Quantum Clustering Algorithms

Esma Aïmeur AIMEUR@IRO.UMONTREAL.CA
Gilles Brassard BRASSARD@IRO.UMONTREAL.CA
Sébastien Gambs GAMBSSEB@IRO.UMONTREAL.CA
Université de Montréal, Département d'informatique et de recherche opérationnelle
C.P. 6128, Succursale Centre-Ville, Montréal (Québec), H3C 3J7 Canada

Abstract

By the term "quantization", we refer to the process of using quantum mechanics in order to improve a classical algorithm, usually by making it go faster. In this paper, we initiate the idea of quantizing clustering algorithms by using variations on a celebrated quantum algorithm due to Grover. After having introduced this novel approach to unsupervised learning, we illustrate it with a quantized version of three standard algorithms: divisive clustering, k-medians and an algorithm for the construction of a neighbourhood graph. We obtain a significant speedup compared to the classical approach.

1. Introduction

Unsupervised learning is the part of machine learning whose purpose is to give to machines the ability to find some structure hidden within data. Typical tasks in unsupervised learning include the discovery of "natural" clusters present in the data (*clustering*), finding a meaningful low dimensional representation of the data (*dimensionality reduction*) or learning explicitly a probability function (also called density function) that represents the true distribution of the data (*density estimation*). Given a training data set, the goal of a clustering algorithm is to group similar datapoints in the same cluster while putting dissimilar datapoints in different clusters. Some possible applications of clustering algorithms include: discovering sociological groups existing within a population, grouping automatically molecules according to their structures, clustering stars according to their galaxies, and gathering news or papers according to their topic.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

Multidisciplinary by nature, *Quantum Information Processing* (QIP) is at the crossroads of computer science, mathematics, physics and engineering. It concerns *the implications of quantum mechanics for information processing purposes* (Nielsen & Chuang, 2000). Quantum information is very different from its classical counterpart: it cannot be measured reliably and it is disturbed by observation, but it can exist in a superposition of classical states. Classical and quantum information can be used together to realize wonders that are out of reach of classical information processing alone, such as being able to factorize efficiently large numbers, with dramatic cryptographic consequences (Shor, 1997), search in a unstructured database with a quadratic speedup compared to the best possible classical algorithms (Grover, 1997) and allow two people to communicate in perfect secrecy under the nose of an eavesdropper having at her disposal unlimited computing power and technology (Bennett & Brassard, 1984).

Machine learning and QIP may seem *a priori* to have little to do with one another. Nevertheless, they have already met in a fruitful manner (see the survey of Bonner & Freivalds, 2002, for instance). In this paper, we seek to speed-up some classical clustering algorithms by drawing on QIP techniques. It is important to have efficient clustering algorithms in domains for which the amount of data is huge such as bioinformatics, astronomy and Web mining. Therefore, it is natural to investigate what could be gained in performing these clustering tasks if we had the availability of a quantum computer.

The outline of the paper is as follows. In Section 2, we review some basic concepts of QIP, in particular Grover's algorithm and its variations, which are at the core of our clustering algorithm quantizations. In Section 3, we introduce the concept of quantization as well as the model we are using. We also briefly explain in that section the quantum subroutines based on Grover's algorithm that we are exploiting in order to

speed-up clustering algorithms. Then, we give a quantized version of divisive clustering, k-medians and the construction of a c-neighbourhood graph, respectively, in Sections 4, 5 and 6. Finally, we conclude in Section 7 with a discussion of the issues that we have raised.

2. Quantum Information Processing

Quantum information processing draws its uncanny power from three quantum resources that have no classical counterpart. *Quantum parallelism* harnesses the superposition principle and the linearity of quantum mechanics in order to compute a function simultaneously on arbitrarily many inputs. *Quantum interference* makes it possible for the logical paths of a computation to interfere in a constructive or destructive manner. As a result of interference, computational paths leading to desired results can reinforce one another, whereas other computational paths that would yield an *un*desired result cancel each other out. Finally, there exist multi-particle quantum states that cannot be described by an independent state for each particle (Einstein, Podolsky & Rosen, 1935). The correlations offered by these states cannot be reproduced classically (Bell, 1964) and constitute an essential resource of QIP called *entanglement*.

2.1. Basic Concepts

In this section, we briefly review some essential notions of QIP. A detailed account of the field can be found in the book of Nielsen and Chuang (2000). 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. Formally, using the Dirac notation, a qubit can be 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 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.

All other operations allowed by quantum mechanics are *reversible* (and even *unitary*). 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 $\frac{1}{\sqrt{2}}|0\rangle - \frac{1}{\sqrt{2}}|1\rangle$. Figure 1 illustrates the notions seen so far, where time flows from left to right. Note that a *single line* carries quantum information, whereas a *double line* carries classical information; $\mathcal{M}$ denotes a measurement.

Figure 1. Example of a simple quantum circuit.

In this very simple example, we apply a Walsh–Hadamard gate to state $|0\rangle$, which yields $\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$. The subsequent measurement produces either 0 or 1, each with probability $|\frac{1}{\sqrt{2}}|^2 = 1/2$, and the state collapses to the observed classical value. This circuit can be seen as a perfect random bit generator.

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 binary encoding of integer i. Unitary operations can also be applied to two or more qubits. Fortunately (for implementation considerations), any unitary operation can always be decomposed in terms of unary and binary gates. However, doing so efficiently (by a polynomial-size circuit) is often nontrivial.

Figure 2 illustrates the process by which a function f is computed by a quantum circuit C. Because unitary operations must be reversible, we cannot in general simply go from $|x\rangle$ to $|f(x)\rangle$. Instead, we must map $|x, b\rangle$ to $|x, b + f(x)\rangle$, where the addition is performed in an appropriate finite group and the second input is a quantum register of sufficient size. In case of a Boolean function, $|b\rangle$ is a single qubit and we use the sum modulo 2, also known as the exclusive-or and denoted "$\oplus$". In all cases, it suffices to set b to zero at the input of the circuit in order to obtain $f(x)$.

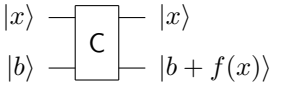

Figure 2. Unitary computation of function f.

When f is a Boolean function, it is often more convenient to compute f in a manner that would have no classical counterpart: if x is the classical input, we *flip* its quantum phase from $+|x\rangle$ to $-|x\rangle$ (or vice versa) precisely when $f(x) = 1$. This process, which is achieved by the circuit given in Fig. 3, is particularly

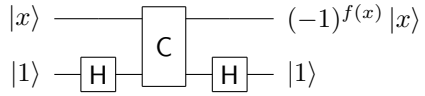

Figure 3. Computing a function by phase flipping.

interesting when it is computed on a superposition of all (or some) inputs. That operation plays a key role in Grover's algorithm (Section 2.2).

2.2. Grover's Algorithm and Variations

In the original version of Grover's algorithm (Grover, 1997), we are given a Boolean function f as a black box and we are promised that there exists a unique x_0 such that $f(x_0) = 1$. Classically, finding that x_0 would require an average of $n/2$ queries of the black box, where n is the number of points in the domain of f. Grover's algorithm solves the same problem after roughly $\sqrt{n}$ accesses to the black box, but of course those accesses are made in quantum superposition.

Grover's algorithm starts by using a stack of Walsh–Hadamard gates on the all-zero state in order to create an equal superposition of all possible inputs. It then proceeds by repeating the so-called *Grover iteration*, which is composed of two steps: a call to the quantum circuit given in Fig. 3, which flips the phase of the unknown x such that $f(x) = 1$ (the "target state") and an *inversion about the average*, which is independent of f. This iteration has to be repeated roughly $\frac{\pi}{4}\sqrt{n}$ times. The effect of a single Grover iteration is to slightly increase the amplitude of the target state, while decreasing the amplitudes of the other states. After the right number of Grover iterations, the amplitude of the target state is very close to 1, so that we are almost certain to obtain it if we measure the register at that time.

Following Grover's original idea, generalizations of his algorithm have been developed that deal with the case in which there are more than a single x so that $f(x) = 1$. In that case, roughly $\frac{\pi}{4}\sqrt{n/t}$ Grover iterations should be applied before measuring (Boyer, Brassard, Høyer & Tapp, 1998), where t is the number of solutions. In case the number t of solutions is unknown, the same paper shows that it remains possible to find one of them in a time proportional to $\sqrt{n/t}$. Other extensions of Grover's algorithm have been developed, in which it is possible to *count* (either exactly or approximately) the number of solutions (Brassard, Høyer, Mosca & Tapp, 2002).

Several applications of Grover's algorithm have been developed to find the minimum of a function (Dürr & Høyer, 1996) and the c smallest values in its image (Dürr, Heiligman, Høyer & Mhalla, 2004) after $\Theta(\sqrt{n}\,)$ and $\Theta(\sqrt{cn}\,)$ calls on the function, respectively. Other applications can approximate the median or related statistics (Nayak & Wu, 1999) with a quadratic gain compared to the best possible classical algorithms.

3. Quantization of Clustering Algorithms

As a motivating example, consider the following scenario, which corresponds to a highly challenging clustering task. Imagine that you are an employee of the Department of Statistics of the United Nations. Your boss comes to you with the complete demographic data of all the inhabitants of Earth and asks you to analyse this data with a clustering algorithm in the hope of discovering meaningful clusters. Seeing how reluctant you seem to be in front of all this data, he tells you not to worry because, in order to help you achieve this task, he was able to "borrow" the prototype of a full-size quantum computer from the National Security Agency. Can this quantum computer be used to speed-up the clustering process?

By the term *quantization*, we refer to the process of starting from a classical algorithm and converting it into a quantum algorithm in order to improve it [1], generally by making it go faster. The first quantized clustering algorithm, although it was not developed for this purpose, is due to Dürr, Heiligman, Høyer and Mhalla (2004). They have studied the quantum query complexity of graph problems and developed among other things a quantized version of Borůvka's algorithm (1926), capable of finding the minimum spanning tree of a graph in a time in $\Theta(n^{3/2})$, where n is the number of vertices in the graph [2]. Suppose that each datapoint x_i of the training set is represented by a vertex and that each pair of vertices (x_i, x_j) is linked by an edge whose weight is proportional to some distance measure $Dist(x_i, x_j)$. Once the minimal spanning tree of this graph has been computed, it is easy to group the datapoints into k clusters by removing the $k-1$ longest edges of this tree.

Although related, the task of quantizing clustering algorithms should not be confused with the design of classical clustering algorithms inspired from quantum mechanics (Horn and Gottlieb 2001; 2002) or the task of performing clustering directly on quantum states (Aïmeur, Brassard & Gambs, 2006).

3.1. The Model

In traditional clustering, the assumption is made that the training data set D_n is composed of n points, denoted $D_n = \{x_1, \ldots, x_n\}$. Each datapoint x corre-

[1] Not to be confused with an alternative meaning of quantization, which is to divide a continuous space into discrete pieces.

[2] In the case of a complete graph, all possible classical algorithms require a time in $\Omega(n^2)$.

sponds to a vector of attributes. For instance, $x \in \mathbb{R}^d$ if points are described by d real attributes. The goal of a clustering algorithm is to partition the set D_n in subsets of points called *clusters*, such that similar objects are grouped together within the same cluster (*intra-similarity*) and dissimilar objects are put in different clusters (*inter-dissimilarity*). A notion of *distance* (or a similarity measure) between each pair of points is assumed to exist and is used by the algorithm to decide on how to form the clusters.

In this paper, we depart from this traditional setting by adopting instead the framework of the *black box model*. Specifically, we assume that our knowledge concerning the distance between points of the training data set is available solely through a black box, also known as an "oracle". We make no *a priori* assumptions on the properties of this distance, except that it is symmetric [3] and non-negative. (In particular, the triangle inequality need not hold.) This model is close in spirit to the one imagined by Angluin (1988), which is used in computational learning theory to study the query complexity of learning a function given by a black box. A quantum analogue of Angluin's model has been defined by Servedio (2001). The main difference between Angluin's model and ours is that we are not interested in learning a function but rather in performing clustering [4].

In the classical black-box setting, a query corresponds to asking for the distance between two points x_i and x_j by providing indexes i and j to the black box. In accordance with the general schema given in Fig. 2, the corresponding quantum black box is illustrated in Fig. 4; we call it O (for "oracle"). In particular, it is possible to query the quantum black box in superposition of entries. For instance, if we apply the Walsh–Hadamard gate to all the input qubits initially set to $|0\rangle$ (but leave the $|b\rangle$ part to $|0\rangle$), we can set the entry to be a superposition of all the pairs of indexes of datapoints. In that case, the resulting output is a superposition of all the triples $|i, j, \mathit{Dist}(x_i, x_j)\rangle$. [5]

[3] If the distance is not symmetric, the algorithms presented here can easily be modified at no significant increase in the running time.

[4] We are not aware of prior work in the study of clustering complexity in Angluin's model, be it in the classical or quantum setting. However, a similar problem has been considered in the classical PAC (Probably Approximately Correct) learning setting (Mishra, Oblinger & Pitt, 2001). The issue was to study the number of queries that are necessary to learn (in the PAC learning sense) a specific clustering from a class of possible clusterings.

[5] Not to be confused with simply a superposition of all the distances between pairs of points, which would make no quantum sense in general.

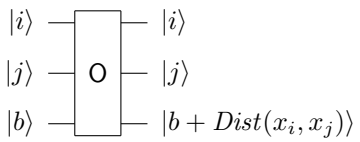

Figure 4. Illustration of the distance oracle: i and j are the indexes of two points from D_n and $\mathit{Dist}(x_i, x_j)$ represents the distance between them. The addition $b + \mathit{Dist}(x_i, x_j)$ is performed in an appropriate finite group between the ancillary register b and the distance $\mathit{Dist}(x_i, x_j)$.

The explicit construction of O from a particular training set D_n is a fundamental issue, which we discuss in Section 7. For now, we simply assume that the clustering instance to be solved is given as a black box, which is the usual paradigm in quantum information processing as well as in Angluin's classical model.

3.2. Quantum Subroutines

In this section, we present three quantum subroutines, which we are going to use in order to accelerate classical clustering algorithms. All these subroutines are variations on Grover's algorithm. In fact, the first two are straightforward applications of former work by Dürr et al. (1996; 2004), although they are fine-tuned for our clustering purposes. The third subroutine is a novel, albeit simple, application of Grover's algorithm.

The quant_find_max algorithm described below (Algorithm 1) is directly inspired by the algorithm of Dürr and Høyer (1996). It serves to find the pair of points that are farthest apart in the data set (the distance between those two points is called the "diameter" of the data set). A similar algorithm, which we do not need in this paper, would find the datapoint that is most distant from one specific point.

Algorithm 1 quant_find_max(D_n)

Choose at random two initial indexes i and j
Set $d_{max} = \mathit{Dist}(x_i, x_j)$
repeat
  Using Grover's algorithm, find new indexes i and j such that $\mathit{Dist}(x_i, x_j) > d_{max}$ provided they exist;
  Set $d_{max} = \mathit{Dist}(x_i, x_j)$
until no new i, j are found
return i, j

The algorithm starts by choosing uniformly at random two indexes i and j. A first guess for the diameter is obtained simply as $d_{max} = \mathit{Dist}(x_i, x_j)$. By virtue of the phase-flipping circuit described in Figures 5 and 6, Grover's algorithm is then used to find a new pair (i, j) of points, if it exists, such that $\mathit{Dist}(x_i, x_j) > d_{max}$. If no such pair exists, we have found the diameter and the algorithm terminates. Otherwise, the tenta-

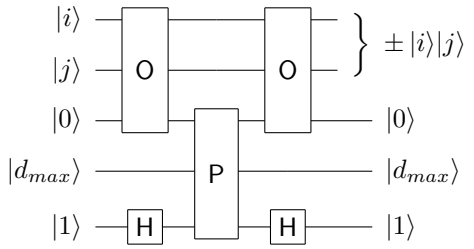

Figure 5. Phase-flipping component of Grover's algorithm, in which the output is identical to the input, except that the global phase of $|i\rangle|j\rangle$ is flipped if and only if $Dist(x_i, x_j) > d_{max}$. See Fig. 6 for definition of P.

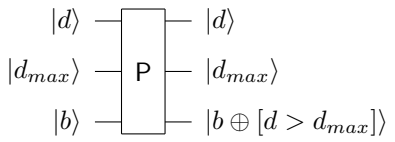

Figure 6. Sub-circuit P for use in Fig. 5, where $[x]$ is the *Iverson bracket* defined by $[x] = 0$ if x is false and $[x] = 1$ otherwise, and "$\oplus$" denotes the exclusive-or.

tive distance d_{max} is updated to be $Dist(x_i, x_j)$ and the procedure is repeated. It follows from the analysis of Dürr and Høyer (1996) that convergence happens after an expected number of queries in the order of $\sqrt{p}$, where $p = n^2$ is the number of pairs of datapoints, hence the total number of queries is in $O(n)$.

The second subroutine we are going to use for the quantization of classical clustering algorithms is directly inspired by the algorithm for finding the c smallest values of a function, due to Dürr, Heiligman, Høyer and Mhalla (2004). We call this subroutine quant_find_c_smallest_values. Finding the minimum of a function can be seen as a special case of the application of this algorithm for the case $c = 1$. Using the approach that we have just explained for quant_find_max, it is possible to adapt this algorithm to search for the c closest neighbours of a point in a time in $\Theta(\sqrt{cn}\,)$.

Our third and last subroutine is a novel algorithm, which we call quant_cluster_median, for computing the median of a set of m points $Q_m = \{z_1, \ldots, z_m\}$. When the z_i's are simply numbers or, more generally, when all the points are colinear, the quantum algorithm of Nayak and Wu (1999) can be used to find the median in a time in $\Theta(\sqrt{m}\,)$. However, we shall need to find medians in the more general case in which all we know about the points is the distance between each pair (the triangle inequality need not hold), when the algorithm of Nayak and Wu (1999) does not apply.

By definition, the median of Q_m is a point within the set whose sum (or average) distance to all the other points is minimum. This notion of median is particularly intuitive in the L_1–norm sense but can be generalized to other situations (see the survey of Small, 1990, for instance).

Finding the median can be done classically by computing for each point inside the set its sum distance to all the other points and taking the minimum. This process requires a time in $\Theta(m^2)$, again when m is the number of points considered. In the general case in which there are no restrictions on the distance function, we are not aware of a more efficient classical approach. Quantum mechanically, we can easily build the quantum circuit illustrated in Fig. 7, which takes $|i\rangle$ as input, $1 \leq i \leq m$, and computes the sum of the distances between z_i and all the other points in Q_m. For this, it suffices to apply the black box of Fig. 4 successively with each value of j, $1 \leq j \leq m$. (We assume that $Dist(z_i, z_i) = 0$.) This takes a time in $\Theta(m)$, but see Section 7 for possible improvements.

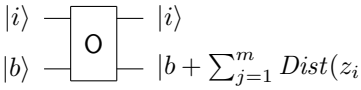

Figure 7. Computing the sum of distances between z_i and all the other points in the set $Q_m = \{z_1, \ldots, z_m\}$.

The minimum-finding algorithm of Dürr and Høyer (1996) can then be used to find the minimum such sum over all possible z_i with $\Theta(\sqrt{m}\,)$ applications of the circuit of Fig. 7. Since each application of the circuit takes a time in $\Theta(m)$, the overall time to compute the median is in $\Theta(m\sqrt{m}\,) = \Theta(m^{3/2})$.

4. Divisive Clustering

One of the simplest ways to build a hierarchy of clusters is by starting with all the points belonging to a single cluster. The next step is to split this cluster into two subclusters. For this purpose, the two datapoints that are the farthest apart are chosen as seeds. Afterwards, all the other points are attached to their nearest seed. This division technique is then applied recursively on the resulting subclusters until all the points contained inside a cluster are sufficiently similar. See Algorithm 2 for details.

The part of this algorithm that is the most costly is to find the two farthest points within the initial data set of n points. If the datapoints are given as vectors in $\mathbb{R}^d$ for an arbitrarily high dimension d, this process generally requires $\Theta(n^2)$ comparisons [6]. Quantum mechanically, however, we can use quant_find_max as a sub-

[6] However, if d is small (such as $d = 1, 2$ or 3) and we are using a metric such as the Euclidean distance, linear or subquadratic algorithms are known to exist.

Algorithm 2 Div_clustering(D)

if points in D are sufficiently similar **then**
  return D as a cluster
else
  Find the two farthest points x_a and x_b in D using quant_find_max
  for each $x \in D$ **do**
    Attach x to the closest between x_a and x_b
  end for
  Set D_a to be all the points attached to x_a
  Set D_b to be all the points attached to x_b
  Call Div_clustering(D_a)
  Call Div_clustering(D_b)
end if

routine to this algorithm, which finds the two farthest points in a time in $\Theta(n)$, as we have seen.

For the sake of simplicity, let us analyse the situation if the algorithm splits the data set in two subclusters of roughly the same size.[7] This leads to the construction of a balanced tree and the algorithm has a global running time $T(n)$ given by asymptotic recurrence $T(n) = 2T(n/2) + \Theta(n)$, which is in $\Theta(n \log n)$.

5. *k*-medians

The k-medians algorithm, also called k-medoids, (Kaufman & Rousseeuw, 1987) is a cousin of the better-known k-means clustering algorithm. It is an iterative algorithm, in which an iteration consists of two steps. During the first step, each datapoint is attached to its closest cluster centre. During the second step, the centre of each cluster is updated by choosing among all the points composing this cluster the one that is its median. The algorithm stops when the centre of the clusters have stabilized (or quasi-stabilized). The algorithm is initialized with k random points chosen as starting centres, where k is a parameter supplied to the algorithm, which corresponds to the desired number of clusters.

The main difference between k-means and k-medians is that k-means is allowed to use a virtual centroid that is simply the average of all the points inside the cluster. In contrast, for k-medians we restrict the centre of the cluster to be a "real" point of the training set. One advantage of k-medians over k-means is that it can be applied even if the only information available about the points is the distance between them, in which case it may be impossible to compute averages, hence to apply the k-means algorithm.

[7] Admittedly, this is not an altogether realistic assumption, especially if the data set contains outliers. However, in that case, we should begin by following the usual classical practice of detecting and removing those outliers before proceeding to divisive clustering.

Algorithm 3 k-medians(D,k)

Choose k points uniformly at random to be the initial centres of the clusters
repeat
  for each datapoint in D **do**
    Attach it to its closest centre
  end for
  for each cluster Q **do**
    Compute the median of the cluster and make it its new centre
  end for
until (quasi-)stabilization of the clusters
return the clusters found and their centres

In order to analyse the efficiency of one iteration of this algorithm, let us assume for simplicity that the clusters have roughly the same size n/k. (If not, the advantage of our quantum algorithm compared to the classical approach will only be more pronounced.) If the medians were computed classically, each of them would need a time in $\Theta((\frac{n}{k})^2)$, for a total of $\Theta(\frac{1}{k}n^2)$ for finding the centres of all k clusters. Quantum mechanically, we have seen that it is possible to compute the median of a cluster of size n/k in a time in $\Theta(\frac{n}{k}\sqrt{\frac{n}{k}})$ using the quant_cluster_median subroutine. This yields a running time in $\Theta(\frac{1}{\sqrt{k}}n^{3/2})$ for one iteration of the quantum k-medians algorithm, which is $\sqrt{n/k}$ times faster than the classical approach, everything else being the same in terms of convergence rate.

6. Construction of a *c*-neighbourhood Graph

The construction of a neighbourhood graph is an important part in several unsupervised learning algorithms such as ISOMAP (Tenenbaum, de Silva & Langford, 2000) or the clustering method by random walk (Harel & Koren, 2001). Suppose that the points of the training set are the vertices of a complete graph, where an edge between two vertices is weighted according to the distance between these two datapoints. A c-neighbourhood graph can be obtained by keeping for each vertex only the edges linking it to its c closest neighbours. Algorithm 4 gives a quantized algorithm for the construction of a c-neighbourhood graph.

For each datapoint, we can find its c closest neighbours in a time in $\Theta(\sqrt{cn})$ using quant_find_c_smallest_values. This leads to a total cost in $\Theta(n^{3/2})$ for computing the global c-neighbourhood graph, provided we set c to be

Algorithm 4 c-neighbourhood_graph_construction(D,c)

for each datapoint x_i of D **do**
  Use quant_find_c_smallest_values to find the c closest neighbours of x_i
  for each c closest neighbours of x_i **do**
    Create an edge between x_i and the current neighbour, which is proportional to the distance between these two points
  end for
end for
return the computed graph

a constant. Classically, if we have to deal with an arbitrary metric and we know only the distance between pairs of points, this would require a time in the order of $\Theta(n^2)$ to find the closest neighbours for each of the n points. However, if we have access for each datapoint to all the d attributes that describe it, it is possible to use Bentley's multidimensional binary search trees, known as kd-trees [8] (Bentley, 1975), to find the c closest neighbours of a specific datapoint in a time in $\Theta(c \log n)$. The construction of the kd-tree requires to sort the datapoints according to each dimension, which can be done in a time in $\Theta(dn \log n)$, where d is the dimensionality of the space in which the datapoints live and n is the number of datapoints.

7. Discussion and Conclusion

In this paper, we have seen how to speed up a selection of classical clustering algorithms by quantizing some of their parts. However, the approach we have used is not necessarily realistic because it requires the availability of a quantum black box that can be used to query the distance between pairs of points in superposition. Even though this is the model commonly used in quantum information processing, we reckon that, in real life, we might not be given directly such a black box. Instead, we would be more likely to be given a training data set D_n that contains the description of n datapoints. An important issue is how to construct ourselves, from this training set, an efficient quantum circuit that has the same functionality as the black box we had assumed throughout this paper. We recognize that this is a fundamental question, but it is currently beyond the scope of this paper.

We believe that our quantized version of the k-medians algorithm (Section 5) can be improved even further by developing a quantum algorithm to estimate the sum of a set of values instead of simply adding them one by one as we propose in Fig. 7. Currently known algorithms to estimate the average (Grover, 1998) cannot be used directly because of precision issues, but methods based on amplitude estimation (Brassard, Høyer, Mosca & Tapp, 2002) are promising.

In order to make a fair comparison between a classical clustering algorithm and its quantized counterpart, it is also important to consider the best possible classical algorithm and the advantage that can be gained *if* we have a full description of the datapoints, rather than just the distance between them. For instance, in the case of the construction of a c-neighbourhood graph, we have seen in Section 6 that classical kd-trees can be used to compute this graph so efficiently that it may not be possible to gain a significant improvement by quantizing the algorithm. It is therefore important to study also the lower bounds that can be achieved for different clustering settings, both classically and quantum mechanically. In particular, in which situation can (or cannot) the quantized version provide a significant improvement? For instance, in the case of clustering with a minimal spanning tree, Dürr, Heiligman, Høyer and Mhalla (2004) have proved that their algorithm is close to optimal. It follows that no clustering algorithm based on the construction of a minimal spanning tree, be it quantum or classical, can do better than $\Omega(n^{3/2})$.

Among the possible extensions to the study initiated in this paper, we note that the quantization approach could be applied to other clustering algorithms. Moreover, this quantization does not need to be restricted to using only variations on Grover's algorithm: it could also use other techniques from the quantician's toolbox, such as quantum random walks (Ambainis, 2003) or quantum Markov chains (Szegedy, 2004). Developing entirely new quantum clustering algorithms instead of simply quantizing some parts of classical algorithms is a most interesting research avenue, which could lead to more spectacular savings. Finally, we believe that the quantization paradigm could also be applied to other domains of machine learning, such as dimensionality reduction and the training of a classifier.

[8] Originally, "kd tree" stands for "k-dimensional tree". Of course those trees would be d-dimensional in our case but it would sound funny to call them "dd trees"!

Acknowledgements

We are grateful to the reviewers and Senior Program Committee for their numerous suggestions. We also thank Alain Tapp for enlightening discussions. This work is supported in part by the Natural Sciences and Engineering Research Council of Canada, the Canada Research Chair programme, the Canadian Institute for Advanced Research and Quantum*Works*.

References

Aïmeur, E., Brassard, G. & Gambs, S. (2006). Machine learning in a quantum world. *Proceedings of Canadian AI 2006* (pp. 433–444).

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

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

Bell, J. (1964). On the Einstein-Podolsky-Rosen paradox. *Physics*, *1(3)*, 195–200.

Bennett, C. H. & Brassard, G. (1984). Quantum cryptography: Public key distribution and coin tossing. *Proceedings of the IEEE Conference on Computers, Systems and Signal Processing, Bangalore, India* (pp. 175–179).

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

Bonner, R. & Freivalds, R. (2002). A survey of quantum learning. *Proceedings of the Workshop on Quantum Computation and Learning* (pp. 106–119).

Borůvka, O. (1926). O jistém problému minimálním. *Práce Moravské Přirodovědecké 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., Høyer, P., Mosca, M. & Tapp, A. (2002). Quantum amplitude amplification and estimation. *Contemporary Mathematics*, *305*, 53–74.

Dürr, C., Heiligman, M., Høyer, P. & Mhalla, M. (2004). Quantum query complexity of some graph problems. *Proceedings of the International Conference on Automata, Languages and Programming: ICALP'04* (pp. 481–493).

Dürr, C. & Høyer, P. (1996). A quantum algorithm for finding the minimum. Available on `http://arxiv.org/quant-ph/9607014`.

Einstein, A., Podolsky, B. & Rosen, N. (1935). Can quantum mechanical description of physical reality be considered complete? *Physical Review*, *47*, 777–780.

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. *Proceedings of the 30th ACM Symposium on Theory of Computing: STOC'98* (pp. 53–62).

Harel, D. & Koren, Y. (2001). On clustering using random walks. *Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science: FSTTCS'01* (pp. 18–41).

Horn, D. & Gottlieb, A. (2001). The method of quantum clustering. *Proceedings of the Neural Information Processing Systems: NIPS'01* (pp. 769–776).

Horn, D. & Gottlieb, A. (2002). Algorithm for data clustering in pattern recognition problems based on quantum mechanics. *Physical Review Letters*, *88(1)*.

Kaufman, L. & Rousseeuw, P. (1987). Clustering by means of medoids. in *Statistical Data Analysis Based on the L_1–Norm and Related Methods*, Y. Dodge (editor), North-Holland, Amsterdam (pp. 405–416).

Mishra, N., Oblinger, D. & Pitt, L. (2001). Sublinear time approximate clustering. *Proceedings of 12th ACM-SIAM Symposium on Discrete Algorithms: SODA'01* (pp. 439–447).

Nayak, A. & Wu, F. (1999). The quantum query complexity of approximating the median and related statistics. *Proceedings of the 31st ACM Symposium on Theory of Computing: STOC'99* (pp. 384–393).

Nielsen, M. & Chuang, I. (Eds.). (2000). *Quantum Computation and Quantum Information*. Cambridge University Press.

Servedio, R. (2001). Separating quantum and classical learning. *Proceedings of the International Conference on Automata, Languages and Programming: ICALP'01* (pp. 1065–1080).

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

Small, C. G. (1990). A survey of multidimensional medians. *International Statistical Review*, *58(3)*, 263–277.

Szegedy, M. (2004). Quantum speed-up of Markov chain based algorithms. *Proceedings of 45th IEEE Symposium on Foundations of Computer Science: FOCS'04* (pp. 32–41).

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