Mach Learn (2013) 90:261–287
DOI 10.1007/s10994-012-5316-5

Quantum speed-up for unsupervised learning

Esma Aïmeur · Gilles Brassard · Sébastien Gambs

Received: 10 February 2012 / Revised: 10 February 2012 / Accepted: 13 July 2012 /
Published online: 31 August 2012

Abstract We show how the quantum paradigm can be used to speed up unsupervised learning algorithms. More precisely, we explain how it is possible to accelerate learning algorithms by quantizing some of their subroutines. Quantization refers to the process that partially or totally converts a classical algorithm to its quantum counterpart in order to improve performance. In particular, we give quantized versions of clustering via minimum spanning tree, divisive clustering and k-medians that are faster than their classical analogues. We also describe a distributed version of k-medians that allows the participants to save on the global communication cost of the protocol compared to the classical version. Finally, we design quantum algorithms for the construction of a neighbourhood graph, outlier detection as well as smart initialization of the cluster centres.

Keywords Unsupervised learning · Clustering · Quantum learning · Quantum information processing · Grover's algorithm

1 Introduction

Consider the following scenario, which illustrates a highly challenging clustering task. Imagine that you are an employee of the Department of Statistics of the United Nations. Your boss gives you the demographic data of all the Earth inhabitants and asks you to anal-

Editor: Shai Shalev-Shwartz.

E. Aïmeur · G. Brassard (✉)
Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, Succursale Centre-Ville, Montréal, Québec H3C 3J7, Canada
e-mail: brassard@iro.umontreal.ca

E. Aïmeur
e-mail: aimeur@iro.umontreal.ca

S. Gambs
Université de Rennes 1-INRIA, IRISA, Campus de Beaulieu, Avenue du Général Leclerc, 35042 Rennes Cedex, France
e-mail: sebastien.gambs@irisa.fr

yse this data with a clustering algorithm in the hope that you might detect interesting trends and groups within this population. Seeing that you seem a bit lost in front of such a big amount of data, he tells you that he was able to "borrow" a fully operational full-scale quantum computer from the National Security Agency. Can this quantum computer help you speed up the clustering process?

Unsupervised learning algorithms are frequently used in *Data Mining* (Witten and Frank 2005) for applications in which the size of the dataset is huge, such as astronomy, bioinformatics or data issued from large-scale networks such as the Internet. For this type of applications, having fast and efficient algorithms is a necessity, and sometimes having a polynomial-time algorithm is not enough to be considered efficient. For instance, even a quadratic-time algorithm can be totally useless in practice on a billion entries, which illustrates the importance of developing algorithms that are as efficient as possible. In particular, we shall see in this paper that the paradigm of quantum computing can be used to speed up several classical unsupervised learning algorithms.

Learning algorithms based on quantum primitives have already been developed in supervised (Anguita et al. 2003; Ezhov and Berman 2003) and reinforcement learning (Dong et al. 2005). However, not much work has been done yet concerning unsupervised learning, with the exception of a quantum algorithm for the minimal spanning tree (Dürr et al. 2004), which can also be used to perform clustering although this was not these authors' intention. See also Li et al. (2009, 2011) and Yu et al. (2010). Almost all the quantum algorithms described in this paper are based on variants of Grover's algorithm (Grover 1997). Our quantum versions of clustering algorithms offer a speed-up compared to their classical counterparts, but do not improve the quality of the resulting clustering process. In particular, if finding the optimal solution for a particular clustering problem is NP-hard (Garey and Johnson 1979), it is generally believed that quantum computers would also be unable to solve the problem exactly in polynomial time (Bennett et al. 1997).

The outline of this paper is as follows. First, Sect. 2 provides an overview of quantum information processing, in particular of Grover's algorithm and its variants. Some of these variants are further detailed because they are used as subroutines in our learning algorithms. Afterwards, Sect. 3 describes the black-box model adapted for the clustering context. The quantized versions of the clustering algorithms based on the minimum spanning tree, on divisive clustering and k-medians (standard and distributed versions) are detailed in Sects. 4, 5 and 6, respectively. Then, Sect. 7 describes a set of algorithms that can be used as tools during the preprocessing steps of other unsupervised learning algorithms: construction of a neighbourhood graph, detection of outliers and initialization of the cluster centres. Finally, Sect. 8 concludes this paper by a discussion and some perspectives on future extensions, such as a quantum version of ISOMAP, an algorithm for dimensionality reduction.

This paper is a thoroughly polished version of our earlier work (Aïmeur et al. 2007), which it extends in two important directions. First, in addition to speeding up the classical k-medians algorithm in Sect. 6.1, we demonstrate in Sect. 6.2 that quantum information can be used to reduce the amount of *communication* required to implement this algorithm in a distributed setting. Second, we have shown how to speed up classical algorithms for outlier detection (Sect. 7.2) and the "smart" initialization of cluster centres (Sect. 7.3).

2 Quantum information processing

In this section, we briefly review the notions of quantum information processing that are relevant to understanding our quantum versions of unsupervised learning algorithms. A detailed account of the field can be found in the book of Nielsen and Chuang (2000).

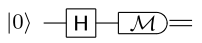

Fig. 1 Simple quantum circuit acting as a perfect random bit generator

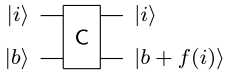

Fig. 2 Unitary computation of function f by quantum circuit C that implements U_f

2.1 Basic notions of quantum information processing

A *qubit* (or *quantum bit*) is the quantum analogue of the classical bit. In contrast with its classical counterpart, a qubit can exist in a *superposition* of states. For instance, an electron can be *simultaneously* on two different orbits of the same atom, which could represent classical bits 0 and 1, respectively. Formally, a qubit is represented as a unit vector in a two-dimensional Hilbert space. Using the Dirac notation, a qubit is described as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, where α and β are complex numbers called the *amplitudes* of classical states $|0\rangle$ and $|1\rangle$, respectively, subject to the *normalization condition* that $|\alpha|^2 + |\beta|^2 = 1$. When the state $|\psi\rangle$ is measured, either $|0\rangle$ or $|1\rangle$ is observed, with probability $|\alpha|^2$ or $|\beta|^2$, respectively. Furthermore, measurements are *irreversible* because the state of the system *collapses* to whichever value ($|0\rangle$ or $|1\rangle$) has been observed, thus losing all memory of former amplitudes α and β. The notion of qubit has a natural extension, which is the *quantum register*. A quantum register $|\psi\rangle$, composed of n qubits, lives in a 2^n-dimensional Hilbert space. Register $|\psi\rangle = \sum_{i=0}^{2^n-1} \alpha_i|i\rangle$ is specified by complex amplitudes $\alpha_0, \alpha_1, \ldots, \alpha_{2^n-1}$ subject to normalization condition $\sum |\alpha_i|^2 = 1$. Here, basis state $|i\rangle$ denotes the n-bit binary encoding of integer i. A quantum state such as $\frac{1}{\sqrt{2}}|01\rangle - \frac{1}{\sqrt{2}}|10\rangle$ is *entangled* if it cannot be described by the state of its individual qubits.

With the exception of measurements, all other operations allowed by quantum mechanics are *unitary* operations on the Hilbert space in which our qubits live. They are represented by *gates*, much as in a classical circuit. For instance, the *Walsh–Hadamard* gate H maps $|0\rangle$ to $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$ and $|1\rangle$ to $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. As an elementary example, consider the circuit shown in Fig. 1, where the single wires carry quantum information while the double wire holds classical information. In this example, we apply a Walsh–Hadamard gate to state $|0\rangle$, which yields $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, and then measure the quantum state. The subsequent measurement produces either 0 or 1, each with probability $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$, and the state collapses to the observed classical value. This circuit acts as a perfect random bit generator.

Fortunately (for implementation considerations), any unitary operation can be decomposed in terms of unary and binary gates. However, doing so efficiently (by a polynomial-size circuit) is often nontrivial and usually impossible. Nevertheless, any function f that can be computed by a classical circuit of size k can be computed by a quantum circuit C of size approximately $2k$, which implements unitary operation U_f. Because unitary operations must be reversible, we cannot in general simply go from $|i\rangle$ to $|f(i)\rangle$. Instead, we must map $|i, b\rangle$ to $|i, b + f(i)\rangle$, as illustrated in Fig. 2, where the addition is performed in an appropriate finite group and the second input is a quantum register of sufficient size to hold the output. It suffices to set b to zero at the input of the circuit in order to obtain $f(i)$.

This quantum circuit can then be applied to a superposition of different inputs: $U_f(\sum_i \alpha_i|i\rangle|0\rangle) = \sum_i \alpha_i|i\rangle|f(i)\rangle$. This gives rise to a potentially exponential amount of

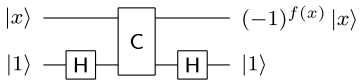

Fig. 3 Circuit to invert the phase of the target state. The output is identical to the input, with the possible exception of the phase, which is inverted if and only if $f(x) = 1$. This process is known as *phase kick-back* because it is in fact the state of the bottom qubit that evolves from $|1\rangle$ to $-|1\rangle$ whenever $f(x) = 1$. However, it is legitimate to think as if the phase flip were applied to the input qubits because $|x\rangle(-|1\rangle) = (-|x\rangle)|1\rangle$

parallelism, which has no classical counterpart. Amongst the most celebrated results, Shor (1997) has developed quantum algorithms for integer factorization and the computation of discrete logarithms, two problems of major importance in the field of cryptography. Another famous algorithm, due to Grover (1997), offers a quadratic speed-up in unstructured search problems. Grover's algorithm was analysed formally and generalized by Boyer et al. (1998). Several extensions of this algorithm are used in this paper as described below.

2.2 Grover's algorithm and its variants

In the original version of Grover's (1997) algorithm, a Boolean function f is given as a black box with the additional promise that there is a unique x_0 such that $f(x_0) = 1$. Classically, finding this x_0, when the function f displays no particular structure, requires on average approximately $\frac{n}{2}$ queries to this black box, for n the number of points in the domain of f. Grover's algorithm solves the same problem with approximatively $\sqrt{n}$ queries to the black box, but those are made in quantum superposition of classical states.

Grover's algorithm starts by applying a Walsh–Hadamard transformation on each qubit of an initial state composed of a sequence of $|0\rangle$'s in order to create an equal superposition of all the possible inputs.[1] Afterwards, the algorithm proceeds by repeating an adequate number of times *Grover's iteration*, which consists in two steps: a call to the quantum circuit shown in Fig. 3 inverts the phase of the *target state*, which is the unknown state $|x\rangle$ such that $f(x) = 1$, and an *inversion around the average* swings the amplitude of each state at the same distance but on the other side of the average amplitude of all states. It will be crucial in Sect. 6.2 that only the first of these two steps depends on the function f under consideration. One application of Grover's iteration has for effect to increase slightly the amplitude of the target state, while decreasing the amplitudes of the other states. After approximately $\frac{\pi}{4}\sqrt{n}$ iterations (Boyer et al. 1998), the amplitude of the target state will have risen very close to 1. Therefore, if the register is measured at that precise moment, the target state will be observed with quasi-certainty. This results in an $O(\sqrt{n})$ computational time.

Starting from the original idea of Grover, several generalizations of his algorithm have been developed for the cases in which there are more than one x such that $f(x) = 1$. For any known number $t > 0$ of solutions, the application of approximately $\frac{\pi}{4}\sqrt{n/t}$ Grover iterations is sufficient to find one of these solutions with high probability (Boyer et al. 1998) or even with certainty if the Grover iterations are modified very slightly, according to Theorem 4 of Brassard et al. (2002). If the number of solutions t is unknown, Boyer et al. (1998) show that it is nevertheless possible to find one of the solutions among the t possible ones in $O(\sqrt{n/t})$ expected time. Moreover, extensions to Grover's algorithm exist that can be used to *count* (exactly or approximately) the number of solutions (Brassard et al. 1998, 2002).

[1] We suppose throughout for simplicity that the number of points under consideration is a power of 2. If it is not, an equal superposition of the points can be obtained either with the use of a quantum Fourier transform or, in case measurement is allowable, by a postselection process.

Other applications of Grover's algorithm find the minimum (or maximum) of a function (and its position) (Dürr and Høyer 1996) or the c smallest (or highest) values of its image (Dürr et al. 2004) after $\Theta(\sqrt{n})$ and $\Theta(\sqrt{cn})$ calls to the function, respectively. Other variants can be used to approximate the median of a set of values or statistics that are related to it (Nayak and Wu 1999) with a quadratic speed-up compared to the best possible classical algorithm. Finally, the generalization of Grover's algorithm known as *amplitude amplification* (Brassard and Høyer 1997; Brassard et al. 2002) can accelerate many classical probabilistic algorithms by a quadratic factor in terms of the number of queries asked to the black box.

3 Quantization of clustering algorithms

Quantization refers to the process that partially or totally converts a classical algorithm to a quantum algorithm in order to improve its performance. In this section, we detail a model and tools that can be used to quantize unsupervised learning algorithms, among which specifically clustering algorithms. Although related, the task of quantizing a clustering algorithm should not be confused with the design of classical algorithms inspired from quantum mechanics (Horn and Gottlieb 2001, 2002) or the task of performing clustering directly on quantum information (Aïmeur et al. 2006).

3.1 Black-box model

Traditionally in clustering, we consider a training dataset composed of n datapoints denoted by $D_n = \{x_1, \ldots, x_n\}$, where each datapoint x_i corresponds to a vector of d attributes. The goal of a clustering algorithm is to partition D_n into subsets of points called *clusters*, such that the objects that are similar are grouped within the same cluster (*intra-similarity*) whereas dissimilar objects are placed in different clusters (*inter-dissimilarity*). One usual assumption is that there exists a notion of *distance* (or a *dissimilarity measure*) that can be evaluated to compare each pair of points. This distance is used by the algorithm to drive the formation of the clusters.

The model considered in most of this paper differs from this traditional machine learning framework and corresponds rather to the *black-box model*. In this model, our knowledge of the dataset comes uniquely from a black box (sometimes called *oracle*) that can be queried to learn the distance between two points. No assumptions are made *a priori* concerning the properties of this distance except that it is non-negative and that $Dist(x_i, x_i) = 0$ for all i. In particular, the triangle inequality need not hold and the distance may not be symmetric.[2] Throughout this paper, we shall nevertheless assume for simplicity that the distance function is symmetric, but most of our algorithms can be adapted *mutatis mutandis* without modifying their computational time significantly if this is not the case. This black-box model is generally used to derive lower bounds for problems for which it is difficult to prove such bounds in a more general context. Note that in this model, the complexity of an algorithm is measured in terms of the numbers of queries to the oracle, which constitutes a lower bound on the computational complexity of this problem. For simplicity, we shall henceforth refer to this number of queries as "time".

This black-box model is comparable to the one introduced by Angluin (1988) in classical computational learning theory, which is used to study the complexity of learning exactly a

[2]To be mathematically rigorous, the term *dissimilarity measure* would be more appropriate than the term *distance* when the symmetry property or the triangle inequality does not hold.

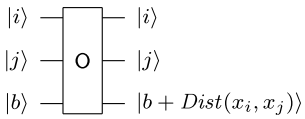

Fig. 4 Illustration of the distance oracle: i and j are the indexes of two points from D_n and $Dist(x_i, x_j)$ represents the distance between these two points. The addition $b + Dist(x_i, x_j)$ is computed in a finite group of appropriate size between the ancillary register b and the distance $Dist(x_i, x_j)$

function given as a black box. A quantum analogue of Angluin's model has been defined by Servedio (2001). The main difference is that our goal is to perform clustering and not to learn exactly a function computed by an oracle. To the best of our knowledge, the complexity of clustering in Angluin's model has not been studied before, whether it be in the classical or the quantum version. However, a similar problem was considered in the classical variant of the PAC setting (*Probably Approximately Correct*) of Valiant (1984), in which the goal was to characterize the time needed to learn a specific clustering (in the PAC sense of the term) among a class of possible clusterings (Mishra et al. 2001).

In the classical variant of the black-box model, a query to the oracle corresponds to asking for the distance between two points x_i and x_j by giving as inputs the indexes i and j of these two points. The corresponding quantum black box is illustrated in Fig. 4, where O stands for "oracle". In order to obey the principles of quantum mechanics, O must be unitary (and therefore reversible). In practice, this is not really a restriction as it is always possible to transform any classical irreversible circuit into a reversible one for a "reasonable" cost (Bennett 1973). Therefore, it is enough to specify the description of the classical irreversible circuit implementing the oracle in order to be able to convert it into a reversible circuit before implementing it quantum mechanically (at least in principle).

If the black box is quantum-mechanical, it can be interrogated in a superposition of inputs. For instance, if the input qubits are initially in state $|0\rangle$ and a Walsh–Hadamard gate is applied on each of these qubits (with the exception of the ancillary register, which is left unchanged to $|0\rangle$), the input will be transformed to an equal superposition of all the pairs of indexes of datapoints. In this specific situation, interrogating the oracle will result in a superposition of all the possible triplets $|i, j, Dist(x_i, x_j)\rangle$.

The assumption that a particular clustering problem is given as a black box is not realistic in practice, even though it is the usual paradigm considered in quantum information processing as well as in Angluin's model. However, Giovannetti et al. (2008) have designed general quantum random access memory (QRAM) architectures, which can be used to construct explicitly and efficiently a quantum circuit playing the role of the oracle.

3.2 Quantum subroutines

This section describes three quantum subroutines used to speed up classical clustering algorithms. All these subroutines are based on variations of Grover's algorithm. In particular, the first two are direct applications of previous work of Dürr and Høyer (1996) and Dürr et al. (2004), respectively. The third subroutine is a novel, albeit simple, application of Grover's algorithm.

The algorithm quant_find_max described thereafter (Algorithm 1) is directly inspired from the algorithm of Dürr and Høyer (1996) for finding the minimum of a function. It can be easily adapted to compute the maximum, hence retrieving the farthest pair of points in the dataset (the distance between these points is called its *diameter*). A similar algorithm can be used to identify the datapoint that is the farthest from a specific point.

Algorithm 1 quant_find_max(D_n)

Initialize $d_{max} = 0$
repeat
  Using Grover's algorithm, try to find indexes i and j such that $Dist(x_i, x_j) > d_{max}$
  Update $d_{max} = Dist(x_i, x_j)$
until no new i, j have been found
Return i, j

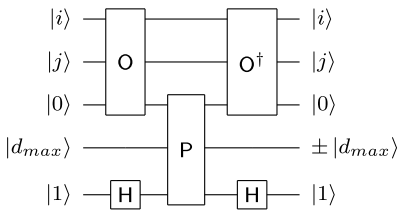

Fig. 5 Circuit implementing the phase flip in the variant of Grover's algorithm that retrieves the pair of points with maximum distance (Algorithm 1). The unitary transformation $\mathsf{O}^\dagger$ is the inverse of O and the subcircuit P is described in the following figure. The output is identical to the input, with the exception of the phase of $|d_{max}\rangle$, which is inverted by phase kick-back if and only if $Dist(x_i, x_j) > d_{max}$

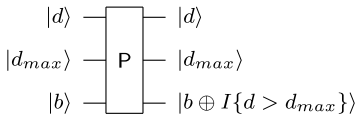

Fig. 6 Subcircuit P used in Fig. 5 to produce the desired phase kick-back, where $I\{\cdot\}$ is the *indicator function*, which evaluates to 1 if its argument is true and 0 otherwise, and "$\oplus$" denotes the sum modulo 2, also known as the exclusive-or

The algorithm starts by initializing d_{max} to zero. By using the circuit realizing the phase flip described in Figs. 5 and 6, Grover's algorithm attempts to find a new pair of points (i, j), if it exists, such that $Dist(x_i, x_j) > d_{max}$. If such a pair is not found, the algorithm terminates. Otherwise, the value of the distance d_{max} is updated to $Dist(x_i, x_j)$ and the process is repeated until the algorithm converges to the diameter with high probability.

Theorem 1 (Convergence of quant_find_max) *With high probability, algorithm* quant_find_max *returns the indexes i and j of the farthest pair of points in $O(\sqrt{p}) = O(n)$ expected time, where $p = n^2$ is the number of pairs in the dataset. In the simpler case, in which we are interested in retrieving the farthest point from a specific point, the corresponding algorithm takes $O(\sqrt{n})$ expected time.*

Proof The convergence proof of quant_find_max is similar to the analysis of Dürr and Høyer (1996) for their algorithm for finding the minimum of a function. □

Dürr et al. (2004) have developed an algorithm for finding the c smallest values of a function with high probability within $O(\sqrt{cn})$ time, for n the number of datapoints in the domain of the function. If we set this function to be the distance between a fixed point and

all the other points, we obtain the second subroutine quant_find_smallest_values, which is a direct application of the algorithm of Dürr et al. (2004) adapted for finding the c closest neighbours of a point.

Theorem 2 (Convergence of quant_find_smallest_values (Dürr et al. 2004)) *With high probability, algorithm* quant_find_smallest_values *finds the c closest neighbours of a point in* $O(\sqrt{cn})$ *time.*

Note that this algorithm for finding the c smallest values is more efficient, albeit more complicated, than simply applying the algorithm for finding the minimum c times, which would have taken $O(c\sqrt{n})$ time instead of $O(\sqrt{cn})$.

The third and last subroutine is a new algorithm, coined quant_find_median, which computes the median within an ensemble of n points $D_n = \{x_1, \ldots, x_n\}$.

Definition 1 (Median) The *median* of a set of points D is one whose sum of distances with all the other points is minimal (ties are broken arbitrarily): it is an $x \in D$ such that

$$\sum_{y \in D} Dist(x, y) \leqslant \sum_{y \in D} Dist(z, y) \tag{1}$$

for all $z \in D$. In the case of multidimensional points, the term *medoid* is sometimes used instead of the term median.

Finding the median can be realized classically by computing, for each point in the ensemble, the sum of its distances with all the other points and by taking the minimum. This process requires $O(n^2)$ time, where n is the number of points considered. In the generic case, where there are no properties of the distance to be used or no structure within the ensemble of points that can be exploited, this naïve solution is the most efficient possible. Indeed, consider a scenario in which all the points are at the same distance from one another, with the exception of two points that are closer to one another. These two points are the medians of this ensemble of points. Classically, it will be necessary to ask almost all the distances between pairs of points to the black box before it is possible to identify one of these two medians. This results in a lower bound of $\Omega(n^2)$ time.

When all the x_i correspond simply to numbers, or more generally when all the points are colinear, the quantum algorithm of Nayak and Wu (1999) can be used to approximate the median in $\Theta(\sqrt{n})$ time. However in the more generic case considered in this paper, our goal is to compute the median by using as sole information the distance between each pair of points, which corresponds to a situation for which the algorithm of Nayak and Wu does not apply.

To solve this problem quantum mechanically, we use the circuit S illustrated in Fig. 7, which takes $|i\rangle$ as input, with $1 \leqslant i \leqslant n$, and computes the sum of distances between x_i and all the other points in D_n. This can be achieved in $O(n)$ time by simply applying the black box described in Fig. 4 successively for each value of j, $1 \leqslant j \leqslant n$ (recall that $Dist(x_i, x_i) = 0$). A more efficient but approximate solution to this problem has recently been proposed by Brassard et al. (2011). We leave it for further research to determine the impact (or not) of this approach on clustering because it is not obvious to determine how detrimental to the *quality* of the resulting clustering would this use of *approximate* medians be.

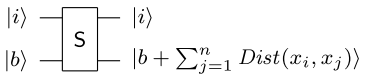

Fig. 7 Computation of the sum of distances between x_i and all the other points of the ensemble $D_n = \{x_1, \ldots, x_n\}$. The oracle S can be obtained by repeating n times the oracle O described in Fig. 4 for j ranging from 1 to n

Afterwards, the algorithm of Dürr and Høyer (1996) can be used to identify the minimal sum for all the possible x_i by using $\Theta(\sqrt{n})$ applications of the circuit of Fig. 7.

Lemma 1 (Convergence of quant_find_median) *With high probability, algorithm* quant_find_median *finds the median among a set of n points in* $O(n^{3/2})$ *time.*

Proof Each application of the circuit illustrated in Fig. 7 takes $O(n)$ time and finding the minimal sum among the n possible ones requires $O(\sqrt{n})$ applications of the circuit by using the quantum algorithm for finding the minimum (Dürr and Høyer 1996). Therefore, the algorithm requires $O(n\sqrt{n}) = O(n^{3/2})$ time to compute the median. □

4 Clustering via minimum spanning tree

Let $G = \langle V, E \rangle$ be a connected undirected graph for which V is the set of n vertices and E is the set of edges. Each edge has a *weight*, which is a positive real value.

Definition 2 (Spanning tree) A *spanning tree* is a subset of $n-1$ edges $T \subseteq E$ such that $\langle V, T \rangle$ constitutes a connected graph.

Definition 3 (Minimum spanning tree) A *minimum spanning tree* is a spanning tree for which the total sum of the weights of the edges is minimum among all the possible spanning trees.

One of the oldest techniques for performing clustering (Zahn 1971) is directly based on the minimum spanning tree. Consider for instance that each datapoint x_i of the dataset is the vertex of a graph and that each pair of vertices (x_i, x_j) is linked by an edge whose weight is proportional to some measure of distance $Dist(x_i, x_j)$. Once a minimum spanning tree for this graph has been constructed, it is easy to group the points into k clusters simply by removing the $k-1$ heaviest edges of this tree.

The clustering based on the minimum spanning tree maximizes a criterion that depends on the minimal distance between each cluster.

Definition 4 (Spacing) Let C_1 and C_2 be two disjoint clusters, the *spacing* between C_1 and C_2 is defined as

$$\text{Spacing}(C_1, C_2) = Dist_{min}(C_1, C_2) = \min_{x \in C_1, y \in C_2} Dist(x, y). \tag{2}$$

In words, it is the distance between the closest pair of points (x, y), for x belonging to the first cluster and y belonging to the second cluster.

The spacing is also sometimes called *minimal distance between two clusters* or *single linkage*.

Definition 5 (k-clustering with maximum spacing) A *k-clustering with maximum spacing* of the dataset D_n is a set of k clusters $C_1, C_2, \ldots, C_k$ partitioning D_n that maximizes

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \text{Spacing}(C_i, C_j). \tag{3}$$

The clustering based on the minimum spanning tree is precisely the one that maximizes this clustering criterion of maximum spacing (Gower and Ross 1969). Therefore, this corresponds to a situation for which a polynomial-time algorithm for maximizing the clustering criterion exists, which is not the case for most of the other clustering criteria.

Classically, when the graph is given as an adjacency matrix, this clustering problem can be solved directly by using Prim's algorithm (1957), which takes $O(n^2)$ time, where n is the number of vertices in the graph (or equivalently the number of points in the dataset). This algorithm is optimal since all classical algorithms require $\Omega(n^2)$ time in the case of a complete graph and an arbitrary metric.

The first quantum algorithm for clustering, although it had not been developed with this purpose in mind, is due to Dürr et al. (2004), who have studied the quantum complexity of some graph problems, including the minimum spanning tree. The models considered in their work are quantum versions of the *adjacency matrix* and the *adjacency list*. The adjacency matrix model, which is the only one considered here, is comparable to the black-box model described in Sect. 3.1. In this model, we give as input the indexes i and j of two vertices and we get as output the weight of the edge linking them (which, in our application, corresponds to the distance).

In practice, we must avoid at all cost the need to build a quantum circuit (or QRAM) that stores explicitly in its memory all the possible values of the adjacency matrix. Indeed, the construction of this circuit would require at least $\Omega(n^2)$ time because one would need to read each entry of the $n \times n$ matrix at least once. This is clearly not better than using directly Prim's algorithm. Intuitively, the situations for which it is interesting to apply Grover's algorithm (or one of its variants) are when it is possible to construct a quantum circuit defining the search space in less time than needed classically for generating the minimum spanning tree.

The algorithm of Dürr et al. (2004) is a quantization of a classical algorithm for the minimum spanning tree due to Borůvka (1926). Taking no account of the practical need to build a quantum circuit for the adjacency matrix (as mentioned above), it assumes that this matrix is given by an oracle.

Theorem 3 (Quantum algorithm for the minimum spanning tree) *The quantization of Borůvka's algorithm due to Dürr et al.* (2004) *can build the minimum spanning tree of a graph in $\Theta(n^{3/2})$ time, where n is the number of vertices of the graph. Moreover, this algorithm is optimal*: *in the case of a complete graph, no quantum algorithm can find the minimum spanning tree in less than $\Omega(n^{3/2})$ time.*

This theorem directly implies that it is possible to obtain a k-clustering with maximum spacing in $\Theta(n^{3/2})$ time, again on condition that an efficient quantum circuit to provide the distance between an arbitrary pair of points can be obtained in $O(n^{3/2})$ time.

Algorithm 2 quant_divisive_clustering(D)

if the points within D are sufficiently similar **then**
  Return D as a cluster
else
  Find points x_a and x_b that are the farthest apart within D using quant_find_max
  for each $x \in D$ **do**
    Attach x to the closest point among x_a and x_b
  end for
  Let D_a be the set of points attached to x_a
  Let D_b be the set of points attached to x_b
  Call quant_divisive_clustering(D_a)
  Call quant_divisive_clustering(D_b)
end if

5 Divisive clustering

One of the simplest ways to build a hierarchy of clusters is to start by assigning all the points to the same super-cluster. Afterwards, *divisive clustering* splits this super-cluster into two subclusters. Two datapoints are chosen to be the initial *seeds* of the two subclusters. A common technique is to choose as seeds the two points that are the farthest apart within the dataset. Afterwards, all the other points are attached to their closest seed. This division technique is then applied recursively on all the subclusters obtained until all the points contained within a cluster are sufficiently similar or some other stopping criterion is reached (see Algorithm 2 for more details).

Divisive clustering seeks to maximize at each step of the recursion a criterion that depends on the maximal distance between two clusters.

Definition 6 (Maximal distance between two clusters) The *maximal distance between two clusters* is the distance between the pair of points (x, y) that are the farthest apart, for x belonging to the first cluster and y belonging to the second cluster:

$$Dist_{max}(C_1, C_2) = \max_{x \in C_1, y \in C_2} Dist(x, y). \tag{4}$$

The costliest part of this algorithm is to identify the two points that are the farthest apart within the initial ensemble of n points. If the datapoints are vectors in $\mathbb{R}^d$ for an arbitrarily high dimension d, this process requires $\Omega(n^2)$ comparisons in the general case,[3] and this even in the case of approximations (Finocchiaro and Pellegrini 2002). Quantum mechanically, it is possible to use quant_find_max as a subroutine in this algorithm to find the two points that are the farthest apart in $O(n)$ time.

Theorem 4 (Quantum divisive clustering) *With high probability, algorithm* quant_divisive_clustering *performs divisive clustering of a set of n points with a gain of at least* $\Omega(\frac{n}{\log n})$ *over the classical version, as measured by the ratio between the classical and quantum times.*

[3]However, if this dimension d is low (such as $d \leqslant 3$), linear-time and sub-quadratic algorithms exist for some metrics such as the Euclidean distance (Preparata and Shamos 1985; Bespamyatnikh 1998).

Proof Suppose at first that at each recursive call, the algorithm splits the dataset into two sub-clusters of approximately the same size. This will lead to the construction of a balanced binary tree and the algorithm will have its execution time $T(n)$ characterized asymptotically by the recurrence $T(n) = 2T(n/2) + O(n)$, which is $O(n \log n)$. Classically, the recurrence for the same case is characterized by $T(n) = 2T(n/2) + \Omega(n^2)$ because of the time required to find the two points that are the farthest apart among the ensemble of points, which results in $\Omega(n^2)$ time. The ratio between the classical and quantum times is therefore

$$\Omega\left(\frac{n^2}{n \log n}\right) = \Omega\left(\frac{n}{\log n}\right).$$

On the other hand, in the unbalanced case in which the algorithm produces one cluster that contains a small number of points and another that concentrates all the global mass, the generated tree will be unbalanced and with a depth of n. In this case, this requires $O(n^2)$ global computational time quantum mechanically compared to of $\Omega(n^3)$ classically, which results in a gain of $\Omega(n^3/n^2) = \Omega(n)$. The gain between the classical and quantum versions is therefore even more pronounced in the unbalanced case, but it is $\Omega(\frac{n}{\log n})$ in all cases. □

In practice, if the clusters thus generated are highly unbalanced, this may be indicative of the presence of *outliers*. In this case, a common technique consists in detecting and removing these outliers before launching divisive clustering. This avoids the formation of sub-clusters that are too unbalanced. We discuss in Sect. 7.2 a quantized version of an algorithm for performing outlier detection.

6 *k*-medians

This section describes two quantum versions of the k-medians algorithm, the "standard" version, in which all the points are physically gathered in the same location, and the *distributed* version, in which the data is shared between two or more participants.

6.1 Standard version

The k-medians algorithm, also called k-medoids (Kaufman and Rousseeuw 1987), is a cousin of the k-means algorithm (Lloyd 1982). The cluster centres are often initialized as k points chosen at random among the n points of the dataset, k being a parameter of the algorithm corresponding to the number of desired clusters. (In Sect. 7.3, a quantum algorithm for initializing the clusters using a "smarter" approach is proposed.) The algorithm is iterative and each iteration is composed of two steps. During the first step, each datapoint is attached to its closest centre. Afterwards, during the second step, the centre of each cluster is updated as the median of all the points belonging to this cluster (recall that this is the point with minimal total distance to all the points in the cluster). The algorithm terminates when all the cluster centres are stabilized (or quasi-stabilized).

The k-medians algorithm seeks to partition the data into k clusters minimizing a distance criterion depending on the distance between all the points belonging to a cluster and the centre of this cluster.

Definition 7 (k-medians criterion) Consider a set of k clusters $C_1, C_2, \ldots, C_k$ partitioning D_n that have points $\mu_1, \mu_2, \ldots, \mu_k \in D_n$ for respective centres. These clusters are *optimal*

Algorithm 3 quantum_k_medians(D_n, k)

Choose uniformly at random k points that will be the initial cluster centres
repeat
  for each datapoint in D_n **do**
    Attach this point to its closest centre
  end for
  for each cluster Q **do**
    Compute the cluster's median using quant_find_median(Q); make it the new centre
  end for
until (quasi-)stabilization of the clusters
Return the clusters found and their respective centres

with respect to the *clustering criterion* of k-medians if they minimize

$$\sum_{i=1}^{k} \sum_{x \in C_i} Dist(x, \mu_i). \tag{5}$$

Contrary to the criteria used for the clustering algorithms described in the previous sections, optimizing the k-medians criterion is NP-hard for all $k \geqslant 2$ even in the Euclidean case (Papadimitriou 1981). Hence, no known polynomial-time algorithm exists for computing an optimal solution to this problem (a set of clusters that minimizes this criterion) even with the use of a quantum computer. The quantum version of the k-medians algorithm (Algorithm 3) is therefore a quantized version of a heuristic for solving an NP-hard problem, but not an algorithm for solving exactly this problem.

The main difference between k-means and k-medians is that the former chooses a virtual point called *centroid* as centre of a cluster, which corresponds to the average of the points belonging to this cluster, whereas the latter restricts the centre to be an actual point of the dataset. The k-means algorithm is guaranteed to converge to a stable assignment of the cluster centres after a finite number of iterations whereas the k-medians algorithm can reach a situation in which it oscillates between two (or more) configurations. This difference of convergence behaviour arises from the fact that the average is always uniquely defined whereas it is possible to have several perfectly valid medians for the same set of points. However, one of the advantages of k-medians over k-means is that it can be used even when the sole information available is the distance between points (and not their descriptions), which renders the computation of the average (and therefore the application of k-means) meaningless. Compared to its more famous cousin, the k-medians algorithm also offers the advantage of being more robust to noise and less sensitive to the presence of outliers within the data (see Sect. 7.2 for a brief explanation).

Theorem 5 (Standard quantum k-medians) *With high probability, algorithm* quantum_k_medians *computes the clustering of a set of n points with a gain of at least $\Omega(\sqrt{n/k})$ over the classical version, as measured by the ratio between the classical and the quantum times, where k is the number of clusters returned by the algorithm.*

Proof In order to analyse the efficiency of one iteration of the algorithm, suppose at first that all the clusters are approximately of the same size n/k. If all the medians were to be computed classically, each would require $\Theta((\frac{n}{k})^2)$ time, for a total of $\Theta(\frac{1}{k}n^2)$ time in order

to identify the centres of the k clusters. Quantum mechanically, the median of a cluster of size n/k can be obtained in $O(\frac{n}{k}\sqrt{\frac{n}{k}})$ time by using the subroutine quant_find_median. This leads to $O(n^{3/2}/\sqrt{k})$ time for one iteration of the quantum version of k-medians, which is $\Omega(\sqrt{n/k})$ times faster than the classical approach. Consider now the unbalanced scenario, in which all the clusters but one are of small constant size, whereas a unique cluster concentrates almost all the mass of datapoints. In this case, finding the median requires $\Omega(n^2)$ time classically, compared to $O(n^{3/2})$ time quantum mechanically, which is $\Omega(\sqrt{n})$ times faster. The gain between the classical and quantum versions is therefore even more pronounced in the unbalanced case, but it is $\Omega(\sqrt{n/k})$ in all cases. □

Note that the use of the quantum or classical version has no impact on the convergence rate of the algorithm, as measured by t, the number of iterations. In practice, however, the number of iterations and the quality of the clustering returned can be improved by performing a more "intelligent" initialization of the cluster centres (see Sect. 7.3).

Between two iterations of the k-medians algorithm, it is possible that the clusters stay relatively stable, meaning that only a small number of points are moved from one cluster to another. In this situation, it is likely that an appropriate (classical) data structure can keep track of the contents of clusters and their medians in order to accelerate the computation of medians from one iteration to the next. However, this potential improvement does not seem to have been the focus of much study in the (classical) literature. If it were, we could use the quantum version of k-medians during the first iterations before using the classical alternative based on the appropriate data structure when the situation becomes more stable. Alternatively, we could design an appropriate *quantum* data structure to bring about even more improvement. This is the topic of further research.

The quantum version of k-medians might also be improved by developing a quantum algorithm to estimate the sum of distances rather than simply adding them one by one as suggested in Fig. 7. Currently existing algorithms for estimating the mean (Grover 1998) do not seem to be appropriate for this purpose because of precision issues, but other methods based on *amplitude estimation* seem promising (Brassard et al. 2011). It is also plausible that the convergence of the algorithm, as measured by the number of iterations t, might be sped up with the help of quantum techniques. We leave this as another avenue for future research.

6.2 Distributed version

In the same manner that it is sometimes possible to speed up an algorithm by exploiting the quantum paradigm, in some cases it is possible to reduce the communication cost of a distributed protocol (de Wolf 2002; Brassard 2003; Buhrman et al. 2009). In a distributed setting, the dataset D_n is not localized in a single place but distributed among two (or several) participants. Equivalently, this can be seen as a situation in which each participant has his own dataset and their goal is to run a learning algorithm on the union of their datasets. For simplicity, we shall concentrate on the case of two participants and we assume that each of the attributes of any given datapoint can be represented with finite precision using a constant number of bits. This problem can be solved simply by gathering all the data in a central site and then running the standard version of the learning algorithm. In this case, the communication cost of the protocol would be of $\Theta(dm)$ bits, for d the dimension of the space in which the datapoints live and m the number of points of the smaller dataset. This can be very inefficient in practice if the size of the dataset is large. The main purpose of the theory of (classical) *communication complexity* (Kushilevitz and Nisan 1997) is to find

efficient protocols that require fewer bits of communication than what communicating the whole input would entail.

In a quantum distributed learning situation, the participants have the possibility of exchanging qubits instead of bits or to share prior entanglement in order to help them in performing their task. The main issue is whether or not the use of quantum information can help in decreasing the communication cost of some distributed protocols. Holevo's (1973) theorem rules out the possibility of transmitting more than n classical bits of information by communicating n qubits unless the participants share prior entanglement, in which case it is possible to transmit $2n$ classical bits by superdense coding (Bennett and Wiesner 1992), but this is the best possible (Cleve et al. 1999). Furthermore, it is impossible to use entanglement for instantaneous communication as this would entail faster-than-light communication. However, these limitations do not preclude the possibility of using quantum information to save significantly (quadratically or even exponentially) on the communication cost of computing some distributed functions (de Wolf 2002; Brassard 2003; Buhrman et al. 2009) or to achieve tasks that are provably impossible classically in a context in which communication between participants is ruled out (Brassard et al. 2005).

Suppose that the dataset D_n is shared between two participants, Alice and Bob. For the sake of simplicity, we assume that they share equally an even number n of datapoints.[4] In order for the quantum version of distributed k-medians to be interesting, it is necessary that its communication cost be lower than that of the trivial protocol, which would require sending all the data to one site, at a cost of $\Theta(dn)$ bits. The points in D_n can be reordered such that the first $n/2$ points correspond to Alice's dataset $D_a = \{x_1, \ldots, x_{n/2}\}$, whereas the rest constitutes Bob's dataset $D_b = \{x_{1+n/2}, \ldots, x_n\}$.

Alice can construct a quantum circuit E_a that encodes her dataset D_a, possibly in the form of a QRAM (Giovannetti et al. 2008). This circuit E_a takes as input the index i of a point from her dataset, for $i \in \{1, \ldots, \frac{n}{2}\}$, and produces the description x_i of this point. Bob can do the same with his own dataset D_b and build a circuit E_b that, from the index $j \in \{\frac{n}{2} + 1, \ldots, n\}$, returns the description x_j of this point. Moreover, if Alice has an implementation of the quantum circuit *Dist*, which computes the distance between two points whose description is given as inputs, she can use it in conjunction with E_a to implement a circuit S_a that computes the sum of distances between an arbitrary point whose description is known (even a point belonging to Bob's dataset) and all the points from her dataset D_a (the same goes for Bob, who can build circuit S_b). Quantum circuit *Dist* can be constructed by first designing an irreversible classical version of the distance circuit and then using Bennett's (1973) technique to make it reversible and thus implementable as a quantum circuit. Moreover, Alice and Bob can also construct $\mathsf{E}_a^\dagger$, $\mathsf{S}_a^\dagger$, $\mathsf{E}_b^\dagger$ and $\mathsf{S}_b^\dagger$, the inverses of E_a, S_a, E_b and S_b, respectively.

In order to implement the k-medians algorithm in a distributed manner, Alice and Bob have to perform distributively Grover's iteration, as used in the subroutine quant_find_median. During this protocol, they use and exchange three quantum registers. The first register, which encodes the index of a datapoint, is called the *index register* and is of size $\Theta(\log n)$. This is the only register that will *not* be exchanged during the protocol. The second register, which contains the description of a datapoint, is called the *description register* and is of size $\Theta(d)$. Finally, the third register, which contain the distance or a sum of

[4] If this assumption does not hold, the algorithm can be adapted without difficulty. However, if the sizes of the datasets of Alice and Bob are exceedingly unbalanced—for example if one of the participants has only $O(\sqrt{n})$ datapoints—the best solution is that the participant with the smallest dataset sends directly all his data to the other participant, who then can run the standard version of the learning algorithm on the whole dataset.

distances between one datapoint and several others, is called the *distance register*. We assume that we know an upper bound sum_dist_{max} on the maximum sum of distances between one point and all the other points[5] so that we can choose the distance register to be of size $\Theta(\log sum\_dist_{max})$. Indexes A and B are attached to the quantum registers to indicate who, respectively between Alice and Bob, has control over this register at a particular moment of the protocol.

Consider an iteration of the k-medians algorithm in which Alice and Bob need to find the median of cluster $Q = Q_a \cup Q_b$, where $Q_a \subseteq D_a$ and $Q_b \subseteq D_b$. Let i_a be the desired index of a point in Q_a whose sum of distances with all the other points of Q is minimal (i_b is defined in the same way for Q_b). The distributed Grover iteration, as given by Algorithm 4, is used to find i_a with quant_find_median, where d_{min} is initialized to $1 + sum\_dist_{max}$ and the index register is initialized to an equal superposition of the indices of all the points in Q_a before the first iteration. Here, circuits S_a and S_b (and their inverses $\mathsf{S}_a^\dagger$ and $\mathsf{S}_b^\dagger$) have been modified to take only account of the points in Q_a and Q_b, respectively.

Lemma 2 *The protocol* distributed_Grover_iteration (*Algorithm* 4), *which implements Grover's iteration in a distributed manner, can be used in the subroutine* quant_find_median. *Its communication cost is of* $O(d + \log sum\_dist_{max})$ *qubits, where* d *is the number of attributes used to describe the datapoints and* sum_dist_{max} *is an upper bound on the maximum sum of distances between one datapoint and all the other datapoints of* D_n.

Proof The step by step description of Algorithm 4 clearly demonstrates that it implements the functionality of Grover's iteration such as described in Sect. 2.2. In particular, the effect of the first part of Grover's iteration (Steps 1 to 8) is to invert the phase of all the points in Q_a whose sum of distances from all the points in Q is below the value d_{min}. The inversion around the average, which is performed during the last step of Grover's iteration, is independent of the functionality considered and can be realized locally by Alice. Concerning communication, only steps 3 and 6 require an exchange of information between Alice and Bob. The sizes of the registers exchanged are $O(d)$ for the description register and $O(\log sum\_dist_{max})$ for the distance register. The global communication cost is therefore $O(d + \log sum\_dist_{max})$ qubits. □

For each iteration of the k-medians algorithm and for each cluster, the subroutine quant_find_median is called in order to find i_a by applying the protocol distributed_Grover_iteration for implementing Grover's algorithm. Before calling distributed_Grover_iteration for the first time, Alice sets the index register in a superposition of all the indexes of the points in Q_a (the current cluster considered by her). Once i_a has been identified by using quant_find_median, i_b can be found by inverting the roles of Alice and Bob. Finally, to determine the median among $Q_a \cup Q_b$, it is sufficient to choose between i_a and i_b the one whose sum of distances with all the points in Q is minimum. Algorithm 5 formalizes the distributed version of k-medians.

Theorem 6 (Quantum k-medians distributed) *With high probability, algorithm* k_medians_distributed *clusters an ensemble of* n *points distributed between two participants at a* communication *cost of* $O(t\sqrt{kn}(d + \log sum\_dist_{max}))$ *qubits, where* n *is the number of points in the dataset,* d *the number of attributes used to describe these points,* k

[5] In the worst case, sum_dist_{max} is $O(n\sqrt{d})$ but a better upper bound could be known.

Algorithm 4 distributed_Grover_iteration($Q = Q_a \cup Q_b$)

[Step 1] Alice uses circuit E_a to produce in the description register the description x_i of a point from index i found in the index register, which will usually be in superposition.
$|i\rangle_A|0\rangle_A^{\otimes\Theta(d)} \overset{\mathsf{E}_a}{\mapsto} |i\rangle_A|x_i\rangle_A$.
(The index register is of size $\Theta(\log n)$ qubits and the description register is of size $\Theta(d)$.)
[Step 2] Alice calls the circuit S_a to compute the sum of distances between the point x_i and all the points in Q_a.
$|i\rangle_A|x_i\rangle_A|0\rangle_A^{\otimes\Theta(\log sum\_dist_{max})} \overset{\mathsf{S}_a}{\mapsto} |i\rangle_A|x_i\rangle_A|\sum_{x_j\in Q_a} Dist(x_i,x_j)\rangle_A$
(The distance register is of size $\Theta(\log sum\_dist_{max})$.)
[Step 3] Alice sends the description and distance registers to Bob.
$|i\rangle_A|x_i\rangle_A|\sum_{x_j\in Q_a} Dist(x_i,x_j)\rangle_A \overset{com}{\mapsto} |i\rangle_A|x_i\rangle_B|\sum_{x_j\in Q_a} Dist(x_i,x_j)\rangle_B$
[Step 4] Bob makes the description register interact with the description of his points by using the circuit S_b. This operation computes the sum of distances between x_i and all the points in $Q = Q_a \cup Q_b$.
$|i\rangle_A|x_i\rangle_B|\sum_{x_j\in Q_a} Dist(x_i,x_j)\rangle_B \overset{\mathsf{S}_b}{\mapsto} |i\rangle_A|x_i\rangle_B|\sum_{x_j\in Q} Dist(x_i,x_j)\rangle_B$
[Step 5] Let f be the Boolean function defined as

$$f(i) = \begin{cases} 1 & \text{if } \sum_{x_j\in Q} Dist(x_i,x_j) < d_{min}, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where d_{min} was initialized to $1 + sum\_dist_{max}$ before the first iteration.
Bob applies the conditional phase flip P, which realizes the following transformation:
$|i\rangle_A|x_i\rangle_B|\sum_{x_j\in Q} Dist(x_i,x_j)\rangle_B \overset{\mathsf{P}}{\mapsto} (-1)^{f(i)}|i\rangle_A|x_i\rangle_B|\sum_{x_j\in Q} Dist(x_i,x_j)\rangle_B$.
[Step 6] Bob applies $\mathsf{S}_b^\dagger$ to disentangle the distance register from his points in Q_b and sends back the description and distance registers to Alice.
$(-1)^{f(i)}|i\rangle_A|x_i\rangle_B|\sum_{x_j\in Q} Dist(x_i,x_j)\rangle_B \overset{\mathsf{S}_b^\dagger + com}{\mapsto} (-1)^{f(i)}|i\rangle_A|x_i\rangle_A|\sum_{x_j\in Q_a} Dist(x_i,x_j)\rangle_A$
[Step 7] Alice disentangles and discards the distance register by applying $\mathsf{S}_a^\dagger$.
$(-1)^{f(i)}|i\rangle_A|x_i\rangle_A|\sum_{x_j\in Q_a} Dist(x_i,x_j)\rangle_A \overset{\mathsf{S}_a^\dagger}{\mapsto} (-1)^{f(i)}|i\rangle_A|x_i\rangle_A$
[Step 8] Alice disentangles and discards the description register by applying $\mathsf{E}_a^\dagger$.
$(-1)^{f(i)}|i\rangle_A|x_i\rangle_A \overset{\mathsf{E}_a^\dagger}{\mapsto} (-1)^{f(i)}|i\rangle_A$
[Step 9] Alice applies locally the inversion around the average to the index register.

the number of clusters returned, t *the number of iterations of the algorithm and* sum_dist_{max} *is an upper bound on the maximum sum of distances between one datapoint and all the other datapoints of* D_n.

Proof Let us consider the two extremes. If the k clusters are approximatively of the same size n/k, each iteration of the algorithm requires that Alice and Bob exchange $O(\sqrt{n/k})$ times a register of size $O(d + \log sum\_dist_{max})$ per cluster, for a total of $O(k\sqrt{n/k}) = O(\sqrt{kn})$ registers exchanged per iteration. On the other hand, if one cluster contains almost all the n points and the others are of negligible size, each iteration of the algorithm requires that Alice and Bob exchange $O(\sqrt{n})$ times a register of the same size as in the other case. We see that the balanced case is the worst in terms of communication (whereas

Algorithm 5 k_medians_distributed(D_a, D_b, k)

Alice and Bob choose k points uniformly at random among D_a and D_b as being the initial cluster centres
repeat
  for each datapoint in D_a and D_b **do**
    Attach this point to its closest centre
  end for
  for each cluster $Q = Q_a \cup Q_b$ **do**
    Alice find i_a by using quant_find_median(Q) with the protocol distributed_Grover_iteration in order to perform Grover's iteration;
    Bob find i_b by using quant_find_median(Q) with the protocol distributed_Grover_iteration (the roles of Alice and Bob are exchanged) for performing Grover's iteration;
    median$(Q) = \arg\min_{i \in \{i_a, i_b\}} \sum_{x_j \in Q} Dist(x_i, x_j)$
  end for
until (quasi-)stabilization of the cluster
Return the clusters (distributed between Alice and Bob) and their centres

it was the best case in terms of computation *time* in the nondistributed setting studied in Sect. 6.1). In all cases, $O(\sqrt{kn})$ exchanges of a register of size $O(d + \log sum\_dist_{max})$ suffice, which results in a communication cost of $O(\sqrt{kn}(d + \log sum\_dist_{max}))$ qubits per iteration. The total communication cost is therefore of $O(t\sqrt{kn}(d + \log sum\_dist_{max}))$ qubits for all t iterations. □

This algorithm can be easily generalized to the multiparty case, for a number of participants $m \geqslant 2$. To do this, it is sufficient for each participant j (for $1 \leqslant j \leqslant m$) to find within his dataset the point i_j whose sum of distances with all the other points is minimum. This can be done by adapting the distributed version of Grover's iteration so that it works with more than 2 participants. Afterwards, the median can be chosen as the point among $\{i_1, \ldots, i_m\}$ that minimizes the sum of distances with all the other points of the cluster.

7 Quantum tools for unsupervised learning algorithms

The algorithms described in this section are not unsupervised learning algorithms by themselves, but they are often used as tools by other algorithms for unsupervised learning. Therefore, faster versions of these algorithms contribute directly to obtaining faster versions of algorithms for unsupervised learning.

7.1 Construction of a neighbourhood graph

The construction of a neighbourhood graph is an important step in the preprocessing of several unsupervised learning algorithms such as the algorithm Isomap for dimensionality reduction (Tenenbaum et al. 2000) and clustering by random walk (Harel and Koren 2001).

Definition 8 (Neighbourhood graph) Consider a complete undirected graph in which vertices correspond to points of a dataset and each edge between two vertices is weighted according to the distance between these two points. A *neighbourhood graph* is built from

Algorithm 6 quant_neighbourhood_graph(D_n, k)

for each datapoint x_i of D_n **do**
  Use quant_find_smallest_values for finding the k closest neighbours of x_i
  for each of the k closest neighbours of x_i **do**
    Create an edge between x_i and this neighbour, which is weighted according to the distance between these two points
  end for
end for
Return the constructed graph

this original graph by keeping for each vertex only the edges connecting it to its k closest neighbours.

An alternative notion of a neighbourhood graph could be defined by keeping an edge between two vertices if and only if they are in the close neighbourhood of each other. This approach guarantees that the maximum degree of the graph is less than or equal to k, the number of the closest neighbours considered. However, it would then be possible that some points become isolated (not connected) from the rest of the graph, which may indicate that they are outliers (see next section), but may also be seen as a drawback if the neighbourhood graph is given as input to an algorithm that requires a connected graph. We shall use Definition 8 henceforth.

Algorithm 6 is a quantized version of an algorithm for constructing a neighbourhood graph.

Theorem 7 (Quantum algorithm for constructing a neighbourhood graph) *With high probability, the algorithm* quant_neighbourhood_graph *constructs the neighbourhood graph of a set of n points in $O(\sqrt{k}\, n^{3/2})$ time, for k the number of neighbours considered.*

Proof For each datapoint, it is possible to obtain its k closest neighbours in $O(\sqrt{kn})$ time by using the subroutine quant_find_smallest_values. Therefore, construction of the global neighbourhood graph takes $O(\sqrt{k}\, n^{3/2})$ time. □

Classically, if we use an arbitrary metric and if the only information available is the distance between pairs of points, $\Omega(n^2)$ time is required to generate the neighbourhood graph. However, if we have explicitly access to all the d attributes describing these points and if d is small,[6] an appropriate data structure such as the *binary trees for multidimensional search* (Bentley 1975), also known as *kd-trees*, can be used to obtain the k closest neighbours of a specific point in $\Theta(k \log n)$ time. The building of a kd-tree requires us to sort all the points for each dimension, which can be done in $\Theta(dn \log n)$ time, where d is the dimension of the space in which the datapoints live and n is the number of datapoints. Therefore, it takes $\Theta((k+d)n \log n)$ time to construct the kd-tree representing a dataset *and* use it for finding the k closest neighbours of each of the n points, provided d small.

[6] Kibriya and Frank (2007) review a study comparing different methods and data structures to speed up the search of the closest neighbours. This study empirically demonstrates that for any $d \geqslant 16$, all these methods become sensitive to the *curse of dimensionality* and require a time worse than linear in the number of points for identifying the closest neighbours of a particular point.

7.2 Outlier detection

Definition 9 (Outlier) An *outlier* is an observation that differs significantly from the rest of the data.

In practice, the exact interpretation of this definition depends on the context considered. In some contexts, outliers are considered as being datapoints generated by a noise process and therefore must be removed if possible. For example, the outlier might be a datapoint that has been corrupted, either by a change in the value of some of its attributes or because it has been labelled with a wrong class. It might even be the case that this observation is purely random and does not carry any meaningful information.

When performing classification, it is desirable to be able to detect and remove outliers that are present inside a dataset in order to improve the accuracy of the classifier that will be learnt on this data. Similarly in clustering, the ability to recognize outliers makes it possible to avoid taking them into account during the formation of clusters and therefore to improve the quality of the resulting clustering. However, in some applications, such as fraud detection for credit card usage or intrusion detection, it is important to be able to detect outliers as they might correspond to an unusual behaviour that we want to discover.

If we know in advance the distribution from which the data have been sampled (for instance by a model learned through density estimation or by some *a priori* knowledge), a statistical test can be used to identify the datapoints that diverge significantly from this distribution (Yamanishi et al. 2004). Another way to detect an outlier is by inspecting the attributes of each point and then considering as potential outliers those *differing significantly from the median value*. The median is generally used instead of the mean because it is less influenced by the presence of outliers since the mean might be greatly influenced by some extreme values. This also explains why the k-medians algorithm (described in Sect. 6) is generally considered as being less sensitive to outliers than its cousin the k-means algorithm (Massart et al. 1986). Other *density-based methods* (Breunig et al. 2000) inspect the neighbourhood of each point and take the points located in low density areas as potential outliers. In practice, this method gives very good results but is often costly in terms of computational resources. Finally, another approach relies on a *notion of distance*, which labels as outliers the points that are at far distant from their k closest neighbours. Indeed, a possible technique (Angiulli and Pizzuti 2002) consists in identifying, for each point x_i, its k closest neighbours and to give a *neighbourhood score* ω_i to this point equal to

$$\omega_i = \sum_{x_j \in V_k(x_i)} Dist(x_i, x_j), \tag{7}$$

where $V_k(x_i)$ is the subset of D_n consisting in the k closest neighbours of x_i. Once this score has been computed for each point, outliers can be identified as points whose score ω is above a threshold ω_{max} (determined empirically), or as the c points that have the highest score, for a well-chosen constant c. Algorithm 7 is a quantum version of this idea, which computes these scores for each point. Classically, $\Omega(n^2)$ time is needed to determine the neighbourhood scores of all n points in the general case.

Theorem 8 (Quantum algorithm for outlier detection based on distances) *With high probability, the algorithm* quant_detection_outlier *identifies all the outliers, within a dataset of size n, in $O(\sqrt{k}\, n^{3/2})$ time, where k is the number of neighbours considered for each point.*

Algorithm 7 quant_detection_outlier(D_n, k, ω_{max})

Use quant_neighbourhood_graph(D_n, k) to build the neighbourhood graph
for each point x_i in D_n **do**
  Compute ω_i as being the sum of distance between x_i and its k closest neighbours
  Identify x_i as an outlier if $\omega_i > \omega_{max}$
end for
Return the outliers thus identified

Algorithm 8 quantum_initialization_centres(D_n, k)

Choose at random a datapoint in D_n, which is labelled as μ_1
for $i = 2$ to k **do**
  Use quant_find_max for finding $\mu_i = \arg\max_{x \in D_n} \sum_{j=1}^{i-1} Dist(x, \mu_j)$
end for
Return $\mu_1, \ldots, \mu_k$ as initial cluster centres

Proof By virtue of Theorem 7, it takes $O(\sqrt{k}\, n^{3/2})$ time to compute the neighbourhood graph. Once this is done, $O(k)$ time is needed for each of the n nodes to compute its score and decide if it is an outlier. Therefore, the algorithm takes $O(\sqrt{k}\, n^{3/2} + kn)$ total time, which is $O(\sqrt{k}\, n^{3/2})$ since $k \leqslant n$. □

7.3 Initialization of the cluster centres

Traditionally, the initial cluster centres in algorithms such as k-means or k-medians are chosen at random among all the points from the dataset. Starting from two different initial configurations, the algorithm has a non-negligible probability of converging towards two different clusterings. Moreover, as it is NP-hard to optimize the cost functions of k-means and k-medians (Papadimitriou 1981), it is likely that both clusterings thus generated correspond at best to *local* minima. A standard technique for alleviating this problem is to run the algorithm several times from different initial configurations and to keep only the clustering that minimizes the cost function considered.

A different approach is to choose the initial cluster seeds in a "smart" way instead of completely at random. An algorithm of the *max-min* (Dasgupta and Long 2005) type starts by choosing the first centre μ_1 at random among all the points from the dataset. Afterwards, the second centre μ_2 is chosen as the point with maximum distance from μ_1. The following centres $\mu_3, \ldots, \mu_k$ are chosen one by one, always taking as next centre the datapoint whose sum of distances with all the previous centres is maximum. Formally, we define the ith centre μ_i as:

$$\mu_i = \arg\max_{x \in D_n} \sum_{j=1}^{i-1} Dist(x, \mu_j). \tag{8}$$

This method produces initial cluster seeds that are well-scattered and distant from one another. Algorithm 8 is a quantum version of this method for initializing the cluster centres.

Theorem 9 (Quantum algorithm for the initialization of the cluster centres) *With high probability, the algorithm* quantum_initialization_centres *initializes in a "smart" way the k cluster centres of a set of n points in* $O(k^2\sqrt{n})$ *time.*

Proof The algorithm quantum_initialization_centres chooses the first centre at random among all the possible points of the dataset. Afterwards, the second centre is determined as the farthest point from this original centre by using the quant_find_max subroutine, which requires $O(\sqrt{n})$ time. Then for each of the following centres μ_i, for $3 \leqslant i \leqslant k$, we can compute it through $i-1$ applications of the oracle O, and then calling the subroutine quant_find_max on the register of distances, which take $O((i-1)\sqrt{n})$ time. Therefore, this algorithm takes $O(k^2\sqrt{n})$ time in total, which is obtained from summing the arithmetic series $\sum_{i=1}^{k-1} i\sqrt{n}$. □

The algorithm quantum_initialization_centres can be used to find a clustering composed of k disjoint clusters in which the maximum diameter, among all these clusters, is approximately minimized. The main goal here is to create an ensemble of k clusters with small maximum diameter. Formally, this algorithm seeks to minimize

$$\max_{1\leqslant i\leqslant k} \max_{x,y\in C_i} Dist(x,y) \tag{9}$$

over all clusterings $C_1, C_2, \ldots, C_k$. This criterion is NP-hard to optimize in its exact version, but an approximation due to González (1985) returns a set of k clusters whose maximum diameter is at most twice that of the optimal k-clustering of minimal maximum diameter, provided the distance function is defined in a Euclidean space. González's algorithm starts by choosing the k cluster centres as explained previously, which classically requires $O(k^2 n)$ time. Afterwards, each point of the dataset is attached to its closest centre. The quantum version of this algorithm requires $O(k^2\sqrt{n})$ time to identify the cluster centres.

8 Conclusion and future avenues of research

As seen in this paper, some unsupervised learning algorithms can be accelerated beyond what can be achieved classically by making some of their subroutines quantum mechanical. Moreover, in some distributed learning scenarios, it is possible to save on the communication cost if the participants are allowed to exchange quantum information or if they share prior entanglement, such as the distributed version of k-medians (Sect. 6.2). In this conclusion, we summarize the different results presented in this paper and open new avenues of research for quantum unsupervised learning algorithms.

8.1 Fair comparison between classical and quantum learning algorithms and lower bounds

In order to make a fair comparison between classical and quantum clustering scenarios, it is important to consider the best known classical algorithm when the only information available is the distance between pairs of points as well as when a complete description of the datapoints is available. For instance, when building a neighbourhood graph, the classical algorithm for kd-trees computes this graph so efficiently that quantizing the classical algorithm working only on the distances offers no significant advantage if the dimension d is small (Kibriya and Frank 2007). A fundamental open question is the study of the lower bounds for different clustering scenarios, be they classical or quantum mechanical. In particular, can we characterize precisely the situations in which the quantized version of the clustering algorithm offers a speed-up not only on its classical counterpart but on all possible classical algorithms?

Table 1 Table summarizing the lower and upper bounds in the time (or communication cost) required by unsupervised learning algorithms studied in this paper

Clustering problem	Classical	Quantum mechanical
Minimum spanning tree/k-clustering with maximum spacing	$\Theta(n^2)$	$\Theta(n^{3/2})$
Divisive clustering (in the case of balanced sub-clusters)	$\Theta(n^2)$	$\Omega(n)$, $O(n \log n)$
k-medians (standard)	$\Omega(\frac{n^2}{k})$, $O(t\frac{n^2}{k})$	$\Omega(n)$, $O(t\frac{1}{\sqrt{k}}n^{3/2})$
k-medians (distributed), communication cost	$\Theta(dn)$	$\Omega(d\sqrt{n})$, $O(t\sqrt{kn}(d+\log sum\_dist_{max}))$
Construction of a neighbourhood graph (for d a medium or high dimension)	$\Theta(n^2)$	$\Omega(n)$, $O(\sqrt{k}n^{3/2})$
Outlier detection (based on the neighbourhood graph)	$\Theta(n^2)$	$\Omega(n)$, $O(\sqrt{k}n^{3/2})$
"Smart" initialization of the cluster centres	$\Omega(n)$, $O(k^2n)$	$\Omega(\sqrt{n})$, $O(k^2\sqrt{n})$

For example, in the case of the minimum spanning tree (Sect. 4), Dürr et al. (2004) have proven that their algorithm is optimal. Therefore, if one could give a reduction between the problem of k-clustering with maximum spacing and the construction of the minimum spanning tree, this would imply that no clustering algorithm that tries to maximize the criterion of k-clustering with maximum spacing, whether it be classical or quantum mechanical, can take less than $\Omega(n^{3/2})$ time. Table 1 summarizes the results seen so far in this paper.

These bounds are relatively tight for the clustering via minimum spanning tree, for the divisive clustering and for the "smart" initialization of the cluster centres. Concerning the construction of the neighbourhood graph and the detection of outliers, it seems likely that the true lower bound is closer from $\Omega(n^{3/2})$ than from $\Omega(n)$. If this intuition were formally proven, this would have for consequence that the algorithms presented in this paper are essentially optimal. On the other hand, for the quantum version of the k-medians algorithm, there is still a important gap between the lower bound and the upper bound, which leaves open the possibility of finding a more efficient quantum algorithm. In particular, the impact of the new approximate median-finding algorithm of Brassard et al. (2011) should be investigated.

Another fundamental question is to determine the type of clustering algorithms admitting a distributed version in which the transmission of quantum information can lead to a saving on the communication cost (compared to the classical versions). So far, we have seen that the k-medians algorithm admits an efficient distributed quantum version. The same kind of idea could also be applied to obtain a distributed version of the "smart" initialization of the cluster centres. This simply involves using the distributed version of Grover's iteration in the computation of quant_find_max. Both distributed algorithms have a communication cost of $\tilde{O}(d\sqrt{kn})$ qubits (not taking into account the log factors) and return as output only $O(k)$ bits of information corresponding to the cluster centres. On the other hand, algorithms such as clustering via minimum spanning tree output $\Omega(n)$ bits of information (for instance the description of the minimum spanning tree), which seems not easily compressible. It follows that these algorithms are unlikely candidates for the distributed setting because their communication cost would be at least as much as simply sending directly all the data to a single entity.

8.2 Quantization of dimensionality reduction algorithm

Isomap (Tenenbaum et al. 2000) is an unsupervised algorithm for learning a low-dimensional representation of data coming from non-linear varieties. More specifically, Isomap assumes that the observed data, which live in high dimension, have been generated by a low-dimensional manifold. The main idea of the algorithm is to approximate the *geodesic distance* between two points on this curvature by the length of the shortest path between these two points on a neighbourhood graph. Once the geodesic distance has been estimated for each pair of points of the dataset, multidimensional scaling (Cox and Cox 1994) is used on the distance matrix in order to learn a low-dimensional representation of the data. The computational bottleneck of Isomap, as well as of most of the dimensionality reduction algorithms, stems from the computation of the eigenvalues and eigenvectors of $n \times n$ matrices, which requires in general $O(n^3)$ time. The cost of these eigencomputations dominates the computational complexity of these algorithms. Concerning the determination of the eigenvalues, some quantum algorithms, such as amplitude estimation (Brassard et al. 2002), use the quantum Fourier transform to estimate the eigenvalues more quickly than classically achievable. However, with respect to the determination of the eigenvectors of a matrix, to the best of our knowledge there exist no quantum algorithms that are more efficient than their classical counterparts. We leave as open question the possibility of developing a quantum algorithm for determining the eigenvalues and the eigenvectors of an $n \times n$ matrix more efficiently than it is classically possible, which would directly lead to a panoply of quantum algorithms for dimensionality reduction.

8.3 Other research avenues

Grover's algorithm, and its generalization amplitude amplification (Brassard and Høyer 1997; Brassard et al. 2002), are applicable in a wide range of situations but offer at best a quadratic speed-up compared to the best classical algorithm. More recently, other algorithmic approaches based on *quantum walks* (Kempe 2003; Ambainis 2003) or *quantum Markov chains* (Szegedy 2004; Magniez et al. 2007; Sántha 2008) have emerged. These techniques seem promising and capable of overcoming the intrinsic limits of Grover's algorithm. In particular, there are strong similarities and links between some learning algorithms and some quantum algorithmic techniques, which might be exploited to develop new quantum learning algorithms. An example of promising candidate is a classical clustering algorithm based on random walks (Harel and Koren 2001).

Better than quadratic improvement over classical algorithms are likely to emerge only when intrinsically quantum algorithms, without any classical counterpart, will be developed. A paradigm shift on the computation as well as the representation model, such as adopting the point of view of the measurement-based model (Raussendorf and Briegel 2001) or adiabatic computation (Farhi et al. 2000), might lead to the development of such algorithms. These two models offer the same computational power as the "traditional" model of the quantum circuits (Aharonov et al. 2004; Broadbent and Kashefi 2009), but bring a different perspective and might result in the discovery of new algorithms that can more naturally be phrased in these models than in the form of circuits. Indeed, the measurement-based model is an ideal framework for expressing parallel or distributed algorithms, while the adiabatic computation model seems more appropriate for building optimization algorithms, such as a quantum version of simulated annealing.

References

Aharonov, D., van Dam, W., Kempe, J., Landau, J., Lloyd, S., & Regev, O. (2004). Adiabatic quantum computation is equivalent to standard quantum computation. In *Proceedings of 45th IEEE symposium on foundations of computer science* (pp. 42–51).

Aïmeur, E., Brassard, G., & Gambs, S. (2006). Machine learning in a quantum world. In *Proceedings of the 19th Canadian conference on artificial intelligence* (pp. 433–444).

Aïmeur, E., Brassard, G., & Gambs, S. (2007). Quantum clustering algorithms. In *Proceedings of the 24th international conference on machine learning* (pp. 1–8).

Ambainis, A. (2003). Quantum walks and their algorithmic applications. *International Journal of Quantum Information*, *1*(4), 507–518.

Angiulli, F., & Pizzuti, C. (2002). Fast outlier detection in high dimensional spaces. In *Proceedings of the 6th European conference on principles and practice of knowledge discovery in databases* (pp. 15–26).

Angluin, D. (1988). Queries and concept learning. *Machine Learning*, *2*, 319–342.

Anguita, D., Ridella, S., Riviecco, F., & Zunino, R. (2003). Quantum optimization for training support vector machines. *Neural Networks*, *16*(1), 763–770.

Bennett, C. H. (1973). Logical reversibility of computation. *IBM Journal of Research and Development*, *17*, 525–532.

Bennett, C. H., & Wiesner, S. J. (1992). Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, *69*, 2881–2884.

Bennett, C. H., Bernstein, E., Brassard, G., & Vazirani, U. (1997). Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, *26*(5), 1510–1523.

Bentley, J. L. (1975). Multidimensional binary search tree used for associative searching. *Communications of the ACM*, *18*(9), 509–517.

Bespamyatnikh, S. N. (1998). An efficient algorithm for the three-dimensional diameter problem. In *Proceedings of the 9th symposium on discrete algorithms* (pp. 137–146).

Borůvka, O. (1926). O jistém problému minimálním. *Práce Moravské Přírodovědecké Společnosti*, *3*, 37–58.

Boyer, M., Brassard, G., Høyer, P., & Tapp, A. (1998). Tight bounds on quantum searching. *Fortschritte der Physik*, *46*, 493–505.

Brassard, G. (2003). Quantum communication complexity. *Foundations of Physics*, *33*(11), 1593–1616.

Brassard, G., & Høyer, P. (1997). An exact quantum polynomial-time algorithm for Simon's problem. In *Proceedings of the 5th Israel symposium on theory of computing and systems* (pp. 12–23).

Brassard, G., Høyer, P., & Tapp, A. (1998). Quantum counting. In *Proceedings of the 25th international conference on automata, languages and programming* (pp. 820–831).

Brassard, G., Høyer, P., Mosca, M., & Tapp, A. (2002). Quantum amplitude amplification and estimation. In S. J. Lomonaco Jr. (Ed.), *Quantum computation and quantum information* (pp. 53–74).

Brassard, G., Broadbent, A., & Tapp, A. (2005). Quantum pseudo-telepathy. *Foundations of Physics*, *35*, 1877–1907.

Brassard, G., Dupuis, F., Gambs, S., & Tapp, A. (2011). An optimal quantum algorithm to approximate the mean and its application for approximating the median of a set of points over an arbitrary distance. arXiv:1106.4267.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on management of data* (pp. 93–104).

Broadbent, A., & Kashefi, E. (2009). Parallelizing quantum circuits. *Theoretical Computer Science*, *410*(26), 2489–2510.

Buhrman, H., Cleve, R., Massar, S., & de Wolf, R. (2009). Non-locality and communication complexity. arXiv:0907.3584.

Cleve, R., van Dam, W., Nielsen, M., & Tapp, A. (1999). Quantum entanglement and the communication complexity of the inner product function. In *Proceedings of the first NASA international conference on quantum computing and quantum communications* (pp. 61–74).

Cox, T., & Cox, M. (1994). *Multidimensional scaling*. London: Chapman and Hall.

Dasgupta, S., & Long, P. M. (2005). Performance guarantee for hierarchical clustering. *Journal of Computer and System Sciences*, *70*(4), 555–569.

de Wolf, R. (2002). Quantum communication and complexity. *Theoretical Computer Science*, *287*(1), 337–353.

Dong, D., Chen, C., & Chen, Z. (2005). Quantum reinforcement learning. In *Proceedings of the first international conference on advances in natural computation* (pp. 686–689).

Dürr, C., & Høyer, P. (1996). A quantum algorithm for finding the minimum. arXiv:quant-ph/9607014.

Dürr, C., Heiligman, M., Høyer, P., & Mhalla, M. (2004). Quantum query complexity of some graph problems. In *Proceedings of the 31st international conference on automata, languages and programming* (pp. 481–493).

Ezhov, A. A., & Berman, G. P. (2003). *Introduction to quantum neural technologies*. Princeton: Rinton Press.
Farhi, E., Goldsone, J., Gutmann, S., & Sipser, M. (2000). Quantum computation by adiabatic evolution. arXiv:quant-ph/0001106.
Finocchiaro, D. V., & Pellegrini, M. (2002). On computing the diameter of a point set in high dimensional Euclidean space. *Theoretical Computer Science*, *287*(2), 501–514.
Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. New York: Freeman.
Giovannetti, V., Lloyd, S., & Maccone, L. (2008). Architectures for a quantum random access memory. *Physical Review A*, *78*(5), 052310.
González, T. F. (1985). Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, *38*, 293–306.
Gower, J. C., & Ross, G. J. S. (1969). Minimum spanning trees and single linkage cluster analysis. *Applied Statistics*, *18*(1), 54–64.
Grover, L. K. (1997). Quantum mechanics helps in searching for a needle in a haystack. *Physical Review Letters*, *79*(2), 325–328.
Grover, L. K. (1998). A framework for fast quantum mechanical algorithms. In *Proceedings of the 30th ACM symposium on theory of computing* (pp. 53–62).
Harel, D., & Koren, Y. (2001). On clustering using random walks. In *Proceedings of the 21st conference on foundations of software technology and theoretical computer science* (pp. 18–41).
Holevo, A. S. (1973). Bounds for the quantity of information transmitted by a quantum mechanical channel. *Problems of Information Transmission*, *9*, 177–183.
Horn, D., & Gottlieb, A. (2001). The method of quantum clustering. In *Proceedings of the neural information processing systems* (pp. 769–776).
Horn, D., & Gottlieb, A. (2002). Algorithms for data clustering in pattern recognition problems based on quantum mechanics. *Physical Review Letters*, *88*(1), 018702.
Kaufman, L., & Rousseeuw, P. (1987). Clustering by means of medoids. In Y. Dodge (Ed.), *Statistical data analysis based on the L_1-norm and related methods* (pp. 405–416).
Kempe, J. (2003). Quantum random walks—An introductory overview. *Contemporary Physics*, *44*(4), 307–327.
Kibriya, A. M., & Frank, E. (2007). An empirical comparison of exact nearest neighbour algorithms. In *Proceedings of the 11th European conference on principles and practice of knowledge discovery in databases* (pp. 140–151).
Kushilevitz, E., & Nisan, N. (1997). *Communication complexity*. Cambridge: Cambridge University Press.
Li, Q., He, Y., & Jiang, J.-p. (2009). A novel clustering algorithm based on quantum games. *Journal of Physics A: Mathematical and Theoretical*, *42*, 445303.
Li, Q., He, Y., & Jiang, J.-p. (2011). A hybrid classical-quantum clustering algorithm based on quantum walks. *Quantum Information Processing*, *10*(1), 13–26.
Lloyd, S. P. (1982). Least square quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137. First made available as a Bell Telephone Laboratories Paper (1957).
Magniez, F., Nayak, A., Roland, J., & Sántha, M. (2007). Search via quantum walk. In *Proceedings of the 39th ACM symposium on theory of computing* (pp. 575–584).
Massart, D. L., Kaufman, L., & Rousseeuw, P. J. (1986). Least median of squares: a robust method for outlier and model error detection in regression and calibration. *Analytica Chimica Acta*, *187*, 171–179.
Mishra, N., Oblinger, D., & Pitt, L. (2001). Sublinear time approximate clustering. In *Proceedings of the 12th symposium on discrete algorithms* (pp. 439–447).
Nayak, A., & Wu, F. (1999). The quantum query complexity of approximating the median and related statistics. In *Proceedings of the 31st ACM symposium on theory of computing* (pp. 384–393).
Nielsen, M. A., & Chuang, I. L. (2000). *Quantum computation and quantum information*. Cambridge: Cambridge University Press.
Papadimitriou, C. H. (1981). Worst-case and probabilistic analysis of a geometric location problem. *SIAM Journal on Computing*, *10*(3), 542–557.
Preparata, F. P., & Shamos, M. I. (1985). *Computational geometry: an introduction*. New York: Springer.
Prim, R. C. (1957). Shortest connecting networks and some generalizations. *The Bell System Technical Journal*, *36*(6), 1389–1401.
Raussendorf, R., & Briegel, H. J. (2001). A one-way quantum computer. *Physical Review Letters*, *88*(22), 5188–5191.
Sántha, M. (2008). Quantum walk based search algorithms. In *Proceedings of 5th international conference on theory and applications of models of computation* (pp. 31–46).
Servedio, R. (2001). Separating quantum and classical learning. In *Proceedings of the 28th international conference on automata, languages and programming* (pp. 1065–1080).

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, *26*(5), 1484–1509.

Szegedy, M. (2004). Quantum speed-up of Markov chain based algorithms. In *Proceedings of 45th IEEE symposium on foundations of computer science* (pp. 32–41).

Tenenbaum, J. B., de Silva, V., & Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, *290*(5500), 2319–2323.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*, 1134–1142.

Witten, I. H., & Frank, E. (2005). *Data mining: practical machine learning tools and techniques* (2nd ed.). San Mateo: Morgan Kaufmann.

Yamanishi, K., Takeuchi, J., Williams, G. J., & Milne, P. (2004). On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. *Data Mining and Knowledge Discovery*, *8*(3), 275–300.

Yu, Y., Qian, F., & Liu, H. (2010). Quantum clustering-based weighted linear programming support vector regression for multivariable nonlinear problem. *Soft Computing*, *14*(9), 921–929.

Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, *20*(1), 68–86.