - c\gamma^2(1+\theta\sqrt{D})\alpha\left\|g'(w^t) - g'(w_*)\right\|^2 \\
& + 2\left(1+\theta\sqrt{D}\right)\left(1+\alpha^{-1}\right)c\gamma^2 L\left[\frac{1}{n}\sum_i g_i(\phi^t_i) - g(w_*) - \frac{1}{n}\sum_i\left\langle g'_i(w_*), \phi^t_i - w_*\right\rangle\right] \\
& + c\gamma^2\theta\sqrt{D} + c\gamma^2\theta^2 D + 2c\gamma\theta\sqrt{D}\|w^t - w_*\|\Bigg\}.
\end{aligned}
$$

As in [DBLJ14, Theorem 1], we pull out a $\frac{1}{\tau}$ factor of T^t and use the above inequalities, taking into account the contributions from the three error terms above:

$$
\begin{aligned}
\mathbb{E}[T^{t+1}] - T^t \le & -\frac{1}{\tau}T^t + \left(\frac{1}{n} - \frac{2c\gamma(L-\mu)}{L} - 2c\gamma^2\mu\alpha(1+\theta\sqrt{D})\right)\left[g(w^t) - g(w_*) - \left\langle g'(w_*), w^t - w_*\right\rangle\right] \\
& + \left(\frac{1}{\tau} + 2(1+\alpha^{-1})(1+\theta\sqrt{D})c\gamma^2 L - \frac{1}{n}\right)\left[\frac{1}{n}\sum_i g_i(\phi^t_i) - g(w_*) - \frac{1}{n}\sum_i\left\langle g'_i(w_*), \phi^t_i - w_*\right\rangle\right] \\
& + \left(\frac{1}{\tau} - \gamma\mu\right)c\left\|w^t - w_*\right\|^2 + \left((1+\alpha)\gamma(1+\theta\sqrt{D}) - \frac{1}{L}\right)c\gamma\mathbb{E}\left\|g'_j(w^t) - g'_j(w_*)\right\|^2 \\
& + \left\{c\gamma^2\theta\sqrt{D} + c\gamma^2\theta^2 D + 2c\gamma\theta\sqrt{D}\|w^t - w_*\|\right\}
\end{aligned}
$$

According to Lemma A.1 in Appendix A, we can ensure that all round parentheses in the first three lines are non-positive by setting the parameters according to

$$
\begin{aligned}
\gamma = \frac{1}{(1+\alpha)(1+\theta\sqrt{D})L}, \quad c = \frac{2}{n\gamma}, \quad \alpha = 8, \quad \frac{1}{\tau} = \min\left\{\frac{1}{2n}, \frac{\gamma\mu}{2}\right\}, \\
\text{and } \theta = \min\left\{\frac{1}{\sqrt{D}}, \frac{\mu\|w^t - w_*\|^2}{2\sqrt{D}\left(\frac{5}{18L} + 2\|w^t - w_*\|\right)}\right\}.
\end{aligned}
\tag{27}
$$

With this setting of the parameters,

$$
\left(\frac{1}{\tau} - \gamma\mu\right)c\left\|w^t - w_*\right\|^2 + \left\{c\gamma^2\theta\sqrt{D} + c\gamma^2\theta^2 D + 2c\gamma\theta\sqrt{D}\|w^t - w_*\|\right\} \le 0\,.
$$

Using the non-negativity of all square brackets completes the proof. □

The next theorem provides the time complexity of optimizing the smooth approximation f^β via A-SAGA depending on the condition number L/μ, where L is the Lipschitz constant of the gradient of f^β.

Theorem IV.7. *Under Conditions 2, and given ϵ as a target precision, A-SAGA finds a point in the ϵ-neighbourhood of w^β_* defined in* (17) *using*

$$
O\left(\left(n + \frac{\beta D\Delta^2 + \ell}{\mu}\right)\left(\log\frac{1}{\epsilon} + \log n - \log(\beta D\Delta^2 + \ell)\right)\right)
$$

gradient evaluations.

Proof. As in [DBLJ14, Corollary 1], we note that $c\left\|w^t - w_*\right\|^2 \le T^t$. Therefore, by chaining the expectations

$$\mathbb{E}\left[\left\|w^t - w_*\right\|^2\right] \le C_0\left(1 - \frac{1}{\tau}\right)^t,$$

where

$$C_0 = \left\|w^0 - w_*\right\|^2 + \frac{1}{c}\left[f(w^0) - \left\langle f'(w_*), w^0 - w_*\right\rangle - f(w_*)\right].$$

Therefore, we should have

$$t \ge \frac{\log\frac{1}{\epsilon} + \log C_0}{-\log\left(1 - \frac{1}{\tau}\right)}.$$

Using the inequality $\log(1-x) \le -x$, it suffices that

$$t \ge \tau\left(\log\frac{1}{\epsilon} + \log C_0\right).$$

From (27), we know that

$$\tau = \max\left\{2n, \frac{2}{\gamma\mu}\right\} \le \max\left\{2n, \frac{36L}{\mu}\right\},$$

where we have used the fact that $\theta \le \frac{1}{\sqrt{D}}$. So, we get

$$t \ge \max\left\{2n, \frac{36L}{\mu}\right\}\left(\log\frac{1}{\epsilon} + \log C_0\right).$$

We recall that $r(w) + \max^{\beta}_{y\in\mathcal{Y}} f_i(y,w) = \max^{\beta}_{y\in\mathcal{Y}} r(w) + f_i(y,w)$ has Lipschitz continuous gradients with parameter $\beta D\Delta^2 + \ell$ (see [BT12]), so the function f^β has Lipschitz continuous gradients with parameter $L = \beta D\Delta^2 + \ell$. We also note that $C_0 = O(1/c) = O(nL) = O(\frac{n}{\beta D\Delta^2 + \ell})$. Therefore, when f^β is sufficiently smooth, that is,

$$\frac{L}{\mu} = \frac{\beta D\Delta^2 + \ell}{\mu} \le \frac{n}{18},$$

we have

$$t = O\left(n\left(\log\frac{1}{\epsilon} + \log n - \log(\beta D\Delta^2 + \ell)\right)\right),$$

and otherwise

$$t = O\left(\frac{\beta D\Delta^2 + \ell}{\mu}\left(\log\frac{1}{\epsilon} + \log n - \log(\beta D\Delta^2 + \ell)\right)\right).$$

We can combine these two bounds into one:

$$t = O\left(\left(n + \frac{\beta D\Delta^2 + \ell}{\mu}\right)\left(\log\frac{1}{\epsilon} + \log n - \log(\beta D\Delta^2 + \ell)\right)\right).$$

This completes the proof. □

Remark. The number of gradient evaluations in Theorem IV.7 is $O\left(\log\frac{1}{\epsilon}\right)$ in terms of ϵ only. Also, based on Equation (27), we have $\theta = O(\epsilon)$.

B. Complexity of Solving the Min-Max Problem

Recall that w_* denotes the minimum of $f(w)$ and w_*^β the minimum of the smooth approximation $f^\beta(w)$. In this section, we analyze the inverse temperature β at which sampling from the quantum Gibbs sampler has to happen in order for w_*^β to be a sufficiently good approximation of the original optimum w_*.

Lemma IV.8. *To solve the original problem* (13) *with ϵ-approximation, it suffices to solve the smooth approximation* (16) *for $\beta > \frac{\log|\mathcal{Y}|}{\epsilon}$ with precision $\epsilon - \frac{\log|\mathcal{Y}|}{\beta}$.*

Proof. The softmax operator $\max^\beta$ is an upper bound on the max function satisfying

$$\max_{y\in\mathcal{Y}} \upsilon(y) \le \max_{y\in\mathcal{Y}}{}^\beta \upsilon(y) \le \max_{y\in\mathcal{Y}} \upsilon(y) + \frac{\log|\mathcal{Y}|}{\beta}, \tag{28}$$

for any function υ [NS16]. Using this inequality and the optimality of w_* and w_*^β, it follows that

$$f(w_*) \le f(w_*^\beta) \le f^\beta(w_*^\beta) \le f^\beta(w_*) \le f(w_*) + \frac{\log|\mathcal{Y}|}{\beta}$$

Therefore, $0 \le f^\beta(w_*^\beta) - f(w_*) \le \frac{\log|\mathcal{Y}|}{\beta}$. So, in order to solve the original problem within an error of ϵ, that is, $f(w^t) - f(w_*) \le \epsilon$, it is sufficient to have $\frac{\log|\mathcal{Y}|}{\beta} < \epsilon$, and

$$f^\beta(w^t) - f^\beta(w_*^\beta) \le \epsilon - \frac{\log|\mathcal{Y}|}{\beta}.$$

resulting in

$$f^\beta(w^t) - f(w_*) \le \epsilon,$$

and using the fact that $f(w^t) \le f^\beta(w^t)$, we can conclude that

$$f(w^t) - f(w_*) \le \epsilon,$$

completing the proof. □

Lemma IV.9. *In solving problem* (17) *with A-SAGA we have*

$$\mathbb{E}\left[f^\beta(w^t) - f^\beta(w_*)\right] \le \frac{L}{2} C_0 \left(1 - \frac{1}{\tau}\right)^t.$$

Proof. By the descent lemma [Nes13, Lemma 1.2.4], we have

$$f^\beta(w) - f^\beta(w_*) \le \langle \nabla f^\beta(w_*), w - w_* \rangle + \frac{L}{2}\|w - w_*\|^2.$$

The smoothness of the function f^β, the optimality of w_*, and the convexity of $\mathcal{W}$ imply that $\langle \nabla f^\beta(w_*), w - w_* \rangle \le 0$, and therefore

$$f^\beta(w) - f^\beta(w_*) \le \frac{L}{2}\|w - w_*\|^2. \tag{29}$$

The result now follows from Theorem IV.6. □

The above two lemmas are useful for achieving an approximation of the optimal *value* of f by doing so for f^β.

Theorem IV.10. *Under Conditions 2, A-SAGA applied to the function f^β at*

$$\beta = \frac{2\log|\mathcal{Y}|}{\epsilon} \tag{30}$$

requires $O\left((\frac{D\Delta^2\log|\mathcal{Y}|}{\mu\epsilon}+\frac{\ell}{\mu})\left(\log\frac{1}{\epsilon}+\log n\right)\right)$ gradient evaluations to find a point at which the value of f is in the ϵ-neighbourhood of the optimal value of f, provided ϵ is sufficiently small.

Proof. Based on Lemma IV.8, it suffices to find a point at which the value of f^β is in the $\left(\epsilon-\frac{\log|\mathcal{Y}|}{2\beta}\right)$-neighbourhood of its optimal value. Using Lemma IV.9, we need

$$\mathbb{E}\left[f(w^t)-f(w_*)\right] \le \frac{L}{2}C_0\left(1-\frac{1}{\tau}\right)^t \le \epsilon - \frac{\log|\mathcal{Y}|}{2\beta} = \frac{\epsilon}{2}\,. \tag{31}$$

Following the same steps as in Theorem IV.7, we conclude that

$$t \ge \frac{\log\frac{2}{\epsilon}+\log\frac{C_0L}{2}}{-\log\left(1-\frac{1}{\tau}\right)}\,. \tag{32}$$

Using the inequality $\log(1-x)\le -x$, it suffices that

$$t \ge \tau\left(\log\frac{2}{\epsilon}+\log\frac{C_0L}{2}\right). \tag{33}$$

From (27), we know that

$$\tau = \max\left\{2n,\frac{2}{\gamma\mu}\right\} \le \max\left\{2n,\frac{36L}{\mu}\right\}, \tag{34}$$

where we have used the fact that $\theta \le \frac{1}{\sqrt{D}}$.

We recall that $r(w)+\max^\beta_{y\in\mathcal{Y}} f_i(y,w) = \max^\beta_{y\in\mathcal{Y}} r(w)+f_i(y,w)$ has Lipschitz continuous gradients with parameter $\beta D\Delta^2+\ell$ (see [BT12]), so the function f^β has Lipschitz continuous gradients with parameter $L=\beta D\Delta^2+\ell$. Hence,

$$\tau \le \max\left\{2n,\frac{36(\beta D\Delta^2+\ell)}{\mu}\right\}. \tag{35}$$

Since $\beta=\frac{2\log|\mathcal{Y}|}{\epsilon}$, for sufficiently small ϵ, the second term dominates and we have

$$\tau \le \frac{36(\beta D\Delta^2+\ell)}{\mu}\,. \tag{36}$$

Replacing the values of L, μ, and τ in the formulae, we get

$$t \ge \frac{36(\frac{2\log|\mathcal{Y}|}{\epsilon}D\Delta^2+\ell)}{\mu}\left(\log\frac{2}{\epsilon}+\log\frac{C_0L}{2}\right). \tag{37}$$

Note that $C_0L=O(\frac{L}{c})=O(n)$, so the time complexity is $t=O\left((\frac{D\Delta^2\log|\mathcal{Y}|}{\mu\epsilon}+\frac{\ell}{\mu})\left(\log\frac{1}{\epsilon}+\log n\right)\right)$, proving the claim. ∎

Remark. The number of gradient evaluations in Theorem IV.10 is $O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$ in terms of ϵ. We should mention that the best complexity (in terms of precision) for optimizing Equation (12) is $O(\frac{1}{\epsilon})$ [SZ13, Nes05], matching the theoretical optimal bound. Our result is close to optimal (up to a logarithmic factor).

It is also interesting to observe that based on Equation (27), we have $\theta = O(\sqrt{\epsilon})$, which means that to optimize f, we do not need as much precision as for optimizing f^β. Surprisingly, the error in gradient evaluations could be orders of magnitude larger than the desired precision and the algorithm would still converge with the same rate as in SAGA.

Example. A special case of practical importance is when the functions f_i are linear in w, that is, in the form of (43). In this case our optimization problem is

$$f(w) = \lambda\frac{\|w\|^2}{2} + \frac{1}{n}\sum_{i=1}^{n}\max_{y\in\mathcal{Y}}\{a_{i,y}w + b_{i,y}\}\,. \tag{38}$$

Let $\mathcal{W} = \mathbb{B}^D(0,1)$ be the unit ball centred at the origin of $\mathbb{R}^D$, where D is the dimension of w. For the linear functions, the Lipschitz constant of the gradients is 0, as the gradient does not change. For the regularizer $\lambda\frac{\|w\|^2}{2}$, the Lipschitz constant of the gradient is λ. Therefore, $\ell = \lambda$. For the bound on the partial derivatives of the functions, we have $\Delta = \lambda + \max_i \max_j \max_y |a_{i,y,j}|$, where $a_{i,y,j}$ is the j-th element of the vector $a_{i,y}$.

A further special case is when the functions f_i remain linear in w but are quadratic in y, e.g., the energy function of an Ising model,

$$f(w) = \lambda\frac{\|w\|^2}{2} + \frac{1}{n}\sum_{i=1}^{n}\max_{y\in\mathcal{Y}}\{yJ_iy^T + h_iy^T\}\,, \tag{39}$$

where $\mathcal{Y} = \{-1,1\}^m$, $J_i \in \mathbb{R}^{m\times m}$, and $h_i \in \mathbb{R}^m$, for an Ising model with m particles. Here the vector w includes all the elements of the matrices J_i and vectors h_i for all i. In this case $\mathcal{W}$ is the unit ball of dimension $D = nm(m+1)$ around the origin. Similar to the previous example, since f_i are linear, we still have $\ell = \lambda$. For the bound on the gradient of the functions, we have $\Delta = \lambda + 1$, where we use the fact that the elements of y are in $\{-1,1\}$.

Finally, we can show convergence of A-SAGA to an approximation of the optimal solution of f.

Corollary. *With the same conditions as Theorem IV.10, A-SAGA finds a point in the ϵ-neighbourhood of w_* with $O\Big(\big(\frac{D\Delta^2\log|\mathcal{Y}|}{\mu\epsilon} + \frac{\ell}{\mu}\big)\big(\log\frac{1}{\epsilon} + \log n - \log\mu\big)\Big)$ gradient evaluations.*

Proof. This follows from the previous theorem and the definition of strong convexity. □

C. Complexity of Original SAGA without Additive Error

To optimize f^β using SAGA with exact gradient evaluations, instead of the parameters from Equation (27), we set

$$\gamma = \frac{1}{2(\mu n + L)}, \quad c = \frac{1}{2\gamma(1-\gamma\mu)n}, \quad \alpha = \frac{2\mu n + L}{L}, \quad \text{and} \quad \frac{1}{\tau} = \gamma\mu \tag{40}$$

according to [DBLJ14], with no assignment of θ (since there are no additive errors after all). Following the same steps as in the proof of Theorem IV.7, Theorem IV.10 and its corollary, we may optimize f^β in order to estimate the optimal solution of f.

Theorem IV.11. *Let f, f^β, r, f_i, ℓ, μ, Δ, and ϵ be given as in Theorem IV.7. Then SAGA uses*

$$O\left(\left(n+\frac{\beta D\Delta^2+\ell}{\mu}\right)\left(\log\frac{1}{\epsilon}+\log n-\log\left(\mu n+\beta D\Delta^2+\ell\right)\right)\right) \tag{41}$$

gradient evaluations to find a point in the ϵ-neighbourhood of w_*^β defined in (17) *and*

$$O\left(\left(n+\frac{D\Delta^2\frac{\log|\mathcal{Y}|}{\epsilon}+\ell}{\mu}\right)\left(\log\frac{1}{\epsilon}+\log n\right)\right) \tag{42}$$

gradient evaluations to find an ϵ-approximation of the optimal value of f.

It is clear that the scaling in Equation (41) with respect to all parameters is similar to Theorem IV.7 and the scaling in Equation (42) is similar to Theorem IV.10, except for an extra n term added in the first parentheses.

Remark. We summarize the results of Theorem IV.7, Theorem IV.10, and Theorem IV.11 by making the remark that with $O(\epsilon)$ and $O(\sqrt{\epsilon})$ additive errors in gradient evaluations, the scaling of SAGA and A-SAGA for respectively optimizing f^β and f remains similar.

D. A Quantum Algorithm for Solving the Smooth Approximation

In Theorem IV.7 and Theorem IV.10, we have assumed that the additive error in calculating the partial derivatives is always at most $\theta/3$. Using the quantum Gibbs sampler from Section III B, we can guarantee such an upper bound only with a non-zero probability of failure. As shown in Theorem III.3, the gradients of the function $\max_y^\beta r(w)+f_i(y,w)$ can be estimated with additive errors of at most θ in all partial derivatives appearing in the gradient with probability at least $1-\zeta$ in $O\left(D^2\sqrt{|\mathcal{Y}|}\Delta\beta K/\zeta\theta\right)$ queries, where Δ is a bound on the norms of the partial derivatives and K is a bound on the function values. We now propose a quantum algorithm, called Q-SAGA, for optimizing the smooth approximation function $f^\beta(w)$ (by combining Theorem III.3 with Theorem IV.7) and for optimizing the original function f (by combining Theorem III.3 with Theorem IV.10), using a quantum Gibbs sampler. Here β is a fixed inverse temperature. The higher this value is, the more accurate the approximation of $f(w)$ via $f^\beta(w)$ will be. This is at the expense of a worse scaling in terms of β.

Lemma IV.12. *Under Conditions 1 and 2, each gradient evaluation takes*

$$O\left(\frac{D^{3.5}\beta(\frac{1}{\beta D\Delta^2+\ell}+\sqrt{\epsilon})\sqrt{|\mathcal{Y}|}\Delta^3}{\zeta\mu^2\epsilon}\right)$$

queries to the oracle for one of the f_i and almost the same order of other quantum gates, where $1-\zeta$ is the probability of the Gibbs sampler returning a gradient estimate whose additive errors in all partial derivatives are at most θ, where θ is determined based on Equation (27)*.*

Proof. Each iteration of A-SAGA requires finding all partial derivatives of f_i for a random choice of i with precision θ. Since ϵ is small, based on Equation (27), we have $\theta=O(\frac{1}{\sqrt{D}}\frac{\mu\epsilon}{\frac{1}{\beta D\Delta^2+\ell}+\sqrt{\epsilon}})$. We also note that Δ, which is a bound on the partial derivatives of $\max_y r(w)+f_i(y,w)$, is also a bound on the partial derivatives of $\max_y{}^{\beta} r(w) + f_i(y, w)$, because $\nabla_w \max_y{}^{\beta} r(w) + f_i(y, w) = \mathbb{E}(\nabla_w[r(w) + f_i(Y_i, w)])$. Also, according to Lemma IV.2, $\frac{2D\Delta^2}{\mu}$ is a bound on the values of $r(w) + f_i(y, w)$. Therefore, each gradient calculation is performed in $O\left(D^{3.5}\beta(\frac{1}{\beta D\Delta^2+\ell} + \sqrt{\epsilon})\sqrt{|\mathcal{Y}|}\Delta^3/\zeta\mu^2\epsilon\right)$ according to Theorem III.3, concluding the proof. $\square$

Theorem IV.13. *Under Conditions 1 and 2, given sufficiently small $\epsilon > 0$ as a target precision, Q-SAGA finds a point in the ϵ-neighbourhood of w_*^{β} defined in (17) with probability at least $1/2$, in*

$$O\left(\frac{D^{3.5}\beta\sqrt{|\mathcal{Y}|}\Delta^3}{(\beta D\Delta^2+\ell)\mu^2\epsilon} n^2 \left(\log\frac{1}{\epsilon} + \log n\right)^2\right)$$

queries to the oracle for one of the f_i and almost the same order of other quantum gates when f^{β} is sufficiently smooth (i.e., the condition number L/μ is sufficiently small), and otherwise in

$$O\left(\frac{D^{3.5}\beta\sqrt{|\mathcal{Y}|}\Delta^3}{\mu^4\epsilon}(\beta D\Delta^2+\ell)\left(\log\frac{1}{\epsilon} + \log n\right)^2\right)$$

queries to the oracle for one of the f_i and almost the same order of other quantum gates. In both cases, the complexity is $O(\frac{1}{\epsilon}\log^2\frac{1}{\epsilon})$ in terms of ϵ only.

Proof. Since ϵ is small, and β, M, and ℓ are fixed, we can simplify the result of Lemma IV.12 and conclude that each gradient could be estimated in $O\left(\frac{D^{3.5}\sqrt{|\mathcal{Y}|}\Delta^3}{(D\Delta^2+\ell/\beta)\zeta\mu^2\epsilon}\right)$ queries to the oracle for one of the f_i and the same order of other quantum gates. Suppose the probability of failure in satisfying the bound $\theta/3$ for all partial derivatives appearing in the gradients and for any single iteration of gradient evaluation is at most $\zeta = \frac{1}{2T}$ for some positive integer T. Then in T iterations of Q-SAGA, the probability of all gradient evaluations satisfying the additive $\theta/3$ upper bound is larger than $(1-\zeta)^T \geq 1 - \left(\frac{1}{2T}\right)T \geq \frac{1}{2}$. The result follows from Theorem IV.7. $\square$

Theorem IV.14. *Under Conditions 1 and 2, given sufficiently small $\epsilon > 0$ as a target precision, Q-SAGA finds a point in the ϵ-neighbourhood of w_* defined in (13) with probability at least $1/2$ in*

$$O\left(\left(\frac{D^{5.5}\Delta^7\sqrt{|\mathcal{Y}|}\log^3|\mathcal{Y}|}{\mu^4\epsilon^{3.5}}\right)\left(\log\frac{1}{\epsilon} + \log n\right)^2\right)$$

queries to the oracle for one of the f_i and almost the same order of other quantum gates. This is $O(\frac{1}{\epsilon^{3.5}}\log^2\frac{1}{\epsilon})$ in terms of ϵ only.

Proof. By replacing the value of β from Equation (30), each gradient evaluation costs

$$O\left(\frac{1}{\zeta\mu\epsilon}D^{2.5}\frac{\log|\mathcal{Y}|}{\epsilon}\left(\frac{1}{\frac{\log|\mathcal{Y}|}{\epsilon}D\Delta^2+\ell} + \sqrt{\epsilon}\right)\sqrt{|\mathcal{Y}|}\Delta\frac{D\Delta^2}{\mu}\right)$$

queries to the oracle for one of the f_i and the same order of other quantum gates according to Lemma IV.12. Using the fact that ϵ is small, this simplifies to $O\left(\frac{D^{3.5}\log|\mathcal{Y}|\sqrt{|\mathcal{Y}|}\Delta^3}{\zeta\mu^2\epsilon^{1.5}}\right)$. From Theorem IV.10, we know that we need $O\left(\left(\frac{D\Delta^2\log|\mathcal{Y}|}{\mu\epsilon} + \frac{\ell}{\mu}\right)\left(\log\frac{1}{\epsilon} + \log n\right)\right)$ gradient evaluations. Using the fact that ϵ is small, this simplifies to $O\left(\left(\frac{D\Delta^2\log|\mathcal{Y}|}{\mu\epsilon}\right)\left(\log\frac{1}{\epsilon} + \log n\right)\right)$. By multiplying the number of gradient estimations with the complexity of each, we get a total complexity of $O\left(\left(\frac{D^{3.5}\log|\mathcal{Y}|\sqrt{|\mathcal{Y}|}\Delta^3}{\zeta\mu^2\epsilon^{1.5}}\right)(\frac{D\Delta^2\log|\mathcal{Y}|}{\mu\epsilon})\left(\log\frac{1}{\epsilon}+\log n\right)\right)$. As with the proof of Theorem IV.13 we should satisfy a failure probability of at most $O(\frac{1}{T})$ and get a total complexity of $O\left(\left(\frac{D^{3.5}\log|\mathcal{Y}|\sqrt{|\mathcal{Y}|}\Delta^3}{\mu^2\epsilon^{1.5}}\right)(\frac{D\Delta^2\log|\mathcal{Y}|}{\mu\epsilon})^2\left(\log\frac{1}{\epsilon}+\log n\right)^2\right)$ which, after simplification, completes the proof. □

V. NUMERICAL EXPERIMENTS

We compare the optimization of the function f, as defined in (12), with its smooth approximation f^β, as defined in (16). To exclude the effects of sampler errors and noise, we restrict our experiments to small instances (i.e., we restrict the size of the sets $\mathcal{Y}$) in order to be able to find the value of the softmax operator and its gradient exactly.

Also, for simplicity, we restrict our experiments to the case where each f_i is a linear function of w, and r is a quadratic function of w:

$$f_i(y,w) = a_{i,y}^T(w-b_i') + b_{i,y}, \quad y\in\mathcal{Y}, w\in\mathbb{R}^D\,, \tag{43}$$

$$r(w) = \frac{\lambda}{2}\|w\|^2, \quad \lambda\in\mathbb{R}^+, w\in\mathbb{R}^D\,. \tag{44}$$

This guarantees the strong convexity of f. Here, the elements $y\in\mathcal{Y}$ are used as indices for their corresponding $a_{i,y}$ and $b_{i,y}$ vectors. All coefficient vectors $a_{i,y}$ and $b_{i,y}$ are randomly generated according to the Cauchy distribution, and all vectors b_i' are randomly generated according to a uniform distribution. The reason we choose the Cauchy distribution for $a_{i,y}$ and $b_{i,y}$, is its thick tail, which results in having occasional extreme values for the coefficients. The reason we choose the uniform distribution for b_i' is to avoid the functions f_i having a similar minimum, which makes the problem easy.

In our experiment, we generate a random objective function with $D = 10$ parameters, i.e., $w\in\mathbb{R}^{10}$, where w is initialized to the vector $w = (10, 10, \ldots, 10)^T$. We use 200 summand functions f_i, i.e., $n = 200$. We set $\lambda = 2$ and $\mathcal{Y} = \{1, 2, \ldots, 100\}$. We generate the vectors b_i' from the uniform distribution over the set $[0, 10000]^{10}$.

We benchmark four gradient descent schemes: (1) stochastic gradient descent (SGD) applied to the smooth approximation f^β; (2) stochastic sub-gradient descent (SubSGD) applied to the original non-smooth function f; (3) stochastic sub-gradient descent with polynomial-decay averaging (SubSGDP) [SZ13] applied to the original non-smooth function f; and (4) SAGA [DBLJ14] applied to the smooth approximation f^β.

All methods have two tunable hyperparameters in common: (1) γ_0, the initial learning rate, i.e., the step size of gradient descent or its variations; and (2) c_γ, a constant indicative of a schedule on γ through the assignment of $\gamma_t = \frac{\gamma_0}{1+tc_\gamma}$ at iteration t. SGD and SAGA are applied to the smooth approximation f^β and, as such, the inverse temperature β is a tunable hyperparmeter in these methods. In contrast, SubSGD and SubSGDP are applied to the original non-smooth objective function. SubSGDP also has an additional hyperparmeter η, which is used to define the polynomial-decay averaging scheme. For each algorithm, we tune the hyperparameters via a grid search with respect to a quantity we call *hyperparameter utility* that is explained below. We use the

Algorithm	β	γ_0	c_γ	η
SGD	10^{-4}	10^{-2}	10^1	N/A
SubSGD	N/A	10^{-2}	10^1	N/A
SubSGDP	N/A	10^{-3}	0	5
SAGA	10^{-4}	10^{-3}	0	N/A
β-10-SAGA	$10^{-7} - 10^{-6}$	10^{-3}	0	N/A

TABLE I: The tuned hyperparameter values.

following values to form a grid in each case:

$$\begin{aligned} \beta &\in \{10^{-7}, 10^{-5}, \ldots, 10^0\}; \\ \gamma_0 &\in \{10^{-7}, 10^{-5}, \ldots, 10^0\}; \\ c_\gamma &\in \{0\} \cup \{10^{-4}, 10^{-3}, \ldots, 10^2\}; \text{ and} \\ \eta &\in \{1, 2, \ldots, 7\}. \end{aligned}$$

We run each algorithm 20 times with different seeds for random number generation, which randomizes the choice of functions f_i for each run, wherein we perform 1000 iterations, and track the progress on the original non-smooth objective function f.

For each algorithm and each hyperparameter setting, we calculate the average objective value over all 20 runs and all 1000 iterations. For each algorithm we calculate the following quantities: (1) total descent—the difference between the initial objective value and the best value found over all 20 trials; (2) absolute ascent—the sum of the values of all ascents between any two consecutive iterations over all iterations of all 20 trials; and (3) hyperparameter utility—the absolute ascent divided by the total descent.

For each algorithm, we choose the hyperparameter setting that minimizes the average objective value over 20 runs and 1000 iterations subject to the constraint that its hyperparameter utility is less than 0.01. We use this constraint to avoid unstable hyperparameter settings. For instance, a very large step size might reduce the objective value very quickly in the beginning but fail to converge to a good solution.

The value of the hyperparameters found by the grid search for each algorithm is reported in Table I. Other than the four methods discussed above, a final row called β-10-SAGA has been included, a description of which will follows.

We can see that for SAGA, we have $c_\gamma = 0$, resulting in a constant step size consistent with the theoretical proof of convergence of SAGA. For SGD and SubSGD, we obtain $c_\gamma = 10$, which is also consistent with the theoretical step sizes of $1/\mu t$ and $\eta/\mu(t+\eta)$, respectively [SZ13]. Note that by the contribution of the regularizer $r(w) = \frac{\lambda \|w\|^2}{2}$, we have $\mu \geq \lambda = 1$. For SubSGDP, we see that the polynomial-decay averaging manages to work with a constant step size, whereas to prove its theoretical convergences, a step size of $\eta/\mu(t+\eta)$ is used.

We see that SGD and SubSGD perform poorly (at least for stable choices of hyperparameters, e.g., their having small step sizes). SubSGDP results in a great improvement, yet SAGA further outperforms it. This is despite the fact that SAGA optimizes the original non-smooth function in $O(\frac{1}{\epsilon}\log\frac{1}{\epsilon})$ once applied to the smooth approximation f^β, whereas SubSGDP converges theoretically in the provably optimal rate of $O(\frac{1}{\epsilon})$.

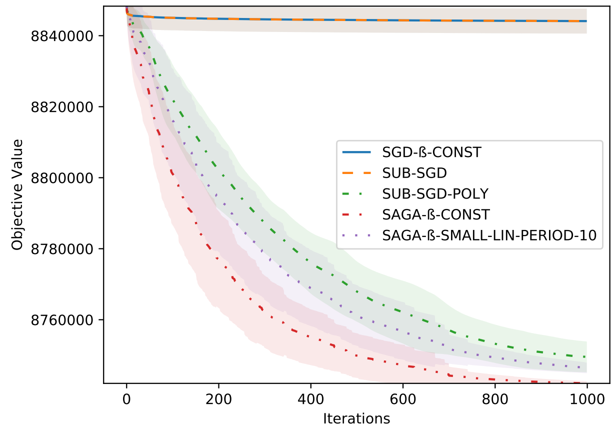

FIG. 1: The average objective value of five algorithms, SGD, SubSGD, SubSGDP, SAGA, and β-10-SAGA, over 20 runs and 1000 iterations. For each iteration the average over 20 runs is shown using dark lines, alongside the standard deviation, shown using shaded regions. We see that SubSGDP is highly effective in improving SubSGD. We also see that SAGA is highly effective in improving SGD, and it even outperforms SubSGDP, despite SubSGDP being provably optimal. β-10-SAGA performs slightly worse, but is more suitable for Gibbs samplers.

We observe that the objective function value is around 8×10^6. After checking the values of w, we see that $r(w) \approx 10^5$. Hence $\frac{1}{n}\sum \max f_i \approx \max f_i \approx 8 \times 10^6$. Therefore, at $\beta = 10^{-4}$, we have $\beta \max f_i \approx 800$. In this regime, the Boltzmann distribution from which we need to sample is very close to the delta function concentrated on the ground states.

In an alternative SAGA experiment, called β-10-SAGA, we have β start from 10^{-7} and in every 10 iterations increase it by 10^{-8}, resulting in a final value of 1.1×10^{-6}. Its performance is slightly worse than that of SAGA, although it is still better than that of SubSGDP. However, $\beta \max f_i$ starts from around 0.8 and approaches 8 in the end, which is more suitable for a Gibbs sampler.

VI. OBJECTIVE FUNCTIONS FOR STRUCTURED PREDICTION

A. S3VM

In this section, we use ideas from Section III A to solve a smooth approximation of SSVMs. We first observe that the constrained optimization problem SSVM as presented in (3) can be rewritten [YJ09] as the minimization of the objective function

$$f_{\mathrm{SSVM}}(w) = \frac{1}{2}\lambda\|w\|^2 + \sum_{(x,y)\in\mathcal{S}} \max_{y'} \left\{\Delta(y', y) + w^T \left[\Phi(x, y') - \Phi(x, y)\right]\right\}, \tag{45}$$

where λ is the regularization parameter for which we have $\lambda = \frac{1}{C}$ with C being the parameter defined in (3). This objective function is a convex upper bound on the risk minimization problem

$$\min_{w} \sum_{(x,y)\in\mathcal{S}} \Delta(\arg\max_{y'} \left(w^T \Phi(x, y')\right), y), \tag{46}$$

as we saw in Section VI. Note that Eq. (45) can easily be rewritten in the form of the function defined in (12). The smoothing of (45) results in the function

$$f_{\mathrm{S3VM}}(w;\beta) = \frac{1}{2}\lambda\|w\|^2 + \sum_{(x,y)\in\mathcal{S}} \max_{y'}{}^{\beta} \left\{\Delta(y',y) + w^T\left[\Phi(x,y') - \Phi(x,y)\right]\right\}, \tag{47}$$

which is a smooth and strongly convex upper bound on the objective function of (46). We use S3VM as an abbreviation for *smooth structured support vector machine.* As a matter of fact, we rediscover the so-called *softmax margin* objective function [GS10b] for structured prediction,

$$f_{\mathrm{SMM}}(w) = \frac{1}{n}\sum_{x,y} \max_{y'}{}^{\beta} \left[\Delta(y',y) + s(x,y',w) - s(x,y,w)\right], \tag{48}$$

which is an upper bound on $f_{\mathrm{MM}}(w)$ from (9). For the gradient of (47), from (18) we have

$$\nabla_w f_{\mathrm{S3VM}}(w;\beta) = \lambda w + \sum_{(x,y)\in\mathcal{S}} \mathbb{E}_Y(\Phi(x,Y)) - \Phi(x,y), \tag{49}$$

where Y is a random variable with the probability distribution

$$p_{B+\Delta}(y'|x;w,\beta) \propto \exp\left(\beta\left[\Delta(y',y) + w^T\Phi(x,y')\right]\right), \quad (x,y)\in\mathcal{S}. \tag{50}$$

One method of calculating Eq. (49) is to use a Monte Carlo estimation by generating samples from the distribution (50).

Smoothing of the maximum-margin problem is not a new idea. This approach was studied for speech recognition tasks [SS07]. In [HU10], the authors also considered the same smoothing approach to SSVMs and gave an approximate inference method based on message passing algorithms. In [GS10a] and [VLZ11], the authors compared S3VM with several other structured prediction objective functions and found S3VM and *loss-inspired conditional log-likelihood* outperformed the rest. Loss-inspired conditional log-likelihood [VLZ11] was introduced as an inexpensive modification to the conditional log-likelihood objective function and later reinvented in [NBJ$^+$16] as *reward augmented maximum likelihood*, but with additional theoretical analysis connecting it to entropy-regularized reinforcement learning.

B. Conditional Log-Likelihood

One approach to obtaining an objective function for structured prediction is to use the conditional log-likelihood

$$\mathcal{L}(w) = -\sum_{(x,y)\in\mathcal{S}} \log p(y|x;w), \tag{51}$$

where $p(y|x;w)$ is a conditional probability density function parameterized by a tunable parameter vector w. One way to define the probability distribution function p is to use the scoring function $s(x,y,w)$ according to the equation

$$p_B(y|x;w,\beta) = \frac{\exp(\beta s(x,y,w))}{Z_B(x;w,\beta)}, \quad (x,y)\in\mathcal{X}\times\mathcal{Y}, \tag{52}$$

where β is a non-tunable parameter separate from w, and $Z_B(x; w, \beta)$ is the normalizing constant.

The *conditional log-likelihood* (CL) objective function is

$$f_{\mathrm{CL}}(w) = -\sum_{(x,y)\in\mathcal{S}} \log p_B(y|x; w, \beta) = -\beta \sum_{(x,y)\in\mathcal{S}} \left[\max_{y'\in\mathcal{Y}}{}^{\beta} s(x, y', w) - s(x, y, w) \right] . \tag{53}$$

To compute the gradient of this objective function, we can use (18) for the gradient of the softmax operator $\max^{\beta}$.

One weakness of this objective function is that it does not take the task-specific loss function Δ into account. If the model satisfies some regularity conditions and the size of the dataset is large, this would not be a problem because of the asymptotic consistency and efficiency of the maximum-likelihood method [NM94]. However, in practice, these conditions are usually not satisfied, so it might be possible to find a better objective function to obtain a solution. S3VM is an example of such a function. We consider other alternatives in the next sections.

There is an interesting connection between the objective function (53) and the principle of maximum entropy. In [BPP96], the authors prove the following. Consider all the conditional probability distributions over the output y given an input x. Among all such distributions, the one that satisfies a specific set of constraints to match the empirical distribution of the data, while simultaneously maximizing the entropy, has a probability function of the form p_B. Furthermore, it is the same distribution that maximizes the conditional log-likelihood f_{CL} of (53).

When the scoring function s corresponds to the negative energy function of an undirected graphical model, the model trained with the conditional log-likelihood objective function (53) is called a *conditional random field* (CRF) [LMP01], an important model used in structured prediction. It has found applications in various areas, including computer vision [HZCP04, KH04, Li09], natural language processing [SP03, ML03], and bioinformatics [BCHP07, DVP$^+$07].

C. Loss-Targeted Conditional Log-Likelihood

Instead of using conditional log-likelihood, we may consider a source and a target probability density function p and q and minimize a notion of distance between them [VLZ11, NBJ$^+$16]. The conditional log-likelihood objective function (51) can actually be driven with this approach.

Example. Let us use Kullback–Leibler (KL) divergence as our notion of distance. For p, we use p_B as defined in (52). For the target distribution, we may simply use the Kronecker delta between the predicted label y' and the true label y:

$$q(y'|x) = \delta_{y',y}, \quad (x, y) \in \mathcal{S} . \tag{54}$$

The structured prediction objective function is then

$$\sum_{(x,y)\in\mathcal{S}} \sum_{y'\in\mathcal{Y}} \delta_{y,y'} \log \frac{\delta_{y,y'}}{p_B(y'|x; w)} = -\sum_{(x,y)\in\mathcal{S}} \log p_B(y|x; w) = \mathcal{L}(w) , \tag{55}$$

which was studied in the previous section.

As discussed previously the conditional log-likelihood objective function f_{CL} does not take the task-specific loss Δ into account. One way to resolve this is to use a target distribution q that depends on Δ. In [VLZ11, NBJ+16], the authors propose using the target distribution

$$q_\Delta(y'|x) \propto \exp(-\mu\Delta(y', y)), \quad (x, y) \in \mathcal{S}\,, \tag{56}$$

where $\mu \in \mathbb{R}$ is a parameter adjusting the spread of the distribution. The KL distance between p_B and q_Δ,

$$\sum_{y' \in \mathcal{Y}} \left[q_\Delta(y'|x) \log q_\Delta(y'|x) - q_\Delta(y'|x) \log p_B(y'|x; w)\right]\,,$$

has its first term $q_\Delta(y'|x) \log q_\Delta(y'|x)$ independent of w, so we can ignore it and obtain the *loss-targeted conditional log-likelihood* (LCL) objective function

$$\begin{aligned} f_{\mathrm{LCL}}(w) &= \sum_{(x,y)\in\mathcal{S}} \sum_{y'\in\mathcal{Y}} \left[-q_\Delta(y'|x) \log p_B(y'|x; w, \beta)\right] \\ &= \sum_{(x,y)\in\mathcal{S}} \sum_{y'\in\mathcal{Y}} q_\Delta(y'|x) \left[\log Z_B(x; w, \beta) - \beta s(x, y', w)\right] \\ &= \sum_{(x,y)\in\mathcal{S}} \left\{ \log Z_B(x; w, \beta) - \sum_{y'\in\mathcal{Y}} \beta q_\Delta(y'|x) s(x, y', w) \right\} \\ &= \beta \sum_{(x,y)\in\mathcal{S}} \left\{ \max_{y'}{}^{\beta} s(x, y', w) - \mathbb{E}_{Y_\Delta}\left(s(x, Y_\Delta, w)\right) \right\}, \end{aligned} \tag{57}$$

where Y_Δ is a random variable with the probability function q_Δ.

To find the gradient of f_{LCL}, for the $\max_{y'}{}^{\beta} s(x, y', w)$ terms we can use the gradient formula of the softmax operator from (18), and for the $\mathbb{E}_{Y_\Delta}\left(s(x, Y_\Delta, w)\right)$, since the distribution q_Δ does not depend on w, we have

$$\nabla_w \mathbb{E}_{Y_\Delta}\left(s(x, Y_\Delta, w)\right) = \mathbb{E}_{Y_\Delta}\left(\nabla_w s(x, Y_\Delta, w)\right). \tag{58}$$

Based on the particular formulae selected for the scoring function s and the loss function Δ (see Section II B), we might be able to use combinatorial formulae to compute (58) exactly. Another approach could be Monte Carlo estimation by sampling from the distribution q_Δ. This could be an easy task, depending on the choice of Δ. For example, when the labels are binary vectors and Δ is the Hamming distance, we can group together all the values of the labels that have the same Hamming distance from the true label. We can then find a combinatorial formula for the number of values in each group, and determine the probability of each group exactly. In order to generate samples, we choose one group randomly according to its probability and then choose one of the values in the group uniformly at random.

D. The Jensen Risk Bound

The last approach we discuss for incorporating the task-specific loss Δ is using the Earth mover's distance (EMD). An exact definition, and a linear programming formulation to compute the EMD can be found in [RTG98]. In this approach, the EMD is used (instead of KL distance) to measure the distance between two source and target distributions p and q, and to try to minimize this distance.

For p, we choose the probability density function p_B as in (52) and let q be defined as in (54). With these choices, the objective function is

$$f_{\text{Risk}}(w) = \frac{1}{n\beta} \sum_{(x,y)\in\mathcal{S}} \sum_{y'\in\mathcal{Y}} \Delta(y', y) p_B(y'|x; w, \beta) = \frac{1}{n\beta} \sum_{(x,y)\in\mathcal{S}} \mathbb{E}_{Y_B} \left(\Delta(Y_B, y)\right), \tag{59}$$

where Y_B is a random variable with the probability density function p_B. This objective function is called *risk* because of its close relationship with the empirical risk as defined in (7).

The objective function f_{Risk} incorporates the task specific loss Δ; however, it is non-convex [GS10b] and the computation of its gradient,

$$\begin{aligned} \nabla_w f_{\text{Risk}}(w) =& \frac{1}{n} \sum_{(x,y)\in\mathcal{S}} \mathbb{E}_{Y_B} \left(\Delta(Y_B, y) \nabla_w s(Y_B, x, w)\right) \\ &- \frac{1}{n} \sum_{(x,y)\in\mathcal{S}} \mathbb{E}_{Y_B}(\Delta(Y_B, y)) \mathbb{E}_{Y_B}(\nabla_w s(Y_B, x, w)), \end{aligned}$$

is difficult, because of the term $\mathbb{E}_{Y_B} \left(\Delta(Y, y) \nabla_w s(Y_B, x, w)\right)$ [GS10b]. For example, a Monte Carlo estimation of $\mathbb{E}_{Y_B} \left(\Delta(Y_B, y) \nabla_w s(Y_B, x, w)\right)$ would have much greater variance for the same number of samples, compared to the estimation of $\mathbb{E}_{Y_B} \left(\nabla_w s(Y_B, x, w)\right)$, which is what we need in most objective functions, for example, f_{CL}, f_{LCL}, and f_{S3VM}.

A solution to this issue is provided in [GS10a], where the authors have introduced the new objective function f_{JRB}, called the Jensen risk bound, which is an upper bound on f_{Risk}, and has gradients that are easier to calculate:

$$f_{\text{JRB}}(w) = \frac{1}{n\beta} \sum_{(x,y)\in\mathcal{S}} \log \mathbb{E}_{Y_B} \left(\beta \exp(\Delta(Y_B, y))\right) \tag{60}$$

To see why f_{JRB} is an upper bound on f_{Risk}, note that

$$\begin{aligned} \mathbb{E}_{Y_B}(\Delta(Y_B, y)) &= \frac{1}{\beta} \log \exp\left(\beta(\mathbb{E}_{Y_B}(\Delta(Y_B, y))\right) \\ &\leq \frac{1}{\beta} \log\left(\mathbb{E}_{Y_B}(\exp(\beta\Delta(Y_B, y)))\right), \end{aligned}$$

by convexity of the exponential function and Jensen's inequality.

For the gradient formula for f_{JRB}, we have

$$\nabla_w f_{\text{JRB}}(w) = \frac{1}{n\beta} \sum_{(x,y)\in\mathcal{S}} \mathbb{E}_{Y_{B+\Delta}}(\nabla_w s(x, Y_{B+\Delta}, w)) - \mathbb{E}_{Y_B}(\nabla_w s(x, Y_B, w)),$$

where $Y_{B+\Delta}$ is a random variable with the probability density function

$$p_{B+\Delta}(y'|x; w, \beta) \propto \exp(\beta[\Delta(y', y) + s(x, y', w)]), \quad y' \in \mathcal{Y},\ (x, y) \in \mathcal{S}, \tag{61}$$

and Y_B is a random variable with the probability function p_B as defined in (52).

Although f_{JRB} has easier gradients to calculate, it is still a non-convex function. The EMD used here gives rise to a new interpretation of the well-known objective function of risk, which was used successfully for a long period of time by the speech recognition and natural language processing communities [KHK00, PW02, GS10a]. In [GS10a], the authors have introduced the Jensen risk bound objective of (60), as an easier-to-optimize upper bound on the risk objective function.

VII. IMAGE TAGGING AS A STRUCTURED-PREDICTION TASK

Recall the notation used in Section II A. In our image tagging task, let $\mathcal{X}$ be the set of all possible images, and $\mathcal{Y}$ is the set of all possible labels. The labels are ℓ-dimensional binary vectors. In other words, $\mathcal{Y} \subseteq \{-1, 1\}^\ell$. Each dimension of y denotes the presence or absence of a tag in the image (e.g., "cat", "dog", "nature").

We would like to find the feature function $\Phi(x, y, w_0)$ with parameter w_0. Let $\Phi_0 : \mathcal{X} \times \mathcal{W}_0 \to \mathbb{R}^\eta$ be a feature function, where the first argument from $\mathcal{X}$ is an image, the second argument from $\mathcal{W}_0$ is a parameter, and the output is a real vector with $\eta \in \mathbb{N}$ dimensions. The function $\Phi_0(x, w_0)$ serves as a base feature function in the construction of $\Phi(x, y, w_0)$. The function $\Phi_0(x, w_0)$ can be any function. In our experiments, we use a convolutional neural network (CNN) as a feature extractor for this purpose, with w_0 denoting its weights.

One way to define Φ based on Φ_0 is as follows: we design Φ_0 (i.e., the CNN) such that the dimension of its output is identical to the size of the labels: $\eta = \ell$. Let "triu" denote the vectorized upper triangle of its square matrix argument. We then define

$$\Phi(x, y, w_0) = \begin{pmatrix} \mathrm{triu}(yy^T) \\ \Phi_0(x, w_0) \circ y \\ y \end{pmatrix}, \tag{62}$$

where $\circ$ is the element-wise product. Note that $\Phi_0(x, w_0) \circ y$ is well-defined because $\eta = \ell$ and the two vectors $\Phi_0(x, w_0)$ and y have identical dimensions.

The result is $\Phi(x, y, w_0) \in \mathbb{R}^d$ for some $d \in \mathbb{N}$. Let $w \in \mathbb{R}^d$ be the parameter vector of our structured-prediction model. We then define the scoring function s as

$$\begin{aligned} s(x, y, w) = w^T \Phi(x, y, w_0) &= \begin{pmatrix} \theta_1^T & \theta_2^T & \theta_3^T \end{pmatrix} \Phi(x, y, w_0) \\ &= \theta_1^T \mathrm{triu}(yy^T) + \theta_2^T \left[\Phi_0(x, w_0) \circ y\right] + \theta_3^T y \,. \end{aligned} \tag{63}$$

One can then interpret θ_1 as control parameters on the relationship between pairs of labels y_i and y_j. The parameter vector θ_2 controls the effect of the features extracted from the CNN. The parameter vector θ_3 controls the bias of the values of y_i, as some tags are less likely to be present and some are more likely. Note that the formula $s(x, y, w)$ in (63) is quadratic in y.

We choose the function Δ to be the Hamming distance

$$\Delta(y', y) = \mathrm{Hamming}(y', y) \tag{64}$$

for two reasons. Firstly, the error in the predictions made in image tagging is also calculated using the Hamming distance between the true label and the predicted label. Secondly, the Hamming distance is a linear function of y', and therefore $\Delta(y', y) + s(x, y', w)$ remains quadratic in y'. This reduces the inference step of the optimization of f_{S3VM} and f_{JRB} (i.e., sampling from the distribution $p_{B+\Delta}$ of (50) and (61)) to sampling from an Ising model.

Model	Validation Error	Test Error	γ	λ	β	β_{eff}
baseline	2.6844	2.7052	N/A	N/A	N/A	N/A
baseline + S3VM	**2.6568**	**2.69**	10^{-7}	0.0	3^1	$[60.372, 133.0482]$
baseline + CL	2.6696	2.6996	10^{-6}	10^{-6}	3^1	$[53.0406, 118.1979]$
baseline + JRB	2.658	2.6956	10^{-7}	10^{-6}	3^1	$[55.4559, 122.7675]$
baseline + FC	2.7236	2.7656	10^{-2}	0.0	N/A	N/A

TABLE II: Image tagging results. The baseline architecture is that of AlexNet. The three subsequent lines report the performance of extensions of the baseline with an Ising model trained using different objective functions. The last row is an extension of the baseline with a single feedforward fully connected layer with sigmoid activations and the binary cross entropy objective.

A. Numerical Results

We use the MIRFLICKR dataset [HL08], which consists of 25,000 images and 38 tags. This dataset consists of an extended tag set with more than 1000 words. Since the sampling step for the Monte Carlo estimation of the gradient of the objective functions is very slow on CPUs, we restrict the tags to the smaller set of 38. We randomly selected 20,000 images for the training set, 2500 images for the validation set, and the remaining 2500 images for the test set.

We train a pre-trained AlexNet [KSH12], a convolutional neural network, on the training data, to predict the tags. We train AlexNet using the binary cross entropy objective function between its output layer and the true labels. We call this model a *baseline* in what follows. We fix the baseline and feed its output to an Ising model which acts as a denoiser. We then train the weights of the Ising model with three different objective functions, namely f_{CL}, f_{S3VM}, and f_{JRB}. This is inspired by [CSYU15], wherein the output of an AlexNet network is fed to a CRF in a very similar fashion. The architecture of the model is shown in Figure 2.

In the training mode, we use the standard stochastic gradient descent algorithm, with a parameter λ adjusting the L_2 regularizer of $\lambda\|w\|^2/2$ that is added to the objective functions, and a parameter γ as the learning rate, which is kept constant during the training. We consider four training epochs, where, in each epoch, we go through each data point of the training data exactly once, in a random order. In this experiment, we use single-spin flip Gibbs sampling at a constant inverse temperature β as our sampling subroutine to compute a Monte Carlo estimation of the objective function's gradient. Due to our choice of using only a subset of tags to train and test over, our Ising model

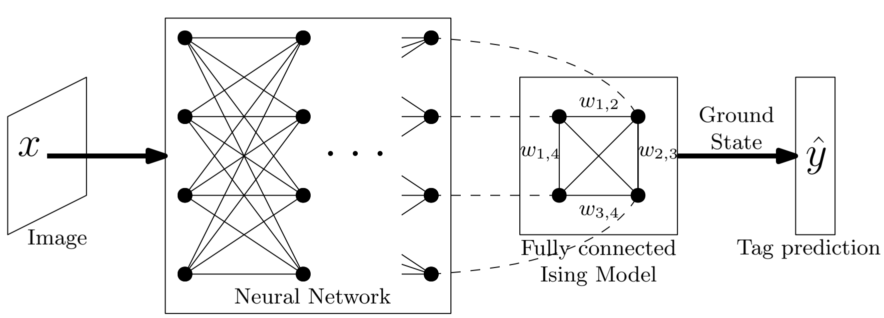

FIG. 2: Image tagging architecture. The image x is fed to a neural network to extract features. The features then are passed to an Ising model the ground state of which determines the prediction.

(a) Test image 7520 (b) Test image 10177 (c) Test image 21851

FIG. 3: Sample tags generated by the different models. In Figure 3a, Figure 3b, and Figure 3c we see that S3VM has respectively decreased, increased, and did not change the error, compared to the baseline.

image number	7520	10177	21851
true labels	plant_life, sky, structures, tree	night, sky, structures, transport	night, sky, structures
baseline	people, plant_life, sky, structures, tree	night, plant_life, sky, structures, sunset, transport, tree	indoor, male, people, structures
baseline+S3VM	plant_life, sky, structures, tree	night, plant_life, sky, structures, sunset, tree	indoor, male, people, structures
baseline+CL	people, plant_life, sky, structures, tree	night, plant_life, sky, structures, sunset, transport, tree	male, people, structures
baseline+JRB	plant_life, sky, structures, tree	night, plant_life, sky, structures, sunset, tree	indoor, male, people, structures
baseline+FC	male, people, plant_life, sky, structures, tree	night, plant_life, sky, structures, sunset, transport, tree	male, people, sky, structures

instances consist of 38 variables and a fully connected architecture. For each instance, we perform 200 sweeps and collect 200 samples.

So, in total, we have three hyperparameters, namely γ, λ, and β. We tune the hyperparameters by performing a grid search over the values

$$\gamma = \{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}\}, \quad \lambda = \{0.0, 10^{-6}, 10^{-4}, 10^{-2}\}, \quad \text{and} \quad \beta = \{3^{-1}, 3^{0}, 3^{1}, 3^{2}\}\,.$$

A last architecture considered is that of an extension of the baseline with a fully connected feedforward layer with sigmoid activations. This model has been added in order to compare the extensions of the baseline with undirected architectures (e.g., the Ising model) versus a feedforward layer using a similar number of parameters. The Ising model has a fully connected graph with $\binom{38}{2} + 38 = 741$ parameters and we use a fully connected feedforward layer with 38 nodes, which amounts to $38^2 + 38 = 1482$ parameters. We use the Adam algorithm for optimization [KB14] implemented in the PyTorch library [PGC$^+$17] with 300 epochs. We tune the learning rate parameter γ using a gridsearch over the values

$$\gamma = \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\},$$

while all other hyperparameters of the Adam optimization algorithm are left at their default values ($\beta_1 = 0.9, \beta_2 = 0.999$).

In Table II, we summarize the performance of the various methods and values of tuned hyperparameters. The reported error is the average Hamming distance between the predicted labels and the true labels in terms of the number of bits. We observe that all three extensions of the baseline with an Ising model improve the baseline, with the S3VM objective function resulting in the greatest improvement.

We observe that the values of λ in all cases are either 0 or very small. However, this might be an artifact of having small numbers of parameters in our model ($\binom{38}{2} + 38 = 741$), making the model immune to over-fitting.

In the final column of Table II, we report the range of the effective thermodynamic β denoted by β_{eff} for each method. The effective β is the product of the nominal value β and the absolute value of the ground state energy of the Ising model over different images. The interval reported in this table is the range of β_{eff} over the images in the test set.

In Figure 3, we see three examples from the test set. Finally, we wish to remark that we would have needed to solve much larger problems and perform many more sweeps of Monte Carlo simulations had we used the complete set of tags. The fully connected architecture is not imposed by the problem we are solving. The use of much sparser connectivity graphs could result in viable feature extractors as well. These are future areas of development that can be explored using high-performance computing platforms.

VIII. CONCLUSION

In this paper, we introduced a quantum algorithm for solving the min-max optimization problem that appears in machine learning applications using a variant of SAGA (which we call *A-SAGA*) that takes into account an additive error on the calculation of gradients. This has allowed us to use a quantum Gibbs sampler as a subroutine of A-SAGA to provide estimations of the gradients and optimize the smooth approximation of the min-max problem. We called the conjunction of A-SAGA with the quantum Gibbs sampler *Q-SAGA*.

We have shown that A-SAGA can give an approximation of the solution of the smooth approximation of the original min-max problem in $O(\log \frac{1}{\epsilon})$ gradient evaluations, provided the additive error is in $O(\epsilon)$. This scaling is, in fact, optimal [DBLJ14, SLRB17]. We then used A-SAGA to solve the original min-max problem in $O(\frac{1}{\epsilon} \log \frac{1}{\epsilon})$ gradient evaluations. We remark that the best algorithms [SZ13, Nes05] for solving the original min-max problem use $O(\frac{1}{\epsilon})$ gradient evaluations. This is the case if the gradients are calculated exactly. We conclude that in the presence of additive errors in estimating the gradients, our results are close to optimal.

The quantum algorithm Q-SAGA solves the smooth approximation of the original min-max problem in $O(\frac{1}{\epsilon} \log^2 \frac{1}{\epsilon})$ queries to the associated quantum oracles and the same order of other quantum gates. Q-SAGA solves the original min-max problem in $O(\frac{1}{\epsilon^{3.5}} \log^2 \frac{1}{\epsilon})$. Despite a worse scaling in terms of ϵ, this quantum algorithm provides a speedup in terms of other parameters indicative of the size of the problem. For example, where the problem is a model for structured prediction using an SSVM, the scaling is $\widetilde{O}(D^{2.5}\sqrt{|\mathcal{Y}|}\epsilon^{-3.5})$, where $\mathcal{Y}$ is the set of all possible predictions and D is the number of tunable parameters.

We have also provided results from several numerical experiments. In particular, we compared the performance of SGD in two cases: with all sampling subroutines performed at a constant temperature, and with the temperature decreasing across iterations according to a schedule. We observed that the scheduled temperature slightly improves the performance of SGD. We believe that studying the temperature schedule would be an interesting avenue of research. In particular, it would be beneficial to gain an understanding of the best practices in scheduling temperature during SGD. It would also be interesting to provide a theoretical analysis of the effect of the temperature schedule in SGD.

As we have seen in our experiments, using a temperature schedule seems not to be consistent with SAGA since the cache of old gradients then comes from other temperatures. Another avenue of future research would be to adapt or modify SAGA so as to overcome this caveat.

Our successful image tagging experiments used only 38 English words as candidate tags. The MIRFLICKR dataset provides a thousand English words as candidate tags, but conducting an experiment of this size was not feasible with the computational resources available to us. Our goal is to pursue efficient Gibbs sampling approaches in quantum and high-performance computation in order to achieve similar results in larger image tagging tasks. In fact, our work proposes a general approach for quantum machine learning using a quantum Gibbs sampler. In this approach, the network architecture consists of a leading directed neural network serving as a *feature extractor*, and a trailing undirected neural network responsible for smooth prediction based on the feature vectors.

IX. ACKNOWLEDGEMENT

The authors thank Mark Schmidt, who motivated our initiation of this project and provided technical feedback throughout. We further thank Ronald de Wolf, Matthias Troyer, Austin Roberts, and Reza Babanezhad for useful technical discussions, and Marko Bucyk for helpful comments and for reviewing and editing the manuscript. This project was fully funded by 1QBit. P. R. further acknowledges the support of the government of Ontario and Innovation, Science and Economic Development Canada.

[ABLZ12] Tameem Albash, Sergio Boixo, Daniel A Lidar, and Paolo Zanardi. Quantum adiabatic markovian master equations. *New Journal of Physics*, 14(12):123016, 2012.

[AFGG12] JE Avron, M Fraas, GM Graf, and P Grech. Adiabatic theorems for generators of contracting evolutions. *Communications in mathematical physics*, 314(1):163–191, 2012.

[AGGW17] Joran van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum sdp-solvers: Better upper and lower bounds. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 403–414. IEEE, 2017.

[ATH03] Yasemin Altun, Ioannis Tsochantaridis, and Thomas Hofmann. Hidden markov support vector machines. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 3–10, 2003.

[BCHP07] Axel Bernal, Koby Crammer, Artemis Hatzigeorgiou, and Fernando Pereira. Global discriminative learning for higher-accuracy computational gene prediction. *PLoS computational biology*, 3(3):e54, 2007.

[BCTM05] Peter L Bartlett, Michael Collins, Ben Taskar, and David A McAllester. Exponentiated gradient algorithms for large-margin structured classification. In *Advances in neural information processing systems*, pages 113–120, 2005.

[BDRF16] S Bachmann, W De Roeck, and Martin Fraas. The adiabatic theorem for many-body quantum systems. *Preprint*, 2016.

[BHMT02] Gilles Brassard, Peter Hoyer, Michele Mosca, and Alain Tapp. Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74, 2002.

[BKL+17] F. G. S. L. Brandão, A. Kalev, T. Li, C. Yen-Yu Lin, K. M. Svore, and X. Wu. Quantum sdp solvers: Large speed-ups, optimality, and applications to quantum learning. *arXiv preprint arXiv:1710.02581*, 2017.

[BPP96] Adam L Berger, Vincent J Della Pietra, and Stephen A Della Pietra. A maximum entropy approach to natural language processing. *Computational linguistics*, 22(1):39–71, 1996.

[BS17] Fernando GSL Brandao and Krysta M Svore. Quantum speed-ups for solving semidefinite programs. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 415–426. IEEE, 2017.

[BT12] Amir Beck and Marc Teboulle. Smoothing and first order methods: A unified framework. *SIAM Journal on Optimization*, 22(2):557–580, 2012.

[CGK+08] Michael Collins, Amir Globerson, Terry Koo, Xavier Carreras, and Peter L Bartlett. Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *Journal of Machine Learning Research*, 9(Aug):1775–1822, 2008.

[CLG+16] Daniel Crawford, Anna Levit, Navid Ghadermarzy, Jaspreet S Oberoi, and Pooya Ronagh. Reinforcement learning using quantum boltzmann machines. *arXiv preprint arXiv:1612.05695*, 2016.

[Col02] Michael Collins. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 1–8. Association for Computational Linguistics, 2002.

[CS16] Anirban Narayan Chowdhury and Rolando D Somma. Quantum algorithms for gibbs sampling and hitting-time estimation. *arXiv preprint arXiv:1603.02940*, 2016.

[CSYU15] Liang-Chieh Chen, Alexander Schwing, Alan Yuille, and Raquel Urtasun. Learning deep structured models. In *International Conference on Machine Learning*, pages 1785–1794, 2015.

[DBLJ14] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in neural information processing systems*, pages 1646–1654, 2014.

[DVP+07] David DeCaprio, Jade P Vinson, Matthew D Pearson, Philip Montgomery, Matthew Doherty, and James E Galagan. Conrad: gene prediction using conditional random fields. *Genome research*, 17(9):000–000, 2007.

[FISS03] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

[GP17] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[GS10a] Kevin Gimpel and Noah A Smith. Softmax-margin crfs: Training log-linear models with cost functions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 733–736. Association for Computational Linguistics, 2010.

[GS10b] Kevin Gimpel and Noah A Smith. Softmax-margin training for structured log-linear models. 2010.

[HL08] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.

[HU10] Tamir Hazan and Raquel Urtasun. A primal-dual message-passing algorithm for approximated large scale structured prediction. In *Advances in Neural Information Processing Systems*, pages 838–846, 2010.

[HZCP04] Xuming He, Richard S Zemel, and Miguel Á Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on*, volume 2, pages II–II. IEEE, 2004.

[JFY09] Thorsten Joachims, Thomas Finley, and Chun-Nam John Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[Joa06] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226. ACM, 2006.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[KB16] Michael J Kastoryano and Fernando GSL Brandao. Quantum gibbs samplers: the commuting case. *Communications in Mathematical Physics*, 344(3):915–957, 2016.

[KH04] Sanjiv Kumar and Martial Hebert. Discriminative fields for modeling spatial dependencies in natural images. In *Advances in neural information processing systems*, pages 1531–1538, 2004.

[KHK00] Janez Kaiser, Bogomir Horvat, and Zdravko Kacic. A novel loss function for the overall risk criterion based discriminative training of hmm models. In *Sixth International Conference on Spoken Language Processing*, 2000.

[KR17] Sahar Karimi and Pooya Ronagh. A subgradient approach for constrained binary optimization via quantum adiabatic evolution. *Quantum Information Processing*, 16(8):185, 2017.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LCG+17] Anna Levit, Daniel Crawford, Navid Ghadermarzy, Jaspreet S Oberoi, Ehsan Zahedinejad, and Pooya Ronagh. Free energy-based reinforcement learning using a quantum processor. *arXiv preprint arXiv:1706.00074*, 2017.

[Li09] Stan Z Li. *Markov random field modeling in image analysis*. Springer Science & Business Media, 2009.

[LJJSP12] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. *arXiv preprint arXiv:1207.4747*, 2012.

[LMP01] John Lafferty, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[LSV08] Quoc V Le, Alex J Smola, and Svn Vishwanathan. Bundle methods for machine learning. In *Advances in neural information processing systems*, pages 1377–1384, 2008.

[ML03] Andrew McCallum and Wei Li. Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 188–191. Association for Computational Linguistics, 2003.

[MTT+17] Satoshi Matsubara, Hirotaka Tamura, Motomu Takatsu, Danny Yoo, Behraz Vatankhahghadim, Hironobu Yamasaki, Toshiyuki Miyazawa, Sanroku Tsukamoto, Yasuhiro Watanabe, Kazuya Takemoto, et al. Ising-model optimizer with parallel-trial bit-sieve engine. In *Conference on Complex, Intelligent, and Software Intensive Systems*, pages 432–438. Springer, 2017.

[NBJ+16] Mohammad Norouzi, Samy Bengio, Navdeep Jaitly, Mike Schuster, Yonghui Wu, Dale Schuurmans, et al. Reward augmented maximum likelihood for neural structured prediction. In *Advances In Neural Information Processing Systems*, pages 1723–1731, 2016.

[Nes05] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[Nes13] Yurii Nesterov. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer Science & Business Media, 2013.

[Ng10] A Ng. Support vector machines (part v of cs229 machine learning course materials). 2010.

[NM94] Whitney K Newey and Daniel McFadden. Large sample estimation and hypothesis testing. *Handbook of econometrics*, 4:2111–2245, 1994.

[NS16] Frank Nielsen and Ke Sun. Guaranteed bounds on information-theoretic measures of univariate mixtures using piecewise log-sum-exp inequalities. *Entropy*, 18(12):442, 2016.

[OHY17] Takuya Okuyama, Masato Hayashi, and Masanao Yamaoka. An ising computer based on simulated quantum annealing by path integral monte carlo method. In *Rebooting Computing (ICRC), 2017 IEEE International Conference on*, pages 1–6. IEEE, 2017.

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[PW02] Daniel Povey and Philip C Woodland. Minimum phone error and i-smoothing for improved discriminative training. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 1, pages I–105. IEEE, 2002.

[PW09] David Poulin and Pawel Wocjan. Sampling from the thermal quantum gibbs state and evaluating partition functions with a quantum computer. *Physical review letters*, 103(22):220502, 2009.

[RTG98] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. A metric for distributions with applications to image databases. In *Computer Vision, 1998. Sixth International Conference on*, pages 59–66. IEEE, 1998.

[RWI16] Pooya Ronagh, Brad Woods, and Ehsan Iranmanesh. Solving constrained quadratic binary problems via quantum adiabatic evolution. *Quantum Information & Computation*, 16(11-12):1029–1047, 2016.

[Sch09] Mark Schmidt. A note on structural extensions of svms. 2009.

[SFB+98] Robert E Schapire, Yoav Freund, Peter Bartlett, Wee Sun Lee, et al. Boosting the margin: A new explanation for the effectiveness of voting methods. *The annals of statistics*, 26(5):1651–1686, 1998.

[SL05] MS Sarandy and DA Lidar. Adiabatic approximation in open quantum systems. *Physical Review A*, 71(1):012331, 2005.

[SLRB17] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162(1-2):83–112, 2017.

[SP03] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 134–141. Association for Computational Linguistics, 2003.

[SS07] Fei Sha and Lawrence K Saul. Large margin hidden markov models for automatic speech recognition. In *Advances in neural information processing systems*, pages 1249–1256, 2007.

[SSSSC11] Shai Shalev-Shwartz, Yoram Singer, Nathan Srebro, and Andrew Cotter. Pegasos: Primal estimated sub-gradient solver for svm. *Mathematical programming*, 127(1):3–30, 2011.

[SZ13] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes. In *International Conference on Machine Learning*, pages 71–79, 2013.

[TCK04] Ben Taskar, Vassil Chatalbashev, and Daphne Koller. Learning associative markov networks. In *Proceedings of the twenty-first international conference on Machine learning*, page 102. ACM, 2004.

[TD00] Barbara M Terhal and David P DiVincenzo. Problem of equilibration and the computation of correlation functions on a quantum computer. *Physical Review A*, 61(2):022301, 2000.

[TGK04] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. In *Advances in neural information processing systems*, pages 25–32, 2004.

[THJA04] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[TLJJ06a] Ben Taskar, Simon Lacoste-Julien, and Michael I Jordan. Structured prediction, dual extragradient and bregman projections. *Journal of Machine Learning Research*, 7(Jul):1627–1653, 2006.

[TLJJ06b] Ben Taskar, Simon Lacoste-Julien, and Michael I Jordan. Structured prediction via the extragradient method. In *Advances in neural information processing systems*, pages 1345–1352, 2006.

[TOV+11] Kristan Temme, Tobias J Osborne, Karl G Vollbrecht, David Poulin, and Frank Verstraete. Quantum metropolis sampling. *Nature*, 471(7336):87, 2011.

[TSVL07] Choon Hui Teo, Alex Smola, SVN Vishwanathan, and Quoc Viet Le. A scalable modular convex solver for regularized risk minimization. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 727–736. ACM, 2007.

[TTY+17] Yutaka Takeda, Shuhei Tamate, Yoshihisa Yamamoto, Hiroki Takesue, Takahiro Inagaki, and Shoko Utsunomiya. Boltzmann sampling for an xy model using a non-degenerate optical parametric oscillator network. *Quantum Science and Technology*, 3(1):014004, 2017.

[VALZ16] Lorenzo Campos Venuti, Tameem Albash, Daniel A Lidar, and Paolo Zanardi. Adiabaticity in open quantum systems. *Physical Review A*, 93(3):032118, 2016.

[Vap63] Vladimir Vapnik. Pattern recognition using generalized portrait method. *Automation and remote control*, 24:774–780, 1963.

[VLZ11] Maksims N Volkovs, Hugo Larochelle, and Richard S Zemel. Loss-sensitive training of probabilistic conditional random fields. *arXiv preprint arXiv:1107.1805*, 2011.

[WKS14] Nathan Wiebe, Ashish Kapoor, and Krysta M Svore. Quantum deep learning. *arXiv preprint arXiv:1412.3489*, 2014.

[WW+99] Jason Weston, Chris Watkins, et al. Support vector machines for multi-class pattern recognition. In *Esann*, volume 99, pages 219–224, 1999.

[YJ09] Chun-Nam John Yu and Thorsten Joachims. Learning structural svms with latent variables. In *Proceedings of the 26th annual international conference on machine learning*, pages 1169–1176. ACM, 2009.

[Yu11] Chun Nam Yu. Improved learning of structural support vector machines: training with latent variables and nonlinear kernels. 2011.

[Zha04] Tong Zhang. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, page 116. ACM, 2004.

Appendix A: Convergence of SAGA with Additive Error

Lemma A.1. *Let $\delta = (1+\theta\sqrt{D})$. In order to satisfy all the inequalities*

$$\frac{1}{n} - 2c\gamma\left(\frac{L-\mu}{L} + \gamma\mu\alpha\delta\right) \le 0\,, \tag{A1}$$

$$\frac{1}{\tau} + 2\left(1+\frac{1}{\alpha}\right)\delta c\gamma^2 L - \frac{1}{n} \le 0\,, \tag{A2}$$

$$\left(\frac{1}{\tau} - \gamma\mu\right)\|w^t - w_*\|^2 + 2\gamma^2\theta\sqrt{D} + \gamma^2\theta^2 D + 2\gamma\theta\sqrt{D}\|w^t - w_*\| \le 0\,, \tag{A3}$$

$$(1+\alpha)\gamma\delta - \frac{1}{L} \le 0\,, \tag{A4}$$

it is sufficient to have

$$\gamma = \frac{1}{(1+\alpha)\delta L}\,, \quad c = \frac{2}{n\gamma}\,, \quad \alpha = 8\,, \quad \frac{1}{\tau} = \min\left\{\frac{1}{2n}, \frac{\gamma\mu}{2}\right\}\,, \quad \theta = \min\left\{\frac{1}{\sqrt{D}}, \frac{\mu\|w^t - w_*\|^2}{2\sqrt{D}\left(\frac{2}{9L} + 2\|w^t - w_*\|\right)}\right\}.$$

Proof. In what follows, we enumerate the steps required to satisfy all inequalities in the statement.
Equation (A4). We set

$$\gamma = \frac{1}{(1+\alpha)\delta L}. \tag{†1}$$

Equation (A1). We consider the two cases of $\frac{L}{\mu} > 2$ and $\frac{L}{\mu} \le 2$. When $\frac{L}{\mu} > 2$,

$$\frac{1}{n} - 2c\gamma\left(\frac{L-\mu}{L} + \gamma\mu\alpha\delta\right) \le \frac{1}{n} - 2c\gamma\left(\frac{L-\mu}{L}\right) < \frac{1}{n} - c\gamma\,.$$

It therefore suffices to have

$$c \ge \frac{1}{n\gamma}. \tag{A5}$$

Alternatively, if $\frac{L}{\mu} \le 2$,

$$\frac{1}{n} - 2c\gamma\left(\frac{L-\mu}{L} + \gamma\mu\alpha\delta\right) \le \frac{1}{n} - 2c\gamma\left(\gamma\mu\alpha\delta\right) = \frac{1}{n} - 2c\gamma\left(\frac{1}{(1+\alpha)\delta L}\mu\alpha\delta\right) = \frac{1}{n} - 2c\gamma\left(\frac{\alpha}{1+\alpha}\frac{\mu}{L}\right)$$
$$\le \frac{1}{n} - 2c\gamma\left(\frac{\alpha}{1+\alpha}\frac{1}{2}\right) \le \frac{1}{n} - \frac{c\gamma}{2}\,,$$

where in the last line we used $\frac{L}{\mu} \leq 2$ and in the last inequality we made the assumption that

$$\alpha \geq 1\,, \tag{A6}$$

resulting in $\frac{\alpha}{1+\alpha} \geq \frac{1}{2}$. Consequently, to satisfy Equation (A1) it suffices to have

$$c \geq \frac{2}{n\gamma}\,. \tag{A7}$$

By combining Equations (A5) and (A7), we set

$$c = \frac{2}{n\gamma}\,. \tag{†2}$$

Equation (A2). We require that

$$2\left(1+\frac{1}{\alpha}\right)\delta c\gamma^2 L - \frac{1}{n} < 0\,, \tag{A8}$$

in which the inequality is strict (in order to assure $\frac{1}{\tau}$ is strictly positive). Plugging in the values of c from Equation (†2) and γ from Equation (†1), we have

$$2\left(\frac{1+\alpha}{\alpha}\right)\delta\left(\frac{2}{n\gamma}\right)\gamma^2 L - \frac{1}{n} = \frac{4}{\alpha n} - \frac{1}{n}\,.$$

So, in order to satisfy Equation (A8), it suffices to have $\frac{4}{\alpha n} - \frac{1}{n} < 0$, resulting in $\alpha > 4$. We may therefore set

$$\alpha = 8 \tag{†3}$$

in order to leave room for $\frac{1}{\tau}$ to be larger in the next step. Note that this automatically satisfies Equation (A6). With this setting of α, the left-hand side of Equation (A2) is equal to

$$\frac{1}{\tau} + 2\left(1+\frac{1}{\alpha}\right)\delta c\gamma^2 L - \frac{1}{n} = \frac{1}{\tau} - \frac{1}{2n}\,.$$

To satisfy Equation (A2), it is sufficient to require that

$$\frac{1}{\tau} \leq \frac{1}{2n}\,. \tag{A9}$$

Equation (A3). We need

$$\frac{1}{\tau} - \gamma\mu < 0\,,$$

where the inequality is strict. To satisfy this, we set

$$\frac{1}{\tau} \leq \frac{\gamma\mu}{2}\,. \tag{A10}$$

By combining Equations (A9) and (A10), we set

$$\frac{1}{\tau} = \min\left\{\frac{1}{2n}, \frac{\gamma\mu}{2}\right\}. \tag{†4}$$

By Equation (A10), Equation (A3) reads

$$\frac{-\gamma\mu}{2}\|w^t - w_*\|^2 + \gamma^2\theta\sqrt{D} + \gamma^2\theta^2 D + 2\gamma\theta\sqrt{D}\|w^t - w_*\| \leq 0\,.$$

Cancelling a γ term and using the value of γ from Equation (†1), we would like to satisfy

$$\frac{\theta\sqrt{D}L}{9(1+\theta\sqrt{D})} + \frac{\theta^2 D}{9(1+\theta\sqrt{D})L} + 2\theta\sqrt{D}\|w^t - w_*\| \leq \frac{\mu}{2}\|w^t - w_*\|^2\,. \tag{A11}$$

Let

$$\theta \leq \frac{1}{\sqrt{D}}\,. \tag{A12}$$

So, we have

$$\begin{aligned}
\frac{\theta\sqrt{D}}{9(1+\theta\sqrt{D})L} + \frac{\theta^2 D}{9(1+\theta\sqrt{D})L} + 2\theta\sqrt{D}\|w^t - w_*\| &\leq \frac{\theta\sqrt{D}}{9L} + \frac{\theta^2 D}{9L} + 2\theta\sqrt{D}\|w^t - w_*\| \\
&\leq \frac{\theta\sqrt{D}}{9L} + \frac{\theta\frac{1}{\sqrt{D}}D}{9L} + 2\theta\sqrt{D}\|w^t - w_*\| \\
&= \sqrt{D}\left(\frac{2}{9L} + 2\|w^t - w_*\|\right)\theta\,.
\end{aligned}$$

To satisfy Equation (A11), we may assume

$$\theta \leq \frac{\mu\|w^t - w_*\|^2}{2\sqrt{D}\left(\frac{2}{9L} + 2\|w^t - w_*\|\right)}\,,$$

and Equation (A12). Therefore, we set

$$\theta = \min\left\{\frac{1}{\sqrt{D}}, \frac{\mu\|w^t - w_*\|^2}{2\sqrt{D}\left(\frac{2}{9L} + 2\|w^t - w_*\|\right)}\right\}. \tag{†5}$$

□