Quantum Machine Intelligence (2022) 4:25
https://doi.org/10.1007/s42484-022-00078-w

RESEARCH ARTICLE

Quantum algorithms for structured prediction

Behrooz Sepehry[1] · **Ehsan Iranmanesh**[1] · **Michael P. Friedlander**[1,2] · **Pooya Ronagh**[1,3,4]

Received: 21 May 2020 / Accepted: 10 June 2022

Abstract
We introduce two quantum algorithms for solving structured prediction problems. We first show that a stochastic gradient descent that uses the quantum minimum finding algorithm and takes its probabilistic failure into account solves the structured prediction problem with a runtime that scales with the square root of the size of the label space, and in $\tilde{O}(1/\epsilon)$ with respect to the precision, ϵ, of the solution. Motivated by robust inference techniques in machine learning, we then introduce another quantum algorithm that solves a smooth approximation of the structured prediction problem with a similar quantum speedup in the size of the label space and a similar scaling in the precision parameter. In doing so, we analyze a variant of stochastic gradient descent 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 $O(\sqrt{\epsilon})$. This 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. Our numerical results using Monte Carlo simulations on an image tagging task demonstrate the benefit of the approach.

Keywords Quantum algorithms · Statistical learning · Structured prediction · Support vector machines · Computational complexity · Query complexity · Quantum minimum finding · Quantum Gibbs sampling · Stochastic gradient descent · Stochastic subgradient methods · Stochastic average gradient methods · Smooth approximation

1 Introduction

Structured prediction is an area of machine learning in which the aim is to learn an association of the input data to a structured output (Sebastian et al. 2014). Structured prediction tasks arise naturally in many real-world applications. For example, predicting temporal structures (e.g., the parse tree of a sentence, or the coreferences between nouns and pronouns in texts) is important in natural language processing applications (Daume and Marcu 2006). Spatial structures (e.g., the segmentation of an image into meaningful components, the geometry of molecules) are of interest in several areas of application, such as image processing, computer vision, and computational biology (Jiang 2020).

The structures of interest in structured prediction are often discrete and combinatorial in nature. As a result, while a description of an admissible structure can be represented efficiently, the number of possible valid structures is exponentially larger than these representations. As such, viewing structured prediction tasks as classification problems results in exponentially large numbers of labels. This is why classical machine learning has to employ techniques such as generalization of support vector machines (SVM) to structured SVMs (SSVM) (Yu 2011) and generative models (Sohn et al. 2015) to solve structured prediction problems.

The inherently combinatorial nature of structured prediction tasks gives rise to piecewise smooth models, as in the case of SVMs. However, predicting a structured output involves assignment of probabilities to vectors that encode relations between multiple simpler objects or labels. Therefore, the combinatorial aspect of the task is especially egregious in structured prediction. For example, in SSVMs, the number of pieces in the piecewise smooth

✉ Pooya Ronagh
pooya.ronagh@1qbit.com

1 1QB Information Technologies (1QBit), Vancouver, BC, Canada

2 University of British Columbia, Vancouver, BC, Canada

3 Institute for Quantum Computing, Waterloo, ON, Canada

4 University of Waterloo, Waterloo, ON, Canada

Published online: 01 September 2022

model is often exponentially large in terms of the dimension of prediction vectors.

The underlying optimization problem in structured prediction tasks takes the form

$$\min_{w\in\mathbb{R}^D} \frac{1}{n}\sum_{i=1}^{n} g_i(w), \quad \text{with} \quad g_i(w) = \max_{y\in\mathscr{Y}} f_i(y, w), \tag{1}$$

where $f_i : \mathscr{Y}\times\mathbb{R}^D \to \mathbb{R}$ are strongly convex functions, $\mathscr{Y}$ is a finite set, and D is the dimension of the vector of model parameters w. In Sections 3 and 4 we will, respectively, assume that the functions f_i have bounded subgradients or Lipschitz continuous gradients. This formulation can be easily extended to the case in which each function f_i is defined on a distinct domain $\mathscr{Y}_i$. The size of $\mathscr{Y}$ can cause the evaluation of the max operator to be computationally intractable. In SVMs and SSVMs, w represents the trainable weights of the model, the functions f_i represent the margins of the points in a dataset from classifying hyperplanes with additional regularizer terms, and $\mathscr{Y}$ is an exponentially large label set. Problems in the form of (1) also arise in other machine learning applications.

In this paper, we present two methods for solving the above optimization problem following the approaches taken in classical convex optimization for nonsmooth optimization. The first approach relies on *subgradient methods*, which are among the common techniques for optimization of nonsmooth models. We introduce a quantum algorithm based on a subgradient method and the quantum minimum finding (QMF) algorithm (Durr and Hoyer 1996). Another common technique for dealing with nonsmooth models is *smooth approximation*, for example, by using softmax operators. Although these techniques are effective at hiding the nonsmooth aspects of the problem by replacing a piecewise smooth problem with a single smooth approximation, computing that approximation can be intractable when the number of pieces is large. In our second quantum algorithm, we consider a smoothing that combines softmax approximation and quantum Gibbs sampling. The quantum Gibbs sampler is used to estimate expected values of certain observables of the Boltzmann distribution of a many-body system, which in turn provide estimations of gradients of the smooth approximation.

The performance advantages of the quantum algorithms introduced in this paper are as follows.

(a) Both of our quantum algorithms achieve quadratic quantum speedups (up to polylogarithmic factors) in terms of the size of the label space in structured prediction tasks. This is an important speedup for machine learning applications since the techniques for classification with a small number of labels do not translate into performant methods in tasks with large numbers of labels (Bi and Kwok 2013).

(b) From the machine learning point of view, optimizing the smooth approximation of the objective function in (1) itself is of natural interest. The softmax approximation allows for the optimizer to fit a reasonable model to the structured prediction problem while avoiding settling into erroneous minima that are artifacts of limited training data. This may result in better generalization and more-robust learning (Lee and Mangasarian 2001).

(c) Quantum algorithms that achieve quadratic speedups for classical optimization problems often do so at the expense of much worse scaling in terms of the solution precision they achieve. For example, quantum algorithms for linear programming and semi-definite programming incur higher-order polynomial complexities with respect to solution precision (see Section 1.1). The higher-order polynomial scalings in the precision of the solution are obstacles to the practical applicability of these algorithms. In contrast, our quantum algorithms possess linear scalings (up to polylogarithmic factors) in the precision of the solutions they return. This scaling agrees with the classical optimal bounds for first-order methods in the convex optimization of nonsmooth functions (Shamir and Zhang 2013; Nesterov 2005).

(d) None of our quantum algorithms require QRAM (Giovannetti et al. 2008) access to classical data. Instead, classical data may be provided through an online stream of single instances (or via mini-batches). Processing each sample amounts to reprogramming the quantum oracles in QMF and quantum Gibbs sampling or equivalently reprogramming a register of qubits that obtains a binary encoding of the sample in computational bases states.

We note that both quantum algorithms require fault-tolerant quantum computers for execution as they rely on repeated applications of amplitude amplification. Our first algorithm is simpler as it employs QMF directly. The conditions for our complexity theoretic results are also more general as the functions f_i are not required to be differentiable (but they have bounded subgradients; see Condition 1). However, the complexity of this algorithm depends logarithmically on the inverse of the minimum gap attained by the functions $f_i(y, w)$ (as functions of y) when w varies through training. This gap may decrease exponentially fast in real-world applications and therefore the logarithmic dependence on it may not necessarily be negligible. Our second algorithm is more involved as it relies on smooth approximation and quantum Gibbs sampling. It also requires f_i to be differentiable with Lipschitz continuous gradients (see

Condition 3). However, its complexity does not depend on the gap of the functions f_i. On the other hand, the complexity of this algorithm with respect to the precision, ϵ, of the solution, is slightly worse (scaling with $\epsilon^{-1.5}$ up to logarithmic factors, as opposed to ϵ^{-1} for the first algorithm).

1.1 Related literature

Stochastic gradient descent (SGD) is a simple and efficient algorithm that has become the core algorithm for classical large-scale convex optimization and its applications in machine learning. SGD relies on classical queries to an unbiased estimator of the gradients of the objective function. SGD generalizes to non-differentiable convex functions, in which case it suffices to have access to unbiased estimators of the subgadients (i.e., the expected queried vector has to be an element of the subgradient set). SGD can only achieve a $\tilde{O}(1/\epsilon^2)$ convergence rate for nonsmooth functions (Shamir and Zhang 2013), making it provably suboptimal for nonsmooth optimization (Harvey et al. 2018). However, variants of it, such as SGD with *suffix averaging* (Rakhlin et al. 2012) or SGD with *polynomial-decay averaging* (Shamir and Zhang 2013; Lacoste-Julien et al. 2012) achieve the optimal convergence rate of $\tilde{O}(1/\epsilon)$.[1]

In this paper, we use stochastic (sub-)gradient descent with polynomial-decay averaging (SGDP) to solve the structured prediction problem (1). We use the QMF of Dürr and Høyer (Durr and Hoyer 1996) as a subroutine to solve the inner discrete optimization problem in (1) over the label space $\mathscr{Y}$. QMF is based on Grover's search algorithm (Grover 1996). Since QMF has a randomized nature, it may fail to return a correct unbiased estimator of the subgradient in (1). Our analysis shows that the failure of QMF can be overcome if its failure rate is kept at $O(\epsilon)$.

A second approach to solving problems in the form of (1) is to approximate the piecewise smooth problem with a fully smooth one and use variants of gradient descent design for smooth problems, such as SAGA (Defazio et al. 2014). Each function g_i is replaced with an approximation that is strongly convex with a Lipschitz continuous gradient. These smooth approximations typically rely on replacing the max operator with the differentiable softmax operator (Gao and Pavel 2017; Beck and Teboulle 2012), that is, each function g_i is replaced by the smooth approximation

$$g_i^{\beta}(w) := \frac{1}{\beta} \log \sum_{y \in \mathscr{Y}} e^{\beta f_i(y,w)}, \tag{2}$$

which is at least as computationally difficult as evaluating the original max operator. This approximation can be interpreted from a thermodynamic perspective, wherein each g_i^{β} represents the free energy of a system with an energy spectrum described by f_i. This motivates the use of quantum Gibbs samplers for computing such smooth approximations.

It has been speculated for the past 20 years that quantum computers can be used to generate samples from Gibbs states (Terhal and DiVincenzo 2000). Since then, many algorithms for Gibbs sampling based on a quantum-circuit model have been introduced (Poulin and Wocjan 2009; Temme et al. 2011; Kastoryano and Brandao 2016; Chowdhury and Somma 2016; van Apeldoorn et al. 2017). The Gibbs sampler of van Apeldoorn et al. (2017) has a logarithmic dependence on the error of the simulated distribution. The sampler of Chowdhury and Somma (2016) similarly has a logarithmic error dependence, but assumes 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 quantum speedup in Gibbs sampling.

Moreover, numerical and experimental heuristics may provide even better practical performance for Gibbs sampling. The Gibbs sampler of Temme et al. (2011) has an unknown runtime, but has the potential to provide efficient heuristics since it relies on a quantum Metropolis algorithm. Experimentally, 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 (Sarandy and Lidar 2005; Avron et al. 2012; Albash et al. 2012; Bachmann et al. 2016; Venuti et al. 2016). 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. (2017) 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 (Matsubara et al. 2017; Okuyama et al. 2017).

Boltzmann distributions arise naturally in machine learning computations due to the principle of maximum entropy. Sampling from Boltzmann distributions, although computationally challenging, is unavoidable in Markovian models of reasoning. For instance, we refer the reader to the Hammersley–Clifford theorem (Wainwright et al. 2008; Koller and Friedman 2009) in the context of training undirected probabilistic graphical models (UGM). With the success of deep neural networks in many practical applications, improving their performance by combining them with UGMs has become an active area of research. For example, a combination of convolutional

[1] Throughout, the notation $\tilde{O}$ is used to hide polylogarithmic factors.

neural networks (CNN) and fully connected conditional random fields (a type of UGM), has achieved state of the art performance in image segmentation—a critical task in computer vision. In image tagging, another computer vision task, combining CNNs with Ising models (as a UGM with binary random variables and pairwise interactions), has been found to provide improved accuracy (Chen et al. 2015). Kim et al. (2011) formulate image segmentation as a correlation clustering problem and solve it using UGMs with higher-order interactions. Similarly Yu and Joachims (2009) solve the natural language processing problem of noun phrase coreference by formulating it as a correlation clustering problem.

The idea of using Gibbs sampling as a subroutine in quantum machine learning has already been considered. Wiebe et al. (2014) use Gibbs state preparation to propose an improved framework for quantum deep learning. Crawford et al. (2016) and Levit et al. (2017) 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) (Brandao and Svore 2017; Brandão et al. 2017; van Apeldoorn et al. 2017). 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. (2017) propose a quantum algorithm for linear programming that requires $\tilde{O}(\epsilon^{-5})$ queries to the input of the LP, and an algorithm for SDPs that requires $\tilde{O}(\epsilon^{-8})$ queries to the input matrices of the SDP, where ϵ is an additive error on the accuracy of the final solution. Van Apeldoorn and Gilyén (2018) later improved the scaling of their result by further analysis and reduced the dependence on precision parameters to $\tilde{O}(\epsilon^{-4})$ and more recently to $\tilde{O}(\epsilon^{-3.5})$ (van Apeldoorn and Gilyén 2019). Several lower bounds proven in van Apeldoorn et al. (2017) and van Apeldoorn and Gilyén (2018) suggest that these results cannot be improved significantly further. In particular, the polynomial dependence on precision parameters is necessary. In contrast, the quantum algorithms proposed in this paper provide (optimal) $\tilde{O}(\epsilon^{-1})$ scaling in precision of the solutions they return.

1.2 Summary of results

The two classical optimization approaches discussed above for (a) solving the original min-max problem directly with the subgradient method, and (b) solving a smooth approximation of it using SAGA (Defazio et al. 2014), inspired the design and analysis of the two quantum algorithms we present in this paper. Our contributions are summarized as follows.

Theorem 1 We show that stochastic subgradient descent with polynomial-decay averaging (SGDP), under conditions that take the probabilistic errors of quantum minimum finding into account, can solve the optimization problem (1) with the same optimal convergence rate as SGDP under its original conditions.

Theorem 2 We then derive the query complexity of Q-SGDP (SGDP using quantum minimum finding) and observe a quadratic speedup in terms of the size of the discrete set $\mathcal{Y}$. The caveat, however, is that the query complexity reported in this theorem has a factor of $\log \frac{1}{G}$ where G is the minimum gap attained by the functions $f_i(y, w)$ as functions of y as w ranges over its various observed values throughout the algorithm.

Theorem 3 We show that SAGA, under conditions that take the probabilistic errors of quantum Gibbs sampling into account, minimizes a smooth approximation of the objective function in (1) with the same optimal convergence rate as SAGA under its original conditions.

Theorem 4 We then derive the query complexity of Q-SAGA (SAGA using quantum Gibbs sampling) for solving the original nonsmooth optimization problem (1). We conclude that Q-SAGA also achieves a quadratic speedup in terms of $|\mathcal{Y}|$ without a dependence on the minimum gap, at the expense of a slightly worse scaling in terms of ϵ.

Experiments of Section 5.2 To show the real-world applicability of our approach, we formulate image tagging as a structured prediction task. We then train a neural network with leading deep layers and a trailing probabilistic graphical model for this task. Our numerical results show that this hybrid architecture trained with a structured prediction objective function in the form of smooth approximation of the objective in (1) can outperform a purely deep model with the same number of parameters.

The proofs of all propositions can be found in the appendix.

2 Background

We first present a brief account of SVMs and SSVMs. We refer the reader to Ng (2010) for the basics of SVMs and to Yu (2011) for SSVMs. We then introduce the more general framework of structured prediction tasks in machine learning. These models are of particular interest in scenarios where the numbers of labels are very large, for example, when a label can be any of the exponentially many binary vectors of a given dimension.

2.1 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 $w \in \mathbb{R}^D$ and $b \in \mathbb{R}$ defining a separating hyperplane $w^T x + b$. For a point $(x, y) \in \mathcal{S}$, the positivity of the product $y(w^T x + b)$ indicates

the correct classification of x. The SVM optimization problem can be expressed as

$$\begin{aligned} \min_{w,b} \quad & \tfrac{1}{2}\|w\|^2 \\ \text{s.t.} \quad & y\big(w^T x + b\big) \geq 1 \quad \forall (x,y) \in \mathscr{S}. \end{aligned} \tag{3}$$

The constraints ensure that every $(x, y) \in \mathscr{S}$ is classified correctly within a confidence margin. If $y(w^T x + b)$ is positive, one could superficially satisfy $y\big(w^T x + b\big) \geq 1$ by scaling up w and b. To avoid this we minimize the objective $\frac{1}{2}\|w\|^2$. In other words, the constraints ensure that the distance of $\mathscr{S}$ 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 $\xi_{(x,y)}$ for every data point in $\mathscr{S}$:

$$\begin{aligned} \min_{w,b,\xi} \quad & \tfrac{\lambda}{2}\|w\|^2 + \sum_{(x,y)\in\mathscr{S}} \xi_{(x,y)} \\ \text{s.t.} \quad & y\big(w^T x + b\big) \geq 1 - \xi_{(x,y)} \quad \forall (x,y) \in \mathscr{S} \\ & \xi_{(x,y)} \geq 0 \quad \forall (x,y) \in \mathscr{S}. \end{aligned} \tag{4}$$

Here, λ is a hyperparameter adjusting the relative importance of the two terms in the objective. We note that when $1 - y\big(w^T x + b\big) \geq 0$, the minimization of $\xi_{(x,y)}$ in the objective function turns the first constraint in (4) active, that is, $\xi_{(x,y)} = 1 - y\big(w^T x + b\big)$. However, when $1 - y\big(w^T x + b\big) < 0$, the second constraint is active and therefore $\xi_{(x,y)} = 0$. Thus, the optimization problem (4) can be written in the following unconstrained form:

$$\min_{w,b} \quad \tfrac{\lambda}{2}\|w\|^2 + \sum_{(x,y)\in\mathscr{S}} \max\big\{1 - y\big(w^T x + b\big), 0\big\}. \tag{5}$$

For simplicity, we remove the bias b from the rest of our analysis and consider it a trainable feature of x.

Let $\mathscr{Y}$ now contain more than just two labels. The *score* of a label $y \in \mathscr{Y}$ is then represented by a dot product $w_y^T x$ where w_y is a vector of model parameters specific to the label y. The Crammer–Singer formulation of the multi-label SVM problem is

$$\begin{aligned} \min_{w,\xi} \quad & \tfrac{\lambda}{2} \sum_{y\in\mathscr{Y}} \|w_y\|^2 + \sum \xi_{(x,y)} \\ \text{s.t.} \quad & w_y^T x - w_{y'}^T x \geq 1 - \xi_{(x,y)} \\ & \qquad \forall (x,y) \in \mathscr{S}, \forall y' \in \mathscr{Y} \setminus \{y\} \\ & \xi_{(x,y)} \geq 0 \qquad \forall (x,y) \in \mathscr{S}. \end{aligned} \tag{6}$$

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:

$$w^T = (w_1^T, \ldots, w_{|\mathscr{Y}|}^T). \tag{7}$$

We then introduce the *joint feature map*

$$\boldsymbol{\Phi}(x, y) = (0, \ldots, x, \ldots, 0), \tag{8}$$

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

$$\Delta(y', y) = \begin{cases} 1 & y \neq y', \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Then, the model can be rewritten as

$$\begin{aligned} \min_{w,\xi} \quad & \tfrac{\lambda}{2}\|w\|^2 + \sum \xi_{(x,y)} \\ \text{s.t.} \quad & \xi_{(x,y)} \geq \Delta(y', y) - w^T \boldsymbol{\Phi}(x, y) + w^T \boldsymbol{\Phi}(x, y') \\ & \forall (x,y) \in \mathscr{S}, \forall y' \in \mathscr{Y} \\ & \xi_{(x,y)} \geq 0 \quad \forall (x,y) \in \mathscr{S}. \end{aligned} \tag{10}$$

The above model is that of an SSVM in general, with possibly more complicated joint feature maps $\boldsymbol{\Phi}$ and loss functions Δ. This optimization problem can be written in the unconstrained form of $\min_w f_{\text{SSVM}}(w)$ for

$$\begin{aligned} f_{\text{SSVM}}(w) \;=\; & \tfrac{\lambda}{2}\|w\|^2 \\ + \sum_{(x,y)\in\mathscr{S}} \max_{y'} \quad & \big\{\Delta(y', y) + w^T\big[\boldsymbol{\Phi}(x, y') - \boldsymbol{\Phi}(x, y)\big]\big\}, \end{aligned} \tag{11}$$

which is a min-max optimization problem of the form

$$\min_w \left(f(w) = \left\{ \sum_{(x,y)\in\mathscr{S}} \max_{y'} f_{(x,y)}(y'; w) \right\} \right), \tag{12}$$

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

$$\begin{aligned} f_{(x,y)}(y'; w) \;=\; & \tfrac{\lambda}{2|\mathscr{S}|}\|w\|^2 \\ & + \Delta(y', y) + w^T\big[\boldsymbol{\Phi}(x, y') - \boldsymbol{\Phi}(x, y)\big]. \end{aligned} \tag{13}$$

Without the regularizer term, problem (10) is therefore readily of the mathematical form of the Lagrangian dual problems studied in Ronagh et al. (2016) and Karimi and Ronagh (2017), and cutting plane or subgradient methods 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 van Apeldoorn et al. (2017) 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.

2.2 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 (13). We will assume that structured prediction problems are equipped with the following elements.

(a) A training dataset $\mathscr{S} \subseteq \mathscr{X} \times \mathscr{Y}$.
The sets $\mathscr{X}$ and $\mathscr{Y}$, respectively, contain all possible inputs and outputs. The elements of $\mathscr{Y}$ 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 $\mathscr{Y}$ may be exponentially large in the size of the input. This distinguishes structured prediction from multi-label classification.

(b) A score function $s_w : \mathscr{X} \times \mathscr{Y} \to \mathbb{R}$.
The score function $s_w(x, y) = s(x, y; w)$ is indicative of the suitability of a label y for a given input x, where w is a vector of tunable parameters. The *predictor* for the model which is a function mapping the input x and parameter w to an output prediction y is therefore

$$h_w(x) = \operatorname*{argmax}_{y'} s(x, y'; w). \tag{14}$$

(c) A real-valued loss function $\Delta : \mathscr{Y} \times \mathscr{Y} \to \mathbb{R}$.
The goal is to find a predictor h_w like (14) that minimizes the *empirical risk*

$$R(h_w) = \frac{1}{|\mathscr{S}|} \sum_{(x,y)\in\mathscr{S}} \Delta(h_w(x), y). \tag{15}$$

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

$$y = \operatorname*{argmin}_{y'} \Delta(y', y), \tag{16}$$

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

$$\Delta(y, y) = 0, \quad \forall y \in \mathscr{Y}, \tag{17}$$

since we can always shift it to $\Delta'(y', y) = \Delta(y', y) - \Delta(y, y)$. This decreases the empirical risk by the constant

$$\frac{1}{|\mathscr{S}|} \sum_{(x,y)\in\mathscr{S}} \Delta(y, y), \tag{18}$$

which is an invariant of $\mathscr{S}$.

Example 1 In the SSVM framework of Section 2.1, the loss function Δ is simply the Kronecker delta function, in other words $\Delta(y, y') = \delta_{y,y'}$, and the score function is linear in the training parameters: $s(x, y; w) = w^T \Phi(x, y)$. Furthermore, if $\Phi(x, y)$ is quadratic in y and $\mathscr{Y}$ is the set of binary vectors with a fixed length, then $s(x, y; w)$ corresponds to the energy of an Ising model, which appears in many structured prediction tasks as discussed in Section 1. With this score function s, the task of prediction using the predictor h corresponds to finding the ground state of the Ising model.

2.3 A min-max optimization problem

We now present a general mathematical programming model motivated by our machine learning discussion above and in Sections 3 and 4 we propose two quantum algorithms for solving it.

We define the objective function

$$f(w) = \frac{1}{n} \sum_{i=1}^{n} g_i(w), \quad \text{where} \quad g_i(w) = \max_{y\in\mathscr{Y}} f_i(y, w). \tag{19}$$

Here w is a vector of tunable real-valued parameters, n is a positive integer, and all f_i are strongly convex real-valued functions of w with Lipschitz continuous gradients. Furthermore each f_i is defined in its first argument y over a finite set $\mathscr{Y}$. In practical machine learning examples, f_i could be strongly convex because of the addition of a strongly convex regularizer (e.g., L^2 regularizer) to an already convex loss function. We are interested in solving the optimization problem

$$w_* = \operatorname*{argmin}_{w} f(w). \tag{20}$$

Even if 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.

3 Nonsmooth optimization

Here we provide a time complexity analysis on the optimization of problem (19) using a stochastic variant of the subgradient method (Harvey et al. 2018; Shamir and Zhang 2013) that incorporates a gradient-averaging scheme known *polynomial-decay averaging* (Shamir and Zhang 2013; Lacoste-Julien et al. 2012).

Our approach is to use stochastic (sub-)gradient descent with polynomial-decay averaging (SGDP) to optimize the nonsmooth objective function f with quantum minimum finding providing subgradient approximations for SGDP. Because the estimates of the subgradients are not exact, we need to revisit the convergence of SGDP in the presence of errors in calculating the subgradients and do so in the following sections.

3.1 A-SGDP: Approximate SGDP

We assume the following condition about the function f.

Condition 1 Each function f_i is μ-strongly convex, and as a result each g_i is also μ-strongly convex. The vector w is restricted to a convex set $\mathscr{W}$. Furthermore, the subgradients of $f_i(y, w)$ exist and have the bounded norms

$$\sup_{w,i,y} \left\{ \|v\|^2 : v \in \partial f_i(y, w) \right\} \leq M, \tag{21}$$

where the supremum ranges over every index i, every $y \in \mathscr{Y}$, and every $w \in \mathscr{W}$. Finally, each function f_i has an efficient quantum oracle, that is, one that acts on $O(\text{polylog}(\frac{1}{\delta}, |\mathscr{Y}|))$ qubits to compute f with an additive error of δ.

The algorithm is as follows. We use the SGDP algorithm of Shamir and Zhang (2013), where, at each iteration, we compute a maximizer y for a function f_i using the quantum minimum finding algorithm (Durr and Hoyer 1996). In the end, we return the weighted average of w at each iteration according to the polynomial-decay averaging scheme. SGDP, combined with the quantum minimum finding algorithm, yields what we refer to as the *quantum* SGDP (Q-SGDP) algorithm.

Before analyzing Q-SGDP, we introduce and analyze the approximate variant of SGDP, called *approximate* SGDP (A-SGDP) Algorithm 1, in which we account for a probabilistic rate of failure in finding a maximizer for the discrete optimization of f_i over $\mathscr{Y}$.

1: **procedure** A-SGDP($w^0 \in \mathscr{W}$: initial point, $T \in \mathbb{N}$: number of iterations, $f_1, \ldots, f_n$: functions, $\eta > 0$: SGDP parameter)
2: $\bar{w}_\eta^0 := w^0$ ▷ Initialize the polynomial-decay averaging value.
3: **for** $t \in \{0, \ldots, T-1\}$ **do**
4: $i \sim \text{Unif}\{1, \ldots, n\}$ ▷ Draw the index i uniformly at random.
5: $\hat{y}_i^t \sim \widehat{\text{argmax}}_{y \in \mathscr{Y}} f_i(y, w^t)$ ▷ Estimate $\text{argmax}_{y \in \mathscr{Y}} f_i(y, w^t)$ with a success probability of at least $1 - \zeta$.[a]
6: $\widehat{\partial g_i(w^t)} := \partial f_i(\hat{y}_i^t, w^t)$ ▷ Calculate the gradient of g_i defined in (19).
7: $v^{t+1} := w^t - \gamma_t \widehat{\partial g_i(w^t)}$ ▷ Perform a gradient descent step with step size γ_t as in Theorem 1.
8: $w^{t+1} := \Pi_{\mathscr{W}}(v^{t+1})$ ▷ Project onto the convex set $\mathscr{W}$.
9: $\bar{w}_\eta^{t+1} := \frac{t+1}{t+\eta+2} \bar{w}_\eta^t + \frac{\eta+1}{t+\eta+2} w^{t+1}$ ▷ Update the polynomial-decay averaging value.
10: **end for**
11: **return** $\bar{w}_\eta^T$
12: **end procedure**

[a] The hat notation indicates that, with probability at most ζ, the maximization in line 5 and consequently the gradient calculations in lines 6 and 7 may have failed.

Algorithm 1: Pseudocode for the A-SGDP algorithm

Example 2 As an illustrative example, we show how the A-SGDP algorithm can be applied to solving the SVM problem (5). Recall that in SVMs the tunable model parameters are w and b as in Section 2.1 (and generically denoted by w in Algorithm 1 and the rest of the paper). Also recall that for SVMs the set of labels is $\mathscr{Y} = \{-1, 1\}$. Let $(x_i, y_i) \in \mathscr{S}$ be the i-th sample as per line 4 of Algorithm 1. We then have

$$f_i(y, w, b) = \begin{cases} 1 - y_i(w^T x_i + b) & y = 1, \\ 0 & y = -1. \end{cases} \tag{22}$$

Consequently, in line 5 we obtain

$$\hat{y}_i = \begin{cases} 1 & 1 - y_i(w^T x_i + b) > 0, \\ -1 & \text{otherwise}, \end{cases} \tag{23}$$

assuming that our estimate of $\hat{y}_i$ is free of errors (e.g., if $\zeta = 0$). Then, in line 6 we obtain

$$\widehat{\partial_w g_i(w)} = \partial_w g_i(w) = \begin{cases} -y_i x_i & \hat{y}_i = 1, \\ \mathbf{0} & \text{otherwise}, \end{cases} \tag{24}$$

where $\mathbf{0}$ is the vector of all zeroes, and

$$\widehat{\partial_b g_i(w)} = \partial_b g_i(w) = \begin{cases} -y_i & \hat{y}_i = 1, \\ 0 & \text{otherwise}. \end{cases} \tag{25}$$

The remaining steps of Algorithm 1 follow with no particular specifications for SVMs.

We now analyze the complexity of A-SGDP. Recall that a minimizer of f is denoted by w_* in (20).

Theorem 5 Under Condition 1, given a target accuracy $\epsilon > 0$, A-SGDP finds a point $w \in \mathscr{W}$ satisfying $|f(w) - f(w_*)| \leq \epsilon$ with a probability of at least 1/2 in

$$T = O\left(\frac{M}{\mu\epsilon}\right) \tag{26}$$

descent iterations, if the step size at iteration t is chosen as $\gamma_t = \frac{1}{\mu t}$, and the failure probability in solving $\underset{y}{\text{argmax}}\, f_i(y, w^t)$ at iteration t is at most $\zeta = \frac{1}{4T}$.

3.2 Quantum minimum finding

The quantum minimum finding algorithm (QMF) (Durr and Hoyer 1996) relies on coherent queries to an oracle implementing a real-valued function $f : \mathscr{Y} \to \mathbb{R}$ on a discrete domain $\mathscr{Y}$ of size N. QMF performs several iterations of Grover's search on a comparative oracle between the values of f and a classically chosen threshold value to determine a next point $y \in \mathscr{Y}$ on which f attains a smaller value than the threshold. After these iterations, QMF returns an optimizer of f with high probability using $\tilde{O}(\sqrt{N})$ queries. The success probability of QMF can be trivially boosted to any target $1 - \zeta$ by $O(\log(1/\zeta))$ repetitions of the algorithm.

We now provide a statement of the query complexity of QMF following Durr and Hoyer (1996, Theorem 1). As we are interested in the complexity of queries made to the oracles of the functions f_i, we take into account the number of additional elementary gates required to implement the mentioned comparative oracle. This is also done in van Apeldoorn et al. (2017, Theorem 49). Lemma 1 is a statement of QMF query complexity using the notation developed in this paper.

Lemma 1 Let i be a fixed index and f_i be the corresponding function as defined in (1). Let F be a bound on the absolute values of f_i, that is, $F = \max_{y \in \mathscr{Y}} |f_i(y, w)|$, and G be the difference between the maximum value of the function and the second-largest value of f_i. Let U be a unitary that implements f_i and acts on q qubits in order to do so. There exists a quantum algorithm that returns a (not necessarily unique) point $\hat{y}_i \in \operatorname{argmax}_{y \in \mathscr{Y}} f_i(y, w)$, with a probability of at least $1 - \zeta$, in

$$O\left(\sqrt{|\mathscr{Y}|}\log(1/\zeta)\right) \tag{27}$$

queries to the oracle of f_i, and using

$$O\left(\sqrt{|\mathscr{Y}|}\log(F/G)\log(1/\zeta)\right) \tag{28}$$

additional other quantum gates.

3.3 Q-SGDP: Quantum SGDP

We may now analyze the complexity of Q-SGDP, which is the result of combining A-SGDP and the quantum minimum finding algorithm. The query complexity of Q-SGDP can be found using the asymptotic number of descent steps reported in Theorem 5 as follows.

Theorem 6 Under Condition 1, given a target accuracy $\epsilon > 0$, Q-SGDP finds a point $w \in \mathscr{W}$ satisfying $|f(w) - f(w_*)| \leq \epsilon$ with a probability of at least 1/2 in

$$T = \tilde{O}\left(\frac{M\sqrt{|\mathscr{Y}|}}{\mu\epsilon}\right) \tag{29}$$

queries to the quantum oracles of f_i and using

$$\tilde{O}\left(\frac{M\sqrt{|\mathscr{Y}|}}{\mu\epsilon}\log\frac{F}{G}\right) \tag{30}$$

additional quantum gates. Here F is a bound on the absolute values of f_i for all i, y, and w and G is the minimum gap attained by the functions f_i for different values of $y \in \mathscr{Y}$ throughout the runtime.

4 Smooth optimization

In Section 3, we considered the subgradient method for minimizing the nonsmooth objective function (19). In convex optimization, smooth approximation of nonsmooth objective functions is a common method for designing improved gradient-based solvers (Beck and Teboulle 2012). We construct such a smooth approximation of the function f, and find the minimum of the approximation.

One approach to smoothing the max of a set of functions is *softmax smoothing* (Beck and Teboulle 2012). For a finite set $\mathscr{Y}$ and $\beta > 0$, the softmax approximation of the max operator over a set of values $\mathscr{Y}$ is defined as

$$\max_{y \in \mathscr{Y}}{}^{\beta} y = \frac{1}{\beta}\log\sum_{y \in \mathscr{Y}}\exp(\beta y). \tag{31}$$

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 (19) and the resultant summation is called the smooth approximation of f at inverse temperature β, denoted by $f^{\beta}(w)$:

$$f^{\beta}(w) = \frac{1}{n}\sum_{i}\max_{y \in \mathscr{Y}}{}^{\beta} f_i(y, w). \tag{32}$$

We note that $f^{\beta}(w)$ converges uniformly to $f(w)$ in the limit of $\beta \to \infty$ (refer to (129) 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 (20) with

$$w_*^{\beta} = \operatorname*{argmin}_{w} f^{\beta}(w). \tag{33}$$

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

$$\nabla_w f^\beta(w) = \frac{1}{n}\sum_i \mathbb{E}_Y(\nabla_w f_i(Y_i, w)), \tag{34}$$

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

$$\mathbb{P}(Y = y) = \frac{\exp(\beta f_i(y, w))}{\sum_{y' \in \mathscr{Y}} \exp(\beta f_i(y', w))}, \quad y \in \mathscr{Y}. \tag{35}$$

4.1 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 $f_i(-, w) : \mathscr{Y} \to \mathbb{R}$ corresponds (with an opposite sign) to the spectrum of a diagonal Hermitian matrix with the functions $-f_i(y, w)$ forming the diagonal entries and we denote this matrix by $H_i(w)$. We assume we have access to oracles for $H_i(w)$ and its partial derivatives. The oracles act on two registers via

$$|k\rangle|z\rangle \mapsto |k\rangle|z \oplus (H_i(w))_{kk}\rangle \tag{36}$$

and

$$|k\rangle|z\rangle \mapsto |k\rangle|z \oplus (\partial_j H_i(w))_{kk}\rangle \quad \forall j, \tag{37}$$

where k ranges over the elements of $\mathscr{Y}$ and z is any computational basis state. Here and in what follows, ∂_j is used as an abbreviation of the notation of partial derivatives with respect to the vector w, that is, $\partial_j = \partial/\partial w_j$. The notation $\partial_j H_i(w)$ stands for the diagonal matrix with partial derivatives $\partial_j f_i(y, w)$ forming its diagonal.

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

$$\max_{y \in \mathscr{Y}}{}^\beta f_i(y, w) = \frac{1}{\beta} \log \mathrm{Tr}(\exp(-\beta H_i(w))). \tag{38}$$

Applying stochastic gradient descent for minimizing (19) would require calculation of the gradients of $f_i(y,w)$ with respect to w, which is equal to $\mathrm{Tr}(A\rho)$ where $\rho = \frac{\exp(-\beta H)}{\mathrm{Tr}(\exp(-\beta H))}$ is the Gibbs state's density matrix and A is the observable associated with the partial derivatives

$$\partial_k \max_y{}^\beta f_i(y, w) = \mathrm{Tr}\big[\big(-\partial_k H_i(w)\big)\rho\big]. \tag{39}$$

This is exactly the type of quantity studied in van Apeldoorn et al. (2017). They show that for $N \times N$ diagonal matrices H and A, such that $\|A\| \leq 1$ (in the operator norm) 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 van Apeldoorn et al. (2017) for our application and for reference we first state their result.

Proposition 1 (Corollary 12 in van Apeldoorn et al. (2017)) 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.[2]

For our application, we need to include the contribution of the operator norms of A and H in the complexity. We also require control over the success probability of the approximation obtained in the above statement, which can be boosted using the powering lemma for fully polynomial randomized approximation schemes (Jerrum et al. 1986). Given these considerations, the complexity of quantum Gibbs sampling from classical functions is as follows.

Proposition 2 (Quantum Gibbs sampling) Let $A, H \in \mathbb{R}^{n\times n}$ be diagonal matrices with $\|A\| \leq \Delta$ 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 a success probability of at least $1 - \zeta$ using $O(\frac{\sqrt{n}\Delta\beta K}{\theta} \log \frac{1}{\zeta})$ queries to A and H, with the number of other quantum gates being almost of the same order.

We now impose boundedness conditions on the functions f_i in (38) before applying the quantum Gibbs sampling algorithm to compute the partial derivatives (39).

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

We can now derive the computational complexity of using quantum Gibbs sampling to estimate the partial derivatives in (39).

Theorem 7 Let $f_i : \mathscr{Y} \times \mathbb{R}^D \to \mathbb{R}$ be a real-valued function satisfying Condition 2. Then the gradients of (38) with respect to the parameter vector w can be calculated in

$$O\left(\frac{D\sqrt{|\mathscr{Y}|}\Delta\beta F}{\theta} \log \frac{D}{\zeta}\right) \tag{40}$$

queries to the oracles of f_i with the number of other gates being almost of the same order. Here F is a bound on the 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.

[2] In the rest of this paper, such a characterization of the number of quantum gates will be referred to as being "almost of the same order".

4.2 A-SAGA: Approximate SAGA

Stochastic average gradient (SAG) (Schmidt et al. 2017) and its variant SAGA (Defazio et al. 2014) are two optimization methods that are specifically designed for minimizing the sum of finitely many smooth functions. SAG and SAGA usually perform better than standard stochastic gradient descent (SGD) (Robbins and Monro 1985). 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 previously visited points, if the step size is small enough, and the functions are smooth, the gradients in the cache will not be too far from the current gradients; thus, using them will reduce the error of estimation of the full gradient, leading to an improved convergence rate.

Our approach is to use SAGA to minimize the smooth, strongly convex objective function $f^{\beta}(w)$ to approximate the minimum of the original nonsmooth objective function f. The 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 Theorem 7. Consequently, we need to revisit the convergence of SAGA in the presence of errors in calculating the gradients and do so in the following sections. In this section, the notation $\langle -, - \rangle$ is used to represent the inner products of vectors of real numbers.

Condition 3 Each function f_i is μ-strongly convex, resulting in each $g_i(w) = \max_y f_i(y, w)$ being μ-strongly convex. The vector w is restricted to a convex set $\mathscr{W}$. Furthermore, the gradients of $f_i(y, w)$ are ℓ-Lipschitz smooth, and the partial derivatives are bounded by

$$\Delta = \max_{w,i,j,y} \left\| \partial_j [f_i(y, w)] \right\|, \tag{41}$$

where the maximum ranges over every index i, every $y \in \mathscr{Y}$, every $w \in \mathscr{W}$, and every j-th component of w.

We note that Condition 3 has important differences with Condition 1. In Condition 3, the functions f_i have Lipschitz continuous gradients, whereas in Condition 1 there were no such restrictions. Also, in Condition 3, we impose a bound Δ on the partial derivatives, whereas in Condition 1, M is a bound on the subgradients.

In the approximate SAGA algorithm (A-SAGA) presented in Algorithm 2, we have an estimate of the gradient with an additive error of at most $\theta/3$[3] in each partial derivative appearing in the gradient. Here the update rule for SAGA from Defazio et al. (2014, 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[4]). That is,

$$\Theta^{t+1} = \Upsilon_j^{t+1} - \Upsilon_j^t + \frac{1}{n} \sum_{i=1}^{n} \Upsilon_i^t. \tag{42}$$

1: **procedure** A-SAGA($w^0 \in \mathscr{W}$: initial point, $T \in \mathbb{N}$: number of iterations, $g_1, \ldots, g_n$: functions)
2: $\quad \widetilde{g_1}, \ldots, \widetilde{g_n} := 0$ ▷ Initialize the cached values of the gradients to zero.[a]
3: $\quad \widetilde{G} := 0$ ▷ Initialize the summation of the cached gradients to zero.
4: $\quad$ **for** $t \in \{0, \ldots, T-1\}$ **do**
5: $\quad\quad j \sim \mathrm{Unif}\{1, \ldots, n\}$ ▷ Draw the index j uniformly at random.
6: $\quad\quad \widehat{g_j} := \nabla g_j(w^t) + \Upsilon_j^{t+1}$ ▷ Estimate the gradient up to an additive error Υ_j^{t+1}.[b]
7: $\quad\quad v^{t+1} := w^t - \gamma \left[\widehat{g_j} - \widetilde{g_j} + \frac{1}{n} \widetilde{G} \right]$ ▷ Take a descent step according to the SAGA formula.
8: $\quad\quad w^{t+1} := \Pi_{\mathscr{W}}(v^{t+1})$ ▷ Project the point onto the set $\mathscr{W}$.
9: $\quad\quad \widetilde{G} := \widetilde{G} + \widehat{g_j} - \widetilde{g_j}$ ▷ Update the summation of cached gradients with the new gradient.
10: $\quad\quad \widetilde{g_j} := \widehat{g_j}$ ▷ Update the cached gradient with the new gradient.
11: $\quad$ **end for**
12: $\quad$ **return** w^T
13: **end procedure**

[a] The tilde notation is used to indicate cached gradient estimations in lines 2, 3, 7, 9, and 10.
[b] The hat notation is used to indicate approximate gradients in lines 6, 7, 9, and 10.

Algorithm 2: Pseudocode for the A-SAGA algorithm

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 (42), we can conclude that every element of the vector Θ^{t+1} is at most θ.

[3] The division by 3 was chosen to simplify the formulae.

[4] 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 line 7 of Algorithm 2 contribute to the bound on Θ. More precisely, if the Gibbs sampler calculates the derivatives with error $\frac{\theta}{3}$, then $\|\Theta^{t+1}\| \leq \theta$.

Following the approach of Defazio et al. (2014) we find a bound for $\|w^t - w_*\|$ using the Lyapunov function $\mathbb{T}$ defined as

$$\begin{aligned}\mathbb{T}^t &:= \mathbb{T}(w^t, \{\phi_i^t\}_{i=1}^n) \\ &:= \frac{1}{n}\sum_i g_i(\phi_i^t) - f(w_*) \\ &- \frac{1}{n}\sum_i \left\langle \nabla g_i(w_*), \phi_i^t - w_* \right\rangle + c\left\|w^t - w_*\right\|^2,\end{aligned} \tag{43}$$

by proving the inequality $\mathbb{E}[\mathbb{T}^{t+1}] \leq (1 - \frac{1}{\tau})\mathbb{T}^t$. The next theorem proves a similar inequality in the case that an additive error on the gradients exists.

Theorem 8 Let the precision of a subroutine calculating the gradients of g_i at every point be

$$\theta = \min\left\{\frac{1}{\sqrt{D}}, \frac{\mu\|w^t - w_*\|^2}{2\sqrt{D}\left(\frac{3}{34L} + 2\|w^t - w_*\|\right)}\right\}. \tag{44}$$

Then there exists a choice of step sizes γ in line 7 of Algorithm 2, c in (43), and $\tau > 0$ such that for all t, $\mathbb{E}[\mathbb{T}^{t+1}] \leq (1 - \frac{1}{\tau})\mathbb{T}^t$.

Remark 1 As shown in the proof of this theorem in the appendix (refer to (89)), the step size γ does not depend on the strong convexity parameter μ. This is a desirable property called "adaptivity to strong convexity".

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. We let w_*^β denote the minimizer of the smooth function f^β.

Theorem 9 Under Condition 3, and given ϵ as a target precision, A-SAGA finds a point w such that $\mathbb{E}\left[\|w - w_*^\beta\|^2\right] \leq \epsilon$ using

$$O\left(\left(n + \frac{\beta D\Delta^2 + \ell}{\mu}\right)\left(\log\frac{n}{\epsilon(\beta D\Delta^2 + \ell)}\right)\right) \tag{45}$$

iterations.

Remark 2 The number of gradient evaluations in Theorem 9 is $O\left(\log\frac{1}{\epsilon}\right)$ in terms of ϵ only. Also, based on (89), we have $\theta = O(\epsilon)$.

4.3 Using A-SAGA to optimize the nonsmooth objective function

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 2 To solve the original problem (20) with ϵ-approximation, it suffices to optimize the smooth approximation (32) for $\beta > \frac{\log|\mathcal{Y}|}{\epsilon}$ with precision $\epsilon - \frac{\log|\mathcal{Y}|}{\beta}$.

Lemma 3 In solving problem (33) with A-SAGA we have

$$\mathbb{E}\left[f^\beta(w^t) - f^\beta(w_*)\right] \leq \frac{L}{2}C_0\left(1 - \frac{1}{\tau}\right)^t. \tag{46}$$

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

Theorem 10 Under Condition 3, A-SAGA applied to the function f^β at $\beta = \frac{2\log|\mathcal{Y}|}{\epsilon}$ requires

$$O\left(\left(\frac{D\Delta^2\log|\mathcal{Y}|}{\mu\epsilon} + \frac{\ell}{\mu}\right)\left(\log\frac{n}{\epsilon}\right)\right) \tag{47}$$

iterations to find a point w at which the original function value f is ϵ-close to its minimum in expectation, that is, $\mathbb{E}\left[f(w) - f(w_*)\right] \leq \epsilon$, provided ϵ is sufficiently small.

Remark 3 The number of gradient evaluations in Theorem 10 is $O\left(\frac{1}{\epsilon}\log\frac{1}{\epsilon}\right)$ in terms of ϵ. We note that in terms of the precision factor, the optimal scaling for optimizing (19) is $O(\frac{1}{\epsilon})$ (Shamir and Zhang 2013; Nesterov 2005), 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 (89), 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.

Finally, it is easy to use the previous theorem and the definition of strong convexity to show convergence of A-SAGA to an approximation of the optimal solution of f.

Corollary 1 With the same conditions as Theorem 10, A-SAGA finds a point w at which $\mathbb{E}\left[w - w_*\right] \leq \epsilon$, provided ϵ is sufficiently small, using

$$O\left(\left(\frac{D\Delta^2\log|\mathcal{Y}|}{\mu\epsilon} + \frac{\ell}{\mu}\right)\left(\log\frac{n}{\mu\epsilon}\right)\right) \tag{48}$$

iterations.

Example 3 A special case of practical importance is when the functions f_i are a linear function in w plus an L^2 regularizer. In this case our objective function to minimize is

$$f(w) = \lambda\frac{\|w\|^2}{2} + \frac{1}{n}\sum_{i=1}^n \max_{y\in\mathcal{Y}}\{a_{i,y}w + b_{i,y}\}. \tag{49}$$

Let $\mathscr{W} = \mathbb{B}^D(0,1)$ 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 minus the regularizer remain linear in w but are quadratic in y, e.g., the energy function of an Ising model (refer also to Example 1)

$$f(w) = \lambda\frac{\|w\|^2}{2} + \frac{1}{n}\sum_{i=1}^{n} \max_{y\in\mathscr{Y}}\{yJ_iy^T + h_iy^T\}, \tag{50}$$

where $\mathscr{Y} = \{-1,1\}^m$, $J_i \in \mathbb{R}^{m\times m}$, 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 $\mathscr{W}$ is the unit ball of dimension $D = nm(m+1)$ around the origin. Similar to the previous example 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 $\{-1, 1\}$.

4.4 Comparison of SAGA and A-SAGA

We now compare our previous result to the case wherein exact gradients are available. That is, we optimize f^β using SAGA, and use it to approximate the optimal solution of f.

Theorem 11 Under Condition 3, and given ϵ as a target precision, SAGA uses

$$O\left(\left(n + \frac{\beta D\Delta^2+\ell}{\mu}\right)\left(\log\frac{n}{\epsilon(\mu n+\beta D\Delta^2+\ell)}\right)\right) \tag{51}$$

gradient evaluations to find a point in the ϵ-neighbourhood of w_*^β defined in (33) and

$$O\left(\left(n + \frac{D\Delta^2\log|\mathscr{Y}|}{\epsilon\mu} + \frac{\ell}{\mu}\right)\left(\log\frac{n}{\epsilon}\right)\right) \tag{52}$$

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

It is clear that the scaling in (51) with respect to all parameters is similar to Theorem 9 and the scaling in (52) is similar to Theorem 10, except for an extra n term added in the first parentheses.

Remark 4 We summarize the results of Theorems 9, 10, and 11 by observing that with $O(\epsilon)$ and $O(\sqrt{\epsilon})$ additive errors in gradient evaluations in the A-SAGA algorithm, its scaling for optimizing f^β and f remains similar to SAGA, which does not assume any errors in gradient evaluations.

In the next section we introduce a quantum algorithm we call Q-SAGA. We note that A-SAGA (or Approximate SAGA) introduced in Algorithm 2 and discussed above is SAGA with approximate gradients, and Q-SAGA (or Quantum SAGA) is the specific case of A-SAGA wherein the approximate gradients are obtained from quantum Gibbs sampling.

4.5 Q-SAGA: a quantum algorithm for optimizing the smooth approximation

In Theorems 9 and 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 4.1, we can guarantee such an upper bound only with a non-zero probability of failure. As shown in Theorem 7, the gradients of the function $\max_y^\beta f_i(y,w)$ can be estimated with additive errors of at most θ in all partial derivatives appearing in the gradient with a hight probability. We now propose a quantum algorithm, called Q-SAGA, for optimizing the smooth approximation function $f^\beta(w)$ (by combining Theorems 7 and 9) and for optimizing the original function f (by combining Theorems 7 and 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.

Lemma 4 Under Conditions 2 and 3, each gradient evaluation takes

$$O\left(\frac{D^{1.5}\beta F\Delta}{\mu\epsilon}\left(\frac{1}{\beta D\Delta^2+\ell} + \sqrt{\epsilon}\right)\sqrt{|\mathscr{Y}|}\log(|\mathscr{Y}|)\log\frac{D}{\zeta}\right) \tag{53}$$

queries to the oracle for one of the f_i with the number of other quantum gates being almost of the same order, 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 as in (44).

Theorem 12 Under Conditions 2 and 3, given sufficiently small $\epsilon > 0$ as a target precision, Q-SAGA finds a point w satisfying $\mathbb{E}\left[\|w - w_*^\beta\|^2\right] \le \epsilon$, with a probability of at least 3/4, in

$$O\left(\frac{nD^{1.5}\beta F\Delta}{(\beta D\Delta^2+\ell)\mu\epsilon}\sqrt{|\mathscr{Y}|}\log(|\mathscr{Y}|)\left(\log\frac{n}{\epsilon}\right)\log\left(Dn\log\frac{n}{\epsilon}\right)\right) \tag{54}$$

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

$$O\left(\frac{D^{1.5}\beta F\Delta}{\mu^2\epsilon}\sqrt{|\mathscr{Y}|}\log(|\mathscr{Y}|)\left(\log\frac{n}{\epsilon}\right)\log\left(Dn\log\frac{n}{\epsilon}\right)\right) \tag{55}$$

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

Theorem 13 Under Conditions 2 and 3, given sufficiently small $\epsilon > 0$ as a target precision, Q-SAGA finds a point w such that $\mathbb{E}\left[\|w - w_*\|^2\right] \leq \epsilon$, with a probability of at least 3/4 in

$$O\left(\left(\frac{D^{2.5}\Delta^3 F\sqrt{|\mathscr{Y}|}(\log^2|\mathscr{Y}|)(\log D)}{\mu^2\epsilon^{1.5}}\right)\times\left(\log\frac{n}{\epsilon}\right)\log\left(\frac{D\Delta^2\log|\mathscr{Y}|}{\mu\epsilon}\left(\log\frac{n}{\epsilon}\right)\right)\right) \tag{56}$$

queries to the oracle for one of the f_i with the number of other quantum gates being almost of the same order. This is $\tilde{O}(\frac{1}{\epsilon^{1.5}})$ in terms of ϵ only.

5 Numerical experiments

5.1 Synthetic benchmark

We compare the optimization of the function f, as defined in (19), with its smooth approximation f^β, as defined in (32). 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 $\mathscr{Y}$) 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 plus an L^2 regularizer:

$$f_i(y,w) = \frac{\lambda}{2}\|w\|^2 + a_{i,y}^T(w - b_i') + b_{i,y} \quad y \in \mathscr{Y}, \lambda \in \mathbb{R}^+, w \in \mathbb{R}^D. \tag{57}$$

Adding of the L^2 regularizer guarantees the strong convexity of f.

The elements $y \in \mathscr{Y}$ 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' as opposed to normal or Cauchy distributions is to avoid the functions f_i having a similar minimum, which makes the problem easy.

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

Table 1 The tuned hyperparameter values

Algorithm	β	γ_0	c_γ	η
Smooth SGD	10^{-4}	10^{-2}	10^1	N/A
Nonsmooth SGD	N/A	10^{-2}	10^1	N/A
SGDP	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

We benchmark four gradient descent schemes: (1) stochastic gradient descent (SGD) applied to the smooth approximation f^β; (2) SGD applied to the original nonsmooth function f (SubSGD); (3) stochastic subgradient descent with polynomial-decay averaging (SGDP) (Shamir and Zhang 2013) applied to the original nonsmooth function f; and (4) SAGA (Defazio et al. 2014) applied to the smooth approximation f^β.

All methods have two tunable hyperparameters in common: (1) γ_0, the initial step size 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 SGDP are applied to the original nonsmooth objective function. SGDP 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 following values to form a grid in each case:

$$\beta \in \{10^{-7}, 10^{-5}, \ldots, 10^0\}; \tag{58}$$

$$\gamma_0 \in \{10^{-7}, 10^{-5}, \ldots, 10^0\}; \tag{59}$$

$$c_\gamma \in \{0\} \cup \{10^{-4}, 10^{-3}, \ldots, 10^2\}; \text{ and} \tag{60}$$

$$\eta \in \{1, 2, \ldots, 7\}. \tag{61}$$

We run each algorithm 20 times with different seeds for random number generation in the algorithm, which randomizes the choice of index i at each iteration of the algorithm, wherein we perform 1000 iterations, and track the progress on the original nonsmooth objective function f. We emphasize that the objective function remains the same over the 20 runs and the seeds are not used to regenerate the random objective function. We empirically observed that 1000 iterations were enough to see the progress of each

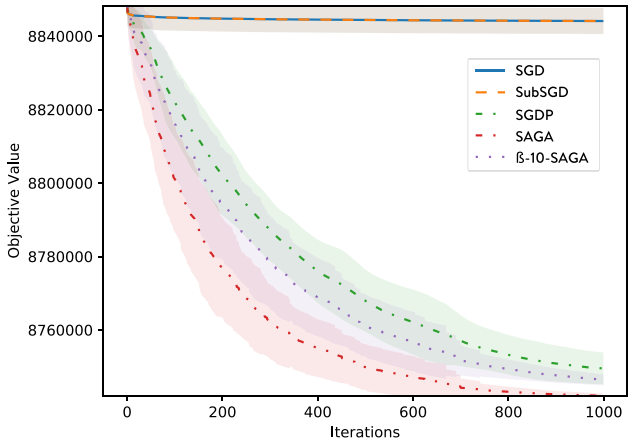


Fig. 1 The average objective value of five algorithms, SGD, SubSGD, SGDP, SAGA, and β-10-SAGA, tracked as a function of gradient descent iterations

optimization algorithm and 20 trials were enough to achieve a qualitative understanding of the error bars.

For each of the gradient descent schemes mentioned above, 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 1. Other than the four methods discussed above, a final row called β-10-SAGA has been included, a description of which 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 $\frac{1}{\mu t}$ and $\frac{\eta}{\mu(t+\eta)}$, respectively (Shamir and Zhang 2013). Note that by the contribution of the regularizer $r(w) = \frac{\lambda\|w\|^2}{2}$, we have $\mu \geq \lambda = 1$. For SGDP, 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 $\frac{\eta}{\mu(t+\eta)}$ is used.

The five gradient descent schemes, SGD, SubSGD, SGDP, SAGA, and β-10-SAGA, are compared in Fig. 1. The objective function value is recorded over 20 trials of each gradient descent scheme. For each gradient descent iteration, the average objective value is shown using dashed and dotted lines, alongside the standard deviation, shown using shaded regions.

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

We observe that the objective function value is around 8×10^6. Hence $\frac{1}{n}\sum \max f_i \approx \max f_i \approx 8\times10^6$. 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 (possibly degenerate) 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}. As shown in Fig. 1, β-10-SAGA performs slightly worse than SAGA, although it is still better than SGDP. However, $\beta\max f_i$ starts from around 0.8 and approaches 8 in the end, which is a more suitable temperature regime for a quantum Gibbs sampler.

5.2 Image tagging as a structured prediction task

In this section we consider the computer vision task of image tagging to demonstrate an application of the smooth min-max objective functions (32). In image tagging, given an image we would like to assign all the relevant tags (possibly more than one) from a predefined set of tags (e.g., "cat", "dog", "river", "nature").

We formulate this problem as a structured prediction problem and train a model on a dataset of pairs of images and tags. To achieve this, we choose three objective functions to benchmark:

1. The smoothed structural support vector machine (S3VM):

$$f_{\text{S3VM}}(w;\beta) = \frac{1}{2}\lambda\|w\|^2 + \sum_{(x,y)\in\mathcal{S}} \max_{y'}{}^{\beta}\left\{\Delta(y',y) + s(x,y';w) - s(x,y;w)\right\}. \quad (62)$$

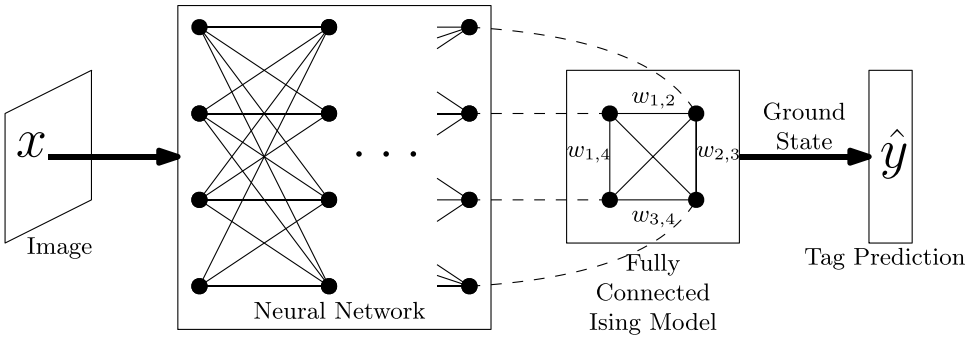


Fig. 2 Image tagging architecture. The image x is fed to a neural network to extract features. The features then are passed to an Ising model the ground state of which determines the prediction

This objective function is the smooth approximation of (11) using the softmax operator (31). Here, s is a score function, as explained in Section 2.2.

2. The conditional log-likelihood (CL) objective function:

$$f_{\mathrm{CL}}(w;\beta) = \tfrac{1}{2}\lambda\|w\|^2 + \sum_{(x,y)\in\mathscr{S}} \max_{y'}{}^{\beta}\big\{s(x,y';w) - s(x,y;w)\big\}. \tag{63}$$

Similar to S3VM, this objective function is in the form of the smooth min-max objective function (32) but does not incorporate the loss function Δ.

3. The Jensen risk bound (JRB) objective function:

$$f_{\mathrm{JRB}}(w;\beta) = \tfrac{1}{2}\lambda\|w\|^2 + \sum_{(x,y)\in\mathscr{S}} \log \mathbb{E}_{Y_B}\big(\beta \exp(\Delta(Y_B,y))\big). \tag{64}$$

This objective function incorporates the score function s and the loss function Δ in a different way. Here Y_B is a random variable that follows a Boltzmann distribution with energy $-s(x,y;w)$ at inverse temperature β. More details about the derivation of the CL and JRB objective functions can be found in Gimpel and Smith (2010).

Recall the notation used in Section 2.1. In our image tagging task, let $\mathscr{X}$ be the set of all possible images, and $\mathscr{Y} = \{-1,1\}^{\ell}$ be the set of all possible labels. The labels are ℓ-dimensional binary vectors, with each component representative of the presence or absence of a tag in the image. We would like to find the feature function $\boldsymbol{\Phi}(x,y;w_0)$ with parameter w_0. Let $\boldsymbol{\Phi}_0 : \mathscr{X}\times\mathscr{W}_0 \to \mathbb{R}^{\eta}$ be a feature function, where the first argument from $\mathscr{X}$ is an image, the second argument from $\mathscr{W}_0$ is a parameter, and the output is a real vector with $\eta \in \mathbb{N}$ dimensions. The function $\boldsymbol{\Phi}_0(x;w_0)$ serves as a base feature function in the construction of $\boldsymbol{\Phi}(x,y;w_0)$. The function $\boldsymbol{\Phi}_0(x;w_0)$ 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 $\boldsymbol{\Phi}$ based on $\boldsymbol{\Phi}_0$ is as follows: we design $\boldsymbol{\Phi}_0$ (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

$$\boldsymbol{\Phi}(x,y;w_0) = \begin{pmatrix} \mathrm{triu}(yy^T) \\ \boldsymbol{\Phi}_0(x;w_0) \circ y \\ y \end{pmatrix}, \tag{65}$$

where $\circ$ is the element-wise product. Note that $\boldsymbol{\Phi}_0(x;w_0) \circ y$ is well-defined because $\eta = \ell$ and the two vectors $\boldsymbol{\Phi}_0(x;w_0)$ and y have identical dimensions. The result is $\boldsymbol{\Phi}(x,y;w_0) \in \mathbb{R}^d$ 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 score function s as

$$\begin{aligned} s(x,y;w) &= w^T\boldsymbol{\Phi}(x,y;w_0) = \begin{pmatrix}\theta_1^T & \theta_2^T & \theta_3^T\end{pmatrix}\boldsymbol{\Phi}(x,y;w_0) \\ &= \theta_1^T\,\mathrm{triu}(yy^T) + \theta_2^T\big[\boldsymbol{\Phi}_0(x;w_0)\circ y\big] + \theta_3^T\,y. \end{aligned} \tag{66}$$

One can then interpret θ_1 as a control parameter 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 (66) is quadratic in y.

We choose the function Δ to be the Hamming distance

$$\Delta(y',y) = \mathrm{Hamming}(y',y) \tag{67}$$

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 simplifies the gradient calculations of both f_{S3VM} and f_{JRB} to expectations with respect to the Boltzmann distribution of an Ising model rather than more complicated distributions.

5.2.1 Numerical results

We use the MIRFLICKR dataset (Huiskes and Lew 2008), which consists of 25,000 images and 38 tags. We randomly

Table 2 Image tagging results

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**	**2.6900**	10^{-7}	0.0	3^1	[60.3720, 133.0482]
baseline + CL	2.6696	2.6996	10^{-6}	10^{-6}	3^1	[53.0406, 118.1979]
baseline + JRB	2.6580	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

select 20,000 images for the training set, 2500 images for the validation set, and the remaining 2500 images for the test set. This dataset consists of an extended tag set with more than 1000 words. For this demonstration, we restrict our numerical experiments to the smaller set of 38 tags to ensure fast convergence of Monte Carlo simulations. However, we note that the extended set of labels in this dataset is well within reach of Monte Carlo simulations using today's high-performance computing platforms.

We train a pre-trained AlexNet (Krizhevsky et al. 2012), a CNN, on the training data, to predict the tags. This is done on an Ubuntu machine with 32 AMD Ryzen CPU cores, 128 GB of memory, and an Nvidia Titan V GPU. We train AlexNet using the binary cross entropy objective function between its output layer and the true labels. We call this model a *baseline*. We fix the baseline, which acts as a feature extractor, and feed its output features to an Ising model which acts as a predictor. 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 Chen et al. (2015), wherein the output of an AlexNet network is fed to an Ising model in a very similar fashion, and trained using the f_{CL} objective function. The architecture of the model is shown in Fig. 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 step size, 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 instances consist of 38 variables and a fully connected architecture. For each instance, we perform 200 sweeps and collect 200 samples. In total, we have three hyperparameters, namely γ, λ, and β. We tune the hyperparameters by performing a grid search over the values

$$\gamma = \{10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}\}, \tag{68}$$

$$\lambda = \{0.0, 10^{-6}, 10^{-4}, 10^{-2}\}, \text{and} \tag{69}$$

$$\beta = \{3^{-1}, 3^0, 3^1, 3^2\}. \tag{70}$$

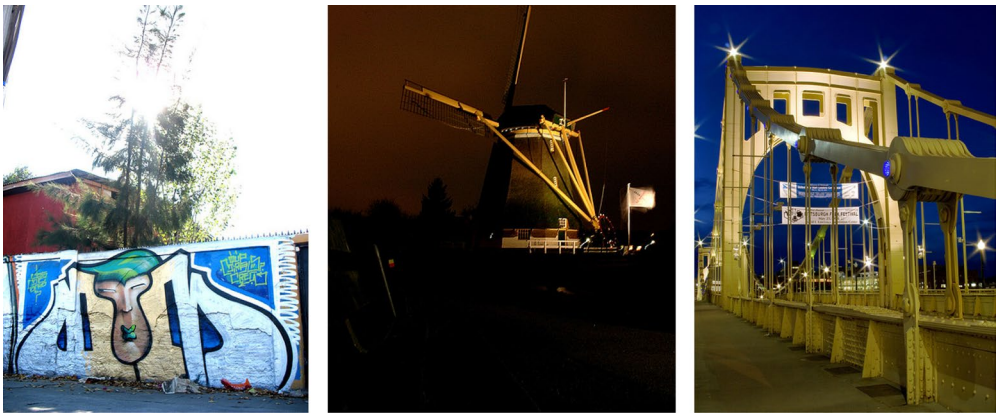
A last architecture considered is that of an extension of the baseline with a fully connected feedforward layer with sigmoid activations. This model has been added in order to compare the extensions of the baseline with undirected architectures (e.g., the Ising model) versus a feedforward layer using 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 (Kingma and Ba 2014) implemented in the PyTorch library (Paszke et al. 2017) with 300 epochs. We tune the step size parameter γ using a grid search over the values

$$\gamma = \{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}, \tag{71}$$

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

In Table 2, we summarize the performance of the various methods and values of tuned hyperparameters. 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 objective functions (62) to (64). The last row is an extension of the baseline with a single feedforward fully connected (labelled "baseline + FC") layer with sigmoid activations and the binary cross entropy objective. The performance of each method is measured via the average Hamming distance between the predicted labels and the true labels in terms of the number of bits. Therefore, the validation and test error values reported in the second and third columns are the average numbers of wrong tags (including the missing tags and the incorrect additional tags).

We observe that all three extensions of the baseline with an Ising model improve the baseline validation errors with the S3VM objective function, resulting in the greatest improvement of $\sim$0.03 tags. The same observation holds for the test errors, although the improvement is smaller in this case, at $\sim$0.015 for S3VM. In contrast, the last row of the table shows that the extension of AlexNet with an additional



Test image 7520 Test image 10177 Test image 21851

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

Fig. 3 Sample tags generated by the different models. In test images 7520, 10177, and 21851 of the dataset we see that S3VM has respectively decreased, increased, and did not affect the error, compared to the baseline

trailing fully connected layer not only increases the training and test errors, but it increases the difference of these two quantities from 0.0208 to 0.042, which hints to a slightly worsened generalization.

We observe that, in all cases, the values of λ are either 0 or very small and therefore high regularization was not required in this case study. 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 2, 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. The large values of effective β suggest that the softmax operators are very close to maximization and the smoothing effect is only material at the close proximity of those values of w where decision boundaries are degenerate.

The main takeaway of the results in Table 2 is that rows 2, 3, and 4 of the table show that the concatenation of the Ising model (as an example of an undirected probabilistic graphical model) to the baseline neural network improved the prediction accuracy of the model. However, the last row shows that this improvement cannot be merely attributed to the larger number of model parameters, since a similar increase in the number of parameters via a fully connected layer does not provide a similar advantage. It remains to be practically verified whether slight improvements observed in this table would become more significant for larger image tagging tasks.

In Fig. 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 quantum computing and classical high-performance computing platforms.

6 Conclusion

In this paper, we introduced quantum algorithms for solving the min-max optimization problem that appears in machine learning applications. We first studied the variant A-SGDP of the subgradient descent with polynomial-decay averaging (SGDP) (Shamir and Zhang 2013) which takes into account incorrect calculations of the subgradients as long as they are bounded. This allows the quantum minimum finding algorithm of Durr and Hoyer (1996)

to find the subgradients used in the subgradient method efficiently while leaving room for a small probability of failure. The combination of A-SGDP and the quantum minimum finding algorithm results in the quantum algorithm Q-SGDP for solving the original (nonsmooth) min-max optimization problem without resorting to smooth approximations. Despite an almost-linear scaling in terms of the precision factor ϵ, this quantum algorithm provides a speedup in terms of the size of the discrete optimization space $\mathscr{Y}$. We showed that Q-SGDP solves the original min-max problem in $\tilde{O}(\frac{1}{\epsilon}\sqrt{|\mathscr{Y}|}\log\frac{1}{G})$, in which the value G is the minimum gap we encounter during the runtime of the algorithm which is unknown beforehand.

Secondly, we studied 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 call 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 (Defazio et al. 2014; Schmidt et al. 2017). We then used A-SAGA to solve the original min-max problem in $\tilde{O}(\frac{1}{\epsilon})$ gradient evaluations. We remark that the best algorithms (Shamir and Zhang 2013; Nesterov 2005) 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 result is close to optimal.

Consequently, the quantum algorithm Q-SAGA solves the smooth approximation of the original min-max problem in $\tilde{O}(\frac{1}{\epsilon})$ queries to the associated quantum oracles. We also analyzed the usage of Q-SAGA, not to solve the smooth prediction problem, but to approximate a solution to the original min-max problem. In order to do this, the temperature has to be assigned proportional to ϵ. In total, this results in $\tilde{O}(\frac{1}{\epsilon^{1.5}}\sqrt{|\mathscr{Y}|})$ queries to the associated oracles. Despite a slightly worse scaling in precision, this algorithm does not have the dependence on the minimum gap G as in Q-SGDP.

Finally, we have 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 issue.

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.

Appendix

A.1. Proof of Theorem 5

Because the failure probability in solving the $\operatorname{argmax}_y f_i(y, w^t)$ in each iteration is $\zeta = 1/(4T)$, the probability of not having any failure in T iterations is at least 3/4 as we can verify

$$\prod_{t=1}^{T}(1-\zeta) = (1-\tfrac{1}{4T})^T \geq 1 - T\tfrac{1}{4T} = 1 - \tfrac{1}{4} = \tfrac{3}{4}. \tag{72}$$

Conditioned on not seeing any failure, we can directly use Theorem 4 of Shamir and Zhang (2013) to conclude that with the step size $\gamma_t = \frac{1}{\mu t}$ we need $T = O(\eta^{1.5}\frac{M}{\mu\epsilon})$ to satisfy $\mathbb{E}\left[|f(w^T) - f(w_*)|\right] \leq \frac{\epsilon}{4}$. Using Markov inequality, we can conclude that

$$\mathbb{P}\left[|f(w^T) - f(w_*)| \leq \epsilon | \text{No failure in } T \text{ iterations}\right] \geq \tfrac{3}{4}. \tag{73}$$

Hence the probability that $|f(w^T) - f(w_*)| \leq \epsilon$ is higher than 1/2 as we can verify

$$\begin{aligned}\mathbb{P}\left[|f(w^T) - f(w_*)| \leq \epsilon\right] &= \mathbb{P}\left[|f(w^T) - f(w_*)| \leq \epsilon | \text{No failure in } T \text{ iterations}\right] \\ &\quad \times \mathbb{P}[\text{No failure in } T \text{ iterations}] \geq \left(\tfrac{3}{4}\right)^2 \geq \tfrac{1}{2}.\end{aligned} \tag{74}$$

Finally hiding the dependence of complexity on η, we get the result.

A.2. Proof of Theorem 6

By multiplying the number of iterations from Theorem 5 by the query complexity found in Lemma 1, we obtain a scaling of

$$O\left(\frac{\eta^{1.5}M}{\mu\epsilon}\sqrt{|\mathscr{Y}|}\log(F/G)\log(1/\zeta)\right), \tag{75}$$

where F is a bound on the absolute values of f_i. Using the value of ζ, the result follows.

A.3. Proof of Proposition 2

This proposition is proven similarly to Corrollary 12 of van Apeldoorn et al. (2017) except that before applying Lemma 9 of van Apeldoorn et al. (2017), the success probability has to be boosted. We restate Lemma 9 of van Apeldoorn et al. (2017).

Lemma 5 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 $\|\psi\|^2 = p \geq p_{\min}$ for some known bound $p_{\min}$. Let $\mu \in (0, 1]$ be the allowed multiplicative error in our estimation of p. Then, with $O\left(\frac{1}{\mu\sqrt{p_{\min}}}\right)$ uses of U and U^{-1}, and using $O\left(\frac{q}{\mu\sqrt{p_{\min}}}\right)$ gates on the q qubits, we obtain a $\tilde{p}$ such that $|p - \tilde{p}| \leq \mu p$ with a probability of at least 4/5.

We note that this lemma provides a multiplicative approximation of p and, as such, it is a fully polynomial randomized approximation scheme as defined in Jerrum et al. (1986). Therefore, for any target rate ζ, the powering lemma (Jerrum et al. 1986, Lemma 6.1) asserts that with $O(\log(1/\zeta))$ repetitions of the algorithm and choosing the median of the returned values as the approximation $\tilde{p}$, its success probability can be boosted to $1 - \zeta$.

A.4. Proof of Theorem 7

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 a probability of at least $1 - \zeta$, we may calculate each of the partial derivatives with a success probability of at least $1 - \zeta/D$, 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 $O\left(\frac{\sqrt{|\mathscr{Y}|}\Delta\beta F}{\theta}\log\frac{D}{\zeta}\right)$ and, since there are D such partial derivatives, the result follows.

A.5. Proof of Theorem 8

Defazio et al. (2014) prove three lemmas. 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. In the following formulae we have used the prime notation as an alternative to ∇ to denote gradients.

Lemma 6 Let $f(w) = \frac{1}{n}\sum_{i=1}^n g_i(w)$. Suppose each g_i is μ-strongly convex and has Lipschitz continuous gradients with the constant L. Then for all w and w_*:

$$\begin{aligned}\langle f'(w), w_* - w\rangle &\leq \tfrac{L-\mu}{L}\left[f(w_*) - f(w)\right] \\ -\tfrac{\mu}{2}\|w_* - w\|^2 - \tfrac{1}{2Ln}\sum_i\left\|g_i'(w_*) - g_i'(w)\right\|^2 &- \tfrac{\mu}{L}\langle f'(w_*), w - w_*\rangle.\end{aligned} \tag{76}$$

Lemma 7 For all ϕ_i and w_*:

$$\tfrac{1}{n}\sum_i\left\|g_i'(\phi_i) - g_i'(w_*)\right\|^2 \leq 2L\left[\tfrac{1}{n}\sum_i g_i(\phi_i) - f(w_*) - \tfrac{1}{n}\sum_i\left\langle g_i'(w_*), \phi_i - w_*\right\rangle\right]. \tag{77}$$

The last lemma in Defazio et al. (2014) is only true if the error in the A-SAGA update rule is disregarded. We therefore restate this lemma as follows.

Lemma 8 For any ϕ_i^t, w_*, w^t, and $\alpha > 0$, with v^{t+1} as defined in SAGA, if

$$X = g_j'(\phi_j^t) - g_j'(w^t) + f'(w_*) - \frac{1}{n}\sum_i g_i'(\phi_i^t), \tag{78}$$

it holds that

$$\mathbb{E}[X] = f'(w^t) - f'(w_*), \quad \text{and} \tag{79}$$

$$\begin{aligned}\mathbb{E}\|X\|^2 &\leq (1+\alpha^{-1})\mathbb{E}\left\|g_j'(\phi_j^t) - g_j'(w_*)\right\|^2 \\ &+ (1+\alpha)\mathbb{E}\left\|g_j'(w^t) - g_j'(w_*)\right\|^2 - \alpha\|f'(w^t) - f'(w_*)\|^2.\end{aligned} \tag{80}$$

We are now ready to state the proof of Theorem 8.

Proof Theorem 8 The first three terms in $\mathbb{T}^{t+1}$ can be bounded in a way similar to the proof of Defazio et al. (2014, Theorem 1):

$$\mathbb{E}\left[\frac{1}{n}\sum_i g_i(\phi_i^{t+1})\right] = \frac{1}{n}f(w^t) + \left(1 - \frac{1}{n}\right)\frac{1}{n}\sum_i g_i(\phi_i^t), \quad \text{and} \tag{81}$$

$$\begin{aligned}\mathbb{E}\left[-\tfrac{1}{n}\textstyle\sum_i\left\langle g_i'(w_*), \phi_i^{t+1} - w_*\right\rangle\right] &= -\tfrac{1}{n}\langle f'(w_*), w^t - w_*\rangle \\ -\left(1 - \tfrac{1}{n}\right)\tfrac{1}{n}\textstyle\sum_i\left\langle g_i'(w_*), \phi_i^t - w_*\right\rangle.&\end{aligned} \tag{82}$$

The last term is bounded by the inequality

$$c\left\|w^{t+1} - w_*\right\|^2 = c\left\|\Pi_{\mathscr{W}}(v^{t+1}) - \Pi_{\mathscr{W}}[w_* - \gamma f'(w_*)]\right\|^2 \leq c\left\|v^{t+1} - w_* + \gamma f'(w_*)\right\|^2, \tag{83}$$

by the optimality of w_* and non-expansiveness of the projection operator $\Pi_{\mathscr{W}}$. 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.

$$\begin{aligned} & c\mathbb{E}\left\|v^{t+1} - w_* + \gamma f'(w_*)\right\|^2 = c\mathbb{E}\left\|w^t - w_* + \gamma X + \gamma\Theta^{t+1}\right\|^2 \\ = \; & c\left\|w^t - w_*\right\|^2 + \left\{2c\mathbb{E}\left[\left\langle \gamma X + \gamma\Theta^{t+1}, w^t - w_*\right\rangle\right] + c\mathbb{E}\left\|\gamma X + \gamma\Theta^{t+1}\right\|^2\right\} \\ = \; & c\left\|w^t - w_*\right\|^2 + \left\{-2c\gamma\left\langle f'(w^t) - f'(w_*), w^t - w_*\right\rangle + 2c\gamma\mathbb{E}\left[\left\langle\Theta^{t+1}, w^t - w_*\right\rangle\right]\right. \\ & \left. + c\gamma^2\mathbb{E}\|X\|^2 + 2c\gamma^2\mathbb{E}\left[\left\langle\Theta^{t+1}, X\right\rangle\right] + c\gamma^2\mathbb{E}\left\|\Theta^{t+1}\right\|^2\right\} \end{aligned} \tag{84}$$

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

$$\mathbb{E}\left[\left\langle\Theta^{t+1}, X\right\rangle\right] \le \theta\sqrt{D}\mathbb{E}[\|X\|] \le \theta\sqrt{D}\sqrt{\mathbb{E}\left[\|X\|^2\right]} \le \frac{\theta\sqrt{D}}{2} + \frac{\theta\sqrt{D}\mathbb{E}\|X\|^2}{2}. \tag{85}$$

We now apply Lemma 8 and the assumption that $\|\Theta^{t+1}\| \le \theta\sqrt{D}$.

$$\begin{aligned} & c\mathbb{E}\left\|v^{t+1} - w_* + \gamma f'(w_*)\right\|^2 \\ \le \; & c\left\|w^t - w_*\right\|^2 + \left\{ -2c\gamma\left\langle f'(w^t) - f'(w_*), w^t - w_*\right\rangle + 2c\gamma\mathbb{E}\left[\left\langle\Theta^{t+1}, w^t - w_*\right\rangle\right]\right. \\ & \left. + \left(c\gamma^2(1+\theta\sqrt{D})\right)\mathbb{E}\|X\|^2 + c\gamma^2\theta\sqrt{D} + c\gamma^2\mathbb{E}\left\|\Theta^{t+1}\right\|^2\right\} \\ \le \; & c\left\|w^t - w_*\right\|^2 + \left\{-2c\gamma\left\langle f'(w^t), w^t - w_*\right\rangle + 2c\gamma\left\langle f'(w_*), w^t - w_*\right\rangle + 2c\gamma\theta\sqrt{D}\|w^t - w_*\|\right. \\ & - \left(c\gamma^2(1+\theta\sqrt{D})\right)\alpha\left\|f'(w^t) - f'(w_*)\right\|^2 \\ & + \left(1+\alpha^{-1}\right)\left(c\gamma^2(1+\theta\sqrt{D})\right)\mathbb{E}\left\|g_j'(\phi_j^t) - g_j'(w_*)\right\|^2 \\ & + (1+\alpha)\left(c\gamma^2(1+\theta\sqrt{D})\right)\mathbb{E}\left\|g_j'(w^t) - g_j'(w_*)\right\|^2 \\ & \left. + c\gamma^2\theta\sqrt{D} + c\gamma^2\theta^2 D\right\}. \end{aligned} \tag{86}$$

We now apply Lemmas 6 and 7 to respectively bound $-2c\gamma\langle f'(w^t), w^t - w_*\rangle$ and $\mathbb{E}\left\|g_j'(\phi_j^t) - g_j'(w_*)\right\|^2$:

$$\begin{aligned} & c\mathbb{E}\left\|w^{t+1} - w_*\right\|^2 \le (c - c\gamma\mu)\left\|w^t - w_*\right\|^2 \\ & + \left\{ \left((1+\theta\sqrt{D})(1+\alpha)c\gamma^2 - \tfrac{c\gamma}{L}\right)\mathbb{E}\left\|g_j'(w^t) - g_j'(w_*)\right\|^2 \right. \\ & - \tfrac{2c\gamma(L-\mu)}{L}\left[f(w^t) - f(w_*) - \langle f'(w_*), w^t - w_*\rangle\right] \\ & - c\gamma^2(1+\theta\sqrt{D})\alpha\left\|f'(w^t) - f'(w_*)\right\|^2 \\ & + 2\left(1+\theta\sqrt{D}\right)\left(1+\alpha^{-1}\right)c\gamma^2 L \\ & \left[\tfrac{1}{n}\sum_i g_i(\phi_i^t) - f(w_*) - \tfrac{1}{n}\sum_i\left\langle g_i'(w_*), \phi_i^t - w_*\right\rangle\right] \\ & \left. + c\gamma^2\theta\sqrt{D} + c\gamma^2\theta^2 D + 2c\gamma\theta\sqrt{D}\|w^t - w_*\|\right\}. \end{aligned} \tag{87}$$

As in Defazio et al. (2014, Theorem 1), we pull out a $\frac{1}{\tau}$ factor of $\mathbb{T}^t$ and use the above inequalities, taking into account the contributions from the three error terms above:

$$\begin{aligned} \mathbb{E}[\mathbb{T}^{t+1}] - \mathbb{T}^t \le \; & -\tfrac{1}{\tau}\mathbb{T}^t + \left(\tfrac{1}{n} - \tfrac{2c\gamma(L-\mu)}{L} - 2c\gamma^2\mu\alpha(1+\theta\sqrt{D})\right) \\ & [f(w^t) - f(w_*) - \langle f'(w_*), w^t - w_*\rangle] \\ & + \left(\tfrac{1}{\tau} + 2(1+\alpha^{-1})(1+\theta\sqrt{D})c\gamma^2 L - \tfrac{1}{n}\right) \\ & \left[\tfrac{1}{n}\sum_i g_i(\phi_i^t) - f(w_*) - \tfrac{1}{n}\sum_i\left\langle g_i'(w_*), \phi_i^t - w_*\right\rangle\right] \\ & + \left(\tfrac{1}{\tau} - \gamma\mu\right)c\left\|w^t - w_*\right\|^2 + \left((1+\alpha)\gamma(1+\theta\sqrt{D}) - \tfrac{1}{L}\right) \\ & c\gamma\mathbb{E}\left\|g_j'(w^t) - g_j'(w_*)\right\|^2 + \left\{c\gamma^2\theta\sqrt{D} + c\gamma^2\theta^2 D + 2c\gamma\theta\sqrt{D}\|w^t - w_*\|\right\} \end{aligned} \tag{88}$$

According to a lemma that will follow (Lemma 9), we can ensure that all round parentheses in the first three lines are non-positive by setting the parameters according to

$$\begin{aligned} & \gamma = \tfrac{1}{2(1+\alpha)L}, \quad c = \tfrac{4}{n\gamma}, \quad \alpha = 16, \quad \tfrac{1}{\tau} = \min\left\{\tfrac{1}{2n}, \tfrac{\gamma\mu}{2}\right\}, \\ & \theta = \min\left\{\tfrac{1}{\sqrt{D}}, \tfrac{\mu\|w^t - w_*\|^2}{2\sqrt{D}\left(\frac{3}{34L} + 2\|w^t - w_*\|\right)}\right\}. \end{aligned} \tag{89}$$

With this setting of the parameters,

$$\left(\tfrac{1}{\tau} - \gamma\mu\right)c\left\|w^t - w_*\right\|^2 + \left\{c\gamma^2\theta\sqrt{D} + c\gamma^2\theta^2 D + 2c\gamma\theta\sqrt{D}\|w^t - w_*\|\right\} \le 0. \tag{90}$$

Using the non-negativity of the expressions in square brackets completes the proof.

In the following lemma we derive an appropriate value for the parameters that we used in Theorem 8 such that all the necessary inequalities in the proof of Theorem 8 are satisfied.

Lemma 9 Let $\delta = (1 + \theta\sqrt{D})$. In order to satisfy all the inequalities

$$\frac{1}{n} - 2c\gamma\left(\frac{L-\mu}{L} + \gamma\mu\alpha\delta\right) \le 0, \tag{91}$$

$$\frac{1}{\tau} + 2\left(1 + \frac{1}{\alpha}\right)\delta c\gamma^2 L - \frac{1}{n} \le 0, \tag{92}$$

$$\left(\tfrac{1}{\tau} - \gamma\mu\right)\|w^t - w_*\|^2 + 2\gamma^2\theta\sqrt{D} + \gamma^2\theta^2 D + 2\gamma\theta\sqrt{D}\|w^t - w_*\| \le 0, \tag{93}$$

$$(1+\alpha)\gamma\delta - \frac{1}{L} \le 0, \tag{94}$$

it is sufficient to have

$$\begin{aligned} & \gamma = \tfrac{1}{2(1+\alpha)L}, \quad c = \tfrac{4}{n\gamma}, \quad \alpha = 16, \quad \tfrac{1}{\tau} = \min\left\{\tfrac{1}{2n}, \tfrac{\gamma\mu}{2}\right\}, \\ & \theta = \min\left\{\tfrac{1}{\sqrt{D}}, \tfrac{\mu\|w^t - w_*\|^2}{2\sqrt{D}\left(\frac{3}{34L} + 2\|w^t - w_*\|\right)}\right\}. \end{aligned} \tag{95}$$

Proof In what follows, we enumerate the steps required to satisfy all inequalities in the statement. Having lower and upper bounds on the value of δ is useful to this end. To impose an upper bound on δ, we assume

$$\theta \le \frac{1}{\sqrt{D}}, \tag{96}$$

resulting in

$$1 \le \delta \le 2. \tag{97}$$

For (94), using the upper bound from (97) we have

$$(1+\alpha)\gamma\delta - \tfrac{1}{L} \le 2(1+\alpha)\gamma - \tfrac{1}{L}. \tag{98}$$

Hence we can satisfy (94) by setting

$$\boxed{\gamma = \frac{1}{2(1+\alpha)L}}. \tag{99}$$

For (91) we consider the two cases of $\frac{L}{\mu} > 2$ and $\frac{L}{\mu} \le 2$. When $\frac{L}{\mu} > 2$,

$$\frac{1}{n} - 2c\gamma\left(\frac{L-\mu}{L} + \gamma\mu\alpha\delta\right) \le \frac{1}{n} - 2c\gamma\left(\frac{L-\mu}{L}\right) < \frac{1}{n} - c\gamma. \tag{100}$$

It therefore suffices to have

$$c \ge \tfrac{1}{n\gamma}. \tag{101}$$

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

$$\begin{aligned}\tfrac{1}{n} - 2c\gamma\left(\tfrac{L-\mu}{L} + \gamma\mu\alpha\delta\right) &\le \tfrac{1}{n} - 2c\gamma(\gamma\mu\alpha\delta) = \tfrac{1}{n} - 2c\gamma\left(\tfrac{1}{2(1+\alpha)L}\mu\alpha\delta\right)\\ &\le \tfrac{1}{n} - 2c\gamma\left(\tfrac{1}{2(1+\alpha)L}\mu\alpha\right) = \tfrac{1}{n} - c\gamma\left(\tfrac{\alpha}{1+\alpha}\tfrac{\mu}{L}\right)\\ &\le \tfrac{1}{n} - c\gamma\left(\tfrac{\alpha}{1+\alpha}\tfrac{1}{2}\right) \le \tfrac{1}{n} - \tfrac{c\gamma}{4},\end{aligned} \tag{102}$$

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

$$\alpha \ge 1, \tag{103}$$

resulting in $\frac{\alpha}{1+\alpha} \ge \frac{1}{2}$. Consequently, to satisfy (91) it suffices to have

$$c \ge \tfrac{4}{n\gamma}. \tag{104}$$

By combining (101) and (104), we set

$$\boxed{c = \frac{4}{n\gamma}}. \tag{105}$$

For (92) we require that

$$2\left(1+\tfrac{1}{\alpha}\right)\delta c\gamma^2 L - \tfrac{1}{n} < 0, \tag{106}$$

in which the inequality is strict (in order to assure $\frac{1}{\tau}$ is strictly positive). Plugging in the values of c from (105) and γ from (99), and the upper bound on δ from (97), we have

$$2\left(\tfrac{1+\alpha}{\alpha}\right)\delta\left(\tfrac{4}{n\gamma}\right)\gamma^2 L - \tfrac{1}{n} \le 4\left(\tfrac{1+\alpha}{\alpha}\right)\left(\tfrac{4}{n\gamma}\right)\gamma^2 L - \tfrac{1}{n} = \tfrac{8}{\alpha n} - \tfrac{1}{n}. \tag{107}$$

So, in order to satisfy (106), it suffices to have $\frac{8}{\alpha n} - \frac{1}{n} < 0$, resulting in $\alpha > 8$. We may therefore set

$$\boxed{\alpha = 16} \tag{108}$$

in order to leave room for $\frac{1}{\tau}$ to be larger in the next step. Note that this automatically satisfies (103). With this setting of α, the left-hand side of (92) is equal to

$$\tfrac{1}{\tau} + 2\left(1+\tfrac{1}{\alpha}\right)\delta c\gamma^2 L - \tfrac{1}{n} = \tfrac{1}{\tau} - \tfrac{1}{2n}. \tag{109}$$

To satisfy (92), it is sufficient to require that

$$\tfrac{1}{\tau} \le \tfrac{1}{2n}. \tag{110}$$

For (93) we need

$$\tfrac{1}{\tau} - \gamma\mu < 0, \tag{111}$$

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

$$\tfrac{1}{\tau} \le \tfrac{\gamma\mu}{2}. \tag{112}$$

By combining (110) and (112), we set

$$\boxed{\frac{1}{\tau} = \min\left\{\frac{1}{2n}, \frac{\gamma\mu}{2}\right\}}. \tag{113}$$

To satisfy (93), using (112), we can instead satisfy

$$\tfrac{-\gamma\mu}{2}\|w^t - w_*\|^2 + \gamma^2\theta\sqrt{D} + \gamma^2\theta^2 D + 2\gamma\theta\sqrt{D}\|w^t - w_*\| \le 0. \tag{114}$$

Cancelling a γ term and using the value of γ from (99), we would like to satisfy

$$\tfrac{\theta\sqrt{D}}{34L} + \tfrac{\theta^2 D}{34L} + 2\theta\sqrt{D}\|w^t - w_*\| \le \tfrac{\mu}{2}\|w^t - w_*\|^2. \tag{115}$$

Using (96), we have

$$\tfrac{\theta\sqrt{D}}{34L} + \tfrac{\theta^2 D}{34L} + 2\theta\sqrt{D}\|w^t - w_*\| \le \tfrac{\theta\sqrt{D}}{34L} + \tfrac{2\theta\sqrt{D}}{34L} + 2\theta\sqrt{D}\|w^t - w_*\|. \tag{116}$$

To satisfy (115), we may assume

$$\theta \le \tfrac{\mu\|w^t - w_*\|^2}{2\sqrt{D}\left(\frac{3}{34L} + 2\|w^t - w_*\|\right)}, \tag{117}$$

and (96). Therefore, we set

$$\boxed{\theta = \min\left\{\frac{1}{\sqrt{D}}, \frac{\mu\|w^t - w_*\|^2}{2\sqrt{D}\left(\frac{3}{34L} + 2\|w^t - w_*\|\right)}\right\}}. \tag{118}$$

This completes the proof of the lemma.

A.6. Proof of Theorem 9

As in Defazio et al. (2014, Corollary 1), we note that $c\left\|w^t - w_*\right\|^2 \leq \mathbb{T}^t$. Therefore, by chaining the expectations

$$\mathbb{E}\left[\left\|w^t - w_*\right\|^2\right] \leq C_0\left(1 - \tfrac{1}{\tau}\right)^t, \tag{119}$$

where

$$C_0 = \left\|w^0 - w_*\right\|^2 + \frac{1}{c}\left[f(w^0) - \left\langle f'(w_*), w^0 - w_*\right\rangle - f(w_*)\right]. \tag{120}$$

Therefore, we should have

$$t \geq \frac{\log\frac{1}{\epsilon} + \log C_0}{-\log\left(1 - \frac{1}{\tau}\right)}. \tag{121}$$

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

$$t \geq \tau\left(\log\tfrac{1}{\epsilon} + \log C_0\right). \tag{122}$$

From (89), we know that

$$\tau = \max\left\{2n, \tfrac{2}{\gamma\mu}\right\} \leq \max\left\{2n, \tfrac{68L}{\mu}\right\}, \tag{123}$$

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

$$t \geq \max\left\{2n, \tfrac{68L}{\mu}\right\}\left(\log\tfrac{1}{\epsilon} + \log C_0\right). \tag{124}$$

In Beck and Teboulle (2012) the authors prove that $\max_{y\in\mathscr{Y}}^{\beta} f_i(y, w)$ has Lipschitz continuous gradients with parameter $\beta D\Delta^2 + \ell$, so the function f^β has Lipschitz continuous gradients with parameter $L = \beta D\Delta^2 + \ell$. We also note that $C_0 = O(1/c) = O(n/L) = O(\frac{n}{\beta D\Delta^2 + \ell})$. Therefore, when f^β is sufficiently smooth, that is,

$$\frac{L}{\mu} = \frac{\beta D\Delta^2 + \ell}{\mu} \leq \frac{n}{34}, \tag{125}$$

we have

$$t = O\left(n\left(\log\tfrac{1}{\epsilon} + \log n - \log(\beta D\Delta^2 + \ell)\right)\right), \tag{126}$$

and otherwise

$$t = O\left(\frac{\beta D\Delta^2 + \ell}{\mu}\left(\log\frac{1}{\epsilon} + \log n - \log(\beta D\Delta^2 + \ell)\right)\right). \tag{127}$$

We can combine these two bounds into one to complete the proof:

$$t = O\left(\left(n + \frac{\beta D\Delta^2 + \ell}{\mu}\right)\left(\log\frac{1}{\epsilon} + \log n - \log(\beta D\Delta^2 + \ell)\right)\right). \tag{128}$$

A.7. Proof of Lemma 2

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

$$\max_{y\in\mathscr{Y}} \upsilon(y) \leq \max_{y\in\mathscr{Y}}^{\beta} \upsilon(y) \leq \max_{y\in\mathscr{Y}} \upsilon(y) + \frac{\log|\mathscr{Y}|}{\beta}, \tag{129}$$

for any function υ (Nielsen and Ke 2016). Using this inequality and the optimality of w_* and w_*^β, it follows that

$$f(w_*) \leq f(w_*^\beta) \leq f^\beta(w_*^\beta) \leq f^\beta(w_*) \leq f(w_*) + \frac{\log|\mathscr{Y}|}{\beta}. \tag{130}$$

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

$$f^\beta(w^t) - f^\beta(w_*^\beta) \leq \epsilon - \frac{\log|\mathscr{Y}|}{\beta}. \tag{131}$$

Therefore $f^\beta(w^t) - f(w_*) \leq \epsilon$, and using the fact that $f(w^t) \leq f^\beta(w^t)$, we can conclude that $f(w^t) - f(w_*) \leq \epsilon$, completing the proof.

A.8. Proof of Lemma 3

By the descent lemma (Nesterov 2013, Lemma 1.2.4), we have

$$f^\beta(w) - f^\beta(w_*) \leq \langle\nabla f^\beta(w_*), w - w_*\rangle + \tfrac{L}{2}\|w - w_*\|^2. \tag{132}$$

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

$$f^\beta(w) - f^\beta(w_*) \leq \tfrac{L}{2}\|w - w_*\|^2. \tag{133}$$

The result now follows from Theorem 8.

A.9. Proof of Theorem 10

Based on Lemma 2, it suffices to find a point at which the value of f^β is in the $\left(\epsilon - \frac{\log|\mathscr{Y}|}{2\beta}\right)$-neighbourhood of its optimal value. Using Lemma 3, we need

$$\mathbb{E}\left[f(w^t) - f(w_*)\right] \leq \frac{L}{2}C_0\left(1 - \frac{1}{\tau}\right)^t \leq \epsilon - \frac{\log|\mathscr{Y}|}{2\beta} = \frac{\epsilon}{2}. \tag{134}$$

Following the same steps as in Theorem 9, we conclude that

$$t \geq \frac{\log\frac{2}{\epsilon} + \log\frac{C_0 L}{2}}{-\log\left(1 - \frac{1}{\tau}\right)}. \tag{135}$$

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

$$t \geq \tau\left(\log\frac{2}{\epsilon} + \log\frac{C_0 L}{2}\right). \tag{136}$$

From (89), we know that

$$\tau = \max\left\{2n, \frac{2}{\gamma\mu}\right\} \leq \max\left\{2n, \frac{68L}{\mu}\right\}, \tag{137}$$

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

We recall that $\max_{y\in\mathscr{Y}}^{\beta} f_i(y,w)$ has Lipschitz continuous gradients with parameter $\beta D\Delta^2 + \ell$ (see Beck and Teboulle (2012)), so the function f^{β} has Lipschitz continuous gradients with parameter $L = \beta D\Delta^2 + \ell$. Hence,

$$\tau \leq \max\left\{2n, \frac{68(\beta D\Delta^2+\ell)}{\mu}\right\}. \tag{138}$$

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

$$\tau \leq \frac{68(\beta D\Delta^2+\ell)}{\mu}. \tag{139}$$

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

$$t \geq \frac{68(\frac{2\log|\mathscr{Y}|}{\epsilon} D\Delta^2+\ell)}{\mu}\left(\log\frac{2}{\epsilon} + \log\frac{C_0 L}{2}\right). \tag{140}$$

Note that $C_0 L = O(\frac{L}{\mu}) = O(n)$, so the time complexity is $t = O\Big((\frac{D\Delta^2\log|\mathscr{Y}|}{\mu\epsilon} + \frac{\ell}{\mu})\Big(\log\frac{1}{\epsilon} + \log n\Big)\Big)$, proving the claim.

A.10. Proof of Lemma 4

Each iteration of SAGA requires finding all partial derivatives of f_i for a random choice of i with precision. Since ϵ is small, based on (89), we have $\theta = O\left(\frac{1}{\sqrt{D}}\frac{\mu\epsilon}{\frac{1}{\beta D\Delta^2+\ell}+\sqrt{\epsilon}}\right)$. We also note that Δ, which is a bound on the partial derivatives of $\max_y f_i(y,w)$, is also a bound on the partial derivatives of $\max_y^{\beta} f_i(y,w)$, because $\nabla_w \max_y^{\beta} f_i(y,w) = \mathbb{E}(\nabla_w[f_i(Y_i,w)])$. From (129) we know that $F\log|\mathscr{Y}|$ is a bound on f_i^{β}. By replacing the values of θ, F, Δ in the cost of gradient calculation the result follows.

A.11. Proof of Theorem 11

To optimize f^{β} using SAGA with exact gradient evaluations, instead of the parameters from (89), we set

$$\gamma = \frac{1}{2(\mu n+L)}, \quad c = \frac{1}{2\gamma(1-\gamma\mu)n}, \quad \alpha = \frac{2\mu n+L}{L}, \quad \text{and} \quad \frac{1}{\tau} = \gamma\mu \tag{141}$$

according to Defazio et al. (2014), with no assignment of θ (since there are no additive errors after all). The rest of the proof follows the same steps as in the proof of Theorem 9, Theorem 10, and its corollary.

A.12. Proof of Theorem 12

Since ϵ is small, and β, M, and ℓ are fixed, we can simplify the result of Lemma 4 and conclude that each gradient could be estimated in $O\Big(\frac{D^{1.5}\beta F\sqrt{|\mathscr{Y}|}\log(|\mathscr{Y}|)\Delta}{\mu\epsilon(\beta D\Delta^2+\ell)}\log\frac{D}{\zeta}\Big)$ queries to the oracle for one of the f_i and the same order of other quantum gates. In T iterations of Q-SAGA, if $\zeta = 1/(4T)$, the probability of all gradient evaluations satisfying the additive θ upper bound is larger than $(1-\zeta)^T \geq 1 - \Big(\frac{1}{4T}\Big)T \geq \frac{3}{4}$. The result follows from Theorem 9.

A.13. Proof of Theorem 13

By replacing the value of β from Theorem 10, each gradient evaluation costs

$$O\left(\left(\frac{1}{D\Delta^2\frac{\log|\mathscr{Y}|}{\epsilon}+\ell} + \sqrt{\epsilon}\right)\frac{D^{1.5}F\sqrt{|\mathscr{Y}|}\log(|\mathscr{Y}|)\Delta}{\mu\epsilon}\log\frac{D}{\zeta}\right) \tag{142}$$

queries to the oracle for one of the f_i and the same order of other quantum gates according to Lemma 4. Using the fact that ϵ is small, this simplifies to $O\Big(\frac{D^{1.5}F\sqrt{|\mathscr{Y}|}\log(|\mathscr{Y}|)\Delta}{\mu\sqrt{\epsilon}}\log\frac{D}{\zeta}\Big)$. From Theorem 10, we know that we need $O\Big((\frac{D\Delta^2\log|\mathscr{Y}|}{\mu\epsilon} + \frac{\ell}{\mu})\Big(\log\frac{n}{\epsilon}\Big)\Big)$ gradient evaluations. Using the fact that ϵ is small, this simplifies to $O\Big((\frac{D\Delta^2\log|\mathscr{Y}|}{\mu\epsilon})\Big(\log\frac{n}{\epsilon}\Big)\Big)$. By multiplying the number of gradient estimations with the complexity of each, we get a total complexity of

$$O\left(\left(\frac{D^{1.5}F\sqrt{|\mathscr{Y}|}\log(|\mathscr{Y}|)\Delta}{\mu\sqrt{\epsilon}}\log\frac{D}{\zeta}\right)\left(\frac{D\Delta^2\log|\mathscr{Y}|}{\mu\epsilon}\right)\left(\log\frac{n}{\epsilon}\right)\right). \tag{143}$$

As with the proof of Theorem 12 we should satisfy a failure probability of at most $O(\frac{1}{T})$ and get a total complexity of $O\Big(\Big(\frac{D^{1.5}F\sqrt{|\mathscr{Y}|}\log(|\mathscr{Y}|)\Delta}{\mu\sqrt{\epsilon}}\log D\Big)\Big(\frac{D\Delta^2\log|\mathscr{Y}|}{\mu\epsilon}\Big)\Big(\log\frac{n}{\epsilon}\Big)\log\Big(\frac{D\Delta^2\log|\mathscr{Y}|}{\mu\epsilon}\log\frac{n}{\epsilon}\Big)\Big)$, which, after simplification, completes the proof.

Acknowledgements 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, Joran van Apeldoorn, András Gilyén, 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.

Declarations

Conflict of interest The authors declare no competing interests.

References

Albash T, Boixo S, Lidar DA, Zanardi P (2012) Quantum adiabatic markovian master equations. New J Phys 14(12):123016

Avron JE, Fraas M, Graf GM, Grech P (2012) Adiabatic theorems for generators of contracting evolutions. Commun Math Phys 314(1):163–191

Bachmann S, De Roeck W, Fraas M (2016) The adiabatic theorem for many-body quantum systems. Preprint

Bi W, Kwok J (2013) Efficient multi-label classification with many labels. In: International conference on machine learning, pp 405–413

Brandão FGSL, Kalev A, Li T, Yen-Yu Lin C, Svore KM, Wu X (2017) Quantum sdp solvers: Large speed-ups, optimality, and applications to quantum learning. arXiv:1710.02581

Brandao FGSL, Svore KM (2017) Quantum speed-ups for solving semidefinite programs. In: Foundations of computer science (FOCS), 2017 IEEE 58th annual symposium on. IEEE, pp 415–426

Beck A, Teboulle M (2012) Smoothing and first-order methods: A unified framework. SIAM J Optim 22(2):557–580

Crawford D, Levit A, Ghadermarzy N, Oberoi JS, Ronagh P (2016) Reinforcement learning using quantum boltzmann machines. arXiv:1612.05695

Chowdhury AN, Somma RD (2016) Quantum algorithms for gibbs sampling and hitting-time estimation. arXiv:1603.02940

Chen L-C, Schwing A, Yuille A, Urtasun R (2015) Learning deep structured models. In: International conference on machine learning, pp 1785–1794

Defazio A, Bach F, Lacoste-Julien S (2014) Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. In: Advances in neural information processing systems, pp 1646–1654

Durr C, Hoyer P (1996) A quantum algorithm for finding the minimum. arXiv:9607014

Daume HC, Marcu D (2006) Practical structured learning techniques for natural language processing. Citeseer

Giovannetti V, Lloyd S, Maccone L (2008) Quantum random access memory. Phys. Rev. Lett. 100(16):160501

Gao B, Pavel L (2017) On the properties of the softmax function with application in game theory and reinforcement learning. arXiv:1704.00805

Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on theory of computing, pp 212–219

Gimpel K, Smith NA (2010) Softmax-margin training for structured log-linear models

Huiskes MJ, Lew MS (2008) The MIR Flickr retrieval evaluation. In: Proceedings of the 1st ACM international conference on multimedia information retrieval. ACM, pp 39–43

Harvey NJA, Liaw C, Plan Y, Randhawa S (2018) Tight analyses for nonsmooth stochastic gradient descent. arXiv:1812.05217

Jiang Z (2020) Spatial structured prediction models: Applications, challenges, and techniques. IEEE Access 8:38714–38727

Jerrum MR, Valiant LG, Vazirani VV (1986) Random generation of combinatorial structures from a uniform distribution. Theor Comput Sci 43:169–188

Kingma DP, Ba J (2014) Adam: A method for stochastic optimization. arXiv:1412.6980

Kastoryano MJ, Brandao FGSL (2016) Quantum gibbs samplers: the commuting case. Commun Math Phys 344(3):915–957

Koller D, Friedman N (2009) Probabilistic graphical models: principles and techniques. MIT Press

Kim S, Nowozin S, Kohli P, Yoo CD (2011) Higher-order correlation clustering for image segmentation. In: Advances in neural information processing systems, pp 1530–1538

Karimi S, Ronagh P (2017) A subgradient approach for constrained binary optimization via quantum adiabatic evolution. Quantum Inf Process 16(8):185

Krizhevsky A, Sutskever I, Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems, pp 1097–1105

Levit A, Crawford D, Ghadermarzy N, Oberoi JS, Zahedinejad E, Ronagh P (2017) Free energy-based reinforcement learning using a quantum processor. arXiv:1706.00074

Lacoste-Julien S, Schmidt M, Bach F (2012) A simpler approach to obtaining an o (1/t) convergence rate for the projected stochastic subgradient method. arXiv:1212.2002

Lee Y-J, Mangasarian OL (2001) Ssvm: A smooth support vector machine for classification. Comput optim Appl 20(1):5–22

Matsubara S, Tamura H, Takatsu M, Yoo D, Vatankhahghadim B, Yamasaki H, Miyazawa T, Tsukamoto S, Watanabe Y, Takemoto K et al (2017) Ising-model optimizer with parallel-trial bit-sieve engine. In: Conference on complex, intelligent, and software intensive systems. Springer, pp 432–438

Nesterov Y (2005) Smooth minimization of nonsmooth functions. Mathematical programming 103(1):127–152

Nesterov Y (2013) Introductory lectures on convex optimization: A basic course, vol 87. Springer Science & Business Media, Berlin

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

Sebastian N, Gehler PV, Jancsary J, Lampert CH (2014) Advanced structured prediction. MIT Press

Nielsen F, Ke S (2016) Guaranteed bounds on information-theoretic measures of univariate mixtures using piecewise log-sum-exp inequalities. Entropy 18(12):442

Okuyama T, Hayashi M, Yamaoka M (2017) An ising computer based on simulated quantum annealing by path integral monte carlo method. In: Rebooting computing (ICRC), 2017 IEEE international conference on. IEEE, pp 1–6

Paszke A, Gross S, Chintala S, Chanan G, Yang E, Devito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic differentiation in pytorch

Poulin D, Wocjan P (2009) Sampling from the thermal quantum gibbs state and evaluating partition functions with a quantum computer. Phys Rev Lett 103(22):220502

Robbins H, Monro S (1985) A stochastic approximation method. In: Herbert robbins selected papers. Springer, pp 102–109

Rakhlin A, Shamir O, Sridharan K et al (2012) Making gradient descent optimal for strongly convex stochastic optimization. In: ICML, vol 12. Citeseer, pp 1571–1578

Ronagh P, Woods B, Iranmanesh E (2016) Solving constrained quadratic binary problems via quantum adiabatic evolution. Quantum Information & Computation 16(11-12):1029–1047

Sarandy MS, Lidar DA (2005) Adiabatic approximation in open quantum systems. Phys Rev A 71(1):012331

Schmidt M, Roux NL, Bach F (2017) Minimizing finite sums with the stochastic average gradient. Math Program 162(1-2):83–112

Sohn K, Lee H, Yan X (2015) Learning structured output representation using deep conditional generative models. Adv Neural Inf Process Syst 28:3483–3491

Shamir O, Zhang T (2013) Stochastic gradient descent for nonsmooth optimization Convergence results and optimal averaging schemes. In: International conference on machine learning, pp 71–79

Terhal BM, DiVincenzo DP (2000) Problem of equilibration and the computation of correlation functions on a quantum computer. Phys Rev A 61(2):022301

Temme K, Osborne TJ, Vollbrecht KG, Poulin D, Verstraete F (2011) Quantum metropolis sampling. Nature 471(7336):87

Takeda Y, Tamate S, Yamamoto Y, Takesue H, Inagaki T, Utsunomiya S (2017) Boltzmann sampling for an xy model using a non-degenerate optical parametric oscillator network. Quantum Science and Technology 3(1):014004

van Apeldoorn J, Gilyén A (2018) Improvements in quantum sdp-solving with applications. arXiv:1804.05058
van Apeldoorn J, Gilyén A (2019) Quantum algorithms for zero-sum games. arXiv:1904.03180
van Apeldoorn J, Gilyén A, Gribling S, De Wolf R (2017) Sdp-solvers: Quantum Better upper and lower bounds. In: Foundations of computer science (FOCS), 2017 IEEE 58th annual symposium on. IEEE, pp 403–414
Venuti LC, Albash T, Lidar DA, Zanardi P (2016) Adiabaticity in open quantum systems. Phys Rev A 93(3):032118
Wainwright MJ, Jordan MI et al (2008) Graphical models, exponential families, and variational inference. Foundations and Trends®; in Machine Learning 1(1–2):1–305
Wiebe N, Kapoor A, Svore KM (2014) Quantum deep learning. arXiv:1412.3489
Yu C-NJ, Joachims T (2009) Learning structural svms with latent variables. In: Proceedings of the 26th annual international conference on machine learning. ACM, pp 1169–1176
Yu CN (2011) Improved learning of structural support vector machines: training with latent variables and nonlinear kernels