Quantum Machine Intelligence (2024) 6:62
https://doi.org/10.1007/s42484-024-00197-6

RESEARCH ARTICLE

The role of encodings and distance metrics for the quantum nearest neighbor

Alessandro Berti[1] · **Anna Bernasconi**[1] · **Gianna M. Del Corso**[1] · **Riccardo Guidotti**[1]

Received: 30 November 2023 / Accepted: 2 September 2024

Abstract
Over the past few years, we observed a rethinking of classical artificial intelligence algorithms from a quantum computing perspective. This trend is driven by the peculiar properties of quantum mechanics, which offer the potential to enhance artificial intelligence capabilities, enabling it to surpass the constraints of classical computing. However, redesigning classical algorithms into their quantum equivalents is not straightforward and poses numerous challenges. In this study, we analyze in-depth two orthogonal designs of the quantum K-nearest neighbor classifier. In particular, we show two solutions based on amplitude encoding and basis encoding of data, respectively. These two types of encoding impact the overall structure of the respective algorithms, which employ different distance metrics and show different performances. By breaking down each quantum algorithm, we clarify and compare implementation aspects ranging from data preparation to classification. Eventually, we discuss the difficulties associated with data preparation, the theoretical advantage of quantum algorithms, and their impact on performance with respect to the classical counterpart.

Keywords Quantum algorithms · K-nearest neighbors · Encoding · Distances · QKNN

1 Introduction

Artificial intelligence (AI) gained attention due to its effective usage in many applications (Tan et al. 2005). Nonetheless, guaranteeing high accuracy and low computational time, qualities often critical for these applications, can be challenging. The rise of quantum computers provides peculiar properties to perform computation, such as superposition and entanglement. Exploiting these properties stems new algorithms that rely on a completely different framework with respect to their classical counterpart (Nielsen et al. 2016). In particular, quantum computing (QC) promises to solve certain problems substantially faster than classical computing.

The AI field can leverage QC properties to fit certain requirements not easily achievable by classical computation. Indeed, quantum artificial intelligence (QAI) summarizes approaches that use synergies between artificial intelligence and quantum computing (Lloyd et al. 2013). Among existing QAI approaches, we focus on quantum algorithms processing classical datasets (Schuld and Petruccione 2018). In this setting, preparing classical data into quantum data can be a non-trivial task. In fact, it requires a classical-quantum "interface" typically realized through ad hoc data transformation procedures.

Among the wide range of QAI algorithms, we focus our analysis on a particular subset known as quantum K-nearest neighbor (QKNN) algorithms, which draw inspiration from the classical K-nearest neighbor (KNN). This supervised learning classifier employs a given distance metric to forecast the grouping of a specific data point. The theoretical advantage of QKNN with respect to KNN is that QKNN calculates the distances between the test instance and all the records in the training set simultaneously. In the literature, we find various versions of QKNN (Schuld et al. 2014, 2017; Ruan et al.

Anna Bernasconi, Gianna M. Del Corso, and Riccardo Guidotti contributed equally to this work.

✉ Alessandro Berti
alessandro.berti@phd.unipi.it

Anna Bernasconi
anna.bernasconi@unipi.it

Gianna M. Del Corso
gianna.delcorso@unipi.it

Riccardo Guidotti
riccardo.guidotti@unipi.it

[1] Department of Computer Science, University of Pisa, Largo B. Pontecorvo, Pisa 56127, Italy

Published online: 17 September 2024

2017; Wiebe et al. 2018; Dang et al. 2018; Afham et al. 2020; Li et al. 2021) implementing different distance functions and different data encoding. However, they offer limited details to replicate the results to comprehensively understand the trade-offs associated with the various QKNNs under investigation. In this work, we study and analyze in depth two QKNNs by examining their complexities and empirically testing them on three datasets. We opt for two distinct variants of QKNN, one that encodes the data in the amplitudes of a quantum state and the other in the basis. This choice allows us to analyze and show the impact in terms of complexity and performance of a different encoding approach. Moreover, we provide a Qiskit[1] implementation for each algorithm to make the results reproducible. Within this study, we also investigate if the accuracy scores of QKNN solutions are promising and employable for real-world tasks by comparing them with the classical KNN.

The results of our study show that QKNN can be comparable to or even better than the classic KNN with appropriate data encoding and training strategies. Nevertheless, due to existing technological constraints in executing quantum algorithms on a real quantum computer and simulating those algorithms on a classical computer, it is currently impractical to empirically achieve the superior theoretical complexity of QKNN. It is also evident from the experiments that the costs of using QKNN methods lie in the data preparation and its encoding. Indeed, developing an efficient state preparation technique or quantum memory is mandatory to cut the cost of data encoding in quantum states, thus guaranteeing a speed-up over its classical counterpart.

This work represents the extended version of the conference paper presented in Berti et al. (2022) where our primary focus was on comparing the performance in terms of the accuracy of the quantum classifiers against the classical KNN. This current version offers a much more comprehensive exploration, extending the analysis not only to the accuracy of the algorithms but also to the complexities involved, including a detailed examination of how different encodings and distance functions influence these factors. Specifically, we provide a thorough and self-contained study of quantum algorithms, clarifying critical aspects of their implementation. This includes detailed explanations of each subroutine, providing examples, analytical descriptions, and circuit illustrations along with the Qiskit implementations. To the best of our knowledge, this work represents the first in-depth exploration and comparison of the impacts of the encodings and distance functions on both the accuracy and complexity of quantum nearest neighbor classifiers.

Hereby, we clarify important aspects related to implementing each quantum algorithm. In particular, we shed light on each subroutine, providing examples, analytical descriptions, and circuit illustrations along with the Qiskit implementations. This manuscript is organized as follows: Sect. 2 introduces the motivation of interest for studying the QKNN, Sect. 3 presents the notations and the classical KNN, Sect. 4 describes different encoding techniques of data into a quantum state, Sect. 5 illustrates two different QKNN techniques based on amplitude encoding and basis encoding, respectively, Sect. 6 shows the performance of the classical KNN and the two QKNNs with respect to three different datasets, and eventually Sect. 7 draws the conclusions over the manuscript.

2 Related works

In this section, we review representative works on QKNN. We refer the reader to Schuld and Petruccione (2018) for a comprehensive overview of key concepts and ideas regarding QAI algorithms.

In Schuld et al. (2014), the authors design one of the first proposals of QKNN that relies on a binary data representation combined with Hamming distance for the pattern classification task. The data is encoded in the qubits using *basis encoding* (see Sect. 4.3). The algorithm runs in polynomial time $O(TMN)$ where N is the number of features, M is the number of training instances, and T is the accuracy threshold. The authors also indicate that assuming an efficient state preparation for the training set, the complexity of the proposed algorithm would be independent of the number of training vectors.

Another QKNN based on Hamming distance and basis encoding is discussed in Ruan et al. (2017). Here, the features of the training set are extracted, stored as bit vectors, and mapped to quantum states (Nielsen et al. 2016). Distances between the instance to classify and the training set are computed leveraging quantum parallelism. Moreover, these distances are calculated as Hamming distances exploiting the adder circuit proposed in Kaye (2004) according to a distance threshold. The time cost is $O(N^3)$, where N is the number of features. Note that this evaluation does not take into account the cost of the initial state preparation. Finally, the authors evaluate the performance of their classifier with respect to the solutions proposed in (Wiebe et al. 2018; Lloyd et al. 2013), showing an improvement in terms of accuracy, but also observing that the complexity of the proposed classifier scales only with respect to the number of features N and not with the dataset size.

Another QKNN classifier based on Hamming distance is proposed in Li et al. (2021), where Hamming distances are computed as described in Ruan et al. (2017) and the nearest neighbor is selected through a quantum sub-algorithm for searching the minimum of an unsorted integer sequence (Durr and Hoyer 1999), which is the novelty of this work. The

[1] https://qiskit.org/

overall time complexity of the proposed quantum algorithm is $O(\sqrt{M}\log N(\log M + N\log^2 N))$, where M is the number of instances, and N is the number of features. Thus, when the M instances lie in a low-dimensional feature space (i.e., $N \ll M$), the algorithm classifies a sample with time complexity $O(\sqrt{M}\log M)$, showing a quadratic speedup over its classical counterpart , but also noting that the proposed solution loses its quadratic speedup when applied to high-dimensional instances.

A QKNN encoding data using *amplitude encoding* (see Sect. 4.2) is proposed in Schuld et al. (2017). The algorithm stores instances in the amplitudes of quantum states using a number of qubits logarithmic in the number of features N and training instances M. Then, the distances are computed by estimating the Euclidean distance through the interference between each training instance and the sample to classify. This distance-based quantum classifier uses a simple interference circuit that moves in the opposite direction of quantum circuits with complex subroutines that are difficult to implement in today's quantum computers. The effectiveness of the classifier is demonstrated through experiments on the IBM Quantum Experience and numerical simulations, showing good performance on simple benchmark tasks.

Another amplitude encoding approach is presented in Wiebe et al. (2018) for s-sparse datasets (i.e., only s features are non-zero). This aspect makes this algorithm useful for classifying images with black background. In particular, the authors demonstrate that their oracle-based solution overcomes the low assignment accuracy that occurs in the nearest-centroid approach of Lloyd et al. (2013). Eventually, they also show that the proposed solution is noise-resistant and performs effectively on typical real-world tasks.

An application of QKNN based on amplitude encoding to image classification is presented in Dang et al. (2018). The algorithm extracts and stores the features of all the images in a quantum state. Next, it computes distances in parallel between the M training images and the image to classify. Eventually, a quantum minimum search subroutine (Durr and Hoyer 1999) is applied to get the k neighbors having minimum distance with respect to the sample. The complexity of the quantum algorithm is $O(\sqrt{kM})$ with respect to the complexity $O(M\log k)$ of the classical process: the larger M, the greater the quantum advantage. This work also examines the complexity of the proposed quantum solution compared to the classical one by varying k. Unlike other approaches, the quantum algorithm encodes values in the amplitude and then transfers the information to the basis using the amplitude estimation algorithm (Brassard et al. 2002), enabling the use of the quantum minimum search subroutine (Durr and Hoyer 1999).

Finally, the QKNN in Afham et al. (2020) employs a controlled swap and two Hadamard gates on an ancilla qubit to estimate the fidelity, a distance measure corresponding to cosine similarity. In particular, the fidelity between each data in the training set and the sample to classify is estimated by measuring an ancilla qubit. The width of the algorithm is polylogarithmic in size M of the training set, and the query complexity to extract the k nearest neighbors is $O(\sqrt{kM})$. Unlike other approaches, the class label is not encoded in qubits and is assigned to the sample by classical majority voting.

From the state of the art, it is clear that QKNN has valuable properties that make it more efficient than the classical KNN, at least from a theoretical perspective. However, most of the aforementioned works only provide a few details about the quantum circuits and the data preprocessing needed to run the QKNN algorithms. Our study not only offers a thorough analysis of the performance of different QKNNs but also clarifies key implementation details, crucial to ensure the reproducibility and comparability of different QKNN methodologies. Moreover, we provide the circuit implementations[2] of the analyzed QKNN algorithms.

3 Preliminaries

A classification dataset $\mathcal{D} = \langle X, Y\rangle$ consists of a set $X = \{x^{(0)}, x^{(1)}, \ldots, x^{(M-1)}\}$ of M instances (or records) described by N features (i.e., $x^{(i)} = \{x_0^{(i)}, x_1^{(i)}, \ldots, x_{N-1}^{(i)}\}$) and a set Y of labels $y^{(i} \in \mathbb{N}$ each assigned to an instance $x^{(i)} \in X$. Each label (or class) $y^{(i)}$ is chosen among L the labels in a set V, i.e., $L = |V|$. In AI, given a dataset $\mathcal{D}$, the objective is to find a function f that assigns to an unseen instance u a label y, i.e., $y = f(u)$, such that $y = \hat{y}$ where $\hat{y}$ is the ground truth of u. The performance of an AI classifier can be measured in terms of *accuracy* that accounts for the number of correspondences between the predicted labels and the ground truth. Our objective is to illustrate and analyze how this problem can be solved with QAI procedures modeling the well-known K-nearest neighbor (KNN) classifier (Tan et al. 2005).

3.1 Classical *K*-nearest neighbors

K-nearest neighbors (KNN) is a supervised AI algorithm implementing a classifying function f. The core idea behind KNN is that instances similar in the feature space must also be similar with respect to the class label. It has been successfully applied to many problems: classification of tabular data (Tan et al. 2005), text (Manning et al. 2008), time series (Fawaz et al. 2019), parameter optimization (Nigsch et al. 2006), and also outlier detection (Ramaswamy et al. 2000).

[2] Code available at: https://github.com/Brotherhood94/exploring_different_encoding_and_distance_metrics_for_a_quantum_instance_based_classifier

KNN takes as input a set of training examples $\mathcal{D}$ and the number of nearest neighbors k. Then, given a distance function d, for each unseen instance u, it computes the distance between u and all the instances in $\mathcal{D}$. Then, it selects from $\mathcal{D}$ the k instances $\mathcal{D}_u \subseteq \mathcal{D}$ having the smallest distance from u (i.e., $\mathcal{D}_u$ are the nearest neighbors of u). Finally, it assigns to u a label y based on the majority vote of the nearest neighbors according to the following formula:

$$y = \arg\max_{v \in V} \sum_{(x^{(i)}, y^{(i)}) \in \mathcal{D}_u} I(v = y^{(i)}),$$

where v is a class label, $y^{(i)}$ is the class label of the nearest neighbors $x^{(i)}$, and $I(\cdot)$ returns 1 if its argument is true, 0 otherwise.

Despite being simple, KNN can be characterized by many variants (Tan et al. 2005). For instance, the choice of the number of neighbors denoted as k determines the sensitivity of KNN and has an impact on the classification results. Additionally, the selection of the distance function, denoted as d, is important. Typically, the Euclidean distance is employed for continuous features, while the Hamming distance is used for categorical features. We highlight that, since KNN relies on distances, if the features are modeled with vastly different scales, normalizing the training data can dramatically improve the accuracy (Tan et al. 2005). Furthermore, the majority vote for the class selection can be weighted.

The main weakness of KNN is the computation of the distances. Indeed, given an instance u, KNN has to compute the distance between u and every instance in the training set $\mathcal{D}$. Thus, the computational complexity of KNN strictly depends on the number of features N describing an instance and on the number of instances M in the training set. This leads to a computational complexity of $O(NM)$. A possible workaround to this issue is to reduce the number of instances in the training set by using *centroids* instead of real data (Manning et al. 2008). In this version, named *nearest centroid classifier*, the unseen instance u is compared with a small set of instances representative of the classes of the problem, typically obtained by averaging the features of the records in each class. However, this approach typically reduces the accuracy of the classifier, which no longer relies on "real" records but on prototypical ones. As an alternative, KNN can refer to a sample $\mathcal{D}' \subset \mathcal{D}$ of instances randomly selected, with $M' < M$ and $M' = |\mathcal{D}'|$.

Regardless, even when reducing the number of training instances, K-nearest neighbors (KNN) still requires the computation of distances between the unseen instance u and the training set. What makes quantum K-nearest neighbors (QKNN) intriguing is its ability to surmount this limitation by simultaneously calculating all distances.

4 Encoding data into quantum states

In this section, we review two different techniques to encode classical information into quantum state typically used for QKNN algorithms (Schuld and Petruccione 2018) and then discuss the general state preparation problem. We remark that the question of data encoding becomes crucial for QAI algorithms as it can heavily impact the performance both in terms of the quality of the results and time.

4.1 Quantum state preparation

Quantum state preparation consists of the initialization of a quantum register that encodes classical data. This is a fundamental and crucial step for practical applications. To preserve the potential advantages of quantum algorithms for big data applications, quantum state preparation must be performed efficiently. However, the initialization of the quantum state may result in computationally expensive in terms of the width and depth of the circuit implementing a given state preparation technique.

Different techniques for loading classical data into a quantum state have been proposed and implemented. A standard approach for quantum state preparation is the method proposed in Shende et al. (2006) and implemented on the IBM Qiskit framework. The idea is to assume that the n-qubit quantum register is already in the desired state $|\psi\rangle$ and to construct a circuit that takes $|\psi\rangle$ to the zero state $|0\rangle^{\otimes n}$ of the computational basis. The state preparation circuit is then the reverse of such a circuit. The arbitrary state $|\psi\rangle$ is transformed into the zero state through an iterative procedure that disentangles the n qubits one by one, starting from the least significant one. Each disentangled qubit state can be taken to $|0\rangle$ using appropriate one-qubit elementary rotation gates. The overall resulting circuit contains at most $2^{n+1} - 2n$ CNOT gates, in addition to the rotation gates, and is asymptotically optimal as it lies a factor of four away from the theoretical lower bound (Shende et al. 2004; Möttönen et al. 2005).

Another method to transform any given n-qubit quantum register into an arbitrary state using uniformly controlled one-qubit rotations is proposed in Möttönen et al. (2005). The circuit sequentially applies $2^{n+2} - 4n - 4$ CNOTs and $2^{n+2} - 5$ one-qubit elementary rotations, in the worst case, and can be simplified in case of initial and target states with suitable symmetries. For instance, if one of the two states coincides with some vector of the computational bases, only half of the gates are needed.

A different approach to state preparation is based on the use of *Quantum Random Access Memory* (QRAM) (Ventura and Martinez 2000; Giovannetti et al. 2008; Park et al.

2019), a device that stores quantum data. A QRAM mimics the functionalities of RAM in classical computers and has the same three basic components as the RAM: a memory array, an address register, and an output register. Address and output registers are composed of qubits instead of bits, while the memory array can be either quantum or classical, depending on the QRAM's usage. Most importantly, the QRAM has the ability to perform memory accesses in coherent quantum superposition: if the quantum computer needs to access a superposition of memory cells, the address register must contain a superposition of addresses, and the QRAM will return a superposition of data in the output register. The possibility of efficiently implementing these devices, which will become an essential component of quantum computers if large quantum computers are eventually built, would yield a speedup for many quantum algorithms over classical data. In the literature, various frameworks implementing this model have been proposed. However, it is worth noting that a hardware implementation of a QRAM is still under development.

A first attempt in this direction is proposed in Ventura and Martinez (2000) for the basis encoding of binary data. This method can construct a quantum state $|\mathcal{D}\rangle$ representing a classical dataset $\mathcal{D}$ of M binary instances, of length ℓ bits each, in time linear in ℓ and M. We adopt this technique to encode data in the QKNN of Sect. 5.2. Section 5.2.2 provides a detailed description of this strategy.

In Giovannetti et al. (2008), the authors introduce a QRAM model, called *bucket brigade*, that requires $O(\log M)$ address qubits, $O(M)$ qutrits (i.e., three-level quantum systems) used as switches for routing, and $O(M)$ classical or quantum memory cells for M binary data. This architecture exponentially reduces the requirements for a memory call: even if it still requires $O(M)$ switches for routing, it only activates $O(\log^2 M)$ of them. This yields a more robust QRAM algorithm, as it entails an exponential reduction in the number of gates that need to be entangled for each memory call, leading to an exponential decrease in the power needed for addressing.

Another QRAM model is the *Flip-Flop* QRAM (FF-QRAM) (Park et al. 2019). The FF-QRAM can read unsorted classical data stored in memory cells and superpose them in the computational basis states with non-uniform probability amplitudes to create a specific quantum state for a given algorithm. FF-QRAM allows the encoding of discrete or continuous classical information as quantum bits or probability amplitudes of a quantum state. For the encoding of classical data consisting of M entries, each with N features, FF-QRAM requires $O(MN)$ quantum operations. We refer the reader to Sect. 5.1.2 for a detailed description of FF-QRAM, as this is the strategy adopted for the initial state preparation in our implementation of the QKNN with amplitude encoding Sect. 5.1.

The paper (Araujo et al. 2021) proposes a new format for data encoding. It exploits the method of Möttonen et al. (2005) and a divide-and-conquer approach using controlled swap gates and ancilla qubits. The method achieves an exponential quantum speedup in time to load an N-dimensional real vector in the amplitude of a quantum state with a quantum circuit of depth $O(\log_2^2(N))$, and using $O(N)$ qubits. Thus, the decrease of the circuit depth is obtained at the expense of the width of the circuit, creating entanglement between data register qubits and an ancillary system. After filtering out the ancilla qubits, we do not get the pure state that one obtains with amplitude encoding but a mixed state where the classical data is still encoded as amplitudes of an orthonormal basis set. As shown in the paper, there are, however, applications where we can work directly with this encoding.

In the literature, other versions of QRAMs are discussed (Möttönen et al. 2005; Araujo et al. 2021; Soklakov and Schack 2006; Long and Sun 2001; Kerenidis and Prakash 2017). See also Phalak et al. (2023) for a general review on this subject.

4.2 Amplitude encoding

Amplitude encoding associates classical data with the amplitudes of a quantum state. Given a vector $x = (x_0, \ldots, x_{N-1})$ of classical data, we can encode the information in the following quantum state:

$$|\psi\rangle = \sum_{j}^{N-1} x_j |j\rangle \,.$$

where the register $|j\rangle$ is a $\lceil \log N \rceil$-qubit register sufficiently large to store all indexes $j = 0, 1, \ldots, N-1$ in superposition. We recall that the norm of a quantum state must be equal to 1; thus, before amplitude encoding a vector x, we need to normalize it, so that $\sum_j |x_j|^2 = 1$. Another detail concerns the length N of the vector. In particular, if the vector length is not a power of two, it must be padded with zeros up to the next power of two. For simplicity, we assume that N is a power of two. Overall, this encoding requires $n = \log_2 N$ qubits.

For instance, let us show how to preprocess $x = (0.3, 0.4, 0.8)$. Since the length of the vector is $N = 3$, we need to pad the vector to the next power of two: $x = (0.3, 0.4, 0.8, 0.0)$. Then, we normalize to unit length (i.e., to divide it by $\sqrt{\sum_j x_j^2}$), landing on the following vector $x = (0.32, 0.42, 0.85, 0.00)$ that can be now amplitude encoded in a quantum state $|\psi\rangle$.

The main advantage of amplitude encoding is that encoding a dataset with M instances and N features only requires $O(\log_2 N + \log_2 M)$ qubits. Further details for loading a

real dataset into a quantum state with amplitude encoding are presented and discussed in Sect. 5.1.2.

4.3 Basis encoding

Basis encoding associates a classical ℓ-bit-string with a computational basis state of an ℓ-qubit system. For instance, we can encode the string 0011 as the basis state $|0011\rangle$. In this way, if we use s bits to encode a feature, each N-feature instance is represented with $\ell = N\,s$ bits. Thus, given a *binary* record $x^{(i)} = (x_0^{(i)} x_1^{(i)} \dots x_{\ell-1}^{(i)})$, we can prepare basis states $|x^{(i)}\rangle$ where each qubit corresponds to a bit of the binary input:

$$|\psi\rangle = |x_0^{(i)} x_1^{(i)} \dots x_{\ell-1}^{(i)}\rangle = |x^{(i)}\rangle. \tag{1}$$

We highlight that turning a dataset $\mathcal{D}$ into a binary one using a basis encoding approach is not necessarily a trivial task, as discussed in Sect. 5.2.2.

5 Quantum KNN algorithms

In this section, we re-design quantum KNN algorithms in the state-of-the-art, highlighting technical details, similarities, and differences between different procedures. There are two fundamental aspects for which QKNN algorithms can differ:

- The *distance function*
- The *data encoding*

The choice of a distance function highly impacts the data encoding; specifically, a distance function that computes the distances on the binary representation of the data is substantially different with respect to a distance function acting on an amplitude encoding of the data. Moreover, it is worth noting that quantum computation evolves the initial quantum state encoding a given dataset. Thus, a new state preparation of this quantum state from scratch occurs to classify every new instance (Schuld et al. 2017). A common feature among the QKNN approaches under investigation is the parallel computation of distances between the training set and the instance to classify. This aspect is what makes QKNN an appealing quantum algorithm.

The implementations of QKNNs in the literature often lack details, making it difficult to reproduce the results. Hence, our goal is to analyze and implement a complete QAI procedure and provide reproducible research. Moreover, we provide a Qiskit implementation of each QKNN algorithm investigated. We highlight that the simulation of quantum algorithms on classical devices requires a significant amount of resources that scale exponentially according to the number of qubits. Therefore, we perform the experiments to fit all the available classical resources.

To simplify the presentation, with a little abuse of notation, we will use the notation $|r\rangle$ to denote the contents of a quantum register r, of one or one qubits.

5.1 QKNN with amplitude encoding

We describe here an implementation of QKNN inspired by Schuld et al. (2017). The fundamental concept involves the encoding of data within the amplitudes of a quantum state, as expounded in Sect. 4.2. The classification of an instance employs an estimate of the *Euclidean distance*. For the sake of conciseness, we shall herein refer to this implementation as *amplitude encoding-based QKNN*, succinctly denoted as *aQKNN*.

5.1.1 Quantum Euclidean distance

Let us consider two quantum states $|\delta\rangle$ and $|\phi\rangle$ that amplitude encodes N features each. We want to compute an estimate of the Euclidean distance between $|\delta\rangle$ and $|\phi\rangle$:

$$d(|\delta\rangle, |\phi\rangle) = \sqrt{\sum_{i=0}^{N-1} (\delta_i - \phi_i)^2} = \||\delta\rangle - |\phi\rangle\|,$$

where δ_i and ϕ_i correspond to the i-th amplitudes of $|\delta\rangle$ and $|\phi\rangle$, respectively.

To compute the Euclidean distance $d(|\delta\rangle, |\phi\rangle)$, we require an ancillary qubit $|0\rangle_e$ and $n = \log_2 N$ qubits to amplitude encode the quantum states $|\delta\rangle$ and $|\phi\rangle$. The idea behind the quantum algorithm for computing the Euclidean distance consists in leveraging the phenomena of interference. In particular, we put in equal superposition the ancillary qubit $H|0\rangle_e = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$, and we amplitude encode and entangle $|\delta\rangle$ ($|\phi\rangle$) with the branch $|1\rangle_e$ ($|0\rangle_e$) of the ancillary qubit. Then, we make the branches interfere by means of a Hadamard gate on the ancillary qubit $|e\rangle$. Eventually, a post-selection on $|e\rangle = |1\rangle$ occurs. The probability of measuring the ancillary qubit $|e\rangle$ in state $|1\rangle$ results in an estimate of the Euclidean distance between $|\delta\rangle$ and $|\phi\rangle$. Figure 1 describes the quan-

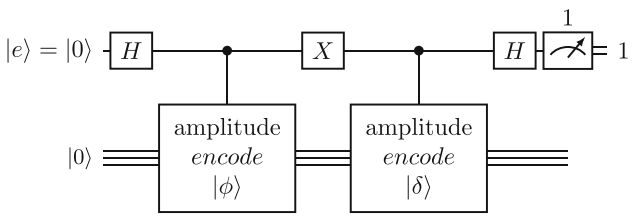

Fig. 1 Circuit computing the Euclidean distance

tum circuit implementing the corresponding algorithm. A detailed description follows.

The initial state corresponds to $|0\rangle_e|0\rangle$. Then, we apply the first Hadamard gate on the qubit in register e:

$$\frac{1}{\sqrt{2}}(|0\rangle_e|0\rangle + |1\rangle_e|0\rangle).$$

Now, we have two branches, $|0\rangle_e$ and $|1\rangle_e$, which we entangle respectively with $|\delta\rangle$ and $|\phi\rangle$ landing in the subsequent quantum state:

$$\frac{1}{\sqrt{2}}(|0\rangle_e|\delta\rangle + |1\rangle_e|\phi\rangle).$$

Then, the last Hadamard gate produces the interference between $|\delta\rangle$ and $|\phi\rangle$:

$$\frac{1}{2}(|0\rangle_e(|\delta\rangle + |\phi\rangle) + |1\rangle_e(|\delta\rangle - |\phi\rangle)).$$

Eventually, the probability of measuring the ancillary qubit $|e\rangle$in state 1 is given by the following:

$$P(|e\rangle = |1\rangle) = \|\frac{1}{2}(|\delta\rangle - |\phi\rangle)\|^2 = \frac{1}{4}\||\delta\rangle - |\phi\rangle\|^2 = \frac{1}{4}d(|\delta\rangle, |\phi\rangle)^2;$$

therefore, we have an estimate of the Euclidean distance:

$$d(|\delta\rangle, |\phi\rangle) = 2\sqrt{P(|e\rangle = |1\rangle)}$$

5.1.2 Encoding a dataset in the amplitudes

Multiple state preparation techniques describe how to encode a dataset in the amplitude of a quantum state, as previously discussed. Hereby, we employ the FF-QRAM (Park et al. 2019) to map a dataset into a quantum state. Given a dataset $\mathcal{D}$ with M instances and a single feature each, we need $m = \log_2 M$ qubits to address a specific instance $x^{(i)}$, where $0 \leq i < M$. In particular, we define a quantum register $|a^{(i)}\rangle = |a_0^{(i)} a_1^{(i)} \dots a_{m-1}^{(i)}\rangle$, and an additional qubit $|r\rangle$ that stores the single feature $x_0^{(i)}$ of $x^{(i)}$ via amplitude encoding. We denote by $|x_0^{(i)}\rangle_r$ the qubit which amplitude-encodes the feature $x_0^{(i)}$, i.e., $|x_0^{(i)}\rangle_r = \sqrt{1-(x_0^{(i)})^2}|0\rangle_r + x_0^{(i)}|1\rangle_r$. The FF-QRAM models the dataset $\mathcal{D}$ as follows:

$$FF\text{–}QRAM(\mathcal{D}) = \sum_{i=0}^{M-1} |a^{(i)}\rangle(\sqrt{1-(x_0^{(i)})^2}|0\rangle_r + x_0^{(i)}|1\rangle_r), \tag{2}$$

where $|a^{(i)}\rangle$ acts as a *register* index that identifies a given memory address. The branch corresponding to $|r\rangle = |1\rangle$ contains our instances while the branch $|r\rangle = |0\rangle$ can be ignored. For example, let us define $|a^{(0)}\rangle = |00\dots0\rangle$ as the register index that locates the memory address storing the feature $x_0^{(0)}$ in qubit $|x_0^{(0)}\rangle_r$. Then, according to this logic, the register index $|a^{(1)}\rangle = |00\dots01\rangle$ locates the feature $x_0^{(1)}$. In general, a register index $|a^{(i)}\rangle$ corresponds to the binary encoding of i on $m = \log_2 M$ bits.

Let us show the step-by-step idea of an FF-QRAM that maps in a quantum state a dataset $\mathcal{D}$ of M instances, one feature each. The first step is to normalize the classical dataset $\mathcal{D}$. Then, we allocate a quantum register index $|a\rangle = |0\rangle^{\otimes m}$ of size $m = \log_2 M$ qubits plus an additional qubit $|0\rangle_r$, so that the initial quantum state $|\psi_0\rangle$ is

$$|\psi_0\rangle = |0\rangle_a^{\otimes m}|0\rangle_r.$$

At this point, we have a single memory address $|0\rangle_a^{\otimes m}$, but we need to store M values. Therefore, we *create* the extra memory addresses by putting each qubit of the quantum register $|0\rangle_a^{\otimes m}$ into equal superposition:

$$|\psi_1\rangle = H|0\rangle_a^{\otimes m}|0\rangle_r = \frac{1}{\sqrt{M}}\sum_{i=0}^{M-1}|a^{(i)}\rangle|0\rangle_r.$$

The last step involves mapping the feature value $x_0^{(i)}$ in the amplitude of the register r. For each normalized instance $x_0^{(i)}$, we compute $\theta_i = 2\arcsin(x_0^{(i)})$. Then, we perform a rotation $R_y(\theta_i)$ controlled on the quantum register $|a^{(i)}\rangle$ for each θ_i. The key point is that by rotating a given qubit by $\theta_i = 2\arcsin(x_0^{(i)})$, we land in the state $R_y(\theta_i)|0\rangle = \sqrt{1-(x_0^{(i)})^2}|0\rangle + x_0^{(i)}|1\rangle$, where the amplitude associated with $|1\rangle$ coincides with our normalized feature $x_0^{(i)}$. The final state that describes the FF-QRAM encoding a dataset $\mathcal{D}$ with M instances and one feature each is therefore given by

$$|\psi_2\rangle = \sum_{i=0}^{M-1}|a^{(i)}\rangle(\sqrt{1-(x_0^{(i)})^2}|0\rangle + x_0^{(i)}|1\rangle) = FF-QRAM(\mathcal{D}).$$

It is worth noting that the FF-QRAM rotates (i.e., stores the feature value $x_0^{(i)}$) the same qubit $|r\rangle$, but it does not overwrite the previous value $x_0^{(i-1)}$ since the register index switched from $|a^{(i-1)}\rangle$ to $|a^{(i)}\rangle$. In other words, the M values $x_0^{(i)}$ are stored in the amplitudes of *different copies* in a superposition of the same qubit $|r\rangle$.

From a circuit perspective, to implement a basic FF-QRAM, we employ the following gates: the Hadamard gate, the X gate, and the Multi Controlled-R_y (MCRy) gate, according to the following intuition. The Hadamard gate creates a superposition of the index register $|a\rangle$; the X gate enables us to target a specific index register $|a^{(i)}\rangle = |\text{bin}(i)_m\rangle$, and the MCR$y$ gate amplitude-encodes the feature value $x_0^{(i)}$ in the register $|r\rangle$.

An example follows. Let us assume $M = 4$, then we need $m = \log_2(4) = 2$ qubits for the register index $|a\rangle$ to address 4 different memory locations. Initially, $|a\rangle = |00\rangle$; thus, we apply H over $|a\rangle$ obtaining the following superposition: $|\psi\rangle = \frac{1}{\sqrt{4}}(|00\rangle + |01\rangle + |10\rangle + |11\rangle)$. We then *select* the appropriate register index $|a^{(i)}\rangle$ to store the feature value $x_0^{(i)}$ in the amplitude of $|r\rangle$. For instance, let us assume $i = 2$, and consider the register index $|a^{(2)}\rangle = |10\rangle$. We move to the memory address $|10\rangle$ by means of X gates. Eventually, we store the feature $x_0^{(2)}$ by *selectively* rotating the qubit $|r\rangle$ associated with the register index superposition $|a^{(2)}\rangle = |10\rangle$. Practically, this last step requires an MCRy gate controlled on $|a\rangle$ and targeting $|r\rangle$ (see Fig. 2).

However, it is unlikely that an instance has only one feature. The logic to extend the FF-QRAM defined in Eq. 2 to handle more than one feature is similar to the one we used to address memory locations. We add a *second level addressing* employing additional qubits. Let us define N as the number of features; then we require $n = \log_2 N$ additional qubits to index a given feature j, where $0 \leq j < N$, of a given instance i, $0 \leq i < M$. We denote these qubits as $|b^i\rangle = |b_0^{(i)} b_1^{(i)} \dots b_{n-1}^{(i)}\rangle$. The overall quantum state is therefore

$$FF-QRAM(\mathcal{D}) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |a^{(i)}\rangle |b_j^{(i)}\rangle |x_j^{(i)}\rangle_r.$$

The last definition deals with an FF-QRAM that also encodes the class of a given instance. Indeed, in the AI field, it is typical to deal with labeled instances. Thus, we complete the whole picture by describing an FF-QRAM that takes into account classes. Let us define L as the total number of distinct classes within a dataset $\mathcal{D}$. Then, according to the same rationale of the previous FF-QRAM definition, we introduce $l = \log_2 L$ qubits to map the classes into a quantum state. We define those qubits as $|c^{(i)}\rangle = |c_0^{(i)} c_1^{(i)} \dots c_{l-1}^{(i)}\rangle$ where $|c^{(i)}\rangle$ is the class index of the instance $x^{(i)}$, for $0 \leq i < M$. Thus, we have

$$FF-QRAM(\mathcal{D}) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |a^{(i)}\rangle |b_j^{(i)}\rangle |x_j^{(i)}\rangle_r |c^{(i)}\rangle. \quad (3)$$

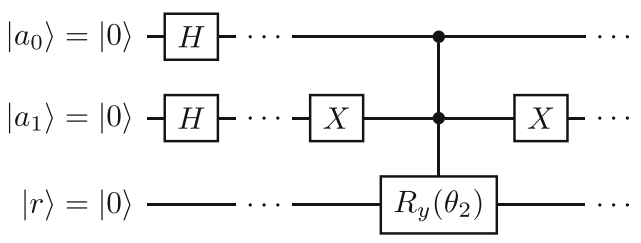

Fig. 2 Amplitude encoding of the feature $x_0^{(2)}$ in index register $|a^{(2)}\rangle = |10\rangle$

In summary, utilizing the FF-QRAM for encoding a dataset $\mathcal{D}$ comprising M training instances, each with N features, and encompassing L distinct classes requires $O(\log_2 N + \log_2 M + \log_2 L)$ qubits.

5.1.3 Quantum classification with Euclidean distance

Given a dataset $\mathcal{D}$, the aQKNN under analysis classifies an instance u leveraging quantum parallelism and employing the quantum Euclidean distance as a distance function. We remind the reader of the notation adopted. In particular:

- M is the number of instances in $\mathcal{D}$, and $m = \log_2 M$ is the number of qubits required to index the instances.
- N is the number of features per instance in $\mathcal{D}$, and $n = \log_2 N$ is the number of qubits required to index each feature of a given instance.
- L is the number of distinct classes in $\mathcal{D}$, and $l = \log_2 L$ is the number of qubits needed to represent all the possible classes.

Then, we denote as $|a\rangle^{\otimes m}$, $|b\rangle^{\otimes n}$, and $|c\rangle^{\otimes l}$ the registers to amplitude encode the instance $t$and the training instances in $\mathcal{D}$. In particular, $|a\rangle^{\otimes m}$ indexes the instances, $|b\rangle^{\otimes n}$ the features, and $|c\rangle^{\otimes l}$ the classes. We refer to $|u\rangle$ and $|\mathcal{D}\rangle$ as the quantum states that amplitude encodes u and $\mathcal{D}$, respectively, following the encoding. Note that the instance u is the one we want to classify; thus, we encode along with u an equal superposition of all the classes instead of a specific one. In some sense, since we do not know the actual class of u, we say that it can be *any class* at the beginning of the algorithm.

To implement the quantum Euclidean distance, we add an ancillary qubit $|e\rangle = 0$, and then we apply an H-gate to it (i.e., $|e\rangle = |+\rangle_e = \frac{|0\rangle_e + |1\rangle_e}{\sqrt{2}}$). Afterwards, we entangle $|0\rangle_e$ with $|u\rangle$, and $|1\rangle_e$ with the quantum state $|\mathcal{D}\rangle$. Eventually, we apply the H-gate on the ancillary qubit $|e\rangle$, thus computing simultaneously all Euclidean distances among $|u\rangle$ and each training instance in $|\mathcal{D}\rangle$. Figure 3 illustrates the circuit implementation.

The last step of the algorithm consists of a post-selection. Indeed, only a subset of superpositions of our final quantum state contributes to the classification. Namely, we post-select on those quantum states such that registers $|r\rangle = 1$ and $|e\rangle = 0$. Then, we measure the register $|c\rangle$, which encodes the predicted class for the instance u. Indeed, the ancilla $|e\rangle$ in the state $|0\rangle_e$ holds the amplitudes depending on the distances between the instance u and all the training ones (see Sect. 5.1.1). A measurement on $|c\rangle$ returns the most probable class for u according to the distances with respect to the training instances.

This method relates to KNN when setting $k = M$, where k defines the number of neighbors (i.e., the training instances)

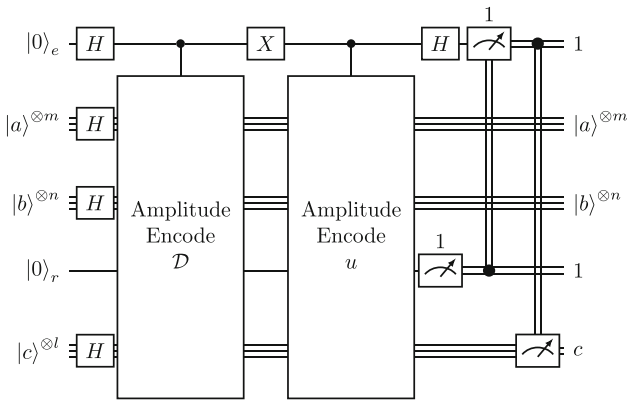

Fig. 3 Overall circuit for the classification of u with Euclidean distance with respect to the dataset $\mathcal{D}$

taken into account, and weighing the neighborhood by the distance measure (Schuld et al. 2017).

Complexity analysis The overall complexity of aQKNN is dominated by the cost of encoding the training set into a quantum state. In fact, employing the FF-QRAM to prepare the quantum state requires $O(\log_2 M + \log_2 N + \log_2 L)$ qubits and $O(MN)$ operations. It is worth noting that the classical KNN also requires the same cost, which, instead, is due to the computation of the distance between the training instances and the instance to classify. Therefore, we will only have an advantage over the classical KNN when a state preparation routine with a (poly)logarithmic cost is effectively available. We also observe that the post-selections on $|r\rangle$ and $|e\rangle$ reduce the success probability of the algorithm, thus requiring a non-negligible number of repetitions of the whole quantum algorithm. In particular, according to Schuld et al. (2017), the success probability of the post-selection on $|e\rangle$ is $\frac{1}{2}$ (i.e., $P(|e\rangle = 1) = \frac{1}{2}$). While the success probability of $|r\rangle$ is data dependent; specifically, $P(|r\rangle = 1) = \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |\sin(x_j^{(i)})|^2$, thus, as discussed in Park et al. (2019) , the number of shots scales as $\frac{1}{P(|r\rangle=1)}$. Therefore, the overall success probability of a given run is $P(|e\rangle = 1) \times P(|r\rangle = 1) = \frac{1}{2} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} |\sin(x_j^{(i)})|^2$.

5.2 QKNN with basis encoding

In this section, we introduce a version of the quantum K-nearest neighbors (QKNN) algorithm, leveraging the Hamming distance as the chosen metric for quantifying similarity and employing the basis encoding technique as elucidated in Sect. 4.3. We denote this algorithm as *bQKNN*. As detailed in Li et al. (2021), this algorithm employs a searching algorithm to find the minimum distance to accomplish the classification task for a given instance.

5.2.1 Quantum Hamming distance

The Hamming distance is a measure of dissimilarity between two binary strings σ and τ with the same length ℓ. It quantifies the number of positions at which the two strings differ. We evaluate the Hamming distance of two strings by computing the bitwise XOR of the two strings and then counting the number of 1 s in the resulting sequence.

The Hamming distance circuit consists of the following steps:

1. Basis encode (see Sect. 4.3) the two binary vectors σ and τ of length ℓ in the two ℓ-qubits quantum registers $|\sigma\rangle$ and $|\tau\rangle$.
2. Perform the bitwise XOR between $|\sigma\rangle$ and $|\tau\rangle$ using CNOT gates. Indeed, $\text{CNOT}(|\sigma_j\rangle, |\tau_j\rangle) = |\sigma_j\rangle|\sigma_j \oplus \tau_j\rangle$, for $0 \leq j < \ell - 1$. Let $|h_j\rangle$ denote $|\sigma_j \oplus \phi_j\rangle$, for $0 \leq j < \ell - 1$.
3. Counting the 1s in $|h\rangle = |h_0, \ldots, h_{\ell-1}\rangle$ returns the Hamming distance. The sum of the 1s in $|h\rangle$ is computed with the circuit discussed in Kaye (2004) that takes in input the register $|h\rangle$ and a register $|g\rangle$ of $p = \lfloor \log_2 \ell \rfloor + 1$ qubits needed to represent the sum of the 1 s in $|h\rangle$.

Figure 4 illustrates the quantum circuit that computes the Hamming distance.

5.2.2 Encoding a dataset in the basis

Basis encoding consists of encoding a given string of ℓ bits in the computational basis of a quantum state of ℓ-qubits (see Sect. 4.3). In this section, we consider the basis encoding technique presented in Ventura and Martinez (2000), and we apply this technique on a binary dataset $\mathcal{D}$ of M instances and N features each. It is worth noting that the number of

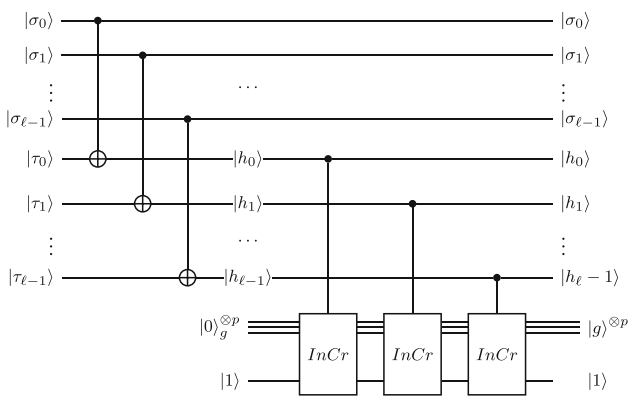

Fig. 4 Quantum circuit computing Hamming distance between two generic quantum states $|\sigma\rangle$ and $|\tau\rangle$

instances impacts the depth of the circuit, while the number of features and their binary representation impacts the width of the circuit.

Once the classical data have been encoded as binary strings into $\mathcal{D}$, we can basis-encode the dataset into a quantum state $|\mathcal{D}\rangle$ according to the strategy of Ventura and Martinez (2000). In order to encode the M binary instances of length ℓ, the quantum circuit requires $\ell+2$ qubits, logically organized as follows:

- an ℓ-qubit register $|x^{(i)}\rangle = |x_0^{(i)} x_1^{(i)} \dots x_{\ell-1}^{(i)}\rangle$ for the M binary instances;
- a 2-qubits ancillary register $|z\rangle$.

Note that the register $|x^{(i)}\rangle$ has as many qubits as the number of bits of the instance intended for encoding. Therefore, we can represent any possible input binary pattern by flipping the qubits from 0 to 1 according to the corresponding bits of the input instance. We describe below the steps to basis-encode the $(i+1)$-th instance $x^{(i)}$, assuming that the previous i instances have already been encoded.

1. Selective-flip the $|x^{(i)}\rangle$ qubits to match the binary encoding of instance $x^{(i)}$ on the ℓ qubits. We perform this operation by applying a CNOT controlled on $|z_1\rangle = |0\rangle$ that targets the qubits corresponding to the bit equals to 1 of the $(i+1)$-th instance $x^{(i)}$.
2. Flip the qubit $|z_0\rangle$ if the state of $|z_1\rangle = |0\rangle$ and apply the controlled-rotation gate R_μ defined as follows:

$$R_\mu = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{\frac{\mu-1}{\mu}} & \frac{-1}{\sqrt{\mu}} \\ 0 & 0 & \frac{1}{\sqrt{\mu}} & \sqrt{\frac{\mu-1}{\mu}} \end{bmatrix} \tag{4}$$

where $\mu = M - i$ is the number of instances, including the current one, yet to be encoded. This gate takes $|z_0\rangle$ as the control qubit and $|z_1\rangle$ as the target qubit. The idea behind this step is to split the quantum state into two branches: a branch b_1 containing the first $i+1$ instances in superposition, each with amplitude $\frac{1}{\sqrt{M}}$, and a branch b_2 with the new instance $x^{(i)}$, with amplitude $\sqrt{1-\frac{i+1}{M}}$. The branch b_1 (where $|z\rangle = |01\rangle$) permanently encodes the binary instances up to the $(i+1)$-th.
3. Eventually, we restore b_2 so that $|x^{(i)}\rangle = |0\rangle^{\otimes \ell}$.
4. Repeat from Step 1 to load the next instance.

Figure 5 illustrates the quantum circuit to encode the binary record 001 with basis encoding. Once all the M binary instances have been encoded, the system ends up in the following state:

$$|\mathcal{D}\rangle = \frac{1}{\sqrt{M}} \sum_{i=0}^{M-1} |x^{(i)}\rangle. \tag{5}$$

The quantum state $|\mathcal{D}\rangle$ has value $\frac{1}{\sqrt{M}}$ for each record $x^{(i)}$, and zero otherwise. Since the number of amplitudes 2^M can be larger than the number of nonzero amplitudes, basis encoding datasets can generate sparse amplitude vectors.

5.2.3 Quantum classification with Hamming distance

Given a dataset $\mathcal{D}$ of M training instances of N features each, we want to classify an instance u exploiting quantum parallelism by means of the Hamming distance. First, we prepare the quantum state $|\mathcal{D}\rangle$ encoding the dataset $\mathcal{D}$ as described in the previous section. Then, we basis-encode the instance u that we want to classify in an additional quantum register of ℓ-qubits, obtaining $|u\rangle = |u_0 u_1 \dots u_{\ell-1}\rangle$. Next, we employ the circuit illustrated in Fig. 4 to compute the Hamming distances in parallel. We recall that, after this step, the register $|h\rangle$ contains the superposition of the Hamming distances between the instance $|u\rangle$ and each training instance $|x^{(i)}\rangle \in \mathcal{D}$.

The last step consists of the classification of the instance $|u\rangle$ (see Fig. 6). As the Hamming distance is a measure of similarity between binary strings, we focus on identifying instances that minimize this distance with respect to u. This preference for a smaller Hamming distance indicates greater similarity between the binary strings. We accomplish this task by applying the strategy presented in Li et al. (2021). In particular, the bQKNN involves leveraging the algorithm proposed in Durr and Hoyer (1999), which finds the minimum of an unordered integer sequence. Indeed, the algorithm returns the instance $|x^{(i)}\rangle$ that minimizes the Hamming dis-

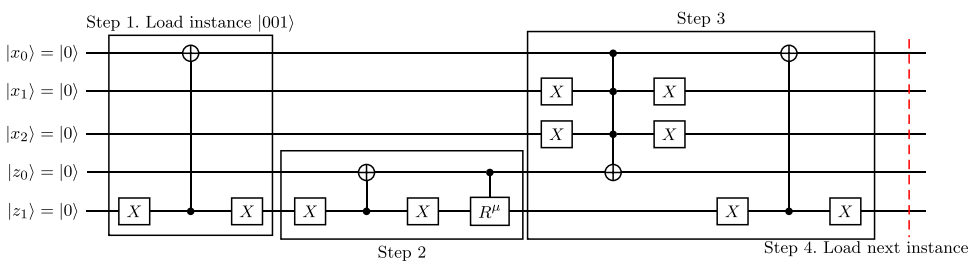

Fig. 5 Basis encoding binary instance $|001\rangle$

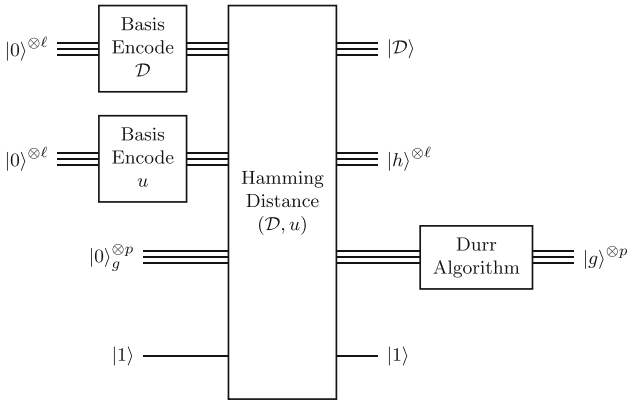

Fig. 6 Overall circuit for the classification of a bit string u with Hamming distance with respect to the dataset $\mathcal{D}$

tance with respect to $|u\rangle$. Eventually, we classify $|u\rangle$ with the same class of $|x^{(i)}\rangle$.

Complexity analysis The computational cost of the bQKNN is of order $O(M\,\ell + \ell\,\log^2\ell + \sqrt{M}\,\log\ell\,\log M)$, where $O(M\,\ell)$ is the cost of initial state preparation, the term $O(\ell\,\log^2\ell)$ accounts for the Hamming distance computation performed in parallel over the M instances where $O(\log^2\ell)$ is the depth of each $InCr$ circuit, and the last term $O(\sqrt{M}\,\log\ell\,\log M)$ is for the computation of the instance with the minimum Hamming distance from the test vector.

We observe that the cost of the state preparation described in Section 5.2.2 is linear in the number of training instances and in the number of qubits. Again, the most expensive step is data preparation, and the algorithm will become competitive with its classical counterpart only when more efficient procedures for the quantum encoding of classical data will be available.

6 Experiments

In this section, we empirically analyze the impact of the various alternatives of different distance functions, quantum circuits, data encoding, and preprocessing on the implementations of QKNN described in the previous sections.

We implement the QKNNs and run the experiments using the Qiskit framework. To ensure the replicability of the experiments, the code has been made publicly available on GitHub.[3]

[3] Code available at https://github.com/Brotherhood94/exploring_different_encoding_and_distance_metrics_for_a_quantum_instance_based_classifier

Table 1 Dataset description

name	Training	Test	Features	Labels
`iris`	105	45	4	3
`cancer`	398	171	30	2
`mnist`	246	106	8×8	2

We run experiments on three open-source datasets[4]:

- `iris`: a dataset where each class refers to a type of iris plant
- `cancer`: a dataset that describes characteristics of the cell nuclei for recognizing breast cancer
- `mnist`: a dataset of handwritten digit grayscale images

We use 70% of the datasets for training and the remaining 30% for evaluation. Details for the datasets are reported in Table 1. We provide in Table 2 the state preparation costs of the techniques discussed in Sects. 5.1.2 and 5.2.2 in terms of width and depth for each dataset we study, assuming we do not partition the datasets. It is evident that this setup would require a significant amount of computational resources, as simulating a quantum algorithm on a classical device demands resources that increase exponentially with the number of qubits. Therefore, we address this matter by training QKNN algorithms on distinct partitions that cover the whole training set.

We set as $N' \leq N$ the number of features, and $M' \leq M$ the number of records used by aQKNN and as $\ell = N's$ the number of bits used by bQKNN. When possible, we perform the experiments with $N' = N$, i.e., we map all the features describing the instances in the dataset to the quantum circuits. However, since this is typically not feasible, we apply a principal component analysis (PCA) (Tan et al. 2005) transformation to turn a dataset with N features into a dataset with $N' < N$ features, where each one of the N' new features is a principal component.

For each instance u we want to classify, we train the models with $M' = L \cdot i$, for $i \in [1, \ldots, 32]$, where L is the number of class labels. For the selection of the M' training instances

[4] `iris`: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_iris.html
`cancer`: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_breast_cancer.html
`mnist`: https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_digits.html. For `mnist`, we focus on the task of classification between a couple of similar digits, i.e., "0 and 8," therefore considering only two classes.

Table 2 State preparation costs in terms of width and depth for each dataset under investigation with respect to the complexities of the amplitude encoding and the basis encoding techniques employed

name	M	N	Amplitude encoding		Basis encoding	
			Width	Depth	Width	Depth
iris	150	4	11	600	$2+\ell$	$150\,\ell$
cancer	569	30	16	17,070	$2+\ell$	$569\,\ell$
mnist	352	8×8	14	5632	$2+\ell$	$352\,\ell$

We observe that in the basis encoding, width and depth vary according to a chosen precision ℓ used to encode the features

out of the M available, we employ two different strategies named *sampling* and *prototypes*, respectively.

Inspired by Schuld et al. (2014); Afham et al. (2020), for the *sampling* strategy, we sample uniformly at random M' training instances. Since the selection of the M' instances is random in this setting, for each test record, we repeat the classification at least 50 times using different samples of M' training instances to have meaningful results. Thus, we report the results by averaging the results observed among the various experiments.

On the other hand, the *prototype* strategy works as follows. When $i = 1$ and $M' = L$, like in Wiebe et al. (2018), we use a single prototypical instance for each class, and we run QKNN using only $M' = L$ records in the training set. In particular, we use as training instances the L centroids obtained from the instances belonging to each class of the training set. Recall that a centroid is computed, taking the average value for each feature. In practice, this approach implements a *quantum nearest centroid classifier*. Instead, when $i > 1$, we use i prototypes for each class as training instances. We derive these prototypes as the centroids returned by centroid-based clustering algorithms (Tan et al. 2005) applied separately to each subset of instances belonging to the same class. In particular, we exploit the K-Means algorithm (Tan et al. 2005) for preparing the prototypes for the amplitude encoding and the K-Modes (Tan et al. 2005) algorithm for the binary encoding. We run K-Means/Mode on the different subsets, searching for a number of clusters equal to i.

We highlight that to have a fair comparison and to better understand the effect of the different types of training relying on sampled and prototypical instances, we adopt the same strategy for the experiments with the classic KNN. Moreover, we consider classification *accuracy* (Tan et al. 2005) as an evaluation metric, while we do not report precision, recall, or f1-measure. In fact, being the datasets balanced with respect to the classes, we have not observed significant discrepancies in these measures from the accuracy in our experiments.

An important issue concerns the problem of the demanding computational resources required for the simulation of quantum algorithms. This issue mainly affects the version of QKNN with basis encoding, whose experimental setting has been defined as follows. Given ℓ qubits, we reserve s bits for each feature, and we reduce the number of features from N to a value N' such that $\ell \geq N's$. Thus, before running bQKNN, we have to compress the information captured by the N' features into ℓ bits. We perform this task employing two distinct strategies that retain information pertaining to the similarity of instances through different methodologies.

The first one consists of the discretization of the numerical attributes through the *Recursive Minimal Entropy Partitioning* (RMEP) method (Dougherty et al. 1995). RMEP recursively divides the numerical values represented in a real interval into partitions that minimize the entropy of the target class until a fixed threshold is reached. The partitions obtained define regions represented by a single representative value. The effect of the discretization on the classification accuracy is negligible (less than 1%), as experimentally shown in Fan and Li (2020). Since every feature is discretized into bins encoded by binary sequences of length s, we need at least N strings, each of s bits. Thus, if the number ℓ of available qubits is lower than $N\,s$, we first run PCA on $\mathcal{D}$ to reduce the number of features to be discretized to $N' = \lfloor \ell/s \rfloor$.

The second strategy consists of using a hash function that preserves information about their similarity. In particular, we adopt *Locality-Sensitive Hashing* (LSH) (Leskovec et al. 2014). LSH hashes similar input items into the same bin with high probability. The number of bins is much smaller than the universe of possible input items.

The problem with RMEP and LSH is that more instances might be associated with the same binary string, and QAI algorithms using basis encoding cannot deal with multiple instances having the same binary representation. Indeed, in the quantum realm, these collisions would introduce interference over instances already loaded with the same binary representation. To deal with this issue, it is possible to associate an index to each instance to discern different instances with the same representation. However, this patch does not solve the problem: in the prediction process, it would be infeasible to identify the correct class of training instances with the same representation that belongs to different classes. We overcome this limitation by removing duplicate instances from the training set. In cases where there are multiple instances with the same encoding and different labels, we adopt a majority vote to determine the class label for that particular training instance.

6.1 Results

We report the performance in terms of *accuracy* (Tan et al. 2005) for QKNN in the amplitude (aQKNN) and basis (bQKNN) versions, compared with the classic KNN for the Aer QASM quantum simulator of Qiskit with 8192 shots. For bQKNN, we consider the two different data preparation approaches presented: RMEP and LSH identified with *-e* and *-h*, respectively. The plots labeled with *sampling* in the title report the results using the sampling strategy, randomly selecting the instances from the training set. On the other hand, the plots labeled with *prototype* in the title report the results obtained with the prototype strategy using prototypical instances calculated on the training set as described in the previous section. We highlight that, for each version using sampling, results are averaged over 50 runs for statistical significance. We evaluate all the methods with the same experimental setting, varying the number of features N' and the number of training instances M'.

6.1.1 Iris

In Fig. 7, we observe the performance for the `iris` dataset. When considering fewer instances, we note that the performance is significantly lower with respect to the classic KNN using the whole training set, as illustrated with the red continuous line. However, in the prototype setting with PCA and $N' = 2$, KNN (represented as a green line with the markers as circles) achieves accuracy comparable with the accuracy on the whole training set for $M' \leq 6$. Also, we observe that for `iris`, the RMEP encoding has always better performance than the LSH one. In each plot, aQKNN is half a point under classical KNN on average, but it still shows promising behavior. Indeed, for $N' = N = 4$, i.e., when we are exploiting all the available features, and thus no PCA is applied, aQKNN is even better than KNN with $M' = 24$ in the prototype setting. This might indicate that the quantum circuit and the necessary data manipulation do not necessarily affect the performance too much with respect to a classic setting. Also, when $N' = 2$ using sampling, we notice that bQKNN-e is comparable to KNN. This is probably due to the binary encoding that collapses different instances into the same representation. On the other hand, when $N' = 2$ using prototypes, bQKNN methods have markedly lower performance than KNN and aQKNN. Finally, we notice an improvement in the performance of bQKNN-h when $M' > 12$. Overall, the performance of QKNN and KNN can be comparable.

On the `iris` dataset, we run experiments on the real quantum machine *ibmq_16_melbourne* with 15 qubits, the one with the highest number of qubits among those at our disposal. Unfortunately, we could only experiment with N'=2, M'=2 in the prototype setting. The performance is not extraordinary, as we reached an accuracy of less than 0.5 versus a value of about 0.6 obtained in the simulator. This is

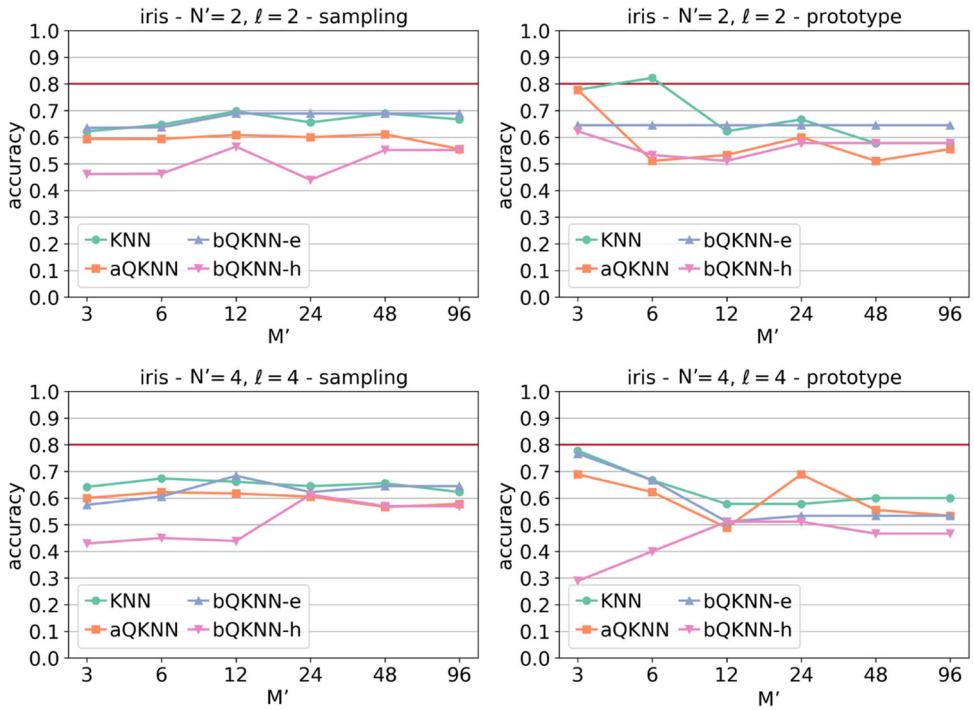

Fig. 7 Performance comparison in terms of accuracy on the `iris` dataset

probably due to the noise that can interfere during the computation.

6.1.2 Breast cancer

In Fig. 8, we show the accuracy for the `cancer` dataset. The first aspect to notice is that the performance of the various approaches is more in line with those of the classic KNN trained on the whole dataset (red continuous line). This is probably due to the fact that we are facing a binary classification problem for the `cancer` dataset. What emerges here is that QKNN approaches are comparable and even better than the classic KNN trained with comparable settings. Moreover, it is clear from the plots using the sampling strategy that when the number of training instances M' increases, the accuracy of both aQKNN and bQKNN increases.

The computational resources allowed us to run experiments for bQKNN with $N' \leq 4$. The results for aQKNN with $N' = 8$ and $N' = 16$ are reported at the bottom of Fig. 8. Unlike the previous results, in this case, KNN is slightly better than aQKNN, but we have an overall drop in accuracy. Therefore, the dimensionality reduction of PCA seems decisive in boosting performance for both classic and quantum cases when fewer training instances are available. Finally,

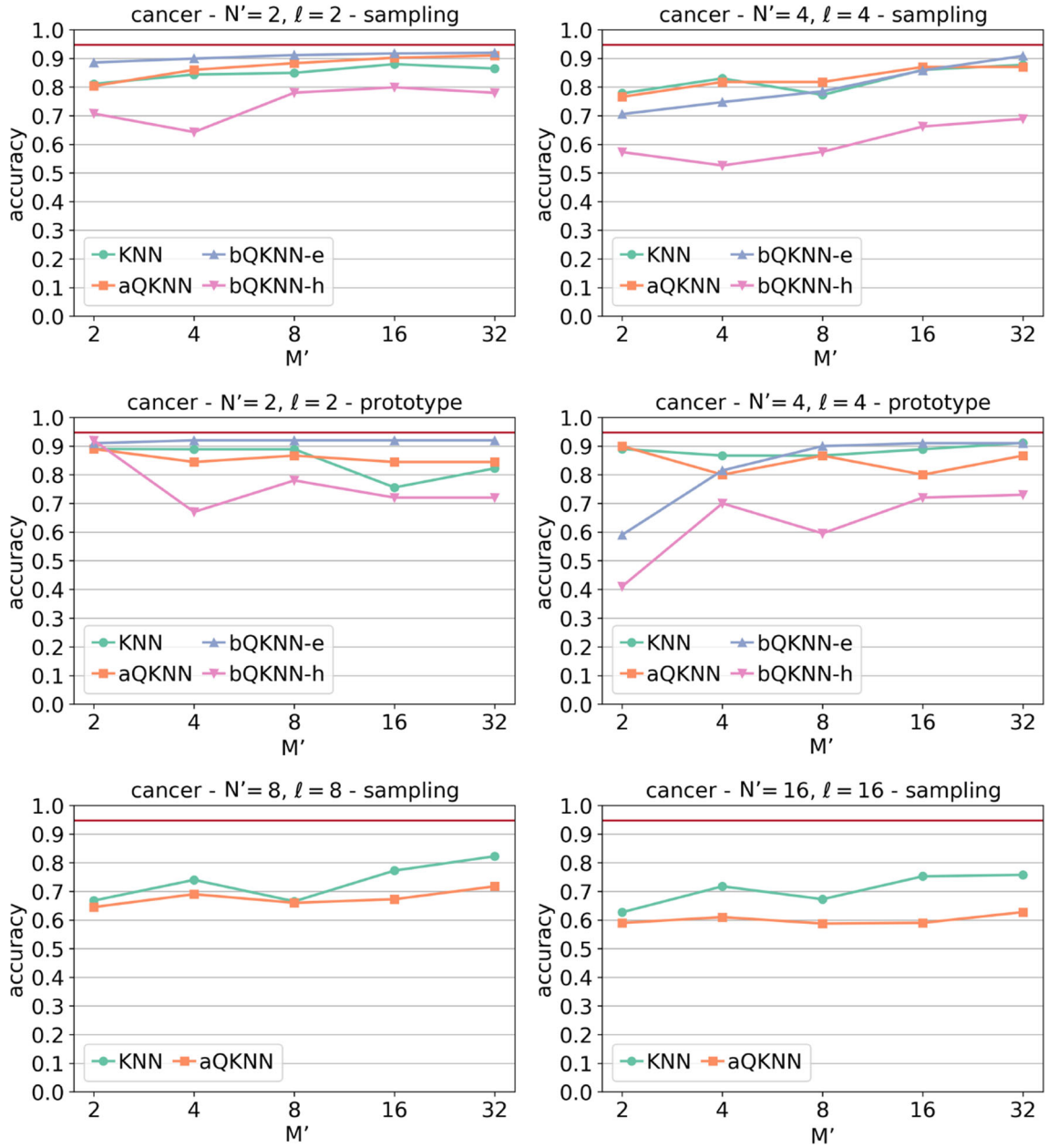

Fig. 8 Performance comparison in terms of accuracy on the `cancer` dataset

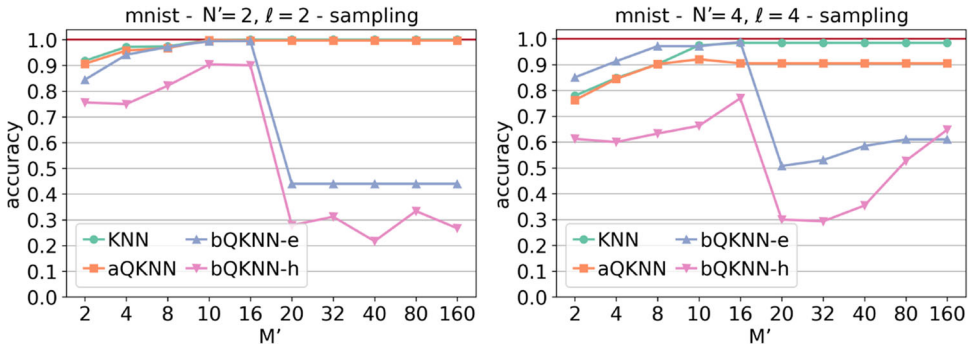

Fig. 9 Performance comparison in terms of accuracy on the `mnist` dataset

as for `iris`, the performance of bQKNN-e is remarkably better than that of bQKNN-h.

6.1.3 Mnist

In Fig. 9, we show the accuracy for the `mnist` dataset. We report results only for the sampling strategy since the prototype, as with the datasets shown above, leads to worse accuracy scores. The first aspect that we notice is that when $M' > 16$, there is a consistent drop in the performance of bQKNN. As before, the LSH version of bQKNN performs worse than the RMEP one. When $M' < 16$, the performance of the various algorithms is comparable, and both aQKNN and bQKNN reach the same level of KNN when $M' = 16$ and $N' = 2$. Thus, the dimensionality reduction adopted, i.e., the PCA, probably helps identify discriminating attributes.

Finally, we observe that results for $N' = 2$ are slightly better than those for $N' = 4$ for aQKNN, suggesting how more information might introduce distortions into the amplitudes of the data used by aQKNN.

7 Conclusion

We presented in detail and discussed similarities and differences between various alternatives for QKNN algorithms, and we experimented with them on different datasets. Results are promising as they show that the QKNN can theoretically overcome current state-of-the-art AI approaches by exploiting quantum parallelism. Also, from an empirical perspective, QKNN has scores comparable to classical KNN. In particular, aQKNN shows more stable accuracy scores with respect to bQKNN-based solutions. This evidence, observed in the context of Recursive Minimal Entropy Partitioning (RMEP) and Locality-Sensitive Hashing (LSH) preprocessing methodologies, may be attributed to the constrained number of qubits available for the representation of a given instance within the basis representation.

Due to the limited computational resources, we had to employ PCA on the features of the three datasets, and we noticed that in general, PCA seems to have a positive impact on classification tasks.

It is important to remark that QKNN has a lower complexity than KNN in computing the distances, but requires a higher cost at data preparation time. It is evident from this work that an efficient quantum state preparation technique, or a device capable of storing quantum data, is mandatory to achieve a speed-up with respect to the classical counterpart. As a first step in this direction, it would be interesting to exploit the forking technique presented in Berti (2023). This technique permits the leverage of the same quantum dataset for multiple classification tasks within the same quantum circuit instead of just one task for the circuit, as presented in the current work. Additionally, we can use the KP-Tree data structure introduced in Kerenidis and Prakash (2016). It operates under the assumption of enabling queries in superposition to a classical data structure and provides a polylogarithmic state preparation cost in the number of records. Eventually, after attempting to run the experiments on the *ibmq_16_melbourne* device without obtaining notable results, we plan to conduct a more in-depth study of the performance of aQKNN and bQKNN on the latest devices available through the Qiskit framework as future work.

Author Contribution All authors contributed equally.

Funding Open access funding provided by Università di Pisa within the CRUI-CARE Agreement. This study was carried out within the National Centre on HPC, Big Data and Quantum Computing - SPOKE 10 (Quantum Computing) and received funding from the European Union Next-GenerationEU - National Recovery and Resilience Plan (NRRP) - MISSION 4 COMPONENT 2, INVESTMENT N. 1.4 - CUP N. I53C22000690001. This manuscript reflects only the authors' views and opinions; neither the European Union nor the European Commission can be considered responsible for them. Partial support was also provided by INdAM-GNCS and by the Italian Project Fondo Italiano per la Scienza FIS00001966 MIMOSA.

Data Availability Data is provided within the manuscript.

Declarations

Conflict of Interest The authors declare no competing interests.

References

Afham A, Basheer A, Goyal SK et al (2020) Quantum k-nearest neighbor machine learning algorithm. arXiv preprint arXiv:2003.09187

Araujo IF, Park KD, Petruccione F, Silva AJ (2021) A divide-and-conquer algorithm for quantum state preparation. Sci Rep 11:6329

Berti A (2023) Logarithmic quantum forking. In: Proceedings of ESANN, pp 251–256. https://doi.org/10.14428/esann/2023.ES2023-93

Berti A, Bernasconi A, Del Corso GM, Guidotti R (2022) Effect of different encodings and distance functions on quantum instance-based classifiers. In: Gama J, Li T, Yu Y, Chen E, Zheng Y, Teng F (eds) Advances in knowledge discovery and data mining - 26th Pacific-Asia conference, PAKDD 2022, Chengdu, China, May 16-19, 2022, Proceedings, Part II. Lecture Notes in Computer Science, vol 13281, pp 96–108. Springer, s.l

Brassard G, Høyer P, Mosca M, Tapp A (2002) Quantum amplitude amplification and estimation. Quantum Comput Inf 53–74

Dang Y, Jiang N, Hu H, Ji Z, Zhang W (2018) Image classification based on quantum k-nearest-neighbor algorithm. Quantum Inf Proc 17(9):239

Dougherty J, Kohavi R, Sahami M (1995) Supervised and unsupervised discretization of continuous features. In: ICML, pp 194–202

Durr C, Hoyer P (1999) A quantum algorithm for finding the minimum. arXiv preprint arXiv:quant-ph/9607014v2

Fan C, Li P (2020) Classification acceleration via merging decision trees. FODS '20: ACM-IMS Foundations of Data Science Conference, Virtual Event, USA, October 19–20, 2020. ACM, s.l, pp 13–22

Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller P-A (2019) Deep learning for time series classification: a review. DAMI 33(4):917–963

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

Kaye P (2004) Reversible addition circuit using one ancillary bit with application to quantum computing. arXiv, 0408173

Kerenidis I, Prakash A (2016) Quantum recommendation systems. arXiv preprint arXiv:1603.08675

Kerenidis I, Prakash A (2017) Quantum recommendation systems. In: Papadimitriou CH (ed.) 8th Innovations in theoretical computer science conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA. LIPIcs, vol 67, pp 49–14921. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, s.l

Leskovec J, Rajaraman A, Ullman JD (2014) Mining of massive datasets, 2nd edn. Cambridge University Press, s.l

Li J, Lin S, Kai Y, Guo G (2021) Quantum k-nearest neighbor classification algorithm based on Hamming distance. arXiv preprint arXiv:2103.04253

Lloyd S, Mohseni M, Rebentrost P (2013) Quantum algorithms for supervised and unsupervised machine learning. arXiv preprint arXiv:1307.0411

Long GL, Sun Y (2001) Efficient scheme for initializing a quantum register with an arbitrary superposed state. Phys Rev A 64:014303

Manning CD, Raghavan P, Schütze H (2008) Introduction to information retrieval. Cambridge University Press, s.l

Möttönen M, Vartiainen JJ, Bergholm V, Salomaa MM (2005) Transformation of quantum states using uniformly controlled rotations. Quantum Inf. Com. 5:467–473

Nielsen MA, Chuang IL (2016) Quantum computation and quantum information (10th. Cambridge University Press, s.l, Anniversary

Nigsch F, Bender A, Buuren B, Tissen J, Nigsch E, Mitchell JBO (2006) Melting point prediction employing k-nearest neighbor algorithms and genetic parameter optimization. J Chem Inf Model 46(6):2412–2422

Park DK, Petruccione F, Rhee J-KK (2019) Circuit-based quantum random access memory for classical data. Sci Rep 9(1):1–8

Phalak K, Chatterjee A, Ghosh S (2023) Quantum random access memory for dummies

Ramaswamy S, Rastogi R, Shim K (2000) Efficient algorithms for mining outliers from large data sets. SIGMOD Conference. ACM, s.l, pp 427–438

Ruan Y, Xue X, Liu H, Tan J, Li X (2017) Quantum algorithm for k-nearest neighbors classification based on the metric of Hamming distance. Int J Theor Phys 56(11):3496–3507

Schuld M, Petruccione F (2018) Supervised learning with quantum computers. Springer, s.l

Schuld M, Fingerhuth M, Petruccione F (2017) Implementing a distance-based classifier with a quantum interference circuit. EPL (Europhysics Let.) 119(6):60002

Schuld M, Sinayskiy I, Petruccione F (2014) Quantum computing for pattern classification. In: PRICAI. LNCS, vol 8862, pp 208–220. Springer, s.l

Shende VV, Markov IL, Bullock SS (2004) Minimal universal two-qubit controlled not-based circuits. Phys Rev A 69

Shende VV, Bullock SS, Markov IL (2006) Synthesis of quantum-logic circuits. IEEE Trans Com Aided Des Int Cir Syst 25(6):1000–1010

Soklakov AN, Schack R (2006) Efficient state preparation for a register of quantum bits. Phys Rev A 73:012307

Tan P, Steinbach MS, Kumar V (2005) Introduction to data mining. Addison-Wesley, s.l

Ventura D, Martinez TR (2000) Quantum associative memory. Inf Sci 124(1–4):273–296

Wiebe N, Kapoor A, Svore KM (2018) Quantum nearest-neighbor algorithms for machine learning. Quantum Inf Comput 15