PROCEEDINGS A

rspa.royalsocietypublishing.org

Review

Cite this article: Ciliberto C, Herbster M, Ialongo AD, Pontil M, Rocchetto A, Severini S, Wossnig L. 2018 Quantum machine learning: a classical perspective. *Proc. R. Soc. A* **474**: 20170551.
http://dx.doi.org/10.1098/rspa.2017.0551

Received: 8 August 2017
Accepted: 7 December 2017

Subject Areas:
artificial intelligence, quantum computing, quantum physics

Keywords:
quantum, machine learning, quantum computing

Author for correspondence:
Andrea Rocchetto
e-mail: andrea.rocchetto@spc.ox.ac.uk

Quantum machine learning: a classical perspective

Carlo Ciliberto[1], Mark Herbster[1], Alessandro Davide Ialongo[2,3], Massimiliano Pontil[1,4], Andrea Rocchetto[1,5], Simone Severini[1,6] and Leonard Wossnig[1,5,7]

[1]Department of Computer Science, University College London, London, UK
[2]Department of Engineering, University of Cambridge, Cambridge, UK
[3]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[4]Computational Statistics and Machine Learning, Istituto Italiano di Tecnologia, Genoa, Italy
[5]Department of Materials, University of Oxford, Oxford, UK
[6]Institute of Natural Sciences, Shanghai Jiao Tong University, Shanghai, People's Republic of China
[7]Theoretische Physik, ETH Zürich, Zurich, Switzerland

AR, 0000-0002-5216-0095

Recently, increased computational power and data availability, as well as algorithmic advances, have led machine learning (ML) techniques to impressive results in regression, classification, data generation and reinforcement learning tasks. Despite these successes, the proximity to the physical limits of chip fabrication alongside the increasing size of datasets is motivating a growing number of researchers to explore the possibility of harnessing the power of quantum computation to speed up classical ML algorithms. Here we review the literature in quantum ML and discuss perspectives for a mixed readership of classical ML and quantum computation experts. Particular emphasis will be placed on clarifying the limitations of quantum algorithms, how they compare with their best classical counterparts and why quantum resources are expected to provide advantages for learning problems. Learning in the presence of noise and certain computationally hard problems in ML are identified as promising directions

THE ROYAL SOCIETY PUBLISHING

for the field. Practical questions, such as how to upload classical data into quantum form, will also be addressed.

1. Introduction

In the last 20 years, owing to increased computational power and the availability of vast amounts of data, *machine learning* (ML) algorithms have achieved remarkable successes in tasks ranging from computer vision [1] to playing complex games such as Go [2]. However, this revolution is beginning to face increasing challenges. With the size of datasets constantly growing and Moore's law coming to an end [3], we might soon reach a point where the current computational tools will no longer be sufficient. Although tailored hardware architectures, like graphics processing units (GPUs) and tensor processing units (TPUs), can significantly improve performance, they might not offer a structural solution to the problem.

Quantum computation is a computational paradigm based on the laws of quantum mechanics. By carefully exploiting quantum effects such as interference or (potentially) entanglement, quantum computers can efficiently solve selected problems [4–6] that are believed to be hard for classical machines. This review covers the intersection of ML and quantum computation, also known as *quantum machine learning* (QML). The term QML has been used to denote different lines of research such as using ML techniques to analyse the output of quantum processes or the design of classical ML algorithms inspired by quantum structures. For the purpose of this review, we refer to QML solely to describe learning models that make use of quantum resources.

The goal of this review is to summarize the major advances in QML for a mixed audience of experts in ML and quantum computation and to serve as a bridge between the two communities. Most problems will be analysed under the lens of computational complexity, which is, possibly, a unifying language for both communities. We do not aim for completeness but rather discuss only the most relevant results in quantum algorithms for learning. For the interested reader there are now a number of resources covering QML in the broader sense of the term [7,8]. For an introduction to quantum algorithms, we refer to the reviews of Montanaro [9] and Bacon & Van Dam [10], while for ML to the books by Bishop [11] and Murphy [12].

Why should an ML expert be interested in quantum computation? And why are we expecting quantum computers to be useful in ML? We can offer two reasons. First, with an ever-growing amount of data, current ML systems are rapidly approaching the limits of classical computational models. In this sense, quantum algorithms offer faster solutions to process information for selected classes of problems. Second, results in quantum learning theory point, under certain assumptions, to a provable separation between classical and quantum learnability. This implies that hard classical problems might benefit significantly from the adoption of quantum-based computational paradigms. But optimism should come with a dose of scepticism. The known quantum algorithms for ML problems suffer from a number of caveats that limit their practical applicability and, to date, it is not yet possible to conclude that quantum methods will have a significant impact in ML. We will cover these caveats in detail and discuss how classical algorithms perform in the light of the same assumptions.

Quantum computation is a rapidly evolving field but the overarching question remains: when will we have a quantum computer? Although it is not within the scope of this review to present a time line for quantum computation it is worth noting that in recent years the worldwide effort to build a quantum computer has gained considerable momentum owing to the support of governments, corporations and academic institutions. It is now the consensus that general purpose quantum computation is within a 15 year time line [13].

The review is structured as follows. We start with §2 by providing some essential concepts in quantum computation for the reader with no prior knowledge of the field. In §3, we introduce the standard models of learning, their major challenges and how they can be addressed using quantum computation. Section 4 surveys results in quantum learning theory that justify why

we expect quantum computation to help in selected learning problems. We proceed in §5 by discussing how to access data with a quantum computer and how these models compare with parallel architectures. We continue by presenting different computational and mathematical techniques that find widespread application in ML, and can be accelerated with a quantum computer. More specifically, we survey quantum subroutines to speed up linear algebra (§6), sampling (§7) and optimization problems (§8). For each section, we discuss the asymptotic scaling of the classical and quantum subroutine and present some learning applications. Section 9 is dedicated to quantum neural networks. Even if neural networks are not a mathematical technique on their own, they are surveyed in a dedicated section due to their prominence in modern ML. The last two sections cover two promising applications of quantum computation in ML. In §10, we consider the case of learning under noise, while in §11 we discuss computationally hard problems in ML. We conclude with an outlook section.

2. Essential quantum computation

Quantum computing focuses on studying the problem of storing, processing and transferring information encoded in quantum mechanical systems. This *mode* of information is consequently called *quantum information*. The book by Nielsen & Chuang [14] is a standard introduction to the field. Loosely speaking, quantum computational models propose a probabilistic version of (time) reversible computation, i.e. computation in which the output is in one-to-one correspondence with the input. According to quantum theory, physical states are mathematically represented by density matrices, which are trace-one, positive-semi-definite matrices that generalize the concept of probability distributions. The logical states used by a quantum computational model are then identified with the physical states of the quantum system that implements them. A computation is executed reversibly by applying a sequence of unitary matrices to an initialized state. A probabilistic output is obtained according to the distribution encoded by the final density matrix.

In this framework, the fundamental unit of quantum information is the state of any quantum system with two degrees of freedom distinguishable by an observer, which then coincide with the usual logical values 0 and 1. This is called *qubit* and, for our purposes, it is a vector $\psi = \alpha_0 e_0 + \alpha_1 e_1$, where $\alpha_0, \alpha_1 \in \mathbb{C}$, $|\alpha_0|^2 + |\alpha_1|^2 = 1$ and, e_i denotes the ith standard basis vector. The values of a distribution on 0 and 1 are given by $\{|\alpha_1|^2, |\alpha_2|^2\}$. It is a basic fact that the information content of a qubit is equivalent to a single bit. Registers of multiple qubits are assembled with the use of a tensor product. Unitary matrices acting on a small number of qubits can then be interpreted as a generalization of logic gates. The induced dynamics is responsible for interference, a key ingredient of quantum computation. By exploiting interference effects, quantum computers are able to simultaneously evaluate a function on every point of its domain. Although the result of this computation is not immediately accessible to a classical observer, the possibility of using quantum dynamics to increase the probability of determining a given property of the function is promised to allow a quantum computer to solve some computational problems exponentially faster than classical devices. Although the true roots, and extents, of quantum speed-ups are still unclear, it is believed that structure, certain symmetries and non-classical correlations play an important role in the observed advantages [15,16].

In the context of the analysis of classical data, we can exploit the encoding of quantum information to efficiently represent classical probability distributions with exponentially many points. For instance, when $v = (v_1, \ldots, v_{2^n})$ is a probability vector of size 2^n, we can write an n-qubit state (register), $\psi = \sum_{i=1}^{2^n} \sqrt{v_i} e_i$.

Finally, when we use the term qubits, we always refer to idealized, error-free, objects. In practice, quantum states are extremely fragile and require extensive error correction to be shielded from the effects of noise. The theory of error correction guarantees that, if the physical errors are below a certain threshold, it is possible to correct the system efficiently. The theory of error correction is reviewed by Preskill [17]. We further discuss the types of error affecting quantum systems in §10.

(a) Comparing the performance of classical and quantum algorithms

Computational complexity studies the resources needed to perform a given computational task. One of the major goals of this theory is to classify problems according to their time complexity, which roughly corresponds to the number of elementary steps necessary to solve the problem as a function of the size of the input. The books by Papadimitrou [18] and Arora & Barak [19] provide extensive introductions.

We define *quantum speed-up* as the advantage in runtime obtained by a quantum algorithm over the classical methods for the same task. We quantify the runtime with the asymptotic scaling of the number of elementary operations used by the algorithm with respect to the size of the input. To compare the performance of algorithms, we use the computer science notation $\mathcal{O}(f(n))$, indicating that the asymptotic scaling of the algorithm is upper-bounded by a function $f(n)$ of the number n of parameters characterizing the problem, i.e. the size of the input. The notation $\tilde{\mathcal{O}}(f(n))$ ignores logarithmic factors.

In computational complexity theory, it is customary to analyse algorithms with respect to the number of queries to some efficiently implementable oracles, which can be either classical or quantum. This approach to analysing algorithms is called the *query model*. In the query model, an algorithm is said to be *efficient* if it queries the oracle a polynomial number of times. Throughout this review many speed-ups are obtained in the query model. A standard oracle for QML assumes that the classical datasets can be efficiently produced in quantum superposition. The quantum random access memory (QRAM) discussed in §5 is a possible implementation of this oracle.

3. Setting the problem: perspectives in machine learning

The term ML refers to a variety of statistical methods to analyse data. The prototypical goal is to infer, from a finite number of observations (the training data), the future behaviour of an unknown and possibly non-deterministic process (such as the dynamics of the stock market or the activation patterns in the human brain). In the last four decades, the field of ML has grown to such an extent that even providing a brief overview of the most prominent ideas and frameworks would require a review on its own. To this end, for the purpose of this review, we mainly focus on one of the most well-established and mature areas of research, namely supervised learning from the perspective of learning theory. To the reader interested in a more satisfying overview of ML we recommend, for instance, [12].

Learning theory aims to place the problem of learning from data on solid mathematical foundations. Typical questions that one asks in this setting are: How many examples are required to learn a given function? How much computational resource is required to perform a learning task? Depending on a number of assumptions about the data access model and on the goal of learning, it is possible to define different learning models. Two prominent ones are the *probably approximately correct* (*PAC*) framework developed by Valiant [20] and the *statistical learning theory* by Vapnik [21]. Here, a learner seeks to approximate an unknown function based on a training set of input–output pairs. Examples in the training set are assumed to be drawn from an unknown probability distribution and predictions are tested on points drawn from the same distribution. PAC and statistical learning theory model the efficiency of an estimator with two quantities: the sample complexity and the time complexity. The *sample complexity* is the minimum number of examples required to learn a function up to some approximation parameters and it is directly related to the capacity of the hypotheses space and the regularity of the data distribution; the *time complexity* corresponds to the runtime of the best learning algorithm. A learning algorithm is said to be efficient if its runtime is polynomial in the dimension of the elements of the domain of the function and inverse polynomial in the error parameters.

In these settings, the goal is to find a model that fits well a set of training examples but that, more importantly, guarantees good prediction performance on new observations. This latter property, also known as *generalization capability* of the learned model, is a key aspect separating ML from the standard optimization literature. Indeed, while data fitting is often approached as

an optimization problem in practice, the focus of ML is to design statistical estimators able to 'fit' well future examples. This question is typically addressed with so-called *regularization* techniques, which essentially limit the expressive power of the learned estimator in order to avoid overfitting the training dataset.

A variety of regularization strategies have been proposed in the literature, each adopting a different perspective on the problem (see [11,21,22] for an introduction on the main ideas). Among the most well-established approaches, it is worth mentioning those that directly impose constraints on the hypotheses class of candidate predictors (either in the form of hard constraints or as a penalty term on the model parameters, such as in Tikhonov regularization) or those that introduce the regularization effect by 'injecting' noise in the problem (see §10). These ideas have led to popular ML approaches currently used in practice, such as regularized least squares [23], Gaussian process (GP) regression and classification [24], logistic regression [11] and support vector machines (SVMs) [21] to name a few.

From a computational perspective, regularization-based methods leverage on optimization techniques to find a solution for the learning problem and typically consist of a sequence of standard linear algebra operations such as matrix multiplication and inversion. In particular, most classical algorithms, such as GPs or SVMs, require a number of operations comparable to that of inverting a square matrix that has size equal to the number N of examples in the training set. This leads, in general, to a time complexity of $\mathcal{O}(N^3)$ which can be improved depending on the sparsity and the conditioning of the specific optimization problem (see §6). However, as the size of modern datasets increases, the above methods are approaching the limits of their practical applicability.

Recently, alternative regularization strategies have been proposed to reduce the computational costs of learning. Instead of considering the optimization problem as a separate process from the statistical one, these methods hinge on the intuition that reducing the computational burden of the learning algorithm can be interpreted as a form of regularization on its own. For instance, *early stopping* approaches perform only a limited number of steps of an iterative optimization algorithm (such as gradient descent) to avoid overfitting the training set. This strategy clearly entails fewer operations (fewer number of steps) but can be shown theoretically to lead to the same generalization performance of approaches such as Tikhonov regularization [22]. A different approach, also known as *divide and conquer*, is based on the idea of distributing portions of the training data onto separate machines, each solving a smaller learning problem, and then combining individual predictors into a joint one. This strategy benefits computationally from both the parallelization and reduced dimension of distributed datasets and it has been shown to achieve the same statistical guarantees of classical methods under suitable partitions of the training data [25]. A third approach that has recently received significant attention from the ML community is based on the idea of constraining the learning problem to a small set of candidate predictors, obtained by randomly sampling directions in a larger, universal hypotheses space (namely a space dense in the space of continuous function). Depending on how such sampling is performed, different methods have been proposed, the most well-known being random features [26] and Nystrom approaches [27,28]. The smaller dimensionality of the hypotheses space automatically provides an improvement in computational complexity. It has been recently shown that it is possible to obtain equivalent generalization performance to classical methods also in these settings [29].

For all these methods, training times can be typically reduced from the $\mathcal{O}(N^3)$ of standard approaches to $\tilde{\mathcal{O}}(N^2)$ while keeping the statistical performance of the learned estimator essentially unaltered.

Because the size of modern datasets is constantly increasing, time complexities of the order of $\tilde{\mathcal{O}}(N^2)$ might still be too demanding for practical applications. In this regard, quantum computation could offer the potential to further improve the efficiency of such methods, allowing them to scale up significantly. Indeed, through a number of quantum algorithms for linear algebra, sampling and optimization techniques, we could in principle obtain up to exponential speed-ups over classical methods. However, as will be discussed in §6, current QML methods require fast memory access and particular data structures that might limit their applicability

in real settings. Nevertheless, as we will discuss in the following section, a number of results in quantum learning theory point, under specific assumptions, to a clear separation between classical and quantum learning paradigms in some specific settings.

4. 'Can we do better?': insights from quantum learning theory

Learning theorists have been interested in studying how quantum resources can affect the efficiency of a learner since the 1990s. Although different learning models have been translated into the quantum realm, here we focus on the quantum version of the PAC model. The reason for this choice is that, in this model, we have results for both the sample and the time complexity. For an extensive overview of the known results in quantum learning theory, we refer the reader to the review by Arunachalam & de Wolf [30].

The quantum PAC model was introduced in [31]. Here, it is assumed that the learner has access to a quantum computer and to an oracle that returns the training set in quantum superposition. In terms of sample complexity, it has been shown in a series of papers, which constantly improved the bounds until reaching provable optimality [32–35], that the quantum PAC model under an unknown distribution and standard PAC are equivalent up to constant factors. This implies that, in general, quantum mechanics does not help to reduce the amount of data required to perform a learning task. However, if one considers a different learning model, such as the exact learning framework developed by Angluin [36], it is possible to prove that quantum learners can be polynomially more efficient than classical learners in terms of the number of queries to the data oracle [32,37].

Although quantum and classical examples are equivalent up to constant factors when learning under general distributions, the quantum PAC model can offer advantages over its classical counterpart in terms of time complexity. One of the central problems studied in the classical literature is the learnability of disjunctive normal forms (DNFs). To date, the time complexity of the best algorithm for learning DNFs under an unknown distribution is exponential [38]. A number of assumptions can be made to relax the hardness of the problem. For instance, if the learner is provided with examples drawn from the uniform distribution, then the runtime of the best learner becomes quasi-polynomial [39]. For the sake of completeness, we note that the methods presented in §3, like SVMs, have been shown to not be able to learn efficiently DNF formulae [40,41]. The learnability of DNF formulae has also been studied in the quantum PAC model [31]. Here DNFs have been shown to be efficiently learnable under the uniform distribution. This quantum speed-up is obtained through an efficient algorithm [42] that allows exponentially faster sampling from the probability distribution described by the coefficients of a Boolean Fourier transform. Interestingly, DNF formulae can be shown to be efficiently learnable under noise. We will return to this point in §10.

Another case where it is believed that learning can be performed efficiently only when the learner has access to quantum resources is based on a class of functions developed by Kearns & Valiant [43]. This class is provably hard to learn under the assumption that factoring Blum integers is also hard (an assumption widely believed to be true; for a brief introduction to the concept of hardness in computational complexity, see §11). Servedio & Gortler [32] noted that owing to Shor's quantum factoring algorithm [4] this class of functions can be learned efficiently in the quantum PAC model.

The results coming from the quantum learning theory literature show that by carefully exploiting quantum mechanical effects, depending on the type of learning model considered, it is possible to have a better generalization error (i.e. we can learn with fewer examples) or we can learn functions that would otherwise be hard for classical learners.

5. Data access, communication and parallelism

One of the roots of the speed-ups theorized in quantum computation is the ability to process information in quantum superposition [14,44]. Because ML is ultimately about analysing vast

amounts of data, it is important to address the question of how data are turned into quantum superposition. We distinguish between two types of algorithms: those that operate on quantum data (i.e. data that are the output of a quantum process, for example a quantum chemistry problem) and those that seek to process data stored in a classical memory. The first case is ideal for QML. The data are ready to be analysed and we do not have to spend computational resources to convert the data into quantum form. The second case is more elaborate as it requires a procedure that encodes the classical information into a quantum state. As we will see, the computational cost of this operation is particularly relevant to determine whether we can obtain quantum speed-ups in ML for classical data.

Let us assume that one wants to process N d-dimensional classical vectors with a quantum algorithm. The *quantum random access memory* (QRAM) [45,46] is a quantum device that can encode in superposition N d-dimensional vectors into $\log(Nd)$ qubits in $\mathcal{O}(\log(Nd))$ time by making use of the so-called 'bucket-brigade' architecture. The idea is to use a tree structure where the Nd leaves contain the entries of the N vectors in $\mathbb{R}^d$. The QRAM, with a runtime of $\mathcal{O}(\log(Nd))$, can return a classical vector in quantum superposition efficiently. However, the number of physical resources it requires scales as $\mathcal{O}(Nd)$. As we will see, this exponential scaling (with respect to the number of qubits) has been used to question whether the QRAM can be built in an experimental setting or whether it can provide a genuine computational advantage [7,47]. Fundamentally, the issue can be related to whether the exponential number of components needs to be continuously 'active'. The proponents of the QRAM [45,46] claim that only $\mathcal{O}(\log(Nd))$ components need to be active, while the others can be considered as 'non-active' and error-free. Whether this assumption holds in an experimental setting is unclear [48]. We now proceed to discuss its implications.

The first issue that appears with QRAM is whether all the components require to be error-corrected (we briefly discuss errors in quantum computation in §10). Indeed, if the exponential physical resources required full error correction, then it would be impractical to build the device in an experimental setting. Arunachalam *et al.* [48] addressed this question and showed that, given a certain error model, algorithms that require to query the memory a polynomial number of times (like the quantum linear system algorithm presented in §6) might not require fault-tolerant components. However, for superpolynomial query algorithms, like Grover's search [49] (a subroutine required, for example, in some of the quantum methods for training-restricted Boltzmann machines discussed in §9), the QRAM requires error-corrected components.

A second problem related to the exponential number of resources in an active memory has been raised by Aaronson [47] and by Steiger & Troyer [50]. The authors argue that the only fair comparison of a system which requires an exponential number of resources is with a parallel architecture with a similar amount of processors. In this case, many linear algebra routines, including solving linear systems and singular value decomposition, can be solved in logarithmic time [51].

A third caveat of the QRAM is the requirement of having data distributed in a relatively uniform manner over the quantum register. As pointed out in [47,52], if that was not the case, the QRAM would violate the search lower bounds proved in [53]. In the case of non-uniformly distributed data, the QRAM is no longer efficient and takes $\mathcal{O}(\sqrt{N})$ to turn the classical dataset into quantum superposition.

As a last comment on the QRAM, the possibility of loading the data in logarithmic time, when the size of the data is considerable, can be controversial due to speed of communication arguments. In fact, as suggested in [47], latency can play a role in big memory structures. In particular, a lower bound on the distance which the information has to travel implies a lower bound on latency, due to considerations on the limits set by the speed of light. In a three-dimensional space, these are given by $\mathcal{O}(\sqrt[3]{Nd})$. In practice, these considerations will only dominate if the amount of memory is extremely large but, because in QML we aim at datasets that surpass the current capability of classical computers, this bound is a potential caveat.

In conclusion, the QRAM allows data to be uploaded efficiently but might be hard to implement experimentally or might not allow a genuine quantum advantage if we take into account all the required resources. Notably, the fast data access guaranteed by the QRAM is only required for QLM algorithms that run in sublinear time. Although many known QML algorithms run in sublinear time, quantum learning theory suggests that for some classically hard problems quantum resources might give exponential advantages. In this case, a memory structure that can prepare a quantum superposition in polynomial time (i.e. in $\mathcal{O}(Nd)$) can still be sufficient to maintain a quantum speed-up compared with the classical runtime. We will discuss the hard classical learning problem in §11.

Finally, we note that, although the QRAM, due to its generality, is the most widely used memory structure in QML algorithms, other protocols to encode classical data in superposition exist. For example, a technique developed by Grover & Rudolph [54] allows one to generate a quantum superposition that encodes an approximate version of a classical probability distribution provided its density is efficiently integrable.

6. Fast linear algebra with quantum mechanics

A significant number of methods in the QML literature are based on fast quantum algorithms for linear algebra. In this section, we present the two main quantum subroutines for linear algebra: a quantum algorithm for matrix inversion and a quantum algorithm for singular value decomposition. We summarize the major applications of these techniques to ML problems and how they compare with classical and parallel implementations in table 1.

(a) Fast matrix inversion: the quantum linear system algorithm

Solving linear systems of equations is a ubiquitous problem in ML. As discussed in §3, many learning problems, such as GPs or SVMs, require the inversion of a matrix. For a system of linear equations $Ax = b$ with $A \in \mathbb{R}^{N\times N}$ and $x, b \in \mathbb{R}^N$, the best classical algorithm has a runtime of $\mathcal{O}(N^{2.373})$ [67]. However, due to a large pre-factor, the algorithm is not used in practice. Standard methods, for example, based on QR-factorization take $\mathcal{O}(N^3)$ steps [68].

The *quantum linear system algorithm* (QLSA) [57], also known as HHL after the three authors Harrow, Hassidim and Lloyd, promises to solve the problem in $\tilde{\mathcal{O}}(\log(N)\kappa^2 s^2/\epsilon)$, where κ is the condition number (defined to be the ratio of the largest to the smallest eigenvalue), s is the sparsity or the maximum number of non-zero entries in a row and column of A, and ϵ is the precision to which the solution is approximated. The precision is defined as the distance of the solution vector $\bar{a}$ to the true result a, which is given by $\|\bar{a} - a\| = \sqrt{1 - 2\mathrm{Re}\langle\bar{a}, a\rangle} \leq \epsilon$. Ambainis [69] and Childs *et al.* [70] improved the runtime dependency of the algorithm in κ and s to linear and the dependency in ϵ to poly-logarithmic.

Although the QLSA solves matrix inversion in logarithmic time, a number of caveats might limit its applicability to practical problems [47]. First, the QLSA requires the matrix A to be sparse. Second, the classical data must be loaded in quantum superposition in logarithmic time. Third, the output of the algorithm is not x itself but a quantum state that encodes the entries of x in superposition. Fourth, the condition number must scale at most sublinearly with N. An interesting problem that satisfies these requirements is discussed in [71].

Recently, Wossnig *et al.* [72] addressed the first caveat. By using a quantum walk-based approach, the authors derived an algorithm that scales as $\tilde{\mathcal{O}}(\kappa^2\|A\|_F \log N/\epsilon)$ and can also be applied to dense matrices (however, in this case, the speed-up is only quadratic because $\|A\|_F = \sqrt{N}$). This result has been improved in [60], currently the best known lower bound for matrices with this property.

The second caveat inherits the same issues of the QRAM discussed in §5: it is an open question whether we can practically load classical data in quantum superposition in logarithmic time.

The third caveat is a common pitfall of quantum algorithms. As pointed out by Childs [73] and Aaronson [47], in order to retrieve classical information from the quantum state, we need at

Table 1. Quantum linear algebra algorithms and their ML applications. When carefully compared with classical versions that take into account the same caveats, quantum algorithms might lose their advantages. *C*, *Q* and *P* indicate, respectively, the asymptotic computational complexity for classical, quantum and parallel computation. We remind the reader that, to date, memory and bandwidth limits in the communication between processors make the implementation of certain parallel algorithms unrealistic. We remark that asymptotic scalings are only an indication of potential runtime differences and solely by benchmarking the algorithms on quantum hardware we will obtain clear insights on their performance. Given an $N \times N$-dimensional matrix A, we denote by k the number of singular values that are computed by the algorithm, by s the sparsity and by κ the condition number. For approximation algorithms ϵ is an approximation parameter. In other cases, it denotes the numerical precision. Classical algorithms return the whole solution vector. Quantum algorithms return a quantum state; in order to extract the classical vector, one needs $\mathcal{O}(N)$ copies on the state.

problem	solving linear system of equations	singular value estimation
scaling	$C : \tilde{\mathcal{O}}(\sqrt{\kappa}\,\hat{s}N \log(1/\epsilon))$ [55][a]	$C : \mathcal{O}(k^2 N \log(1/\delta)/\epsilon)$ [56][d]
	$Q : \tilde{\mathcal{O}}(s^2\kappa^2 \log(N)/\epsilon)$ [57][b]	$Q : \mathcal{O}(\log(N)\epsilon^{-3})$[58]
	$P : \mathcal{O}(\log^2(N) \log(1/\epsilon))$ [51][c]	$P : \mathcal{O}(\log^2(N) \log(1/\epsilon))$ [51][e]
applications	least-squares SVM [59]	recommendation systems [60]
	GP regression [61]	linear regression [62]
	Kernel least squares [62]	principal component analysis [58][f]

[a] An approximate algorithm that can be applied to dense matrices. Here $\hat{s}N$ is the number of entries in the matrix A and $\hat{s}$ alludes to the number of entries per row.

[b] Exact but does not output the solution vector and works only for sparse matrices (more details can be found in §6).

[c] Requires $\mathcal{O}(N^4)$ parallel units and is numerically unstable due to high sensitivity to rounding errors. Stable algorithms such as Gaussian elimination with pivoting or parallel *QR*-decomposition require $\mathcal{O}(N)$ time using $\mathcal{O}(N^2)$ computational units [63].

[d] An approximate algorithm which returns a rank k-approximation with probability $1 - \delta$ and has an additional error $\epsilon \|A\|_F$. Exacts methods for an $N \times M$ matrix scale with $\min\{MN^2, NM^2\}$.

[e] Calculates SVD by computing the eigenvalue decomposition of the symmetric matrix AA^{T}.

[f] Works on dense matrices that are low-rank approximable. Finally, we note that there exist efficient, classical, parallel algorithms for sparse systems, where $s = \mathcal{O}(\log N)$ [64,65]. Probabilistic numerical linear algebra also allows selected problems to be solved more efficiently and, under specific assumptions, even in linear time and with bounded error [66].

least a number of measurements that are proportional to N. This would destroy every exponential speed-up. One way forward is to use the QLSA only to compute certain features of the classical vector, which can be extracted efficiently using quantum mechanics, for example the expected value $x^{\mathrm{T}}Ax$ of a matrix A. A number of other possible applications are discussed in [47].

It is then natural to question how quantum algorithms compare with their classical analogues after all the caveats have been taken into account. For example, it is possible to show that calculating an expectation value of the form $x^{\mathrm{T}}Ax$ can be done in time linear in the sparsity of the matrix A, using classical sampling methods. Furthermore, conjugated gradient descent can obtain the full solution of the linear system (also when A is not sparse) in linear time in the dimensionality and, in most cases, the sparsity of A [55]. We present a general comparison of the asymptotic scalings of classical, quantum and parallel algorithms for linear algebra and their major applications in ML in table 1.

Comparing algorithms based on their worst case running time may not be the right approach when considering their practical applicability, as is commonly done in ML. Indeed, despite its worst case running time, an algorithm solving a given problem will often terminate much faster: average-case complexity can be much lower than worst case. Furthermore, *smoothed analysis* [74,75] provides a framework for studying the time performance of an algorithm in the presence of realistic input noise distributions. This gives another way to quantify the complexity of algorithms. To date, no quantum algorithm has been analysed in the smoothed analysis framework.

Statistical considerations can also lead to interesting insights on the computational hardness of a learning problem. Kernel-regularized least squares provide a good example. Under standard technical assumptions on the target function of the learning problem, computational regularization methods for kernel-regularized least squares [22,29,76] (see §3) achieve the optimal learning rate of $\epsilon = \mathcal{O}(N^{-1/2})$ while requiring only $\tilde{\mathcal{O}}(N^2)$ operations. With optimal learning rates we mean that any learning algorithm cannot achieve better prediction performance (uniformly) on the class of problems considered. Interestingly, such assumptions also allow us to derive estimates for the condition number of the kernel matrix to be of order $\kappa = \mathcal{O}(N^{1/2})$ [77]. The corresponding quantum scaling for the inversion of the kernel matrix is $\tilde{\mathcal{O}}(N^2)$ and it is therefore comparable to that of computational regularization methods implementable on classical machines (which, in addition, provide the full solution vector).

Finally, it is worth comparing the QLSA with classical parallel methods for matrix inversion. In the parallel model of computation [78], inverting an $N \times N$ matrix takes $\mathcal{O}(\log^2(N))$ computational steps using a number of processors which are of order poly(N) (a crude upper-bound of $\mathcal{O}(N^4)$ is given by [51]). Although the parallel model of computation does not resemble the actual behaviour of parallel machines, it can be a fair comparison considering that quantum computers might also face connectivity issues and hence communication overheads among the qubits. In particular when exponentially large amounts of error-corrected qubits are required, as with the QRAM, it is likely that latency issues arise.

To conclude, the QLSA is a logarithmic time quantum algorithm for matrix inversion, a task arising in many learning problems. However, a number of caveats that include the requirement of a logarithmic access time to the memory and the impossibility of retrieving the solution vector with one measurement lead to the question of whether classical or parallel algorithms that make use of the same assumptions obtain similar, or better, runtimes. In this respect, experimental implementations will greatly contribute to assessing the true potential of these methods in realistic scenarios.

(b) Quantum singular value estimation

The *singular value decomposition* (SVD) of an $M \times N$, rank r matrix A is a factorization of the form $A = U\Sigma V^\dagger$, where U and V are, respectively, $M \times M$ and $N \times N$ unitary matrices and Σ is an $M \times N$ diagonal matrix with r positive entries $\sigma_1, \ldots, \sigma_r$ which are called the *singular values* of A.

Singular value estimation is a fundamental tool in many computational problems and applications ranging from matrix inversion for linear regression to matrix approximation [68]. It is also of particular interest for problems of dimensionality reduction such as *principal component analysis* (PCA) [79]. Classically, finding such a decomposition is computationally expensive, and for $M > N$ it takes $\mathcal{O}(MN^2)$ [80].

Prakash and Kerenidis [52,60] introduced the *quantum singular value estimation* (QSVE) algorithm, based on Szegedy's work on quantum walks [81], which runs in time $\mathcal{O}(\|A\|_F \log MN/\epsilon)$. Their algorithm returns an estimate of the singular values $\tilde{\sigma}_i$ such that $|\sigma_i - \tilde{\sigma}_i| \leq \epsilon$. As for the QLSA, the QSVE algorithm outputs the singular values in quantum superposition. As such, in order to read out all the r-values, the algorithm must be run $\mathcal{O}(N \log N)$ times, thus destroying any exponential speed-up. However, it is still possible to construct useful applications of the QSVE algorithm. For example, Kerenidis & Prakash [60] proposed a recommendation system which runs in $\mathcal{O}(\text{poly}(r)\text{poly}\log(MN))$ (assuming a good r-rank approximation of the preference matrix).

We note that the QSVE algorithm requires an oracle that can prepare quantum states that encode the rows and the columns of the matrix A in poly-logarithmic time. It is possible to implement this oracle with the QRAM, and hence it will inherit the caveats discussed in §5.

An alternative method for QSVE has been proposed by Lloyd *et al.* [58]. The scaling of this algorithm is quadratically worse in ϵ but the requirements on the memory structure are less stringent than in [60]. This is advantageous in some applications, like analysing the principal components of kernel matrices [59].

7. Quantum methods for sampling

Many learning problems of practical interest, for example exact inference in graphical models, are intractable with exact methods. We discuss in detail hard learning problems in §11. Sampling methods are a common technique to compute approximations to these intractable quantities [82]. There is a rich literature on sampling methods [83–87]. The most commonly used ones are Monte Carlo methods and in particular the *Markov chain Monte Carlo* (MCMC). The quantum algorithms discussed in this section are devoted to speed up MCMC methods.

MCMC methods [88], like Gibbs sampling or the Metropolis algorithm, allow sampling from a probability distribution Π defined over a state space using a Markov chain that after a number of steps converges to the desired distribution (in practice one will only reach a distribution which is ϵ-close). The number of steps τ required to converge to Π is referred to as the *mixing time*. Estimating the mixing time can be reduced to bounding the spectral gap δ, which is the distance between the largest and the second largest eigenvalue of a stochastic map that evolves the Markov chain. The mixing time satisfies the inequality $\tau \geq (1/2\delta)\log(2\epsilon)^{-1}$ and it is possible to show [88, 89] that, for the classical MCMC algorithm, τ is of the order of $\mathcal{O}(1/(\delta \log(1/\Pi^*)))$, where Π^* is the minimum value of Π.

Recently, there has been a significant interest in quantum algorithms that allow speeding up of the simulations of the stochastic processes used in MCMC. A common feature of these algorithms is a quadratic speed-up in terms of spectral gap, inverse temperature, desired precision or the hitting time. Advances in this field include algorithms for thermal Gibbs state preparation [90–96] which provide polynomial speed-ups in various parameters, such as the spectral gap. Other methods have introduced the concept of quantum hitting time of a quantum walk [81,96–100]. In this framework, it is possible to obtain a polynomial speed-up with respect to most classical variants (this can be exponential for the hitting time). A number of other algorithms accelerate classical Monte Carlo methods applied to the estimation of quantities such as expectation values and partition functions, which play a major role in physics [9,96,101].

8. Quantum optimization

As discussed in §3, optimization methods are a fundamental building block of many ML algorithms. Quantum computation provides tools to solve two broad classes of optimization problems: semi-definite programming and constraint satisfaction problems.

(a) Quantum algorithms for semi-definite programming

Semi-definite programming [102,103] is a framework for solving certain types of convex optimization problems. Semi-definite programs (SDPs) find widespread applications in ML [104–106]. In an SDP, the objective is to minimize a linear function of an $N \times N$ positive-semi-definite matrix X over an affine space defined by a set of m constraints. The best known classical SDP solvers [107] run in time $\mathcal{O}(m(m^2 + n^\omega + mns)\log^{O(1)}(mnR/\epsilon))$, where ϵ is an approximation parameter, $\omega \in [2, 2.373)$ is the optimal exponent for matrix multiplication, s is the sparsity of A and R is a bound on the trace of an optimal X.

Based on a classical algorithm to solve SDPs by Arora & Kale [108], which has a runtime of $\tilde{\mathcal{O}}(nms(Rr/\epsilon)^4 + ns(Rr/\epsilon)^7)$, where r is an upper bound on the sum of the entries of the optimal solution to the dual problem, in 2016 Brandão & Svore [109] developed a quantum algorithm for SDPs that is quadratically faster in m and n. The dependence on the error parameters of this result has been improved in [110]. In this work, the authors obtain a final scaling of $\tilde{\mathcal{O}}\left(\sqrt{mn}s^2(Rr/\epsilon)^8\right)$.

The main problem of these quantum algorithms is that the dependence on R, r, s and $1/\epsilon$ is considerably worse than in [108]. This quantum algorithm thus provides a speed-up only in situations where $R, r, s, 1/\epsilon$ are fairly small compared with mn and, to date, it is unclear whether there are interesting examples of SDPs with these features (for more details, see [110]).

(b) Quantum algorithms for constraint satisfaction problems

In a *constraint satisfaction problem* (CSP), we are given a set of variables, a collection of constraints, and a list of possible assignments to each variable [111]. The task is to find values of the variables that satisfy every constraint. This setting prompts exact and approximate cases. For many families of CSPs efficient algorithms are unlikely to exist. Two quantum algorithms are known for CSPs: the quantum approximate optimization algorithm and the quantum adiabatic algorithm (QAA). Owing to its generality and a profoundly different way of exploiting quantum evolution, the latter algorithm is also regarded as an independent computational model called *adiabatic quantum computation* (AQC). We will provide a brief introduction to AQC in the following paragraphs.

(i) The quantum approximate optimization algorithm

The *quantum approximate optimization algorithm* (QAOA), developed in 2014 by Farhi, Goldstone and Gutman, is a quantum method to approximate CSPs [112]. The algorithm depends on an integer parameter $p \geq 1$ and the approximation improves as p increases. For small values of p, the QAOA can be implemented on a shallow circuit. As argued in [113], this feature makes the QAOA a good candidate for first-generation quantum hardware.

For certain combinatorial optimization problems, the QAOA can give approximation ratios that are better than what can be achieved by random sampling [112] but worse than the best classical solvers. In specific instances of MAX-kXOR the QAOA with $p = 1$ was believed to outperform the best classical solver [114]. This sparked further research in the classical community and Barak *et al.* [115] designed a classical algorithm able to outperform the quantum scaling.

(ii) The quantum adiabatic algorithm

The QAA [116] is an optimization method that operates in the adiabatic model of quantum computation. The QAA can be thought of as a quantum analogue of simulated annealing [117]. The algorithm encodes the solution to a computational problem in the unknown ground state of a quantum system (usually an Ising spin glass Hamiltonian). By starting off in the ground state of a known and easy to implement Hamiltonian, the QAA exploits a slow, time-dependent Hamiltonian dynamics to obtain the solution to the problem. If the evolution is slow enough, the quantum adiabatic theorem [118] guarantees that the system will reach the desired ground state. If the energy barriers have specific configurations (e.g. tall and narrow) and the energy gap between the ground state and the first excited state remains large enough, the algorithm can obtain significant speed-ups over classical simulated annealing [119,120].

Although QAA and AQC are usually considered synonyms in the literature, we shall keep the two concepts distinct to mark the difference between the computational model and the algorithm. Another name which is frequently used in the literature as a synonym of QAA and AQC is *quantum annealing* (QA). Although there is not a clear consensus in the literature over the differences between these three concepts, we refer to QA only when the adiabatic evolution occurs at non-zero temperature.

Aharonov *et al.* [121] showed that AQC is universal for quantum computation, i.e. it is capable of solving any computational problem that can be solved by a quantum computer. Although it is clearly possible to encode NP-hard problems [122], quantum mechanics is not expected to solve these in polynomial time (however, the scaling constants of the quantum algorithm might be smaller). Finally, it is important to note that the adiabatic algorithm lacks worst case upper bounds on its runtime. Its performance has been analysed with numerical experiments [123–128]. However, these are limited to small-size systems and only running the algorithm on actual hardware will be able to determine the strength of this approach.

9. Quantum neural networks

The term *artificial neural network* (ANN) denotes a variety of models which have been widely applied in classification, regression, compression, generative modelling and statistical inference. Their unifying characteristic is the alternation of linear operations with, usually preselected, nonlinear transformations (e.g. sigmoid functions) in a potentially hierarchical fashion.

While in the last decade neural networks have proved successful in many applications, fundamental questions concerning their success remain largely unanswered. Are there any formal guarantees concerning their optimization and the predictions they return? How do they achieve good generalization performance despite the capacity to completely overfit the training data?

ANNs have been extensively studied in the QML literature. The major research trends have focused on accelerating the training of classical models and on the development of networks where all the constituent elements, from single neurons to training algorithms, are executed on a quantum computer (a so-called *quantum neural network*). The first works on quantum neural networks appeared in the 1990s [129] and a number of papers have been published on the topic. However, it is worth noticing that the field has not reached a level of scientific maturity comparable to the other areas of QML discussed in this review. Possible reasons for the difficulties encountered in making progress in this area can be traced to the inherent differences between the linearity of quantum mechanics and the critical role played by nonlinear elements in ANNs or the fast developments occurring in the field of classical ANNs.

The literature on accelerated training of neural networks using quantum resources has mainly focused on *restricted Boltzmann machines* (RBMs). RBMs [130] are generative models (i.e. models that allow new observational data to be generated based on prior observations) that are particularly apt to be studied from a quantum perspective due to their strong connections with the Ising model. It has been shown that computing the log-likelihood and sampling from an RBM is computationally hard [131]. MCMC methods are the standard techniques used to overcome these difficulties. Nonetheless, even with MCMC the cost of drawing samples can be high [132] for models with several neurons. Quantum resources can help to reduce the training cost.

There are two main classes of quantum techniques to train RBMs. The first one is based on methods from quantum linear algebra (discussed in §6) and quantum sampling (discussed in §7). Wiebe *et al.* [133] developed two algorithms to efficiently train an RBM based on amplitude amplification [134] and quantum Gibbs sampling. These obtain a quadratic improvement in the number of examples required to train the RBM, but the scaling of the algorithm is quadratically worse in the number of edges than in contrastive divergence [135]. A further advantage of the approach proposed in [133] is that it can be used to train full Boltzmann machines (a classical version of this algorithm has also been proposed [136]). A full Boltzmann machine is a type of Boltzmann machine where the neurons correspond to the nodes of a complete graph (i.e. they are fully connected). Although full Boltzmann machines have a higher number of parameters with respect to RBMs, they are not used in practice due to the high computational cost of training and, to date, the true potential of large-scale, full Boltzmann machines is not known.

The second direction for training RBMs is based on QA, a model of quantum computation that encodes problems in the energy function of an Ising model (QA was discussed in §8). Specifically, [137,138] make use of the spin configurations generated by a quantum annealer to draw Gibbs samples that can then be used to train an RBM. These types of physical implementations of RBMs present several challenges. Benedetti *et al.* [139] pointed out the difficulties in determining the effective temperature of the physical machine. To overcome this problem, they introduced an algorithm to estimate the effective temperature and benchmarked the performance of a physical device on some simple problems. A second critical analysis of quantum training of RBMs was conducted by Dumoulin *et al.* [132]. Here, the authors showed with numerical models how the limitations that the first-generation quantum machines are likely

to have, in terms of noise, connectivity and parameter tuning, severely limit the applicability of quantum methods.

A hybrid approach between training ANNs and a fully quantum neural network is the quantum Boltzmann machine proposed by Amin *et al.* [140]. In this model, the standard RBM energy function gains a purely quantum term (i.e. off diagonal) that, according to the authors, allows a richer class of problems to be modelled (i.e. problems that would otherwise be hard to model classically such as quantum states). Whether these models can provide any advantage for classical tasks is unknown. Kieferova & Wiebe [141] suggest that quantum Boltzmann machines could provide advantages for tasks like reconstructing the density matrix of a quantum state from a set of measurements (this operation is known in the quantum information literature as *quantum state tomography*).

Although there is no consensus on the defining features of a quantum ANN, the last two decades have seen a variety of works that attempted to build networks whose elements and updating rules are based solely on the laws of quantum mechanics. The review by Schuld *et al.* [142] provides a critical overview of the different strategies employed to build a quantum ANN and highlights how most of the approaches do not meet the requirements of what can be reasonably defined as a quantum ANN. In particular, most of the papers surveyed by Schuld *et al.* failed to reproduce basic features of ANNs (for example, the attractor dynamics in Hopfield networks). On the other hand, it can be argued that the single greatest challenge to a quantum ANN is that the quantum mechanics is linear but ANNs require nonlinearities [143].

Recently, two similar proposals [144,145] have overcome the problem of modelling nonlinearities by using measurements and introducing a number of overhead qubits in the input and output of each node of the network. However, these models still lack some important features of a fully quantum ANN. For example, the model parameters remain classical, and it is not possible to prove that the models can converge with a polynomial number of iterations. The authors of the papers acknowledge that, in their present forms, the most likely applications of these models appear to be learning quantum objects rather than enhancing the learning of classical data. Finally, we note that, to date, there are no attempts to model nonlinearities directly on the amplitudes.

10. Learning with noise

Noise can play different, potentially beneficial roles in learning problems. In a classical setting, it has been shown that noise can alleviate two of the most common model-fitting issues: local optima and generalization performance. Perturbing gradients can help with the former by 'jumping out' of local optima, whereas perturbing training inputs or outputs can improve the latter.

The possibility of exploiting advantageously the effects of noise is particularly interesting in the context of quantum computation. Early quantum computers are expected to have too few qubits to implement full error correction and the community is actively looking for problems where noise not only does not destroy the computation but can play a beneficial role.

The analysis of noisy learning problems from a quantum perspective becomes particularly promising in selected cases. As we will discuss in this section, quantum resources allow efficiently noisy learning problems to be solved that would be otherwise classically hard. Although few results are known in this area, further research in this direction might provide new cases of a separation between the classical and quantum case in a learning setting.

The goal of this section is to inspire future research aimed at understanding how quantum learners behave in noisy settings. We begin by reviewing for quantum scientists a number of classical problems in ML that benefit from noise. We proceed with a brief introduction to standard ways of modelling errors in quantum computing aimed at ML practitioners. We conclude by discussing problems where quantum resources allow tasks to be performed that would be otherwise hard for a classical learner.

(a) Classical learning can benefit from noise

(i) Noisy inputs

The first direct link between the addition of noise to the training inputs $(x_i)_{i=1}^n$ and Tikhonov regularization was drawn in [146]. Here, it is shown that optimizing a feed-forward neural network to minimize the squared error on noisy inputs is equivalent (up to the order of the noise variance) to minimizing the squared error with Tikhonov regularization on noiseless inputs.

Intuitively, this form of regularization forces the gradient of the neural network output $f(x)$ with respect to the input x to be small, essentially constraining the learned function to vary slowly with x: neighbouring inputs are encouraged to have similar outputs.

An [147] also investigated the effects of adding noise to inputs, outputs, weights and weight updates in neural networks and observed that input (and sometimes weight) noise can, in some settings, improve generalization performance.

(ii) Noisy parameter updates

More recently, in [148], the addition of annealed i.i.d. Gaussian noise to the gradients has been empirically shown to help in optimizing complex neural network models. Indeed, stochasticity in the optimization process can also derive from evaluating gradients of the objective function with respect to randomly selected subsets of the training points (as in *stochastic gradient descent*). This can be intuitively compared to simulated annealing [149] because the natural variability in these 'partial' gradients can help local optima (and saddle points) to escape and the (decreasing) gradient step size can be directly compared to the annealing temperature.

The addition of noise to the update of model parameters was also adopted in [150]. There, as well as using random subsets of training points to evaluate gradients, at each iteration the parameter update is perturbed with Gaussian noise (with variance equal to the decreasing step size). After the initial stochastic optimization phase, it can be shown that this method, under specific conditions, will start generating samples from the posterior distribution over model parameters, allowing us to quantify model uncertainty and avoid overfitting at no extra computational cost.

(iii) Noisy outputs

In GP regression [24], on the other hand, noise in the training outputs $(y_i)_{i=1}^n$ helps avoid the inversion of an otherwise potentially ill-conditioned kernel covariance matrix K. Assuming additive isotropic Gaussian noise (with variance σ^2), to evaluate model predictions, we only ever need to invert a matrix of the form $K + \sigma^2 I$. This can be practical as the kernel matrix is singular or ill-conditioned whenever training inputs are repeated or are very close in the Hilbert space associated with the kernel covariance function.

Finally, when training *generative adversarial networks* (GANs [151]), it has been shown that an overconfident 'discriminator' can hinder learning in the 'generator'. In GANs in fact, a generative model (the 'generator') is trained by attempting to 'deceive' a 'discriminator' model into classifying the generated images as coming from the true data distribution. However, especially early on in training, there might be little overlap in the support of the data distribution and the generator. This can result in the discriminator predicting labels with very high confidence and, as well as potentially overfitting, in making the discrimination decision depend very weakly on the generator's parameters. To address this issue, labels (i.e. true, fake) can be 'fuzzied'. Specifically, for each training point, the discriminator will assume that all K labels have probability at least ϵ/K of occurring, with the true label having probability $(1 - \epsilon) + \epsilon/K$. This corresponds to assuming that with probability ϵ/K labels are sampled at random and, indeed, labels can just be flipped randomly in practice. Effectively, this keeps the model from becoming too confident in its predictions by making it suboptimal to shift all the probability mass on the true label. This technique is called label smoothing [152] and it has been shown to help retain the training

signal for the generator [153], as well as increase the robustness of classifiers to adversarial examples [154].

(b) A classical/quantum separation in learning under noise

To address learning under noise in a quantum setting, it is necessary to discuss what type of noise affects quantum computers. The works by Preskill [17] and Breuer & Petruccione [155] cover the topic extensively. A simple model of quantum errors, usually employed in numerical simulation of noisy quantum devices, makes use of a weighted combination of two kinds of error: bit flips and phase flips. We can justify this simple type of modelling because, in the most common error-correcting codes, errors are detected by projecting more complex errors into convex combinations of bit and phase flips. Given a quantum state $\psi = \alpha_0 e_0 + \alpha_1 e_1$, a bit flip error turns the state into $\tilde{\psi} = \alpha_0 e_1 + \alpha_1 e_0$. Similarly, a phase flip error changes the relative phase of a quantum state, i.e. the resulting state is $\tilde{\psi} = \alpha_0 e_0 - \alpha_1 e_1$. More complex and realistic models of errors include amplitude damping, leakage to higher levels and loss.

Many authors have studied how noise affects the learnability of a function in the quantum setting. The already mentioned work by Bshouty & Jackson [31] showed that DNF formulae can be efficiently learned under the uniform distribution using a quantum example oracle. This contrasts with the classical case (although proved in the statistical query model of learning) where Blum *et al*. [156] showed that DNFs are not learnable under noise with respect to the uniform distribution.

Another result that points to a separation between classical and quantum for a noisy learning problem has been recently proved by Cross *et al.* [157]. In this case, the learnability of parity functions under noise is discussed. It is widely believed that learning parity function under noise is not classically efficient [158] and the best classical algorithm is run in subexponential, but superpolynomial, time. Furthermore, the problem is an average-case version of the NP-hard problem of decoding a linear code [159], which is also known to be hard to approximate [160]. Both the classical and quantum problem are easy without noise. In [157], Cross *et al.* showed that in the quantum PAC model parity functions can be learned efficiently under the uniform distribution (with logarithmic overhead over the noiseless runtime). Their results have been generalized to linear functions and to more complex error models by Grilo & Kerenidis [161].

To summarize, in this section, we surveyed a number of classical results showing that noise in the inputs, outputs or in the parameters can have positive effects on learning algorithms. It would be interesting to investigate whether the type of noise encountered in quantum systems has a similar distribution and structure to the one commonly encountered in classical settings. In this case, ML algorithms would become ideally suited to run on non-fault-tolerant quantum hardware. Finally, further research is needed to identify new, noisy problems that only a learner equipped with quantum resources can solve.

11. Computationally hard problems in machine learning

Algorithms whose runtime is upper-bounded by a polynomial function of N are said to be *efficient*. Problems for which there exists an efficient algorithm are *easy*. Conversely, *hard* problems are those where no polynomial algorithm is known. An important class of easy problems is called P. The class of problems that are efficiently solvable by a quantum computer includes some problems that are not known to be in P.

The quantum algorithms surveyed in this review speed up efficient classical algorithms. Two types of speed-ups are obtained: polynomial or exponential. Polynomial speed-ups, although important from a practical point of view, do not prove that quantum computers are able to turn hard learning problems into easy ones. On the other hand, exponential speed-ups of algorithms that are already efficient face important challenges. Indeed, as we have seen for the matrix inversion algorithm discussed in §6, quantum algorithms for the analysis of classical data running

in logarithmic time require an equally fast access to the memory. This can be obtained using a QRAM that, however, presents a number of issues (see §5).

To achieve an exponential speed-up despite the computational costs arising from accessing the memory, we are restricted to hard algorithms. This is because, for these algorithms, the polynomial time construction of the quantum state that encodes the dataset does not dominate over the speed-up. We discussed an example with such a property: the learnability of DNF formulae (§4). Classically, the best algorithm for learning DNFs runs in superpolynomial time. With quantum resources we can learn the same problem polynomially. Although these types of learning problems have limited practical applications, they suggest that an exponential separation between classical and quantum models of learning might hold in real-world problems.

In this section, we present a number of problems in ML that are believed to be computationally hard and are receiving considerable interest in the classical community. We do not expect that these problems, some of which are NP-hard, can be solved efficiently with a quantum computer. Recall that NP-hard is a class of problems for which there is strong evidence of a separation with P [162]. Our hope is to spark interest in the search for hard problems in ML with the kind of structure (see §2) that can be exploited by quantum computers. We also decided to include problems that are not hard in the computational complexity sense but whose high-degree polynomial runtimes make them intractable. For these cases, where slow (i.e. polynomial) memory access times can still be tolerable, even polynomial speed-ups might be of great practical relevance.

(a) Tensor factorization

As modern datasets grow not only in terms of sheer dimension but also in the complexity of the structures required to store such data (e.g. multi-modal data, social networks, recommender systems and relational data [163–165]), it becomes ever more critical to devise methods able to distil concise and interpretable representations of this information. Tensor models offer a powerful way to address these learning problems. For instance, tensors naturally generalize the concept of the adjacency matrix for multi-relational graphs [166]. However, given the intrinsic multi-dimensional nature of these objects, tensor-based learning problems are typically computationally hard and require large amounts of memory; therefore, they become quickly impractical in most applications. To this end, finding low-rank approximations of tensors (or more generally multi-linear operators), a natural generalization of the problem of matrix factorization (see [167] and references therein) has recently received significant attention from the fields of ML, inverse problems and compressive sensing. However, while for the matrix case the problem is amenable to efficient computations, moving to higher orders becomes significantly challenging. Indeed, in contrast to its matrix counterpart, low-rank tensor factorization, even when relaxed to a nuclear norm-regularized optimization problem, has been recently shown to be NP-hard [168]. Approaches have attempted to circumvent these issues by considering further relaxation of the factorization problem [169–174], but to this day a standard solution has yet to be proposed.

(b) Submodular problems

Recently, several ML problems have been addressed via submodular optimization. Examples of such applications are very diverse, such as document summarization [175], social networks [176] or clustering [177] to name but a few. Submodularity characterizes a family of discrete optimization problems, typically entailing cost functions on sets in which the target functional exhibits a structure akin to that of convexity (or rather concavity) for continuous functions. We refer to Bach [178] for an in-depth introduction on the topic. For many submodular problems, it is possible to identify a corresponding convex problem via the so-called *Lovàsz* extension [179]. As a consequence, such problems can be solved using convex optimization methods, leading to efficient learning algorithms. However, for a wide range of these problems, the corresponding computational complexity, albeit polynomial, is of high order (e.g. $\mathcal{O}(n^5)$ with respect to the

number n of the parameters; see for instance [180–182]), making them remarkably slow in practice. In this sense, an exponential (or even polynomial) decrease in the number of computations to solve a submodular problem, analogous to the one observed for fast linear algebra using quantum algorithms, could be the key to tackling practical applications.

(c) Inference in graphical models

Probabilistic models in ML can be encoded in graphs. Graphical models of particular use are Bayesian networks [183] and Markov random fields [184]: directed acyclic and undirected graphs, respectively, where nodes represent random variables and edges denote dependence between variables. Operations like marginalization and conditioning can be performed by algorithms taking into account the specific connectivity of the given graph (i.e. message passing). While this offers a general framework for inference (i.e. evaluating the distribution of latent variables conditioned on observed ones), it has been shown, by reduction to Boolean satisfiability [185], that exact inference in these networks is NP-hard and that evaluating the normalizing constant Z (or partition function) for the joint distribution is in #P (a family of hard counting problems).

12. Conclusion and outlook

In this review, we surveyed a number of different quantum methods to tackle learning problems. Despite a number of promising results, the theoretical evidence presented in the current literature does not yet allow us to conclude that quantum techniques can obtain an exponential advantage in a realistic learning setting. Even in the case of quantum algorithms for linear algebra, where rigorous guarantees are already available, issues related to data access and restrictions on the types of problems that can be solved might hinder their performance in practice. In fact, near-future advances in quantum hardware development will be important to empirically assess the true potential of these techniques. In this regard, we note how the great majority of the QML literature has been developed within the quantum community. We believe that further advances in the field will only come after significant interactions between the two communities. For this reason, we tried to structure this review to present the different topics in a way that is familiar to both quantum scientists and ML researchers. To achieve this goal, we put great emphasis on the computational aspects of ML. Although this perspective has the obvious advantage of providing an agile way for discussing quantum algorithms (that mostly focus on accelerating the runtime with respect to their classical counterparts), the reader should keep in mind that statistical problems (like determining the generalization performance of an algorithm) are equally relevant. The approach taken in this review has also left some interesting papers aside (e.g. [186]). We invite the reader to consult [7,8,187] for a review that includes these works.

In §3, we discussed how the computational cost represents one of the major challenges for the future of ML. In particular, polynomial scaling in the number of data points might not be adequate in the age of large-scale ML. The quantum algorithms presented here allow the complexity of some, currently used, regularization methods to be reduced. We classified the quantum approaches into four main categories: linear algebra, neural networks, sampling and optimization. The QML algorithms based on linear algebra subroutines are those that promise the greatest computational advantages (i.e. exponential). However, it is not clear whether fundamental limitations related to how quickly these algorithms need to access the memory might compromise their ability to speed up the analysis of classical data. Quantum methods for training neural networks, for sampling and for optimization, provide so far mostly quadratic advantages and some of these might be implementable on first-generation quantum computers. Unfortunately, the theoretical framework on which they are based is not yet well established (e.g. the quantum Boltzmann machines described in §9) and only practical experiments will determine their true performance.

To summarize, the works surveyed in this review, including the theoretical evidence presented in §4, suggest the possibility of a quantum speed-up for some ML problems. However, the extent

of these speed-ups, and consequently the impact of these methods on practical problems, remains an open question.

We identified a number of promising directions for the field. First, exploring the trade-offs between noise, generalization performance and hardness in a quantum context (§10). This is particularly interesting for first-generation quantum hardware that most likely will not be fault-tolerant. Second, deepening our understanding of how quantum resources can affect sample and time complexity, even for problems that are already known to be efficient. Significant work has already been done but some areas like statistical learning theory are yet to receive a thorough analysis in a quantum context. Third, determining whether a QRAM of the size required to handle large datasets can be constructed on a physical device (§5). Fourth, understanding whether there exist non-polynomial problems in ML that can be tackled efficiently using quantum resources (§11). This direction is arguably the most relevant for finding quantum algorithms capable of demonstrating an uncontroversial speed-up in a learning context, and this is indeed the general quest of quantum computation.

Data accessibility. This paper has no additional data.

Authors' contributions. A.R. and L.W. conceived the project. All the authors contributed to the literature review. All the authors wrote the manuscript.

Competing interests. The authors declare no competing interests.

Funding. A.D.I. is supported by the Cambridge–Tuebingen Fellowship and the Qualcomm Innovation Fellowship. A.R. is supported by an EPSRC DTP scholarship and by QinetiQ. C.C. and M.P. are supported by EPSRC. S.S. is supported by the Royal Society, EPSRC, Innovate UK, Cambridge Quantum Computing and the National Natural Science Foundation of China.

Acknowledgements. We thank Scott Aaronson, David Barber, Marcello Benedetti, Fernando Brandão, Dan Brown, Carlos González-Guillén, Joshua Lockhart and Alessandro Rudi for helpful comments on the manuscript.

References

1. Krizhevsky A, Sutskever I, Hinton GE. 2012 Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (eds F Pereira, CJC Burges, L Bottou, KQ Weinberger), pp. 1097–1105. Red Hook, NY: Curran Associates, Inc.
2. Silver D *et al.* 2016 Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489. (doi:10.1038/nature16961)
3. Markov IL. 2014 Limits on fundamental limits to computation. *Nature* **512**, 147–154. (doi:10.1038/nature13570)
4. Shor PW. 1997 Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM J. Comput.* **26**, 1484–1509. (doi:10.1137/S0097539795293172)
5. Van Dam W, Hallgren S, Ip L. 2006 Quantum algorithms for some hidden shift problems. *SIAM J. Comput.* **36**, 763–778. (doi:10.1137/S009753970343141X)
6. Hallgren S. 2007 Polynomial-time quantum algorithms for Pell's equation and the principal ideal problem. *J. ACM* **54**, 4. (doi:10.1145/1206035.1206039)
7. Adcock J *et al.* 2015 Advances in quantum machine learning. (http://arxiv.org/abs/1512.02900).
8. Biamonte J, Wittek P, Pancotti N, Rebentrost P, Wiebe N, Lloyd S. 2017 Quantum machine learning. *Nature* **549**, 195–202. (doi:10.1038/nature23474)
9. Montanaro A. 2016 Quantum algorithms: an overview. *NPJ Quantum Inf.* **2**, 15023. (doi:10.1038/npjqi.2015.23)
10. Bacon D, Van Dam W. 2010 Recent progress in quantum algorithms. *Commun. ACM.* **53**, 84–93. (doi:10.1145/1646353.1646375)
11. Bishop CM *et al.* 2006 *Pattern recognition and machine learning*, vol. 1. New York, NY: Springer.
12. Murphy KP. 2012 *Machine learning: a probabilistic perspective*. Cambridge, MA: The MIT Press.
13. de Touzalin A, Heijman F, Cirac I, Murray R, Calarco T. 2016 The quantum manifesto. See http://qurope.eu/manifesto.
14. Nielsen MA, Chuang IL. 2010 *Quantum computation and quantum information*. Cambridge, MA: Cambridge University Press.
15. Jozsa R, Linden N. 2003 On the role of entanglement in quantum-computational speed-up. *Proc. R. Soc. Lond. A* **459**, 2011–2032. (doi:10.1098/rspa.2002.1097).

16. Aaronson S, Ambainis A. 2009 The need for structure in quantum speedups. (http://arxiv.org/abs/0911.0996).
17. Preskill J. 1998 Fault-tolerant quantum computation. (https://arxiv.org/abs/quant-ph/9712048)
18. Papadimitriou CH. 2003 *Computational complexity*. New York, NY: John Wiley and Sons Ltd.
19. Arora S, Barak B. 2009 *Computational complexity: a modern approach*. Cambridge, MA: Cambridge University Press.
20. Valiant LG. 1984 A theory of the learnable. *Commun. ACM.* **27**, 1134–1142. (doi:10.1145/1968.1972)
21. Vapnik VN. 1998 *Statistical learning theory*, vol. 1. New York, NY: Wiley.
22. Bauer F, Pereverzev S, Rosasco L. 2007 On regularization algorithms in learning theory. *J. Complex.* **23**, 52–72. (doi:10.1016/j.jco.2006.07.001)
23. Cucker F, Smale S. 2002 On the mathematical foundations of learning. *Bull. Amer. Math. Soc.* **39**, 1–49. (doi:10.1090/S0273-0979-01-00923-5)
24. Rasmussen CE, Williams CKI. 2006 *Gaussian processes for machine learning*. Cambridge, MA: The MIT Press.
25. Zhang Y, Duchi J, Wainwright M. 2013 Divide and conquer kernel ridge regression. In *Conf. on Learning Theory, Princeton, NJ*, pp. 592–617. Brookline, MA: Microtome Publishing.
26. Rahimi A, Recht B. 2007 Random features for large-scale kernel machines. In *Advances Neural Information Processing Systems* (eds JC Platt, D Koller, Y Singer, ST Roweis), vol. 3, pp. 1177–1184. Red Hook, NY: Curran Associates, Inc.
27. Smola AJ, Schölkopf B. 2000 Sparse greedy matrix approximation for machine learning. In *Proc. of the Int. Conf. on Machine Learning*, pp. 911–918. San Francisco, CA: Morgan Kaufmann Publishers.
28. Williams CK, Seeger M. 2000 Using the Nyström method to speed up kernel machines. In *Proc. of the 13th Int. Conf. on Neural Information Processing Systems, Vancouver, Canada*, pp. 661–667. Cambridge, MA: The MIT Press.
29. Rudi A, Camoriano R, Rosasco L. 2015 Less is more: Nyström computational regularization. In *Advances in Neural Information Processing Systems* (eds C Cortes, ND Lawrence, DD Lee, M Sugiyama, R Garnett), pp. 1657–1665. Red Hook, NY: Curran Associates, Inc.
30. Arunachalam S, de Wolf R. 2017 Guest column: a survey of quantum learning theory. *SIGACT News* **48**, 41–67. (doi:10.1145/3106700.3106710)
31. Bshouty NH, Jackson JC. 1998 Learning DNF over the uniform distribution using a quantum example oracle. *SIAM J. Comput.* **28**, 1136–1153. (doi:10.1137/S0097539795293123)
32. Servedio RA, Gortler SJ. 2004 Equivalences and separations between quantum and classical learnability. *SIAM J. Comput.* **33**, 1067–1092. (doi:10.1137/S0097539704412910)
33. Atici A, Servedio RA. 2005 Improved bounds on quantum learning algorithms. *Quantum Inf. Process.* **4**, 355–386. (doi:10.1007/s11128-005-0001-2)
34. Zhang C. 2010 An improved lower bound on query complexity for quantum PAC learning. *Inf. Process. Lett.* **111**, 40–45. (doi:10.1016/j.ipl.2010.10.007)
35. Arunachalam S, de Wolf R. 2017 Optimal quantum sample complexity of learning algorithms. In *Proc. 32nd Computational Complexity Conference, CCC 2017, Riga, Latvia, 6–9 July 2017*. LIPIcs 79. Wadern, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.
36. Angluin D. 1988 Queries and concept learning. *Mach. Learn.* **2**, 319–342. (doi:10.1023/A:1022821128753)
37. Bshouty NH, Cleve R, Kannan S, Tamon C. 1994 Oracles and queries that are sufficient for exact learning. In *Proc. of the 7th Annu. Conf. on Computational Learning Theory, New Brunswick, NJ*, pp. 130–139. New York, NY: ACM.
38. Klivans AR, Servedio R. 2001 Learning DNF in time. In *Proc. of the 33rd Annu. ACM Symp. on Theory of Computing, Crete, Greece*, pp. 258–265. New York, NY: ACM.
39. Verbeurgt KA. 1990 Learning DNF under the uniform distribution in quasi-polynomial time. In *COLT '90: Proc. of the 3rd Annu. Workshop on Computational Learning Theory*, pp. 314–326. Cambridge, MA: The MIT Press.
40. Ben-David S, Eiron N, Simon HU. 2002 Limitations of learning via embeddings in Euclidean half spaces. *J. Mach. Learn. Res.* **3**, 441–461.
41. Khardon R, Servedio RA. 2005 Maximum margin algorithms with Boolean kernels. *J. Mach. Learn. Res.* **6**, 1405–1429.
42. Bernstein E, Vazirani U. 1997 Quantum complexity theory. *SIAM J. Comput.* **26**, 1411–1473. (doi:10.1137/S0097539796300921)

43. Kearns M, Valiant L. 1994 Cryptographic limitations on learning Boolean formulae and finite automata. *J. ACM* **41**, 67–95. (doi:10.1145/174644.174647)
44. Aaronson S. 2013 *Quantum computing since Democritus*. Cambridge, MA: Cambridge University Press.
45. Giovannetti V, Lloyd S, Maccone L. 2008 Quantum random access memory. *Phys. Rev. Lett.* **100**, 160501. (doi:10.1103/PhysRevLett.100.160501)
46. Giovannetti V, Lloyd S, Maccone L. 2008 Architectures for a quantum random access memory. *Phys. Rev. A* **78**, 052310. (doi:10.1103/PhysRevA.78.052310)
47. Aaronson S. 2015 Read the fine print. *Nat. Phys.* **11**, 291–293. (doi:10.1038/nphys3272)
48. Arunachalam S, Gheorghiu V, Jochym-O'Connor T, Mosca M, Srinivasan PV. 2015 On the robustness of bucket brigade quantum RAM. *New. J. Phys.* **17**, 123010. (doi:10.1088/1367-2630/17/12/123010)
49. Grover LK. 1996 A fast quantum mechanical algorithm for database search. In *Proc. of the 28th Annu. ACM Symp. on Theory of Computing, Philadelphia, PA*, pp. 212–219. New York, NY: ACM.
50. Steiger DS, Troyer M. 2016 *Racing in parallel: quantum versus classical*. Quantum Machine Learning Workshop. Waterloo, Canada: Perimeter Institute for Theoretical Physics. See http://pirsa.org/displayFlash.php?id=16080019.
51. Csanky L. 1976 Fast parallel matrix inversion algorithms. *SIAM J. Comput.* **5**, 618–623. (doi:10.1137/0205040)
52. Prakash A. 2014 Quantum algorithms for linear algebra and machine learning. PhD thesis, University of California, Berkeley, CA, USA.
53. Bennett CH, Bernstein E, Brassard G, Vazirani U. 1997 Strengths and weaknesses of quantum computing. *SIAM J. Comput.* **26**, 1510–1523. (doi:10.1137/S0097539796300933)
54. Grover L, Rudolph T. 2002 Creating superpositions that correspond to efficiently integrable probability distributions. (http://arxiv.org/abs/quant-ph/0208112).
55. Shewchuk JR. 1994 An introduction to the conjugate gradient method without the agonizing pain. Technical Report no. ICG:865018. Carnegie-Mellon University, Department of Computer Science, Pittsburgh, PA, USA.
56. Frieze A, Kannan R, Vempala S. 2004 Fast Monte-Carlo algorithms for finding low-rank approximations. *J. ACM* **51**, 1025–1041. (doi:10.1145/1039488.1039494)
57. Harrow AW, Hassidim A, Lloyd S. 2009 Quantum algorithm for linear systems of equations. *Phys. Rev. Lett.* **103**, 150502. (doi:10.1103/PhysRevLett.103.150502)
58. Lloyd S, Mohseni M, Rebentrost P. 2014 Quantum principal component analysis. *Nat. Phys.* **10**, 631–633. (doi:10.1038/nphys3029)
59. Rebentrost P, Mohseni M, Lloyd S. 2014 Quantum support vector machine for big data classification. *Phys. Rev. Lett.* **113**, 130503. (doi:10.1103/PhysRevLett.113.130503)
60. Kerenidis I, Prakash A. 2017 Quantum recommendation systems. In *Proc. 8th Innovations in Theoretical Computer Science Conf., ITCS 2017, Berkeley, CA, 9–11 January 2017*. LIPIcs 67. Wadern, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.
61. Zhao Z, Fitzsimons JK, Fitzsimons JF. 2015 Quantum assisted Gaussian process regression. (http://arxiv.org/abs/1512.03929).
62. Schuld M, Sinayskiy I, Petruccione F. 2016 Prediction by linear regression on a quantum computer. *Phys. Rev. A* **94**, 022342. (doi:10.1103/PhysRevA.94.022342)
63. Cosnard M, Robert Y. 1986 Complexity of parallel QR factorization. *J. ACM* **33**, 712–723. (doi:10.1145/6490.214102)
64. Li S. 2009 *Fast algorithms for sparse matrix inverse computations. PhD Thesis, Stanford University, Stanford, CA, USA*.
65. Li S, Darve E. 2012 Extension and optimization of the find algorithm: computing Green's and less-than Green's functions. *J. Comput. Phys.* **231**, 1121–1139. (doi:1016/j.jcp.2011.05.027)
66. Halko N, Martinsson P-G, Tropp JA. 2011 Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.* **53**, 217–288. (doi:10.1137/090771806)
67. Coppersmith D, Winograd S. 1990 Matrix multiplication via arithmetic progressions. *J. Symb. Comput.* **9**, 251–280. (doi:10.1016/S0747-7171(08)80013-2)
68. Golub GH, Van Loan CF. 2012 *Matrix computations*, vol. 3. Baltimore, MD: JHU Press.
69. Ambainis A. 2012 Variable time amplitude amplification and quantum algorithms for linear algebra problems. In *Proc. 29th Int. Symp. on Theoretical Aspects of Computer Science, STACS*

2012, Paris, France, 29 February–3 March 2012. LIPIcs 14. Wadern, Germany: Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik.
70. Childs AM, Kothari R, Somma RD. 2015 Quantum linear systems algorithm with exponentially improved dependence on precision. (http://arxiv.org/abs/1511.02306).
71. Clader BD, Jacobs BC, Sprouse CR. 2013 Preconditioned quantum linear system algorithm. *Phys. Rev. Lett.* **110**, 250504. (doi:10.1103/PhysRevLett.110.250504)
72. Wossnig L, Zhao Z, Prakash A. 2017 A quantum linear system algorithm for dense matrices. (http://arxiv.org/abs/1704.06174).
73. Childs AM. 2009 Quantum algorithms: equation solving by simulation. *Nat. Phys.* **5**, 861–861. (doi:10.1038/nphys1473)
74. Spielman DA, Teng S-H. 2004 Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time. *J. ACM* **51**, 385–463. (doi:10.1145/990308.990310)
75. Spielman DA, Teng S-H. 2009 Smoothed analysis: an attempt to explain the behavior of algorithms in practice. *Commun. ACM* **52**, 76–84. (doi:10.1145/1562764.1562785)
76. Duchi JC, Mackey LW, Jordan MI. 2010 On the consistency of ranking algorithms. In *Proc. of the 27th Int. Conf. on Machine Learning* (*ICML-10*), *Haifa, Israel*, pp. 327–334. Brookline, MA: Microtome Publishing.
77. Caponnetto A, De Vito E. 2007 Optimal rates for the regularized least-squares algorithm. *Found. Comput. Math.* **7**, 331–368. (doi:10.1007/s10208-006-0196-8)
78. Heller D. 1978 A survey of parallel algorithms in numerical linear algebra. *SIAM Rev.* **20**, 740–777. (doi:10.1137/1020096)
79. Jolliffe IT. 1986 *Principal component analysis*, pp. 115–128. New York, NY: Springer.
80. Trefethen LN, Bau III D. 1997 *Numerical linear algebra*, vol. 50. Philadelphia, PA: SIAM.
81. Szegedy M. 2004 Quantum speed-up of Markov chain based algorithms. In *Proc. 45th Annu. IEEE Symp. on Foundations of Computer Science, Rome, Italy*, pp. 32–41. New York, NY: IEEE.
82. Neal RM. 1993 Probabilistic inference using Markov chain Monte Carlo methods. Technical Report no. CRG-TR-93-1. Department of Computer Science, University of Toronto, Toronto, Canada.
83. Neal RM. 2001 Annealed importance sampling. *Stat. Comput.* **11**, 125–139. (doi:10.1023/A:1008923215028)
84. Gilks WR, Wild P. 1992 Adaptive rejection sampling for Gibbs sampling. *Appl. Stat.* **41**, 337–348. (doi:10.2307/2347565)
85. Propp JG, Wilson DB. 1996 Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Struct. Algorithms* **9**, 223–252. (doi:10.1002/(SICI)1098-2418(199608/09)9:1/2<223::AID-RSA14>3.0.CO;2-O)
86. Doucet A, De Freitas N, Gordon N. 2001 *Sequential Monte Carlo methods in practice*, pp. 3–14. New York, NY: Springer.
87. Del Moral P, Doucet A, Jasra A. 2006 Sequential Monte Carlo samplers. *J. R. Stat. Soc.* **68**, 411–436. (doi:10.1111/j.1467-9868.2006.00553.x)
88. Sinclair A. 1993 Markov chains and rapid mixing. In *Algorithms for random generation and counting: a Markov chain approach*, pp. 42–62. Berlin, Germany: Springer.
89. Wocjan P, Chiang C-F, Nagaj D, Abeyesinghe A. 2009 Quantum algorithm for approximating partition functions. *Phys. Rev. A* **80**, 022340. (doi:10.1103/PhysRevA.80.022340)
90. Somma R, Boixo S, Barnum H, Knill E. 2008 Quantum simulations of classical annealing processes. *Phys. Rev. Lett.* **101**, 130504. (doi:10.1103/PhysRevLett.101.130504)
91. Poulin D, Wocjan P. 2009 Sampling from the thermal quantum Gibbs state and evaluating partition functions with a quantum computer. *Phys. Rev. Lett.* **103**, 220502. (doi:10.1103/PhysRevLett.103.220502)
92. Chiang C-F, Wocjan P. 2010 Quantum algorithm for preparing thermal Gibbs states—detailed analysis. *Quantum Cryptography Computing* **26**, 138–147.
93. Bilgin E, Boixo S. 2010 Preparing thermal states of quantum systems by dimension reduction. *Phys. Rev. Lett.* **105**, 170405. (doi:10.1103/PhysRevLett.105.170405)
94. Temme K, Osborne TJ, Vollbrecht KG, Poulin D, Verstraete F. 2011 Quantum metropolis sampling. *Nature* **471**, 87–90. (doi:10.1038/nature09770)
95. Schwarz M, Temme K, Verstraete F. 2012 Preparing projected entangled pair states on a quantum computer. *Phys. Rev. Lett.* **108**, 110502. (doi:10.1103/PhysRevLett.108.110502)
96. Chowdhury AN, Somma RD. 2016 Quantum algorithms for Gibbs sampling and hitting-time estimation. (http://arxiv.org/abs/1603.02940).

97. Ambainis A, Kempe J, Rivosh A. 2005 Coins make quantum walks faster. In *Proc. of the 16th Annu. ACM-SIAM Symp. on Discrete algorithms, Stockholm, Sweden*, pp. 1099–1108. Philadelphia, PA: Society for Industrial and Applied Mathematics.
98. Krovi H, Brun TA. 2006 Hitting time for quantum walks on the hypercube. *Phys. Rev. A* **73**, 032341. (doi:10.1103/PhysRevA.73.032341)
99. Krovi H, Ozols M, Roland J. 2010 Adiabatic condition and the quantum hitting time of Markov chains. *Phys. Rev. A* **82**, 022333. (doi:10.1103/PhysRevA.82.022333)
100. Magniez F, Nayak A, Roland J, Santha M. 2011 Search via quantum walk. *SIAM J. Comput.* **40**, 142–164. (doi:10.1137/090745854)
101. Knill E, Ortiz G, Somma RD. 2007 Optimal quantum measurements of expectation values of observables. *Phys. Rev. A* **75**, 012328. (doi:10.1103/PhysRevA.75.012328)
102. Vandenberghe L, Boyd S. 1996 Semidefinite programming. *SIAM Rev.* **38**, 49–95. (doi:10.1137/1038003)
103. Grötschel M, Lovász L, Schrijver A. 2012 *Geometric algorithms and combinatorial optimization*, vol. 2. Berlin, Germany: Springer Science & Business Media.
104. Lanckriet GR, Cristianini N, Bartlett P, Ghaoui LE, Jordan MI. 2004 Learning the kernel matrix with semidefinite programming. *J. Mach. Learn. Res.* **5**, 27–72.
105. Weinberger KQ, Sha F, Zhu Q, Saul LK. 2007 Graph Laplacian regularization for large-scale semidefinite programming. In *Advances in Neural Information Processing Systems* (eds JC Platt, D Koller, Y Singer, ST Roweis), pp. 1489–1496. Red Hook, NY: Curran Associates, Inc.
106. Jacob L, Obozinski G, Vert J-P. 2009 Group Lasso with overlap and graph Lasso. In *Proc. of the 26th Annu. Int. Conf. on Machine Learning, Montreal, Canada*, pp. 433–440. New York, NY: ACM.
107. Lee YT, Sidford A, Wong SC-W. 2015 A faster cutting plane method and its implications for combinatorial and convex optimization. In *Proc. 2015 IEEE 56th Annu. Symp. on Foundations of Computer Science (FOCS), Berkeley, CA*, pp. 1049–1065. New York, NY: IEEE.
108. Arora S, Kale S. 2007 A combinatorial, primal-dual approach to semidefinite programs. In *Proc. of the 39th Annu. ACM Symp. on Theory of Computing, San Diego, CA*, pp. 227–236. New York, NY: ACM.
109. Brandão FGSL, Svore K. 2016 Quantum speed-ups for semidefinite programming. (http://arxiv.org/abs/1609.05537).
110. van Apeldoorn J, Gilyén A, Gribling S, de Wolf R. 2017 Quantum SDP-solvers: better upper and lower bounds. (http://arxiv.org/abs/1705.01843).
111. Creignou N, Khanna S, Sudan M. 2001 *Complexity classifications of Boolean constraint satisfaction problems*. Philadelphia, PA: SIAM.
112. Farhi E, Goldstone J, Gutmann S. 2014 A quantum approximate optimization algorithm. (http://arxiv.org/abs/1411.4028).
113. Farhi E, Goldstone J, Gutmann S, Neven H. 2017 Quantum algorithms for fixed qubit architectures. (http://arxiv.org/abs/1703.06199).
114. Farhi E, Goldstone J, Gutmann S. 2014 A quantum approximate optimization algorithm applied to a bounded occurrence constraint problem. (http://arxiv.org/abs/1412.6062).
115. Barak B *et al.* 2015 Beating the random assignment on constraint satisfaction problems of bounded degree. (http://arxiv.org/abs/1505.03424).
116. Farhi E, Goldstone J, Gutmann S, Sipser M. 2000 Quantum computation by adiabatic evolution. (http://arxiv.org/abs/quant-ph/0001106).
117. Kirkpatrick S. 1984 Optimization by simulated annealing: quantitative studies. *J. Stat. Phys.* **34**, 975–986. (doi:10.1007/BF01009452)
118. Messiah A. 1958 *Quantum mechanics*, vol. 1. New York, NY: Interscience.
119. Reichardt BW. 2004 The quantum adiabatic optimization algorithm and local minima. In *Proc. of the 36th Annu. ACM Symp. on Theory of Computing, Chicago, IL*, pp. 502–510. New York, NY: ACM.
120. Crosson E, Harrow AW. 2016 Simulated quantum annealing can be exponentially faster than classical simulated annealing. In *Proc. 2016 IEEE 57th Annu. Symp. on Foundations of Computer Science (FOCS), New Brunswick, NJ*, pp. 714–723. New York, NY: IEEE.
121. Aharonov D, Van Dam W, Kempe J, Landau Z, Lloyd S, Regev O. 2008 Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Rev.* **50**, 755–787. (doi:10.1137/080734479)
122. Barahona F. 1982 On the computational complexity of Ising spin glass models. *J. Phys. A* **15**, 3241–3253. (doi:10.1088/0305-4470/15/10/028)

123. Farhi E, Goldstone J, Gutmann S. 2000 A numerical study of the performance of a quantum adiabatic evolution algorithm for satisfiability. (http://arxiv.org/abs/quant-ph/0007071).
124. Farhi E, Goldstone J, Gutmann S, Lapan J, Lundgren A, Preda D. 2001 A quantum adiabatic evolution algorithm applied to random instances of an NP-complete problem. *Science* **292**, 472–475. (doi:10.1126/science.1057726)
125. Farhi E, Gosset D, Hen I, Sandvik A, Shor P, Young A, Zamponi F. 2012 Performance of the quantum adiabatic algorithm on random instances of two optimization problems on regular hypergraphs. *Phys. Rev. A* **86**, 052334. (doi:10.1103/PhysRevA.86.052334)
126. Hen I, Young A. 2011 Exponential complexity of the quantum adiabatic algorithm for certain satisfiability problems. *Phys. Rev. E* **84**, 061152. (doi:10.1103/PhysRevE.84.061152)
127. Young AP, Knysh S, Smelyanskiy VN. 2008 Size dependence of the minimum excitation gap in the quantum adiabatic algorithm. *Phys. Rev. Lett.* **101**, 170503. (doi:10.1103/PhysRevLett.101.170503)
128. Young A, Knysh S, Smelyanskiy V. 2010 First-order phase transition in the quantum adiabatic algorithm. *Phys. Rev. Lett.* **104**, 020502. (doi:10.1103/PhysRevLett.104.020502)
129. Kak S. 1995 On quantum neural computing. *Inf. Sci. (NY)* **83**, 143–160. (doi:10.1016/0020-0255(94)00095-S)
130. Smolensky P. 1986 Information processing in dynamical systems: foundations of harmony theory. Technical Report no. CU-CS-321-86. University of Colorado Boulder, Department of Computer Science, Boulder, CO, USA.
131. Long PM, Servedio R. 2010 Restricted Boltzmann machines are hard to approximately evaluate or simulate. In *Proc. of the 27th Int. Conf. on Machine Learning (ICML-10), Haifa, Israel*, pp. 703–710. Brookline, MA: Microtome Publishing.
132. Dumoulin V, Goodfellow IJ, Courville A, Bengio Y. 2014 On the challenges of physical implementations of RBMs. In *Proc. 28th AAAI Conf. on Artificial Intelligence, Quebec City, Canada, 27–31 July 2014*. Palo Alto, CA: The AAAI Press.
133. Wiebe N, Kapoor A, Svore KM. 2014 Quantum deep learning. (http://arxiv.org/abs/1412.3489).
134. Brassard G, Hoyer P, Mosca M, Tapp A. 2002 Quantum amplitude amplification and estimation. *Contemp. Math.* **305**, 53–74. (doi:10.1090/conm/305/05215)
135. Hinton GE. 2002 Training products of experts by minimizing contrastive divergence. *Neural. Comput.* **14**, 1771–1800. (doi:10.1162/089976602760128018)
136. Wiebe N, Kapoor A, Granade C, Svore KM. 2015 Quantum inspired training for Boltzmann machines. (http://arxiv.org/abs/1507.02642).
137. Adachi SH, Henderson MP. 2015 Application of quantum annealing to training of deep neural networks. (http://arxiv.org/abs/1510.06356).
138. Denil M, De Freitas N. 2011 Toward the implementation of a quantum RBM. Paper presented at the Learning and Unsupervised Feature Learning Workshop of the 25th Ann. Conf. on Neural Information Processing Systems (NIPS), Granada, Spain, 2011.
139. Benedetti M, Realpe-Gómez J, Biswas R, Perdomo-Ortiz A. 2016 Estimation of effective temperatures in quantum annealers for sampling applications: a case study with possible applications in deep learning. *Phys. Rev. A* **94**, 022308. (doi:10.1103/PhysRevA.94.022308)
140. Amin MH, Andriyash E, Rolfe J, Kulchytskyy B, Melko R. 2016 Quantum Boltzmann machine. (http://arxiv.org/abs/1601.02036).
141. Kieferova M, Wiebe N. 2016 Tomography and generative data modeling via quantum Boltzmann training. (http://arxiv.org/abs/1612.05204).
142. Schuld M, Sinayskiy I, Petruccione F. 2014 The quest for a quantum neural network. *Quantum Inf. Process.* **13**, 2567–2586. (doi:10.1007/s11128-014-0809-8)
143. Cybenko G. 1989 Approximation by superpositions of a sigmoidal function. *Math. Control Signals Syst.* **2**, 303–314. (doi:10.1007/BF02551274)
144. Wan KH, Dahlsten O, Kristjánsson H, Gardner R, Kim M. 2016 Quantum generalisation of feedforward neural networks. (http://arxiv.org/abs/1612.01045).
145. Romero J, Olson J, Aspuru-Guzik A. 2017 Quantum autoencoders for efficient compression of quantum data. *Quantum Sci. Technol.* **2**, 045001. (doi:10.1088/2058-9565/aa8072)
146. Bishop CM. 1995 Training with noise is equivalent to Tikhonov regularization. *Neural. Comput.* **7**, 108–116. (doi:10.1162/neco.1995.7.1.108)
147. An G. 1996 The effects of adding noise during backpropagation training on a generalization performance. *Neural. Comput.* **8**, 643–674. (doi:10.1162/neco.1996.8.3.643)

148. Neelakantan A, Vilnis L, Le QV, Sutskever I, Kaiser L, Kurach K, Martens J. 2015 Adding gradient noise improves learning for very deep networks. (http://arxiv.org/abs/1511.06807).
149. Bottou L. 1991 Stochastic gradient learning in neural networks. In *Proc. Neuro-Nımes. 91*. Nimes, France: EC2.
150. Welling M, Teh YW. 2011 Bayesian learning via stochastic gradient Langevin dynamics. In *Proc. of the 28th Int. Conf. on Machine Learning* (*ICML-11*), *Bellevue, WA*, pp. 681–688. Brookline, MA: Microtome Publishing.
151. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y. 2014 Generative adversarial nets. In *Advances in Neural Information Processing Systems* (eds Z Ghahramani, M Welling, C Cortes, ND Lawrence, KQ Weinberger), pp. 2672–2680. Red Hook, NY: Curran Associates, Inc.
152. Szegedy C, Vanhoucke V, Ioffe S, Shlens J, Wojna Z. 2016 Rethinking the inception architecture for computer vision. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Las Vegas, NV*, pp. 2818–2826. New York, NY: IEEE.
153. Salimans T, Goodfellow I, Zaremba W, Cheung V, Radford A, Chen X. 2016 Improved techniques for training gans. In *Advances in Neural Information Processing Systems* (eds DD Lee, M Sugiyama, UV Luxburg, I Guyon, R Garnett), pp. 2234–2242. Red Hook, NY: Curran Associates, Inc.
154. Warde-Farley D, Goodfellow I. 2016 Adversarial perturbations of deep neural networks. In *Perturbation, optimization, and statistics* (eds T Hazan, G Papandreou, D Tarlow), pp. 311–342. Cambridge, MA: The MIT Press.
155. Breuer H-P, Petruccione F. 2002 *The theory of open quantum systems*. Oxford, UK: Oxford University Press.
156. Blum A, Furst M, Jackson J, Kearns M, Mansour Y, Rudich S. 1994 Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *Proc. of the 26th Annu. ACM Symp. on Theory of Computing, Santa Fe, NM*, pp. 253–262. New York, NY: ACM.
157. Cross AW, Smith G, Smolin JA. 2015 Quantum learning robust against noise. *Phys. Rev. A* **92**, 012327. (doi:10.1103/PhysRevA.92.012327)
158. Lyubashevsky V. 2005 The parity problem in the presence of noise, decoding random linear codes, and the subset sum problem. In *Approximation, randomization and combinatorial optimization. Algorithms and techniques* (eds M Goemans, K Jansen, JDP Rolim, L Trevisan), pp. 378–389. Berlin, Germany: Springer.
159. Berlekamp E, McEliece R, Van Tilborg H. 1978 On the inherent intractability of certain coding problems (corresp.). *IEEE Trans. Inf. Theory* **24**, 384–386. (doi:10.1109/TIT.1978.1055873)
160. Håstad J. 2001 Some optimal inapproximability results. *J. ACM* **48**, 798–859. (doi:10.1145/502090.502098)
161. Grilo AB, Kerenidis I. 2017 Learning with errors is easy with quantum samples. (http://arxiv.org/abs/1702.08255).
162. Fortnow L. 2009 The status of the P versus NP problem. *Commun. ACM* **52**, 78–86. (doi:10.1145/1562164.1562186)
163. Suchanek FM, Kasneci G, Weikum G. 2007 Yago: a core of semantic knowledge. In *Proc. of the 16th Int. Conf. on World Wide Web, Banff, Canada*, pp. 697–706. New York, NY: ACM.
164. Dong X, Gabrilovich E, Heitz G, Horn W, Lao N, Murphy K, Strohmann T, Sun S, Zhang W. 2014 Knowledge vault: a web-scale approach to probabilistic knowledge fusion. In *Proc. of the 20th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, New York, NY*, pp. 601–610. New York, NY: ACM.
165. Carlson A, Betteridge J, Kisiel B, Settles B, Hruschka Jr ER, Mitchell TM. 2010 Toward an architecture for never-ending language learning. In *Proc. of the 24th AAAI Conf. on Artificial Intelligence, Atlanta, GA, 11–15 July 2010*, vol. 5, p. 3. Palo Alto, CA: Association for the Advancement of Artificial Intelligence.
166. Getoor L, Taskar B. 2007 *Introduction to statistical relational learning*. Cambridge, MA: The MIT Press.
167. Kolda TG, Bader BW. 2009 Tensor decompositions and applications. *SIAM Rev.* **51**, 455–500. (doi:10.1137/07070111X)
168. Hillar CJ, Lim L-H. 2013 Most tensor problems are NP-hard. *J. ACM* **60**, 45. (doi:10.1145/2512329)

169. Mu C, Huang B, Wright J, Goldfarb D. 2014 Square deal: lower bounds and improved relaxations for tensor recovery. In *Proc. Int. Conf. on Machine Learning, Beijing, China*, pp. 73–81. Brookline, MA: Microtome Publishing.
170. Richard E, Montanari A. 2014 A statistical model for tensor PCA. In *Advances in Neural Information Processing Systems* (eds Z Ghahramani, M Welling, C Cortes, ND Lawrence, KQ Weinberger), pp. 2897–2905. Red Hook, NY: Curran Associates, Inc.
171. Romera-Paredes B, Pontil M. 2013 A new convex relaxation for tensor completion. In *Advances in Neural Information Processing Systems* (eds CJC Burges, L Bottou, M Welling, Z Ghahramani, KQ Weinberger), pp. 2967–2975. Red Hook, NY: Curran Associates, Inc.
172. Romera-Paredes B, Aung H, Bianchi-Berthouze N, Pontil M. 2013 Multilinear multitask learning. In *Proc. Int. Conf. on Machine Learning, Atlanta, GA*, pp. 1444–1452. Brookline, MA: Microtome Publishing.
173. Signoretto M, Dinh QT, De Lathauwer L, Suykens JA. 2014 Learning with tensors: a framework based on convex optimization and spectral regularization. *Mach. Learn.* **94**, 303–351. (doi:10.1007/s10994-013-5366-3)
174. Gandy S, Recht B, Yamada I. 2011 Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse. Probl.* **27**, 025010. (doi:10.1088/0266-5611/27/2/025010)
175. Lin H, Bilmes J. 2011 A class of submodular functions for document summarization. In *Proc. of the 49th Annu. Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1, Association for Computational Linguistics, Portland, OR*, pp. 510–520. Cambridge, MA: The MIT Press.
176. Kempe D, Kleinberg J, Tardos É. 2003 Maximizing the spread of influence through a social network. In *Proc. of the 9th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Washington, DC*, pp. 137–146. New York, NY: ACM.
177. Narasimhan M, Bilmes JA. 2007 Local search for balanced submodular clusterings. In *IJCAI'07: Proc. of the 20th Int. Joint Conf. on Artificial Intelligence, Hyderabad, India, 6–12 January 2007*, pp. 981–986. San Francisco, CA: Morgan Kaufmann Publishers.
178. Bach F. 2015 Submodular functions: from discrete to continous domains. (http://arxiv.org/abs/1511.00394).
179. Lovász L. 1983 Submodular functions and convexity. In *Mathematical programming: the state of the art* (eds A Bachem, B Korte, M Grötschel), pp. 235–257. Berlin, Germany: Springer.
180. Schrijver A. 2000 A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Comb. Theory* **80**, 346–355. (doi:10.1006/jctb.2000.1989)
181. Iwata S, Fleischer L, Fujishige S. 2001 A combinatorial strongly polynomial algorithm for minimizing submodular functions. *J. ACM* **48**, 761–777. (doi:10.1145/502090.502096)
182. Orlin JB. 2009 A faster strongly polynomial time algorithm for submodular function minimization. *Math. Program.* **118**, 237–251. (doi:10.1007/s10107-007-0189-2)
183. Pearl J. 1985 Bayesian networks: a model of self-activated memory for evidential reasoning. In *Proc. of the 7th Conf. of the Cognitive Science Society, Irvine, CA, 15–17 August 1985*, pp. 329–334. Austin, TX: Cognitive Science Society.
184. Kindermann R, Snell L. 1980 *Markov random fields and their applications*, vol. 1. Providence, RI: American Mathematical Society.
185. Cooper GF. 1990 The computational complexity of probabilistic inference using Bayesian belief networks. *Artif. Intell.* **42**, 393–405. (doi:10.1016/0004-3702(90)90060-D)
186. Lloyd S, Garnerone S, Zanardi P. 2014 Quantum algorithms for topological and geometric analysis of big data. (http://arxiv.org/abs/1408.3106).
187. Schuld M, Sinayskiy I, Petruccione F. 2014 An introduction to quantum machine learning. *Contemp. Phys.* **56**, 172–185. (doi:10.1080/00107514.2014.964942)