RESEARCH ARTICLE

Quantum density peak clustering

Duarte Magano[1,2] · Lorenzo Buffoni[3] · Yasser Omar[1,3,4]

Abstract

Clustering algorithms are of fundamental importance when dealing with large unstructured datasets and discovering new patterns and correlations therein, with applications ranging from scientific research to medical imaging and marketing analysis. In this work, we introduce a quantum version of the density peak clustering algorithm, built upon a quantum routine for minimum finding. We prove a quantum speedup for a decision version of density peak clustering depending on the structure of the dataset. Specifically, the speedup is dependent on the heights of the trees of the induced graph of nearest-highers, i.e. the graph of connections to the nearest elements with higher density. We discuss this condition, showing that our algorithm is particularly suitable for high-dimensional datasets. Finally, we benchmark our proposal with a toy problem on a real quantum device.

Keywords Quantum algorithms · Quantum machine learning · Clustering

1 Introduction

Machine learning (ML) (Bishop 2011; Hastie et al. 2009) is a field with an exceptional cross-disciplinary breadth of applications and studies. The aim of ML is to develop computer algorithms that improve automatically through experience by learning from data, so as to identify distinctive patterns and make decisions with minimal human intervention. The applications of ML that are already possible today are extremely compelling and diverse (Sutton and Barto 2018; Graves et al. 2013; Sebe et al. 2005), and still growing at a steady pace. However, the training and deployment of these models, involving an ever increasing amount of data, face computational challenges (Tieleman 2008) that are only partially met by the development of special purpose classical computing units such as GPUs.

This challenge posed by ML algorithms has led to a recent interest in applying quantum computing to machine learning tasks (Schuld et al. 2015; Wittek 2014; Adcock et al. 2015; Arunachalam and de Wolf 2017; Biamonte et al. 2017) that sparked the field of quantum machine learning (QML). So far, there have been many proposals for different QML algorithms. Several of them (Lloyd et al. 2020; Vinci et al. 2020; Otterbach et al. 2017) have shown the potential to accelerate ML tasks, but largely rely on heuristic methods, and *proving* an advantage in terms of computational complexity for these particular algorithms is generally hard. On the other hand, there have been a number of works that, by using quantum algorithms with known complexity as subroutines (Wiebe et al. 2012; Lloyd et al. 2014), could prove the existence of a quantum advantage in QML. Indeed, it has been conjectured that QML algorithms could provide the first breakthrough algorithms on near-term quantum devices, given the inherent robustness of these algorithms to noise and perturbations.

Most of the literature on QML has focused on *supervised* learning (Goodfellow et al. 2016) problems. This particular subset of ML has a number of appealing characteristics as it is more immediate to implement and allows for feedback loops and minimisation of well-known and well-behaved cost functions. Unlike supervised learning, unsupervised

✉ Duarte Magano
duartemagano@tecnico.ulisboa.com

1 Instituto Superior Técnico, Universidade de Lisboa, Lisbon, Portugal

2 Instituto de Telecomunicações, Lisbon, Portugal

3 Portuguese Quantum Institute, Lisbon, Portugal

4 Centro de Física e Engenharia de Materiais Avançados (CeFEMA), Physics of Information and Quantum Technologies Group, Lisbon, Portugal

learning is a much harder, and still largely unsolved, problem. And yet, it has the appealing potential to learn the hidden statistical correlations of large unlabeled datasets (Vincent et al. 2008; Hinton et al. 1995), which constitute the vast majority of data being available today.

Amongst all unsupervised learning problems, clustering is one of the most popular. In clustering, we consider a dataset D composed of n elements,

$$D = \{x_0, \ldots, x_{n-1}\}, \tag{1}$$

in which we are given a notion of *distance* between every two elements in the dataset $x_i, x_j \in D$,

$$\text{dist}(x_i x_j). \tag{2}$$

Interpreting this distance as a similarity measure, the (ill-defined) problem of clustering can be formulated as follows.

Clustering Given a dataset D and a distance $\text{dist}(\cdot, \cdot)$ between each pair of elements of D, partition D into sets called *clusters*, such that similar elements belong to the same cluster and dissimilar elements belong to distinct clusters.

Clustering algorithms thus need to separate unlabeled data into different classes (or clusters) without any external labeling and supervision. Solutions to the clustering problem based on quantum computing have been proposed for a long time, resorting to a plethora of different strategies (Aïmeur et al. 2007; Yu et al. 2010; Li et al. 2011; Aïmeur et al. 2013; Lloyd et al. 2013; Otterbach et al. 2017; Bauckhage et al. 2017; Daskin 2017; Kerenidis et al. 2019; Li and Kais 2021; Kerenidis and Landman 2021; Pires et al. 2021). For example, in Ref. Li et al. (2011), the data points are treated as interacting quantum walkers on a lattice, whereas in Ref. Kerenidis et al. (2019), quantum subroutines for distance estimation and matrix arithmetics are employed to develop an efficient quantum version of a classical standard algorithm, k-means clustering. In Otterbach et al. (2017), the clustering problem was reformulated as an optimisation problem and solved by applying a hybrid optimisation algorithm on a Rigetti quantum processor, hinting to the possibility of realising such advantage on near term quantum devices.

In this work, we adopt the black-box/oracular model of clustering introduced in Aïmeur et al. (2007), in which the information concerning the distances between points in the dataset is available only through oracle queries. Under this framework, references (Aïmeur et al. 2007; 2013), using variants of Grover's search (Grover 1997), quantise typical subroutines of learning algorithms, such as finding the largest distance in a dataset, computing the median, or constructing the c-neighbourhood graph. These subroutines are then used to accelerate standard clustering

methods, namely, divisive clustering, k-medians clustering, and clustering via minimum spanning tree. Nevertheless, the state of the art in clustering has been steadily evolving in recent years, and so the question arises: can we also use quantum computing to speedup modern clustering algorithms?

We consider a recent and very popular clustering method, usually referred to as density peak clustering (DPC) (Rodriguez and Laio 2014). In a nutshell, the idea is to attribute a "density" value to every element of the dataset based on their distances to all other elements, and then assign each element to the same cluster as the nearest neighbour that has a density greater than itself (referred to as its *nearest-higher*). Relying on a variant of quantum search known as quantum minimum finding (Durr and Hoyer 1996), we show that, for any element of the dataset, we can find its nearest-higher (up to bounded-error probability)in time $\mathcal{O}\left(n^{3/2}\right)$, as opposed to the classical $\mathcal{O}\left(n^2\right)$ complexity. Unfortunately, to fully solve DPC, we would need to repeat this subroutine n times. This would provide no advantage as we can classically implement DPC in $\mathcal{O}\left(n^2\right)$ time.

Motivated by this, in this work, we consider a simpler variant of the clustering problem, which we refer to a *decision* clustering.

Decision Clustering Given a dataset D, a distance $\text{dist}(\cdot, \cdot)$ between each pair of elements of D, and two elements $x_i, x_j \in D$, decide if x_i and x_j are in the same cluster.

Evidently, any solution of the clustering problem contains the answer to decision clustering, but not the other way around. This problem is relevant in situations where one is interested in establishing connections between particular elements but does not need to know the cluster structure of the entire dataset. For example, we may want to know if two users of social media platform are friends, or if an individual fits a particular consumer segment. Moreover, this can also be useful when one has to decide to which cluster a new datapoint belongs without having to re-cluster all the data. Finally, decision clustering can be iteratively applied to cluster small subsets of the data.

Classically solving decision clustering with DPC has the same complexity as solving the full clustering problem, $\mathcal{O}\left(n^2\right)$. However, in the quantum setting, we show that we can solve it in $\tilde{\mathcal{O}}\left(n^{3/2}H\right)$ time, where H is the maximum height of all trees in the graph of nearest-highers (to be properly defined in Section 3). The factor H is not known a priori, depending on the specific structure of the dataset. Nevertheless, we argue that for high-dimensional datasets H scales as $\mathcal{O}(n^{1/d_{\text{eff}}})$, where d_{eff} is a constant greater than 2, and confirm this hypothesis with numerical simulations.

When this holds, quantum density peak decision clustering provides a speedup over its classical counterpart.

Finally, we benchmark our quantum algorithm using a toy problem on a real quantum device, the ibm-perth 7-qubit quantum processor. As expected, the hardware errors severely mitigate any possible quantum advantage. Nevertheless, the noiseless simulations confirm the potential of our approach.

The article is structured as follows. Section 2 provides summary of background material that is necessary for understanding our quantum algorithm. Namely, we introduce the data model and review the quantum minimum finding algorithm. Section 3 explains the classical DPC algorithm, assuming that the reader is not yet familiar with it. Then, in Section 4, we present our quantum algorithm. The complexity of the algorithm depends on the H factor, and so in Section 5, we study how H behaves for different datasets. In Section 6, we show the results of the implementation of our proposal on a real quantum device. Section 7 concludes the article with a brief summary and discussion.

2 Preliminaries

2.1 Data model

In this article, we work in a black-box model. We assume that our knowledge about the data comes uniquely from querying an "oracle" that returns the distance between pairs of points

$$(i, j) \xrightarrow{\text{query}} \text{dist}(x_i, x_j). \tag{3}$$

We make no assumptions about this distance besides that it is non-negative and symmetric. That is, it does not need to be a distance by a proper mathematical definition. Our complexity measure is the number of performed queries (this is known as query complexity).

This model is an abstraction that reasonably fits a number of problems. For example, the oracle may represent accessing a database with the distances between a group of cities, or a routine that estimates the dissimilarity between pairs of images. The query complexity is a good estimate of the total time complexity whenever determining the distance between elements is the most computationally intensive part of the algorithm. Nevertheless, we also point out that this model may not be a good description of other common situations. For example, if we know the coordinates of a set of points in $\mathbb{R}^d$ (for some integer d), we already have more structure than in the black-box model and there are geometrical methods that allow significant speedups for certain tasks.

In the quantum setting, we assume that we can query the distances between elements in quantum superposition. That

is, we assume access to a unitary Q (the quantum oracle) such that

$$Q|i, j, 0^{\otimes q}\rangle = |i, j, \text{dist}(x_i, x_j)\rangle, \tag{4}$$

where q is the number of bits necessary to store the distances up to desired accuracy. In particular, given a superposition $\sum_{ij} \alpha_{ij}|i, j\rangle$ (for any set of normalised complex amplitudes $\{\alpha_{ij}\}_{ij}$), Q acts as

$$Q\left(\sum_{ij} \alpha_{ij}|i, j\rangle\right)|0^{\otimes q}\rangle = \sum_{ij} \alpha_{ij}|i, j, \text{dist}(x_i, x_j)\rangle. \tag{5}$$

The quantum query complexity is counted as the number of applications of the unitary Q (for more details on the quantum query model, refer to Ambainis (2017)). When describing classical data, the oracle may be realised with a quantum random access memory (qRAM) architecture (Giovannetti et al. 2008a). In the end, our results are critically dependent on the existence of a qRAM with the above-mentioned properties, representing a common setting in theoretical work on quantum algorithms, and in quantum machine learning in particular. Nevertheless, even though there have been proposals of physical architectures for implementing QRAM (Giovannetti et al. 2008b; Park et al. 2019), there are still significant challenges to overcome before such a device can be practically realised (Matteo et al. 2020).

2.2 Quantum minimum finding

The key quantum routine we use in our work is the well-known quantum minimum finding algorithm of Durr and Hoyer (1996), which in itself is a specific application of the quantum search algorithm (Grover 1997).

Begin by considering a boolean function F defined on a domain of size n, $F : \{0, 1, \ldots, n-1\} \rightarrow \{0, 1\}$. Our goal is to find an element x such that $F(x) = 1$, assuming one exists. F is provided as a black box, that is, we can only gain information about the function by evaluating it on given elements. In the worst-case scenario, we may need to query all n possible inputs before succeeding.

Now suppose that we have access to a unitary O_F that marks the 1-inputs with a -1 phase,

$$O_F|i\rangle = \begin{cases} +|i\rangle, & \text{if } F(i) = 0 \\ -|i\rangle, & \text{if } F(i) = 1 \end{cases}. \tag{6}$$

The unitary O_F is an oracle for F, and an application of O_F is referred to as a quantum query to F as introduced in the section above. Lets now consider $|\Psi\rangle$ to be the uniform superposition of all input states,

$$|\Psi\rangle = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} |i\rangle. \tag{7}$$

If we measure $|\Psi\rangle$, we obtain every input with equal probability. Grover's algorithm (Grover 1997) is based on the observation that we can amplify the probability of measuring 1-input states via the repeated application of the operator

$$G_F = (2|\Psi\rangle\langle\Psi| - I) \cdot O_F \qquad (8)$$

to an initial state $|\Psi\rangle$. Roughly, the amplitudes of the 1-input states grow linearly with each application of G_F, while their measurement probabilities grow quadratically. This means that $\sim \sqrt{n}$ quantum queries are sufficient to measure an input state with high probability. Boyer et al. (1998), generalising Grover's algorithm (Grover 1997), prove the following theorem.

Theorem 1 (Quantum search, Boyer et al. (1998)) *Let F : $\{0,\dots,n-1\} \to \{0,1\}$ and let $t = |\{x \in \{0,\dots,n-1\} : F(x) = 1\}|$ (which does not need to be known a priori). Then, we can find an element x such that $F(x) = 1$ with an expected number of $\mathcal{O}(\sqrt{n/t})$ quantum queries to F.*

Quantum minimum finding calls quantum search as a subroutine. Suppose that we want to find a minimiser of a black-box function $f : \{0,\dots,n-1\} \to \mathbb{N}$. For any element $i \in [n]$, define the boolean function

$$F_i(j) = \begin{cases} 0, & \text{if } f(j) \geq f(i) \\ 1, & \text{if } f(j) < f(i) \end{cases}, \qquad (9)$$

which can be evaluated with two queries to f. Let O_{F_i} be a unitary that evaluates F_i, as in expression (6). The quantum minimum finding algorithm starts by choosing a threshold element i uniformly at random between 0 and $n-1$. Employing quantum search with O_{F_i} as oracle, we look an element j such that $F(j) < F(i)$. We then repeat this process, updating j as the threshold element, until the probability that the selected threshold is the minimum of f is sufficiently large. With this algorithm, Durr and Hoyer (1996) reach the following result.

Theorem 2 (Quantum minimum finding, Durr and Hoyer (1996)) *Let $f : \{x \in \{0,\dots,n-1\} \to \mathbb{N}$ and $\epsilon \in [0,1[$. Then, we can find the minimiser of f with probability at least $1 - \epsilon$ using $\mathcal{O}(\sqrt{n}\log(1/\epsilon))$ quantum queries to f.*

3 Density peak clustering

Since its introduction in 2014 (Rodriguez and Laio 2014), density peak clustering (DPC) has been widely studied and applied (Tu et al. 2019; Cheng et al. 2016; Shi et al. 2019). This algorithm, albeit remaining quite simple in the concept and implementation, presents some interesting features that are absent from most of the simpler clustering algorithms

discussed in the QML literature. As an example, it does not make assumptions on the number of clusters present in the data (unlike simpler algorithms such as k-means (MacQueen and et.al 1967)), but it can infer this information from the data itself. Linked to this first property, DPC does not require any prior hypothesis on the shape of the dataset (i.e. gaussian distributed or symmetric) and works well with datasets of virtually any shape (Rodriguez and Laio 2014; Fang et al. 2020). Finally, DPC is able to detect outliers in the dataset, that is, elements that do not belong to any cluster. This last property opens the possibility of using DPC beyond the usual scope of clustering problems, such as anomaly detection (Tu et al. 2020; Shi et al. 2019). We will now introduce this algorithm from a mathematical point of view and have a deeper look at its implementation to understand its capabilities and its criticalities.

In density peak clustering, the first step is to compute the density $\rho(x_i)$ of every element $x_i \in D$,

$$\rho(x_i) = \sum_{x_j \in D} \chi(\text{dist}(x_i, x_j)), \qquad (10)$$

where χ is a convolutional kernel that can be optimised according to the specific application. Common choices include the step kernel $\chi(x) = \Theta(x - d_c)$ or the Gaussian kernel $\chi(x) = \exp(-x^2/d_c^2)$, for some normalisation parameter d_c. Next, for each element x_i, we find its "nearest-higher" $h(x_i)$, defined as the closest element to x_i with higher density than x_i,

$$h(x_i) = \operatorname*{arg\,min}_{x_j : \rho(x_j) > \rho(x_i)} \text{dist}(x_i, x_j). \qquad (11)$$

The nearest-higher separation $\delta(x_i)$ is the distance between x_i and its nearest-higher,

$$\delta(x_i) = \text{dist}(x_i, h(x_i)). \qquad (12)$$

Naturally, if the point in question is the one with highest density in the entire dataset, then it does not have a nearest-higher. In that case, by convention, the nearest-higher separation is $+\infty$.

The key observation at the basis of DPC is that, for the elements that are local maxima of the density, the nearest-higher separation is much larger than the typical nearest-neighbour distance. This idea is illustrated in Fig. 1, where we consider a small dataset embedded in $\mathbb{R}^2$ with the similarity measure provided by the Euclidean distance. In Fig. 1b, by plotting the density versus the nearest-higher separation for all elements in the dataset, it becomes clear that the elements 7, 11, and 15 stand out from their neighbours. These three elements are classified as *roots*, and each root will originate its own cluster. The elements 0 and 16, despite also having large nearest-higher separations, also have low densities and so are classified as *outliers*. In more detail, for some choice of threshold ρ_c and δ_c (to be
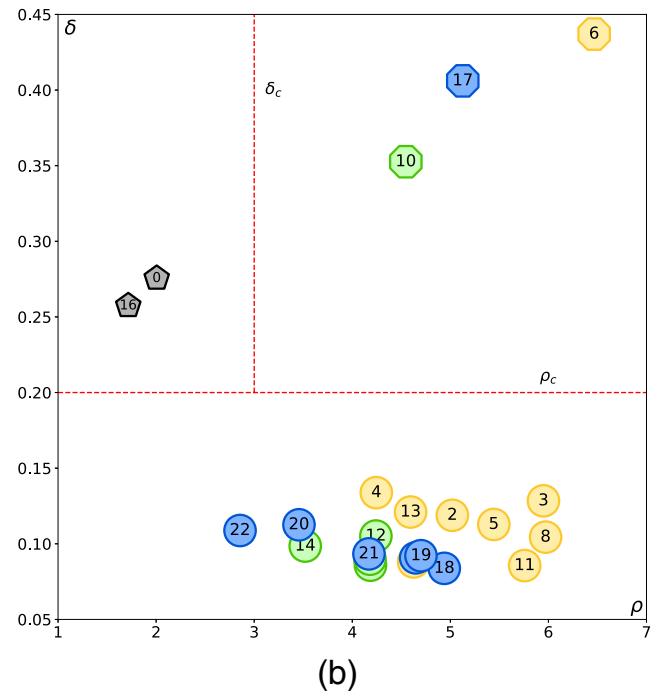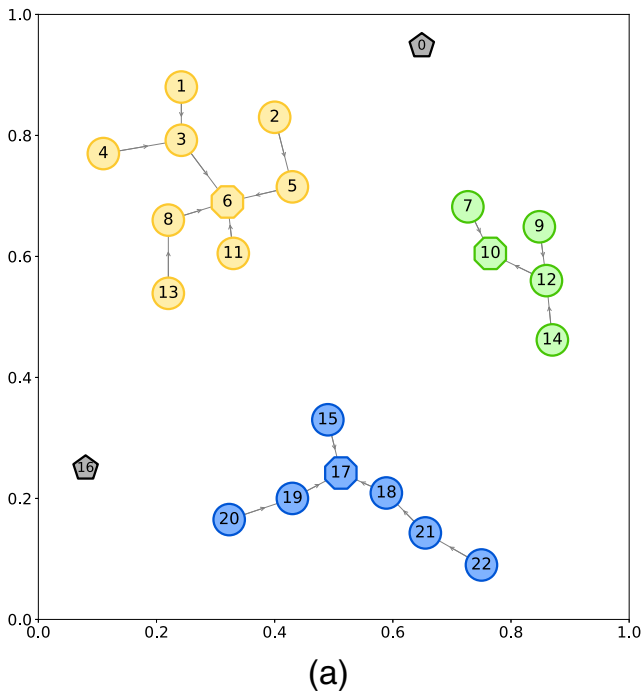
(a)

(b)

Fig. 1 Density peak clustering. In **a**, we represent a dataset of 23 elements embedded in $\mathbb{R}^2$, with the similarity measure provided by the Euclidean distance. The roots are represented by octagons, the outliers by pentagons, and the rest of the dataset by circles. The elements are coloured by cluster (with the outliers coloured in black). The gray arrows indicate the edges of the directed graph of nearest-higher connections. This graph forms a forest, with one tree per distinct cluster. In **b**, we plot the nearest-higher separation as a function of density for all points in the dataset. The elements satisfying $\rho(x_i) > \rho_c \wedge \delta(x_i) > \delta_c$ are promoted to roots, while the ones satisfying $\rho(x_i) < \rho_c \wedge \delta(x_i) > \delta_c$ are demoted to outliers

specified according to the typical scales of the dataset), we promote to roots the elements x_i satisfying

$$\rho(x_i) > \rho_c \quad \text{and} \quad \delta(x_i) > \delta_c, \tag{13}$$

and demote to outliers those who obey

$$\rho(x_i) < \rho_c \quad \text{and} \quad \delta(x_i) > \delta_c. \tag{14}$$

Then, the rest of the elements are assigned to the same cluster as their nearest-higher. As evidenced in Fig. 2c, the directed graph of nearest-highers (with edges originated from roots removed) forms a forest and each tree corresponds to a distinct cluster, the root nodes of the tree being the elements that were promoted to roots.

The main steps of density peak clustering are summarised in Algorithm 1. The complexity of the algorithm becomes clear immediately from the first step. Indeed, in order to compute the density of a given element, we need to query the distance to every other element in the dataset. Repeating this for all elements in the dataset means that we end up querying all $(n^2 - n)/2$ distances.

Claim 1 *The density peak clustering algorithm (Algorithm 1) has $\mathcal{O}(n^2)$ query complexity, where n is the size of the dataset.*

Now consider a *decision* version of the clustering problem, where the goal is, given two elements $x_i, x_j \in D$, to decide if they belong to the same cluster. Evidently, we could solve the full clustering problem and then verify if x_i and x_j were assigned to the same cluster. But in the context of DPC, there is a more direct approach, relying on

---

input : dataset D, density threshold ρ_c, and
nearest-higher separation threshold δ_c (and
possibly normalization parameter d_c,
depending on the form of the kernel)

output: clusters

1 For all $x_i \in D$, compute density
$\rho(x_i) = \sum_j \chi(\text{dist}(x_i, x_j))$;

2 For all $x_i \in D$, compute nearest-higher
$h(x_i) = \arg\min_{j:\rho(x_j) > \rho(x_i)} \text{dist}(x_i, x_j)$ and
respective separation $\delta(x_i) = \text{dist}(x_i, h(x_i))$;

3 For all $x_i \in D$, promote to root if
$\rho(x_i) > \rho_c \wedge \delta(x_i) > \delta_c$ and to outlier if
$\rho(x_i) < \rho_c \wedge \delta(x_i) > \delta_c$;

4 For all $x_i \in D$ that is not a root nor an outlier, assign
x_i to the same cluster as $h(x_i)$;

Algorithm 1 Density peak clustering (originally proposed in Ref. Rodriguez and Laio (2014)).

```
input  : dataset D, x_i, x_j ∈ D, density threshold ρ_c,
         and nearest-higher separation threshold δ_c
         (and possibly normalization parameter d_c,
         depending on the form of the kernel)
output: yes/no
1 def FindRoot(x):
2     if x is a root then
3         output x;
4     else
5         output FindRoot(h(x));
6 if x_i or x_j are outliers then
7     output no;
8 if FindRoot(x_i) = FindRoot(x_j) then
9     output yes;
10 else
11    output no;
```

Algorithm 2 Decision version of density peak clustering.

the following simple observation: the two elements belong to the same cluster if and only if the sequences of nearest-highers starting from x_i and x_j lead to the same root. In other words, the problem is reduced to finding the roots of the respective trees in the graph of nearest-highers. This can be solved with Algorithm 2, where we just walk up the trees node by node by computing the corresponding nearest-higher. We know that we have reached a root when the nearest-higher separation is larger than δ_c.

Unfortunately, this approach comes with no significant complexity advantage as every step we take up the tree has essentially the same computational cost of the full clustering. To see this, just note that computing a nearest-higher of an element x_i requires knowing how the density of all other element compares to $\rho(x_i)$ (*cf.* (11)). So, it seems unavoidable to compute all the densities.

Claim 2 *The decision version of density peak clustering (Algorithm 2) has $\mathcal{O}(n^2)$ query complexity, where n is the size of the dataset.*

In the next section, we show that we can circumvent this cost with quantum search, establishing a quantum advantage for the decision version of density peak clustering.

4 Quantum algorithm

We now have all the elements to introduce a quantum algorithm to solve the decision version of density peak clustering. The quantum algorithm will use the same strategy as in Algorithm 2, calling the quantum minimum finding routine (Section 2.2) to determine the nearest-highers.

For each $i \in \{0, \ldots, n-1\}$, consider the function

$$f_i(j) = \begin{cases} \text{dist}(x_i, x_j), & \text{if } \rho_j > \rho_i \\ +\infty, & \text{if } \rho_j \leq \rho_i. \end{cases} \tag{15}$$

From the definition of nearest-higher (11), it is clear that finding the minimiser of f_i is equivalent to determining the nearest-higher of x_i,

$$h(x_i) = \arg\min_j f_i(j). \tag{16}$$

For any element in the dataset, we can evaluate its density (10) with $n-1$ distance queries. So, we can compute $f_i(j)$, for any $j \in \{0, \ldots, n-1\}$, with $\mathcal{O}(n)$ queries. Since any classical computation can be simulated on a quantum computer (Nielsen and Chuang 2010), there exists a unitary U_i that uses $\mathcal{O}(n)$ quantum queries and achieves the transformation

$$U_i|j\rangle|0^{\otimes q+1}\rangle = |j\rangle|f_i(j)\rangle \tag{17}$$

for any $j \in \{0, \ldots, n-1\}$, where q is the number of qubits to store the distance up to the desired accuracy and we add an extra flag qubit to indicate if f_i evaluates to $+\infty$. Therefore, by Theorem 2, we can use the quantum minimum finding algorithm to find the minimiser of f_i (i.e. the nearest-higher of x_i) with probability at least $1 - \epsilon$ with $\mathcal{O}(\sqrt{n}\log(1/\epsilon))$ applications of U_i, amounting to a total of

$$\mathcal{O}\left(n^{3/2}\log(1/\epsilon)\right) \tag{18}$$

quantum queries.

Let H be the maximum height of any tree in the graph of nearest-highers. In the worst case scenario, one may need to call quantum minimum finding $2H$ times before reaching the root node for both inputs. Moreover, we want to make sure that it finds the correct nearest-highers every single call with high probability, which can be done by setting the constant ϵ in (18) to be sufficiently small. A simple calculation concludes the following.

Claim 3 *Let D be a dataset of n elements and let H be the maximum height of any tree in the graph of nearest-highers. For any two elements $x_i, x_j \in D$, we can solve the decision version of density peak clustering[1] with success probability at least $1-\epsilon$ (for any $\epsilon > 0$) with quantum query complexity*

$$\mathcal{O}\left(n^{3/2}H\log(H/\epsilon)\right). \tag{19}$$

---

[1]To be more precise, we mean that we provide the same answer to the decision version of the clustering problem as we would if we had run the classical density peak algorithm (up to error probability ϵ).

Recall that the classical complexity of the decision version of density peak clustering is $\mathcal{O}(n^2)$ (Claim 2), irrespective of the value of H.

5 Height of trees

We have seen that we can reach a lower quantum query complexity for the density peak decision clustering problem depending on the factor H, the maximum height of the trees in the graph of nearest-highers. Indeed, there is quantum speedup when H scales as $\mathcal{O}(n^a)$ for some $a < 1/2$. In contrast, there is no speedup when $H = \Omega(n^a)$ with $a > 1/2$, as we may solve clustering classically in $\mathcal{O}(n^2)$ time.

To understand how the factor H scales, we can start by considering very simple data model. We assume that the dataset is generated by sampling n points uniformly at random from a bounded region of $\mathbb{R}^d$. In this case, we expect to see only one cluster that spans the entire region. A straightforward calculation reveals that the expected nearest-neighbour distance $\langle nn \rangle$ scales as

$$\langle nn \rangle = \mathcal{O}\left(\frac{1}{n^{1/d}}\right). \tag{20}$$

While the nearest-higher does not always coincide with the nearest-neighbour, the nearest-higher separation is most likely not much larger than the typical nearest-neighbour distance (except for the root node). So, we should have

$$\langle \delta \rangle \sim \langle nn \rangle. \tag{21}$$

Now consider a leaf node of the graph of nearest-highers at a distance, say, L from the root node. As it probably lies close the edge of the region, L characterises the "size" of the cluster. Its nearest-higher is found roughly along the direction of the centre of the cluster. That is, the nearest-higher is at a distance $\sim \delta$ closer to the root node. Repeating this reasoning for all nodes in the path to the root, we conclude that

$$\langle H \rangle \sim \frac{L}{\langle \delta \rangle} \sim \frac{L}{\langle nn \rangle} = \mathcal{O}\left(n^{1/d}\right). \tag{22}$$

Admittedly, we have merely provided an informal argument for the expression (22), not a fully rigorous proof. Still, we can verify this behaviour numerically. We generate such artificial datasets by sampling uniformly from d-balls of radius one, for different dimensions d. The results, shown in Fig. 2a, confirm that indeed $\langle H \rangle$ scales as $n^{1/d}$.

To study how H behaves in more general settings, we consider two other types of datasets:

- *Well-separated, Gaussian-shaped clusters* (Fig. 2b). For different values of d, we pick ten centroids at random in range $[0, 100]^d$, associating to each a randomly chosen covariance matrix. We then generate artificial datasets by drawing from the corresponding Gaussian distributions. With high probability, the clusters will be well-separated.

- *Real-world dataset* (Fig. 2c). We randomly sample entries from the Forest Cover Type dataset (Blackard and Dean 1999). This dataset originally contains fifty-five features, both numerical and categorical, conveying information about the Roosevelt National Forest in Colorado, namely, tree types, shadow coverage, distance to nearby landmarks, soil type, and local topography. We preprocess the dataset by numerically encoding the
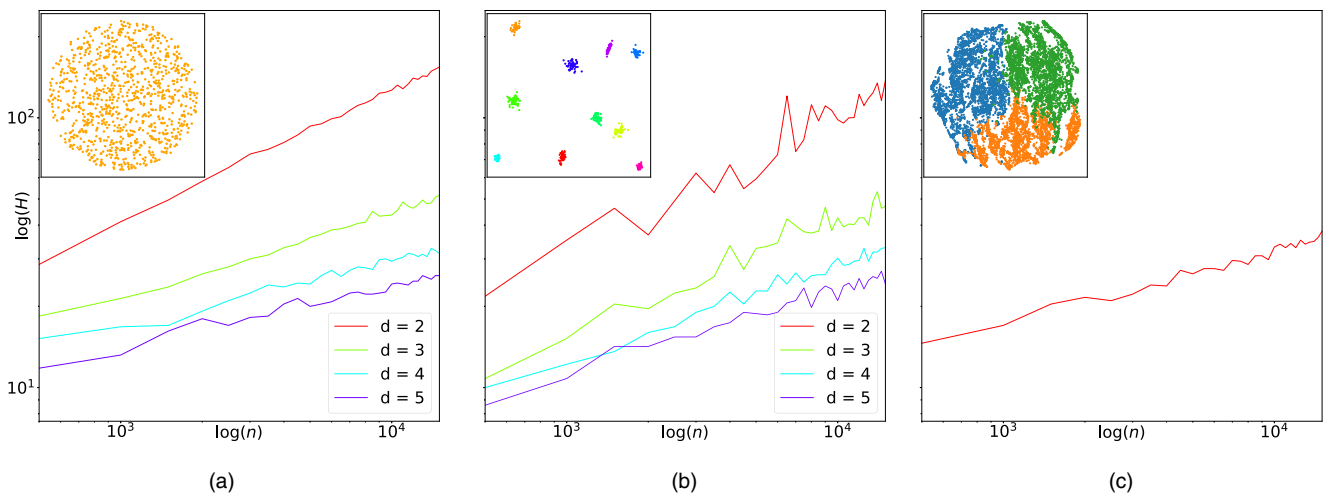

Fig. 2 Heights of trees. We plot the maximum height of any tree in the graphs of nearest-highers, H, for different types of datasets (we show the average over five runs). Both axes are shown in logarithmic scale. For plots **a** and **b**, we artificially generate uniform and Gaussian datasets (respectively) for several dimensions d with varying number of elements n. In the upper left corners, we show examples of such datasets in two dimensions. For plot **c**, we take n random samples from the Forest Cover Type dataset (Blackard and Dean 1999). To visualise the dataset in two dimensions, we used a t-distributed stochastic neighbour embedding (van der Maaten and Hinton 2008)

categorical data, and then rescaling the numerical variables such that each has mean zero and variance one. We define the clustering distance as the Euclidean distance between the first ten principal components of the data.

For both cases, we observe that

$$\langle H \rangle \sim n^{1/d_{\text{eff}}}, \tag{23}$$

for some parameter d_{eff}. We interpret d_{eff} as an "effective dimension" of the dataset, which can be smaller than the number of features of the data. For example, for the Gaussian datasets, a simple polynomial fit reveals $d_{\text{eff}} = 1.94, 2.36, 2.80, 3.22$ for $d = 2, 3, 4, 5$, respectively. For the Forest Cover Type dataset, we find $d_{\text{eff}} = 3.71$. An interesting research question (outside the scope of the present article) would be to properly understand how this effective dimension arises from the structure of the data.

For the cases considered where the datasets had more than two features, we have verified that the effective dimension was greater than two, entering the regime where our density peak decision clustering algorithm shows a quantum speedup. This is evidence that our quantum algorithm could be suitable for high-dimensional, real-world problems. Our conclusions are summarised in the following claim.

Claim 4 *Let D be a dataset of n elements and let the maximum height of any tree in the graph of nearest-highers scale as $\mathcal{O}\left(n^{1/d_{\text{eff}}}\right)$ for some parameter d_{eff}. Then, for any two elements $x_i, x_j \in D$, we can solve density peak decision*

clustering with constant error probability with quantum query complexity

$$\tilde{\mathcal{O}}\left(n^{3/2+1/d_{\text{eff}}}\right). \tag{24}$$

In particular, if $d_{\text{eff}} > 2$, quantum query complexity is better than the classical query complexity $\mathcal{O}\left(n^2\right)$.

We would like to stress that our speedup shows a dependence on a geometric property of the dataset that, to the best of our knowledge, has not yet been seen in the literature. While it is a common idea that quantum speedups in machine learning may rely on the structure of the dataset, it is usually hard to rigorously characterise the necessary structure. In contrast, in this work, we were able prove that we have speedup if H scales better than $\sqrt{n}$ (or, in other words, if the effective dimension is larger than 2).

6 Experimental implementation

In this section, we test the proposed quantum density peak clustering on a real quantum processor. Specifically, we implement the main quantum routine, minimum finding. Since we are limited by the size of the devices, we solve a synthetic clustering problem involving just eight elements — see the dataset depicted in Fig. 3a. We can encode each element of the dataset in $\lceil \log(8) \rceil = 3$ qubits, being suitable to run on NISQ machines. The first problem to address is implementing the oracle in a suitable manner.
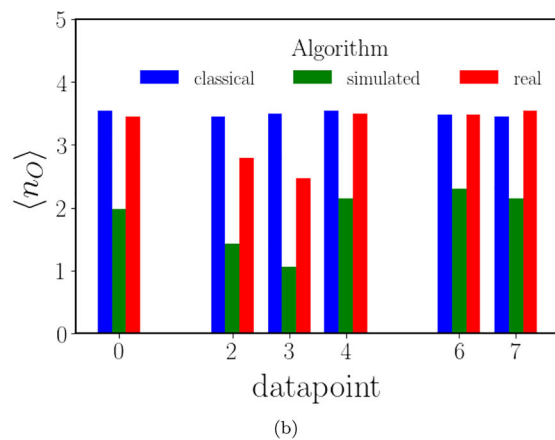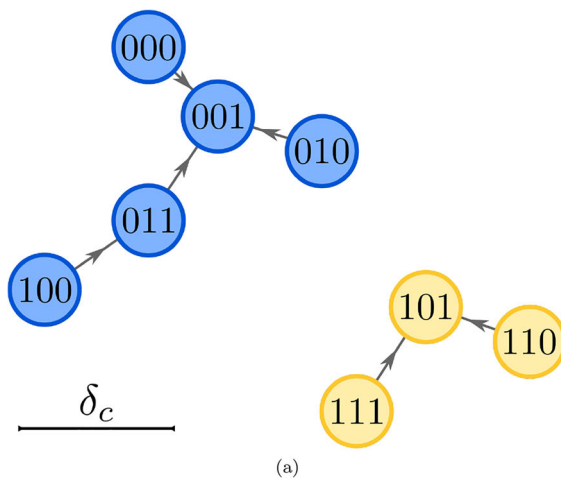

(a)

(b)

Fig. 3 Experimental implementation of toy problem. **a** Toy dataset of 8 points. The two colours (blue and yellow) denote the two distinct clusters in the dataset. The arrows denote the nearest-higher (except for the elements 1 and 5 which are the cluster centers). In plot **b**, we plot the average number of iterations, or average number of oracle calls $\langle n_O \rangle$, to find the nearest-higher for each of the points in the dataset. We reported a classical random search strategy, an ideal simulated quantum minimum finding subroutine, and the implementation of the same

quantum subroutine in a real IBM quantum processor. The average is taken over 1000 consecutive runs of each strategy. The datapoints reported in the x-axis of the bar graph are the decimal representation of the binary points in the dataset (e.g. datapoint 3 is the element 011 of the dataset), notice that datapoints 1 and 5 are missing from the bar graph as they are the cluster centers and their number of iterations is fixed

Given the density ρ_i computed for each $i \in \{0, \ldots, 7\}$ in our synthetic dataset, consider the function introduced in (15). The quantum minimum finding calls an oracle $O_{i,j}$ that implements the Boolean function

$$F_{i,j}(k) = \begin{cases} 0, & \text{if } f_i(k) \geq f_i(j) \\ 1, & \text{if } f_i(k) < f_i(j) \end{cases} \tag{25}$$

as

$$O_{i,j}|k\rangle = \begin{cases} +|k\rangle, & \text{if } F_{i,j}(k) = 0 \\ -|k\rangle, & \text{if } F_{i,j}(k) = 1 \end{cases}. \tag{26}$$

Present-day quantum machines do not have qRAM access, nor can perform complex arithmetic operations such as computing densities. So, we classically pre-compute $F_{i,j}(k)$ for every values of i, j, and k and construct a circuit for each oracle $O_{i,j}$ following scheme outlined in Figgatt et al. (2017). While this is not how the algorithm is meant to be implemented in practice (cf. Section 4), we believe that this approach is suited for the purposes of a proof-of-concept.

At this point, we have all the ingredients to start the quantum nearest-higher search. Given a point i, we start by selecting a random threshold j and call the oracle dictionary $F_{i,j}$. We then mark the states that have $F_{i,j}(k) = 1$ following the same marking scheme outlined in Figgatt et al. (2017) and apply the amplitude amplification subroutine. At the end, we measure a state $|k\rangle$ that is selected as the new threshold $j' = k$ if $f_i(k) \geq f_i(j)$; otherwise, a new threshold j' is selected at random. This procedure is iterated until the nearest higher is found for every point i in the dataset.

We benchmark this strategy against the classical method of nearest-higher search that is just a random sampling of j. In either case, the figure of merit is the average number of oracles calls before finding the nearest-higher, $\langle n_O \rangle$, i.e. the quantum query complexity.

In Fig. 3b, we show the results for $\langle n_O \rangle$ taken over 1000 run of the algorithm by using both the classical strategy and the quantum routine. As expected, the classical nearest-higher algorithm (i.e. random search) always takes on average iterations ~ 3.5 (blue bars in Fig. 3b), irrespectively of the point as one would expect given the size of the dataset.

Regarding the quantum search, we first used the Qiskit package (M.S.A.et al 2021) to simulate the algorithm without noise. In general, we can have multiple rounds of amplitude amplification in each step of our quantum minimum finding subroutine. However, here we opted by always running just one round. The simulations (green bars in Fig. 3b) clearly demonstrate quantum speedup, as the average number of oracle calls before convergence is lower than in the classical case for all points. We can observe that there are some points with speedups more pronounced than others as, for each specific datapoint, this depends on the number of states marked by the oracle.

Finally, we ran our Qiskit program on a real quantum computer by IBM, the ibm-perth 7-qubit processor (see red bars in Fig. 3b). In this case, the analysis is more subtle because of the real-world noise on top of our ideal quantum algorithm. Indeed, for points whose oracles mark several states, the depth search circuits is quite large, making the computations more sensible to noise. On the other hand, there are some points for which the number of states marked by the oracle is low and thus the circuit is small enough that we can still observe an advantage over the classical case (even if it is less pronounced than in the ideal quantum simulation due to noise). This observation is very relevant as it is generally difficult to observe such quantum advantages running quantum machine learning subroutines on real hardware, even if for a toy problem as the one explored here. We can thus hope that, with some improvement in coherences and maybe error correcting codes being built in real quantum processors, we can start observing some advantages for real-world problems.

7 Conclusions

In this work, we have introduced a quantum version of the density peak clustering algorithm, specifically aimed at its decision version. Our proposed algorithm builds upon the well-known quantum minimum finding algorithm, giving us the possibility of computing the query complexity of quantum density peak clustering. Indeed, while the classical query complexity of density peak clustering is $\mathcal{O}\left(n^2\right)$, we prove that our proposed quantum algorithm has complexity $\mathcal{O}\left(n^{3/2+1/d_{\text{eff}}}\right)$, for a parameter d_{eff} that depends on the structure of the dataset. For values of $d_{\text{eff}} > 2$, we have a quantum speedup, albeit a modest one. This provable dependence of the complexity on this geometric property of the dataset constitutes by itself an notable result. Indeed, while it is widely accepted that quantum speedups for machine learning may depend on the structure of the data, it is often difficult to precisely characterise this dependence. As discussed in Section 5, in our case, we interpret the parameter d_{eff} as an "effective dimension" of the dataset, making quantum density peak clustering specially suited for high-dimensional problems. The successful implementation of a toy problem in a real quantum computer and the observation of an advantage, even in the presence of the noise typical of a NISQ device, hints at the concrete possibility of exploiting the capability of quantum density peak clustering in near-term quantum computers.

The original version of the density peak clustering algorithm (Rodriguez and Laio 2014) has its shortcomings. For example, it does not provide a method to choose the parameters δ_c and ρ_c. Moreover, for data with complex shapes and densities, multiple local density peaks can

be mistaken for cluster centres. Several extensions to the original algorithm have been put forward (e.g. Fang et al. (2020)) to try to fix these issues, considerably extending the class of datasets where the method can be successfully applied. A future research direction could be to find quantum formulations of these new versions of density peak clustering.

To conclude, we would like to raise two points that are relevant not only for the quantum density peak clustering, but also for the quantum machine learning community at large. The first one regards the efficient implementation of the classical computation routines, which in this work were included in the oracles. For example, while comparing two distances consumes $\mathcal{O}(1)$ time, the overhead of this calculation prohibits its implementation on present-day quanutm hardware. The second point is to understand how the effective dimension of the dataset d_{eff} scales in general for an arbitrary dataset (and if we can always define an effective dimension given the scaling of nearest neighbours). Going even deeper, one could ask if d_{eff} represents some fundamental property of the data and its structure (and can thus be exploited further) or if it is just a scaling parameter. We believe that answering these questions could be crucial for quantum density peak clustering and its future as a viable clustering algorithm for quantum machine learning on near-term quantum devices.

Declarations

Ethics approval and consent to participate This work does not involve human participants and presents no ethical concerns.

Consent for publication Not applicable.

Competing interests The authors declare no competing interests.

References

Adcock J., Allen E., Day M., Frick S., Hinchliff J., Johnson M., Morley-Short S., Pallister S., Price A., Stanisic S. (2015) Advances in quantum machine learning. arXiv: 1512.02900, [quant-ph]

Aïmeur E., Brassard G., Gambs S. (2007). In: ACM International Conference Proceeding Series, vol 227, (1). https://doi.org/10.1145/1273496.1273497

Aïmeur E., Brassard G., Gambs S. (2013) Mach Learn 90:261. https://doi.org/10.1007/s10994-012-5316-5

Ambainis A. (2017) Understanding quantum algorithms via query complexity. arXiv: 1712.06349 [quant-ph]

Arunachalam S., de Wolf R. (2017) arXiv:1701.06806, [quant-ph]

Bauckhage C., Brito E., Cvejoski K., Ojeda C., Sifa R., Wrobel S. (2017) Adiabatic quantum computing for binary clustering. arXiv: 1706.05528 [quant-ph]

Biamonte J., Wittek P., Pancotti N., Rebentrost P., Wiebe N., Lloyd S. (2017) Nature 549:195. https://doi.org/10.1038/nature23474

Bishop CM (2011) Pattern recognition and machine learning, 1st ed. Springer, New York. https://doi.org/978-0-387-31073-2

Blackard J., Dean D. (1999) Comput Electron Agric 24:131. https://doi.org/10.1016/S0168-1699(99)00046-0

Boyer M., Brassard G., HØyer P., Tapp A. (1998) Fortschr Phys 46:493

Cheng S., Quan T., Liu X., Zeng S. (2016) BMC bioinformatics 17:1. https://doi.org/10.1109/CCDC.2019.8833427

Daskin A. (2017) Quantum spectral clustering through a biased phase estimation algorithm. arXiv: 1703.05568 [quant-ph]

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

Fang F., Qiu L., Yuan S. (2020) Pattern Recogn. 107:107452

Figgatt C., Maslov D., Landsman K. A., Linke N. M., Debnath S., Monroe C. (2017) Nature communications 8:1. https://doi.org/10.1038/s41467-017-01904-7

Giovannetti V., Lloyd S., Maccone L. (2008) Phys Rev Lett 100:160501. https://doi.org/10.1103/PhysRevLett.100.160501

Giovannetti V., Lloyd S., Maccone L. (2008) Phys Rev A 78:052310. https://doi.org/10.1103/PhysRevA.78.052310

Goodfellow I., Bengio Y., Courville A. (2016) Deep learning. MIT Press, Cambridge. http://www.deeplearningbook.org

Graves A., Mohamed A. R., Hinton G. (2013) In: 2013 IEEE international conference on acoustics, speech and signal processing pp. 6645. https://doi.org/10.1109/ICASSP.2013.6638947

Grover L. (1997) Phys. Rev. Lett. 79:325. https://doi.org/10.1103/PhysRevLett.79.325

Hastie T., Tibshirani R., Friedman J. (2009) The elements of statistical learning. Data Mining, Inference, and Prediction, Springer Science & Business Media. https://doi.org/978-0-387-84857-0

Hinton G. E., Dayan P., Frey B. J., Neal R. M. (1995) https://doi.org/10.1126/science.7761831

Kerenidis I., Landman J. (2021) Phys. Rev. A 103:042415. https://doi.org/10.1103/PhysRevA.103.042415

Kerenidis I., Landman J., Luongo A., Prakash A. (2019). In: Proceedings of the 33rd international conference on neural information processing systems vol 372 https://doi.org/10.5555/3454287.3454659

Li Q., He Y., Jiang J. P. (2011) Quantum Inf Process 10:13. https://doi.org/10.1007/s11128-010-0169-y

Li J., Kais S. (2021) Quantum cluster algorithm for data classification, arXiv: 2106.07078 [quant-ph]

Lloyd S., Mohseni M., Rebentrost P. (2013) Quantum algorithms for supervised and unsupervised machine learning. arXiv:1307.0411v2 [quant-ph]

Lloyd S., Mohseni M., Rebentrost P. (2014) Nature Physics. https://doi.org/10.1038/nphys3029

Lloyd S., Schuld M., Ijaz A., Izaac J., Killoran N. (2020) arXiv: 2001.03622, [quanth-ph]

M.S.A.et al (2021) Qiskit: an open-source framework for quantum computing. https://doi.org/10.5281/zenodo.2573505

MacQueen J., et.al (1967) In: Proceedings of the fifth berkeley symposium on mathematical statistics and probability Vol 1 organization Oakland, CA, USA, pp 281-297

Matteo O. D., Gheorghiu V., Mosca M. (2020) IEEE Transactions on Quantum Engineering 1:1–13. https://doi.org/10.1109/TQE.2020.2965803

Nielsen M., Chuang I. (2010) Quantum computation and quantum information, Cambridge University Press. https://doi.org/10.1017/CBO9780511976667

Otterbach J., Manenti R., Alidoust N., Bestwick A., Block M., Bloom B., Caldwell S., Didier N., Fried E. S., Hong S., et.al (2017) Unsupervised machine learning on a hybrid quantum computer. arXiv: 1712.05771 [quant-ph]

Park D. K., Petruccione F., Rhee J. K. K. (2019) Sci Rep 9:1–8. https://doi.org/10.1038/s41598-019-40439-3

Pires D., Bargassa P., Omar Y., Seixas J. (2021) A digital quantum algorithm for jet clustering in high-energy physics, arXiv: 2101.05618 [quant-ph]

Rodriguez A., Laio A. (2014) Science 344:1492. https://doi.org/10.1126/science.1242072

Schuld M., Sinayskiy I., Petruccione F. (2015) Contemp Phys 56:172. https://doi.org/10.1007/s11128-014-0809-8

Sebe N., Cohen I., Garg A., Huang T. S. (2005) Machine learning in computer vision Vol 29, Springer Science & Business Media. https://doi.org/10.1016/j.procs.2020.03.355

Shi W., Lu N., Jiang B., Zhi Y., Xu Z. (2019). In: 2019 Chinese control and decision conference (CCDC) organization IEEE. pp 1954–1959

Sutton R. S., Barto A. G. (2018) Reinforcement learning. An introduction MIT Press, Cambridge. https://doi.org/978-0262193986

Tieleman T. (2008) In: Inproceedings of the 25th international conference on machine learning organization ACM pp. 1064–1071. https://doi.org/10.1145/1390156.1390290

Tu B., Yang X., Li N., Zhou C., He D. (2020) Pattern Recogn Lett 129:144. https://doi.org/10.1016/j.patrec.2019.11.022

Tu B., Zhang X., Kang X., Wang J., Benediktsson J. A. (2019) IEEE Trans Geosci Remote Sens 57:5085. https://doi.org/10.1109/TGRS.2019.2896471

Vincent P., Larochelle H., Bengio Y., Manzagol P. A. (2008) Proceedings of the 25th international conference on Machine learning organization ACM pp. 1096–1103. https://doi.org/10.1145/1390156.1390294

Vinci W., Buffoni L., Sadeghi H., Khoshaman A., Andriyash E., Amin M. (2020) Machine learning: science and technology. https://doi.org/10.1088/2632-2153/aba220

Wiebe N., Braun D., Lloyd S. (2012) Physical review letters 109:050505. https://doi.org/10.1103/PhysRevLett.109.050505

Wittek P. (2014) Quantum machine learning: what quantum computing means to data mining. Academic Press, New York. https://doi.org/978-0128009536

Yu Y., Qian F., Liu H. (2010) Soft. Comput. 14:921

van der Maaten L., Hinton G. (2008) J. Mach. Learn. Res. 9:2579