Quantum Machine Intelligence (2024) 6:23
https://doi.org/10.1007/s42484-024-00155-2

RESEARCH ARTICLE

A quantum k-nearest neighbors algorithm based on the Euclidean distance estimation

Enrico Zardini[1] · **Enrico Blanzieri**[1,2] · **Davide Pastorello**[2,3]

Received: 9 May 2023 / Accepted: 28 February 2024

Abstract
The k-nearest neighbors (k-NN) is a basic machine learning (ML) algorithm, and several quantum versions of it, employing different distance metrics, have been presented in the last few years. Although the Euclidean distance is one of the most widely used distance metrics in ML, it has not received much consideration in the development of these quantum variants. In this article, a novel quantum k-NN algorithm based on the Euclidean distance is introduced. Specifically, the algorithm is characterized by a quantum encoding requiring a low number of qubits and a simple quantum circuit not involving oracles, aspects that favor its realization. In addition to the mathematical formulation and some complexity observations, a detailed empirical evaluation with simulations is presented. In particular, the results have shown the correctness of the formulation, a drop in the performance of the algorithm when the number of measurements is limited, the competitiveness with respect to some classical baseline methods in the ideal case, and the possibility of improving the performance by increasing the number of measurements.

Keywords Quantum computing · Quantum machine learning · k-Nearest neighbors · Euclidean distance

1 Introduction

Quantum machine learning (QML) is one of the most recent and most popular directions of scientific investigation in the area of quantum computing. In particular, the application of quantum computation to machine learning (ML) tasks offers some interesting solutions characterized by a quantum advantage with respect to the classical counterparts, at least on a theoretical level. Furthermore, QML seems to be a good way to exploit existing prototypes of quantum computers for tackling real-world problems. In this sense, a general "practical" approach consists in executing quantum algorithms as subroutines of more complex learning schemes, in which a quantum machine is used as a co-processor within a hybrid architecture. This approach is an interesting alternative to the development of quantum algorithms that fully accomplish ML tasks under the (strong) assumptions of ideality and universality of the quantum hardware.

In the last decade, several interesting QML algorithms have been proposed and characterized from a theoretical viewpoint; sometimes, they have been also empirically tested. Remarkable examples are the quantum SVM proposed by Rebentrost et al. (2014), the distance-based classifiers like the one defined by Schuld et al. (2017), and the quantum neural networks, whose performance have been discussed by Abbas et al. (2021). In particular, several quantum versions of the k-nearest neighbors (k-NN) algorithm have been proposed (see Section 2). In ML, the k-NN is a very simple and widely used classification algorithm that assigns a label to an unclassified data instance according to the labels of the k nearest training instances. To do so, a suitable reference distance in the space in which the data are represented must be selected. In the classical realm, typical

✉ Enrico Zardini
enrico.zardini@unitn.it

Enrico Blanzieri
enrico.blanzieri@unitn.it

Davide Pastorello
davide.pastorello3@unibo.it

[1] Department of Information Engineering and Computer Science, University of Trento, via Sommarive 9, Povo 38123, Trento, Italy

[2] Trento Institute for Fundamental Physics and Applications, via Sommarive 14, Povo 38123, Trento, Italy

[3] Department of Mathematics, Alma Mater Studiorum - Universitá di Bologna, Piazza di Porta San Donato 5, Bologna 40126, Italy

Published online: 23 April 2024

choices are the Hamming distance and the Euclidean distance; instead, in the quantum realm (considering only the quantum k-NNs), the Euclidean distance has not received much consideration. In this work, we propose a quantum k-nearest neighbors algorithm in which the calculation of the Euclidean distances is based on a novel quantum encoding with low qubit requirements and a simple quantum circuit, making the implementation particularly advantageous. As with other algorithms involving the quantum computation of the Euclidean distance (e.g., by means of the SWAP test), an exponential speedup over a classical calculation can be obtained assuming the availability of a quantum random access memory (QRAM, Giovannetti et al. 2008) for data retrieval. Otherwise, there is not a true quantum advantage in terms of time complexity. From a practical viewpoint, in this article, we analyze the performance of the proposed quantum k-NN in terms of classification accuracy and correctness of the nearest neighbors found (evaluated through the Jaccard index). The algorithm has been implemented using Qiskit and run with three different execution modalities: *classical*, *statevector*, and *simulation*. Instead, the empirical evaluation on a real quantum machine was prevented by the number of qubits required by the considered experiments.

The article is structured as follows: Section 2 provides some background information; Section 3 presents the new quantum k-nearest neighbors algorithm based on the Euclidean distance metric; Section 4 deals with the implementation of the algorithm; Section 5 describes the experimental evaluation and the results obtained; Section 6 concludes the article.

2 Background

This section presents background information about quantum machine learning, the quantum k-nearest neighbors algorithms available in the literature, and the usages of the (squared) Euclidean distance in the field of quantum machine learning.

2.1 Quantum machine learning

In general, ML is the automation of methodologies for extracting information from collected data. If the data analysis techniques are implemented on conventional digital computers, we refer to classical ML; if quantum machines are employed, we refer to quantum ML. A general reason justifying the efforts in developing new QML schemes is suggested by Biamonte et al. (2017): since (even small) quantum systems are difficult to simulate with classical computers, we can conjecture that (even small) quantum processors can find structures in data that are difficult to discover classically. Therefore, QML could be the right path towards non-trivial applications of the small-scaled quantum machines available today and in the near future. On the other hand, under strong assumptions of universality, large scale, and fault tolerance, it is possible to formulate several QML algorithms that outperform their classical counterparts. This is very important for the comprehension of the foundations of quantum computing and for showing the actual potential of quantum computers. However, to promote the advent of quantum technologies in the near term, taking into account the limitations of the current quantum hardware is useful while seeking new QML schemes.

From the mathematical viewpoint, there is another relevant motivation for developing ML algorithms to be executed by quantum machines, given by a formal analogy between quantum mechanics and ML: both fields rely on matrix operations in high-dimensional vector spaces. In practice, the Hilbert spaces, in which physical quantum systems are described, can be used as feature spaces for data representations. In this framework, linear algebraic operations are physically realized by the time evolution of quantum states; for instance, in the circuit model of quantum computation, the evolution is described as the action of quantum gates, i.e., unitary operators. In addition, representing data into quantum states is advantageous also in terms of space resources, since the dimension of the Hilbert space of a multi-qubit system is exponential in the number of qubits. Then, the controlled dynamics of a small number of qubits towards a target state may correspond to the application of a complex linear algebraic operation on the considered feature space.

A crucial notion in QML is *quantum encoding*, which is any procedure that encodes classical data (e.g., a list of symbols) into quantum states. In particular, loading efficiently large amounts of data into quantum architectures is a serious bottleneck at the current status of QML; indeed, the state preparation required for running several well-known QML algorithms can be done efficiently only under the strong assumption of the availability of a QRAM. More in detail, given an n-qubit register, let $\{|i\rangle\}_{i=0,\ldots,2^n-1}$ be a fixed orthonormal basis of the corresponding Hilbert space that we call *computational basis*. The simplest quantum encoding is the *basis encoding*, in which the bit strings of length n are encoded into the states that form the computational basis. Therefore, n qubits are used to encode n bits of classical information with interesting quantum opportunities, like creating superpositions of data and enabling non-classical correlations via entanglement. Instead, a more efficient quantum encoding in terms of space resources is the *amplitude encoding*, in which a data instance represented by a normalized complex vector $\mathbf{x} \in \mathbb{C}^{2^n}$ is encoded into the coordinates

(or amplitudes) of a quantum state with respect to the computational basis, namely,

$$|\psi\rangle = \sum_{i=0}^{2^n-1} x_i |i\rangle \qquad (n\text{-qubit state}).$$

The amplitude encoding exploits the exponential storing capacity of a quantum memory, but it does not allow the direct retrieval of the stored data. Indeed, the amplitudes cannot be observed, and only the probabilities $|x_i|^2$ can be estimated. The encoding procedure used in this work is based on amplitude encoding.

Let us conclude this introductory section by arguing that QML is probably the most promising way to find out effective applications of the existing small-scale quantum computers. In particular, one can also drop the requirement that an ML task must be entirely accomplished by quantum computations in favor of hybrid approaches, in which quantum co-processors efficiently solve specific subproblems within more complex learning schemes. Moreover, the quantum speedup is not the unique quantum advantage that can be pursued. Accuracy in prediction, expressive power, generalization capability, and the ability to avoid plateaus in training are also noteworthy figures of merit in evaluating the learning performance of quantum machines.

2.2 Quantum k-NN

The k-nearest neighbors (Fix and Hodges 1951) is a classification algorithm that consists of three steps: the computation of the distance with respect to the training elements; the identification of the k nearest neighbors, i.e., the k elements closest to the test instance; the prediction of the class label through a majority voting. Several quantum variants with different distance measures have been proposed, but a common aspect to all of them is the exploitation of a superposition state in order to perform parallel operations, such as computing the distance from the training elements simultaneously (quantum parallelism).

First of all, quantum k-NN algorithms employing the Hamming distance, thus requiring binary features, have been proposed by Schuld et al. (2014), Wiśniewska and Sawerwain (2018), Ruan et al. (2017), Zhou et al. (2021), and Li et al. (2021). Specifically, the first two works compute the Hamming distances by encoding the sums of the qubits differences (differences obtained through controlled-NOT gates) into the amplitudes by means of a unitary operation (an idea proposed first by Trugenberger 2002). Then, the classification is performed directly by measuring without explicitly selecting the nearest neighbors. Instead, the other works exploit the incrementation circuit presented by Kaye (2004) in order to obtain the distance values in basis encoding. After that, Ruan et al. (2017) select the data with a distance lower than a given threshold by means of an OR gate and a projection operation to directly perform the classification, Zhou et al. (2021) exploit Dürr's minimization algorithm (Dürr and Høyer 1999) to find the k minimum distance values, while Li et al. (2021) apply a novel quantum search procedure inspired by a binary search in order to identify the minimum.

Concerning non-binary features, distance measures related to the angle between vectors, such as the cosine distance, are widely used. For instance, Dang et al. (2018) and Wang et al. (2019) have applied a quantum k-NN variant of this type to image classification tasks. In particular, the SWAP test (Buhrman et al. 2001) without measurements is used to compute the distances, whose values are then transferred to the qubits states through the amplitude estimation algorithm (Brassard et al. 2002). Finally, the nearest neighbors are found by means of Dürr's algorithm. This workflow has been presented first by Wiebe et al. (2015), although for finding only the nearest neighbor. Instead, Afham et al. (2020) and Ma et al. (2021) have proposed a conceptually simpler variant, which consists in iterating SWAP tests and measurements in order to estimate a quantity proportional to the squared cosine similarity with respect to the training instances. In addition, the model allows processing multiple test instances in parallel, as shown by Ma et al. (2021). Actually, Afham et al. have recently proposed another variant (Basheer et al. 2021) whose workflow, however, is not so different from that of the previously described works. Indeed, it involves the SWAP test, a quantum analog-to-digital conversion algorithm (Mitarai et al. 2019), and a variation of Dürr's algorithm.

Other interesting distance measures for non-binary features are the Euclidean, the Mahalanobis, and the polar distances. Specifically, the Euclidean distance is dealt with in-depth in Section 2.3. Regarding the other ones, Gao et al. (2022) have proposed a variant based on the Mahalanobis distance, while Feng et al. (2023) have presented a quantum k-NN based on the polar distance, which combines angle and module length information through an adjustable parameter. In detail, the Mahalanobis distance is computed by exploiting the phase estimation algorithm (Cleve et al. 1998), combined with Hamiltonian simulation (Rebentrost et al. 2018), and a controlled rotation; instead, the polar distance is calculated through the SWAP test without measurements and a pair of Toffoli gates (one of which extended). After that, in both works, the distances are encoded in the qubits states by applying the amplitude estimation algorithm (or its coherent version, proposed by Wiebe et al. 2015) and the nearest neighbors are retrieved through Dürr's algorithm (or an algorithm based on it, proposed by Miyamoto et al. 2019). To conclude, it is also worth mentioning the quantum k-NN, based on a quantum sorting subroutine, that has been proposed by Quezada et al. (2022). It requires a metric operator

computing distances and encoding them in qubits states, an oracle that identifies sorted sequences, and Grover's algorithm (Grover 1996); as in other works, the classification is performed directly without identifying the k nearest neighbors.

2.3 Quantum Euclidean distance

The Euclidean distance is a well-known distance metric in ML. Here, the definition of its squared version is provided, since it will be useful in the following sections. In particular, given two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$, the squared Euclidean distance between them, $d^2(\mathbf{u}, \mathbf{v})$, is defined as

$$d^2(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|^2 = \|\mathbf{u}\|^2 - 2\langle \mathbf{u}, \mathbf{v}\rangle + \|\mathbf{v}\|^2, \tag{1}$$

where $\langle \mathbf{u}, \mathbf{v}\rangle$ is the scalar product between $\mathbf{u}$ and $\mathbf{v}$.

The distance metric in question has been employed also in the field of QML. For instance, Lloyd et al. (2013) have proposed a quantum procedure to estimate the squared Euclidean distance between a data point and the centroid of a cluster, i.e., the mean of the elements contained in a group of data. Specifically, the algorithm relies on the SWAP test, which is applied to the index registers, and does not require the input vectors to have unit norms. An analogous procedure has been used by Sarma et al. (2020) to provide a hybrid k-means clustering algorithm (in which the centroids are classically computed), and by Getachew (2020) for a hybrid version of the k-medians one. Instead, Yu et al. (2020) have proposed three quantum algorithms to estimate three similarity measurements, based on the squared Euclidean distance, for datasets. In particular, all procedures do not require unit-norm input vectors, exploit the quantum interference (given by the change of basis), and use the amplitude estimation algorithm to determine the similarity measures. Finally, it is worth mentioning the quantum binary classifier devised by Schuld et al. (2017). In detail, the classifier circuit consists of a Hadamard gate (necessary for the quantum interference), a conditional measurement, and a final measurement. By iterating the procedure just described, a probability value related to the squared Euclidean distances is estimated for each class. In this last work, input vectors with unit norms are considered.

Regarding the quantum k-NN model, as far as the authors know, the only variant based on the Euclidean distance available in the literature has been presented by Fastovets et al. (2019); it exploits the procedure proposed by Lloyd et al. (2013) to estimate the pairwise distance values, and Dürr's minimization algorithm to find the k nearest neighbors. The main drawback lies in the need of multiple iterations for each of these steps, since both involve a final measurement. In addition, Dürr's algorithm requires an oracle, i.e., a black-box function, to be used. Actually, the nearest neighbor algorithm proposed by Wiebe et al. (2015) admits also the Euclidean distance as the distance metric. However, the workflow, which has been described in Section 2.2, is quite complex to be implemented. Finally, the computation of the single linkage, namely, one of the set similarity measures considered by Yu et al. (2020), could be seen as a generalization of the nearest neighbor search. Nevertheless, their quantum algorithm uses the reciprocals of the input vectors; as a consequence, theoretically, the original distance relationships are not preserved.

3 Method

In this section, the new quantum k-NN algorithm based on the Euclidean distance metric is presented. In addition, a brief discussion of the algorithm's complexity compared with that of the classical counterpart is provided.

3.1 Algorithm

In the quantum k-NN algorithm introduced in this work, a quantity related to the squared Euclidean distance is computed in parallel for all training instances by means of a novel encoding and a simple quantum circuit, which performs a SWAP-test-like procedure without controlled-SWAP gates. In practice, the algorithm exploits the quantum interference and encodes these distance-related values, which are then estimated through measurements, in the amplitudes of the quantum states. It is worth highlighting that the input vectors do not undergo a unit-norm normalization, which would result in a significant information loss. In addition, the number of qubits needed is low, and no oracle is involved, making the implementation feasible. A more detailed and formal description of the steps of this new algorithm is provided below.

3.1.1 Data preprocessing

Let us consider a training set $\mathcal{U} = \{\mathbf{u}_0, ..., \mathbf{u}_{N-1}\}$ of real-valued data instances $\mathbf{u}_j \in \mathbb{R}^d$, and let $\mathcal{L} = \{l_0, ..., l_{N-1}\}$ be the set of corresponding labels. In addition, let us consider a test instance $\mathbf{u}' \in \mathbb{R}^d$, whose label is unknown.

The preprocessing step of the algorithm consists in centering and normalizing the data features into the range $\left[-\frac{1}{2\sqrt{d}}, \frac{1}{2\sqrt{d}}\right]$ (procedure detailed in Section 4). In this way, the maximum norm of the resulting vectors turns out to be $\frac{1}{2}$ and the maximum (squared) Euclidean distance turns out to be 1.

3.1.2 Initial state and encoding(s)

Let $\mathcal{V} = \{\mathbf{v}_0, ..., \mathbf{v}_{N-1}\}$ and $\mathbf{v}'$ be the training set and the test instance after the preprocessing step described above. The quantum circuit is then initialized in the state

$$|\psi\rangle = |0\rangle \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle|\alpha\rangle + |1\rangle|\beta\rangle)\right), \tag{2}$$

where

$$|\alpha\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle \sum_{i=0}^{F-1} x_{ji}|i\rangle,$$

$$|\beta\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} |j\rangle \sum_{i=0}^{F-1} x'_{ji}|i\rangle.$$

Here, F is a positive integer value depending on the encoding used, while $\mathbf{x}_j = \{x_{ji}\}_{i=0,...,F-1}$ and $\mathbf{x}'_j = \{x'_{ji}\}_{i=0,...,F-1}$ represent the quantum encoded versions of the preprocessed training and test data, respectively. Therefore, the number of qubits required is $2 + \lceil \log_2 N \rceil + \lceil \log_2 F \rceil$. In particular, two encodings, whose advantages are discussed in the next sections, have been devised and tested in this work: *extension* and *translation*. Let us look at their definitions. As regards the *extension* encoding, $F = 2d + 3$ and

$$x_{ji} = \begin{cases} \frac{2}{\sqrt{3}} v_{ji} \\ \frac{2}{\sqrt{3}} v_{j(i-d)} \\ \frac{2}{\sqrt{3}} \|\mathbf{v}_j\| \\ 0 \\ \sqrt{1 - 4\|\mathbf{v}_j\|^2} \end{cases} \quad x'_{ji} = \begin{cases} -\frac{2}{\sqrt{3}} v'_i & 0 \le i < d \\ -\frac{2}{\sqrt{3}} v'_{(i-d)} & d \le i < 2d \\ \frac{2}{\sqrt{3}} \|\mathbf{v}_j\| & i = 2d \\ \sqrt{1 - \frac{4}{3}(2\|\mathbf{v}'\|^2 + \|\mathbf{v}_j\|^2)} & i = 2d+1 \\ 0 & i = 2d+2, \end{cases}$$

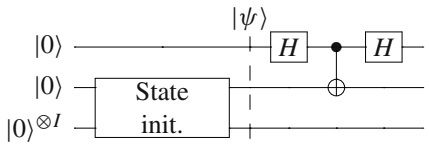
with v_{ji} being the i-th feature of the j-th preprocessed training instance, and v'_i being the i-th feature of the preprocessed test instance. Instead, for the *translation* encoding, $F = 2d + 4$ and

$$x_{ji} = \begin{cases} v_{ji} \\ v_{j(i-d)} \\ \|\mathbf{v}_j\| \\ \frac{1}{2} \\ 0 \\ \sqrt{\frac{3}{4} - 3\|\mathbf{v}_j\|^2} \end{cases} \quad x'_{ji} = \begin{cases} -v'_i & 0 \le i < d \\ -v'_{(i-d)} & d \le i < 2d \\ \|\mathbf{v}_j\| & i = 2d \\ -\frac{1}{2} & i = 2d+1 \\ \sqrt{\frac{3}{4} - (2\|\mathbf{v}'\|^2 + \|\mathbf{v}_j\|^2)} & i = 2d+2 \\ 0 & i = 2d+3. \end{cases}$$

As a consequence, the number of qubits required is the same for both encodings. It is also worth highlighting that, in both cases, $\mathbf{x}_j$ (and therefore $|\alpha\rangle$) is independent of the preprocessed test instance $\mathbf{v}'$, whereas $\mathbf{x}'_j$ (and therefore $|\beta\rangle$) depends on the preprocessed training set $\mathcal{V}$.

3.1.3 Bell-H operation and final state

After the initial state preparation, an operation denoted here as *Bell-H* is performed. In detail, the *Bell-H* corresponds to a SWAP-test-like procedure in which the states of interest ($|\alpha\rangle$ and $|\beta\rangle$), initially prepared in superposition, interfere by means of a controlled-NOT (CNOT) gate. The corresponding quantum circuit, including also the initial state preparation (separated by a dashed vertical line), is the following:



where $I = \lceil \log_2 N \rceil + \lceil \log_2 F \rceil$. In practice, the *Bell-H* circuit consists of a Hadamard gate applied to the first qubit, a CNOT gate with the first qubit as control and the second qubit as target, and another Hadamard gate applied to the first qubit. Hence, the difference with respect to a standard Bell circuit, commonly used to generate Bell states, lies in the presence of an additional downstream Hadamard gate. A significant advantage with respect to the standard SWAP test lies in the constant number of elementary gates required (three), independently of the size of the states involved; as a drawback, the preparation of the input state is more complex, especially without the availability of a QRAM.

The output state obtained after the *Bell-H* operation is

$$|\gamma\rangle = \frac{1}{2}\left(|0\rangle \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle|\alpha\rangle + |0\rangle|\beta\rangle + |1\rangle|\beta\rangle + |1\rangle|\alpha\rangle)\right) + |1\rangle \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle|\alpha\rangle - |0\rangle|\beta\rangle + |1\rangle|\beta\rangle - |1\rangle|\alpha\rangle)\right)\right), \tag{3}$$

and the probability of measuring 1 on the first qubit is $\frac{1}{2}(1 - \langle\alpha|\beta\rangle)$ (the derivation is shown in Appendix A.1). By pulling out the summation on the index register ($|j\rangle$ inside $|\alpha\rangle$ and $|\beta\rangle$) and tracing out (i.e., discarding) the second qubit and the features register ($|i\rangle$ inside $|\alpha\rangle$ and $|\beta\rangle$), it is possible to write the final state as

$$|\delta\rangle = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \left[\sqrt{1 - s(\mathbf{v}_j, \mathbf{v}')}|0\rangle + \sqrt{s(\mathbf{v}_j, \mathbf{v}')}|1\rangle\right]|j\rangle, \tag{4}$$

with $s(\mathbf{v}_j, \mathbf{v}')$ being a similarity measure related to the squared Euclidean distance between $\mathbf{v}_j$ and $\mathbf{v}'$; hence, the lower the distance, the higher the $s(\mathbf{v}_j, \mathbf{v}')$ value. Specifically, $s(\mathbf{v}_j, \mathbf{v}')$ is given by

$$s(\mathbf{v}_j, \mathbf{v}') = P(qubit_1 = 1 \mid j) = \frac{1}{2}(1 - \langle \mathbf{x}_j, \mathbf{x}'_j \rangle), \tag{5}$$

where $qubit_1$ is the first qubit in the circuit, and the value of $\langle \mathbf{x}_j, \mathbf{x}'_j \rangle$ depends on the encoding used, as shown in Table 1 (a more detailed description of how Eq. 4 is obtained is provided in Appendix A.2).

Looking at the first row of Table 1 and recalling Eq. (1), it is possible to notice two aspects: $\langle \mathbf{x}_j, \mathbf{x}'_j \rangle$ is strictly related to the squared Euclidean distance between $\mathbf{v}_j$ and $\mathbf{v}'$ for both encodings; the term $\|\mathbf{v}'\|^2$ does not appear. However, the latter is not an issue, since $\|\mathbf{v}'\|^2$ is the same for all training instances. The other information contained in the table allow understanding the strong point of each encoding. In particular, the *extension* encoding maximizes the range of possible values of $s(\mathbf{v}_j, \mathbf{v}')$, allowing a better representation of the similarity values, namely, a representation less sensitive to the presence of noise. Instead, the *translation* encoding maximizes the probability of measuring 1 on the first qubit, a favorable situation for reasons that will become clear in the next section. Eventually, it is worth highlighting that the range of $s(\mathbf{v}_j, \mathbf{v}')$ is determined by $\mathbf{v}'$; in detail, the minimum range corresponds to a test instance with norm 0, whereas the maximum range corresponds to a test instance with norm equal to $\frac{1}{2}$ (the maximum possible value).

3.1.4 Measurements and distance estimate(s)

After the *Bell-H* operation, the state of the qubits shown in Eq. (4), i.e., the first qubit in the circuit and the index register $|j\rangle$, is measured. In particular, the first qubit is measured first. As a consequence, when 1 (0) is obtained, the indices of the nearest neighbors will have the highest (lowest) probability. If the index register were measured first, the indices would be uniformly sampled. By iterating the circuit execution and the measurement process, the joint probabilities $P(0, j)$ and $P(1, j)$ are estimated as relative frequencies, allowing in turn the estimation of the Euclidean distances. Indeed, the following relationships hold:

$$P(0, j) = \frac{1+\langle \mathbf{x}_j, \mathbf{x}'_j \rangle}{2N} \Longrightarrow \langle \mathbf{x}_j, \mathbf{x}'_j \rangle = 2N \times P(0, j) - 1, \quad (6)$$

$$P(1, j) = \frac{1-\langle \mathbf{x}_j, \mathbf{x}'_j \rangle}{2N} \Longrightarrow \langle \mathbf{x}_j, \mathbf{x}'_j \rangle = 1 - 2N \times P(1, j). \quad (7)$$

Moreover, for the *extension* encoding (see Table 1),

$$d(\mathbf{v}_j, \mathbf{v}') = \sqrt{\frac{3}{4} \langle \mathbf{x}_j, \mathbf{x}'_j \rangle + \|\mathbf{v}'\|^2}, \quad (8)$$

where $d(\mathbf{v}_j, \mathbf{v}')$ is the Euclidean distance between $\mathbf{v}_j$ and $\mathbf{v}'$, while, for the *translation* encoding,

$$d(\mathbf{v}_j, \mathbf{v}') = \sqrt{\langle \mathbf{x}_j, \mathbf{x}'_j \rangle + \frac{1}{4} + \|\mathbf{v}'\|^2}. \quad (9)$$

Regardless of the encoding used, the Euclidean distances $d(\mathbf{v}_j, \mathbf{v}')$ can be estimated from either $P(0, j)$ or $P(1, j)$. In this work, two ways of combining the information of the two joint probabilities have been devised and tested: *avg* and *diff*. In detail, the *avg* distance estimate is the average of the Euclidean distance estimated from $P(0, j)$ and the Euclidean distance estimated from $P(1, j)$. Instead, for the *diff* distance estimate, the scalar product value is obtained as

$$\langle \mathbf{x}_j, \mathbf{x}'_j \rangle = N \times (P(0, j) - P(1, j)),$$

and the Euclidean distance is retrieved through Eq. (8) or (9), depending on the encoding used.

Eventually, it is worth making two observations: given the estimates of $P(0, j)$ and $P(1, j)$ for two indices $j_1, j_2 \in \{0, ..., N-1\}$, the corresponding training instances might be sorted differently according to the *avg* and *diff* distance estimates; if the distance estimates can be mathematically retrieved (i.e., the arguments of the square roots are non-negative), the *avg* distance estimate is always lower than or equal to the corresponding *diff* estimate. More details about these observations can be found in Appendix B.

3.1.5 *k* nearest neighbors and classification

Once all the Euclidean distances $d(\mathbf{v}_j, \mathbf{v}')$ have been estimated, the training elements are classically sorted according to them. Then, the k nearest neighbors are identified, and the test instance is classified by means of a majority voting on the labels of the nearest neighbors.

Table 1 Properties of the two encodings

	Extension	Translation
$\langle \mathbf{x}_j, \mathbf{x}'_j \rangle$ value	$\frac{4}{3}(\|\mathbf{v}_j\|^2 - 2\langle \mathbf{v}_j, \mathbf{v}' \rangle)$	$\|\mathbf{v}_j\|^2 - 2\langle \mathbf{v}_j, \mathbf{v}' \rangle - \frac{1}{4}$
Minimum $s(\mathbf{v}_j, \mathbf{v}')$ range	[0.333, 0.5]	[0.5, 0.625]
Maximum $s(\mathbf{v}_j, \mathbf{v}')$ range	[0, 0.666]	[0.25, 0.75]

Notice that the range of values of $s(\mathbf{v}_j, \mathbf{v}')$ is determined by the preprocessed test instance $\mathbf{v}'$

3.2 Complexity observations

In terms of complexity, the difference between the proposed quantum k-NN algorithm and its classical counterpart lies in the estimation/computation of the Euclidean distances. Indeed, the data preprocessing, the k nearest neighbors identification (performed through a distance sorting operation), and the classification are done classically in both cases. Specifically, the data preprocessing has complexity $O(Nd)$, and the identification of the k nearest neighbors has complexity $O(N \log N + k)$, while the classification step has complexity $O(k)$.

In the proposed algorithm, the Euclidean distances are not computed exactly but are estimated. To this end, a certain number of *shots*, namely, measurements, iterations, is needed. In particular, each iteration requires to prepare the initial state, run the *Bell-H* quantum circuit, and measure the state of the qubits. The initial state preparation is a quite complex operation that can be efficiently accomplished if a QRAM is available.

Proposition 1 *Assuming the availability of a QRAM, the estimation of the Euclidean distances in the proposed quantum k-NN algorithm has a complexity of $O(shots \times (\log N + \log d) + N)$.*

Proof Let x_{ji} and x'_{ji} be real numbers stored in the QRAM as classical floating point numbers. Then, the initial state can be prepared with complexity $O(\log(NF))$. Indeed, the states $|\alpha\rangle$ and $|\beta\rangle$ can be retrieved from the QRAM with complexity $O(\log(NF))$ by definition of QRAM. Instead, the *Bell-H* quantum circuit consists of a constant number of elementary gates; therefore, its complexity is $O(1)$. The measurement step also has constant complexity, as the index qubits are measured simultaneously. Eventually, given the state counts, the computation of the distance estimates has cost $O(N)$. Hence, the complexity of the Euclidean distances estimation is $O(shots \times \log(NF) + N)$, which is equal to $O(shots \times (\log N + \log d) + N)$. □

Instead, if a QRAM is not available, it is necessary to prepare the desired state starting from $|0\rangle^{\otimes(1+I)}$, and the number of gates needed depends on the architecture of the quantum processor. Notice that, since the index register states have non-uniform probabilities that depend also on the outcome of the first qubit measurement, it is not possible to obtain a predefined precision for all the distance estimates. Moreover, *shots* accounts for the estimation of all N Euclidean distances.

In the classical k-NN algorithm, the computation of the Euclidean distances has complexity $O(Nd)$. If we assume that *shots* is a constant value, the complexity of the Euclidean distances estimation in the proposed quantum k-NN turns out to be $O(\log d + N)$, which is lower than $O(Nd)$. However, practically, the higher N, the higher the number of shots needed to properly estimate the Euclidean distances. If, for instance, we assume that *shots* depends logarithmically on N, the complexity turns out to be $O(\log N \log d + N)$, which is still lower than $O(Nd)$. Therefore, under different assumptions on the complexity of *shots*, the proposed quantum k-NN algorithm exhibits a lower complexity than its classical counterpart.

4 Implementation

This section deals with the implementation of the algorithm presented in Section 3. In detail, the Euclidean distance quantum k-NN has been implemented in Python using Qiskit, the open-source SDK provided by IBM (Anis et al. 2021). The code, which is publicly available at https://github.com/ZarHenry96/euclidean-quantum-k-nn, supports different execution modalities, among which:

- *classical*, which does not involve quantum circuits but runs a classical k-NN with the Euclidean distance metric, after the preprocessing step described in Section 3.1.1;
- *statevector*, which processes the final state vector of the circuit and, in actual fact, represents an ideal execution with infinite iterations (in this case, no measurement is performed);
- *simulation* (*local simulation* in the code), which samples from the final probability distribution of the circuit in order to provide state counts.

None of these modalities takes into account the presence of noise. Furthermore, a sample circuit (for the *simulation* modality) is shown in Fig. 1.

In general, the implementation of the algorithm adheres to the description provided in Section 3.1, but some technical aspects deserve to be mentioned. Concerning the preprocessing step, each feature is normalized by: subtracting the average of the maximum and the minimum feature values in the training set; dividing by the feature range (computed on the training set, and set to 1 if the feature is constant) multiplied by $\sqrt{d}$. In addition, if a feature of the test instance exceeds the target range after the normalization, it is clipped to the exceeded edge value. Then, let us focus on the execution modalities involving quantum circuits, since the functioning of the *classical* one is quite straightforward. Specifically, the preparation of the initial state $|\psi\rangle$ (Eq. 2) is done by providing the initialization function supplied by Qiskit with the amplitudes of all qubits except the first one, which is in state $|0\rangle$ by default. As regards the indices not associated with any training instance, which exist if N is not a power of 2, they are excluded when computing the distance values (hence, they are kept in the joint probabilities estima-

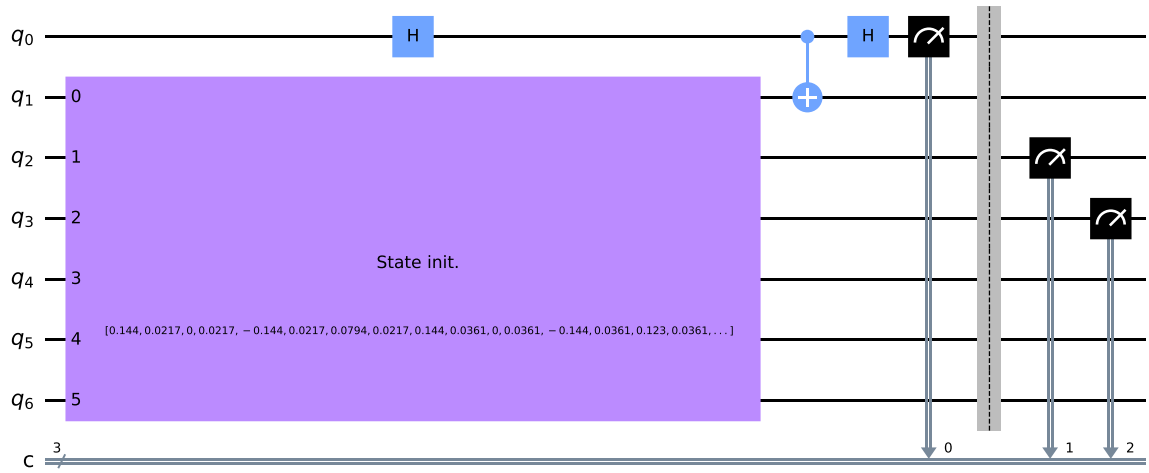


Fig. 1 Example of quantum circuit for the quantum k-NN based on the Euclidean distance. In detail, $N = 4$, $d = 2$, and the execution modality is *simulation* (*statevector* does not include the final measurements)

tion). In addition, if the argument of the square root in Eq. (8) or (9) is negative or larger than 1 due to the state counts distribution obtained, the distance value is approximated to 0 and 1, respectively. Eventually, the training instances with the same Euclidean distance are sorted by increasing index in the training set (this holds also for the *classical* execution modality).

To conclude, it is worth highlighting that the Laplace smoothing (Wilson 1927) has been applied to the estimation of the joint probabilities in the *simulation* modality. In practice, given a number of counts c for the state $|a\rangle|j\rangle$, with $a \in \{0, 1\}$, the probability $P(a, j)$ is estimated as

$$P(a, j) = \frac{c + p}{shots + 2Np},$$

where p is the number of pseudocounts added (for each state), and *shots* is the total number of measurements. In detail, the pseudocounts are summed only to the counts of the significant indices, i.e., the indices actually associated with training instances.

5 Empirical evaluation

In this section, the methods tested, the datasets used, the experimental setup employed, and the results obtained are presented. In particular, the experiments have been run on a shared machine with an Intel Xeon Gold 6238R processor running at 2.20 GHz and 125 GB of RAM.

5.1 Methods

The quantum k-NN based on the Euclidean distance introduced in Section 3 has been tested with different execution modalities and under different (*encoding*, *distance estimate*) configurations, which are reported in Table 2. Runs on real quantum devices have not been performed due to the lack of free-access devices with enough qubits. In addition, for comparison, some classical baseline methods (listed in the same table) have been considered; in particular, the results data for these ones have been fetched from the article by Zardini et al. (2023) (more precisely, from Zardini 2023b).

5.2 Datasets

The datasets used in all the experiments have been taken from the article by Zardini et al. (2023) (more precisely, from Zardini 2023a), mainly for the comparability of the results with the baseline methods. Specifically, the properties of these datasets are reported in Table 3. As explained in the aforementioned article, the original versions of the datasets have been picked from the UCI Machine Learning Repository (Dua and Graff 2017) according to specific criteria, such as numerical features. Then, they have been prepro-

Table 2 Methods tested

Quantum k-NN with Euclidean distance		
Execution modality	**Encoding**	**Distance estimate**
classical	–	–
statevector	extension, translation	avg, diff
simulation	extension, translation	avg, diff
	Baseline methods	
	k-NN with cosine distance	
	Random forest (100 trees)	
	SVM with {Gaussian, linear} kernel	

Table 3 Properties of the datasets used

Name	Classes	Size	Features
01_iris_setosa_versicolor	2	100	4
01_iris_setosa_virginica	2	100	4
01_iris_versicolor_virginica	2	100	4
02_transfusion (Yeh et al. 2009)	2	748	4
03_vertebral_column_2C	2	310	6
04_seeds_1_2	2	140	7
05_ecoli_cp_im	2	220	7
06_glasses_1_2	2	146	9
07_breast_tissue_adi_fadmasgla	2	71	9
08_breast_cancer (Patrício et al. 2018)	2	116	9
09_accent_recognition_uk_us	2	210	12
10_leaf_11_9 (Silva et al. 2013)	2	30	14

Note that the dataset names are links leading to the UCI pages of the original versions of the datasets

cessed in order to meet precise requirements, like binary class labels. The datasets used here, which are available together with the code at https://github.com/ZarHenry96/euclidean-quantum-k-nn, are the ones obtained after the reduction of the number of classes, without subsampling. Additional information about the selection criteria and the preprocessing procedure can be found in the original article.

5.3 Experimental setup

In all experiments, the stratified k-fold cross-validation has been adopted as the validation technique. In practice, each dataset is split into k folds, i.e., subsets. Then, $k-1$ folds form the training set, whereas the leftover represents the test set, and this last step is executed k times so that each subset is used once as the test set. The adjective "stratified" implies that the ratio between classes in the folds has been kept as close as possible to the one of the given dataset. In addition, the same seed has been used for the folds generation in all experiments; in this way, all methods have been evaluated on the same folds.

The parameter values employed in the quantum k-NN experiments are reported in Table 4. In particular, for all execution modalities, the number of folds in the k-fold cross-validation (*folds*) has been set to 5 (a common value in ML), and four different numbers of nearest neighbors selected (k) have been considered. Concerning the *simulation* modality, all (*encoding*, *distance estimate*) configurations have been tested with 1024 measurements (*shots*), which is the default value provided by Qiskit, and the best one has been evaluated also varying this parameter value. In addition, the number of pseudocounts for the Laplace smoothing has been arbitrarily set to 10, and five runs with different seeds have been performed in order to gain statistical evidence. Specifically, the simulation seed for each test instance is randomly generated starting from a "root" run seed. Eventually, it is worth highlighting that the different k values have been evaluated on different seeds, while the *avg* and *diff* distance estimates have been evaluated on the same seeds (namely, for each test instance, the two distance estimates are computed using the same state counts).

Regarding the baseline methods considered for comparison (Zardini et al. 2023), the normalization procedure applied to the input data features is a canonical min-max normalization, whose output range is [0, 1]. The number of folds, the folds generation seed, and the k values considered for the classical k-NN with cosine distance are the same as those used in the quantum k-NN experiments. Eventually, the number of runs for the random forest is 5 (it is a stochastic method).

5.4 Results

The results are presented by means of scatterplots and boxplots. In particular, the quantum k-NN has been evaluated in terms of classification accuracy and correctness of the nearest neighbors found, whereas, for the baseline methods, only the classification accuracy has been considered. More in detail, given a fold, the accuracy is defined as

$$accuracy = \frac{number\ of\ correctly\ classified\ instances\ in\ the\ fold}{total\ number\ of\ instances\ in\ the\ fold};$$

in the case of multiple runs, the average value across runs is reported. Instead, regarding the correctness of the nearest neighbors found, the Jaccard index and the Average Jaccard score (Greene et al. 2014) have been taken into account. Specifically, given a test instance, the Jaccard index (JI) is

Table 4 Parameter setting for the quantum k-NN experiments

Common parameters		Simulation parameters	
Folds	5	Shots	512[a], 1024, 2048[a], 4096[a], 8192[a]
k	3, 5, 7, 9	Pseudocounts	10
		Runs	5

[a]Only the best (*encoding*, *distance estimate*) configuration of the quantum k-NN has been tested with this number of shots

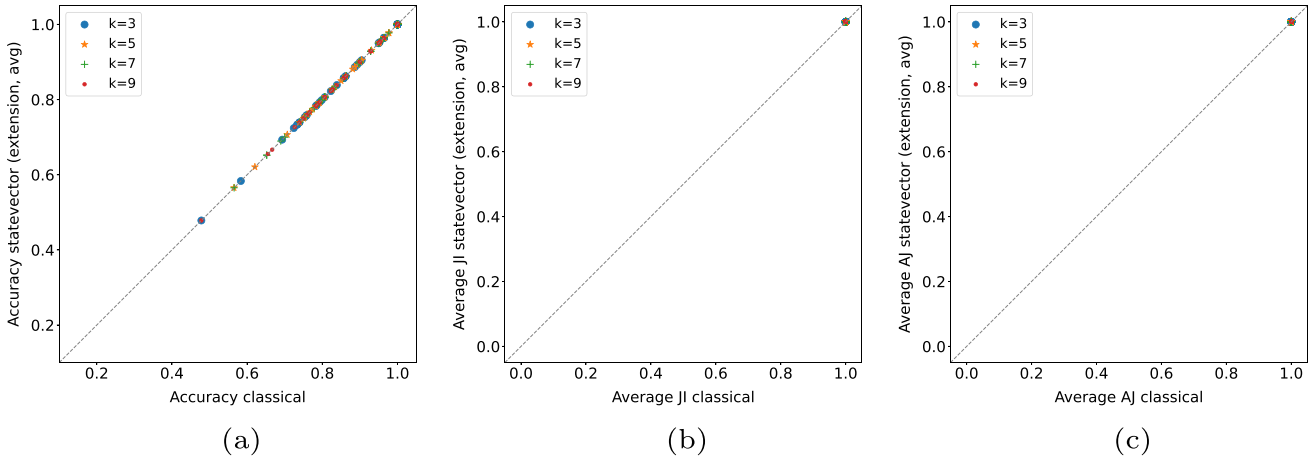


Fig. 2 Comparison between *classical* and *statevector* execution modalities in terms of accuracy (**a**), Jaccard index (**b**), and Average Jaccard score (**c**). The configuration used for *statevector* is (*extension*, *avg*), but the results are the same for all configurations. Each point is related to a dataset fold

defined as

$$Jaccard\ index\ (JI) = \frac{|\mathcal{S}_c \cap \mathcal{S}_f|}{|\mathcal{S}_c \cup \mathcal{S}_f|},$$

where $\mathcal{S}_c$ is the set of correct nearest neighbors (classically computed), and $\mathcal{S}_f$ is the set of nearest neighbors found. Since there is a Jaccard index value for each test instance, the average value has been considered for each fold, and the average of this average value across runs is reported. The same is done for the Average Jaccard (AJ) score, which, given a test instance, is defined as

$$Average\ Jaccard\ (AJ) = \frac{1}{k}\sum_{m=1}^{k} h(\mathcal{S}_{cm}, \mathcal{S}_{fm}),$$

where k is the number of nearest neighbors selected, h is the function computing the Jaccard index, and $\mathcal{S}_{cm}$ is the set containing the correct nearest neighbors up to the m-th most similar item (the same for $\mathcal{S}_{fm}$). Eventually, the statistical significance of the results obtained has been assessed through the Wilcoxon signed-rank test (Wilcoxon 1945), as the data are paired; in some cases (difference boxplots), also the one-sample T-test (Gosset 1908) has been taken into account.

Table 5 Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the distributions shown in Fig. 2

	k=3	k=5	k=7	k=9
Figure 2a	1.000	1.000	1.000	1.000
Figure 2b	1.000	1.000	1.000	1.000
Figure 2c	1.000	1.000	1.000	1.000

The values reported in the table are the p-values obtained

5.4.1 Execution modalities comparison

Let us consider first the *classical* and *statevector* execution modalities. As shown in Fig. 2, the two modalities are equivalent in terms of accuracy (Fig. 2a), Jaccard index (Fig. 2b), and Average Jaccard score (Fig. 2c); the absence of a statistical difference is certified by the Wilcoxon signed-rank test (Table 5). Only one *statevector* configuration, i.e., (*extension*, *avg*), is shown here, but the results are identical for all of them[1]. This confirms that the algorithm presented in Section 3 is correct; indeed, in the ideal case, it is able to achieve the same results as its classical counterpart. It is also worth remembering that the advantage of the quantum algorithm with respect to its classical counterpart lies in the execution time.

Then, let us focus on the *statevector* and *simulation* execution modalities. Specifically, Fig. 3 shows the comparison in terms of accuracy (Fig. 3a), Jaccard index (Fig. 3b), and Average Jaccard score (Fig. 3c) for the (*extension*, *avg*) configuration. As expected, the limited number of measurements (1024) leads to a substantial performance worsening, and *statevector* turns out to statistically outperform the *simulation* modality in both classification accuracy and correctness of the nearest neighbors found, as reported in Table 6. In particular, the drop in performance is more marked for the Jaccard index and the Average Jaccard score. These observations hold for all the (*encoding*, *distance estimate*) configurations; analogous plots and related significance tables for the other configurations are available in Appendix C.1.

[1] For the *translation* encoding, in one fold of one dataset, there are cases in which two nearest neighbors are swapped due to the numerical approximation of the distance values. Therefore, the Average Jaccard score for that fold turns out to be slightly lower than that of the *classical*, but the difference is not statistically significant (p-value=0.317 for all k values)

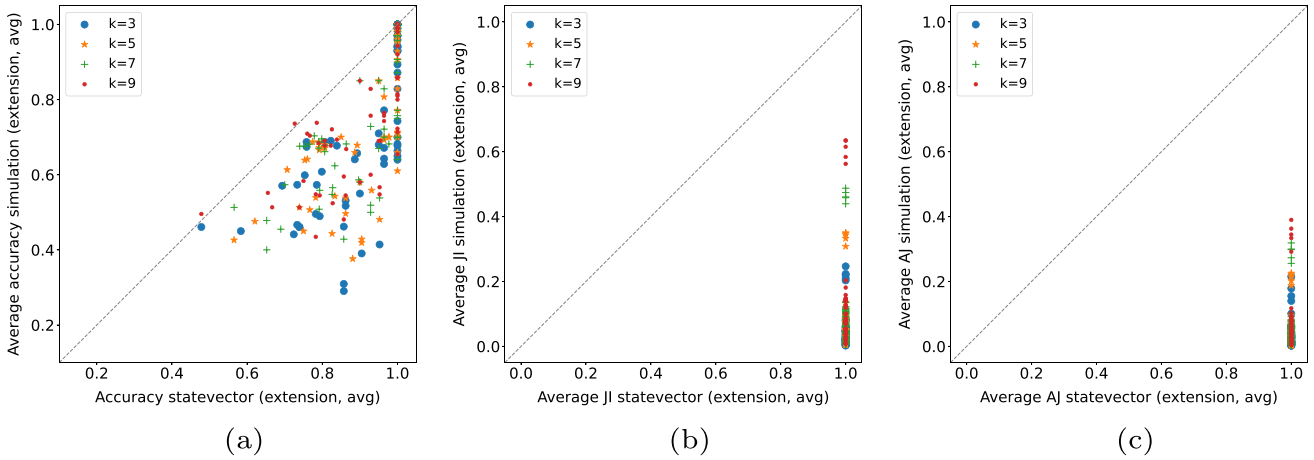


Fig. 3 Comparison between *statevector* (*extension*, *avg*) and *simulation* (*extension*, *avg*) in terms of accuracy (**a**), Jaccard index (**b**), and Average Jaccard score (**c**). The number of shots for *simulation* is 1024, and each point is related to a dataset fold

5.4.2 Encodings and distance estimates comparison

Since the various (*encoding*, *distance estimate*) configurations have achieved the same results for the *statevector* execution modality (the Euclidean distance estimates are exact), only *simulation* (with 1024 shots) is taken into account here. In detail, the configurations are compared by means of difference boxplots, in which each data point represents the difference for a (*dataset fold*, *k value*) pair. The comparisons in accuracy and Jaccard index are shown in Fig. 4a and 4b, respectively, while the plot for the Average Jaccard score is available in Appendix C.2 (Fig. 10) for space reasons.

Concerning the classification accuracy, the configuration that has achieved the best results is (*translation*, *avg*), which has statistically outperformed all the others, as confirmed by Table 7a. In general, the *translation* encoding has performed better than the *extension* encoding in accuracy, and, with an equal encoding, the *avg* distance estimate has outperformed the *diff* distance estimate. Moreover, the differences are statistically significant in terms of both median (Wilcoxon signed-rank test) and mean (one-sample T-test), except for the (*extension*, *avg*) - (*translation*, *diff*) comparison in terms of median.

Surprisingly, the configuration with the highest classification accuracy is not the one that has found the best nearest neighbors. Indeed, the configuration that has achieved the best results in Jaccard index is (*extension*, *avg*), and the differences with respect to the other ones are almost always significant as reported in Table 7b. In general, the *extension* encoding has outperformed the *translation* encoding in Jaccard index, while, with an equal encoding, there is not a clear winning distance estimate: the *avg* distance estimate has performed better with the *extension* encoding, whereas the *diff* distance estimate has achieved better results with the *translation* encoding. The differences are almost all significant in terms of both median and mean; the only exceptions are the (*extension*, *avg*) - (*extension*, *diff*) comparison in mean, and the (*translation*, *avg*) - (*translation*, *diff*) comparison for both statistics. Regarding the Average Jaccard score (Fig. 10), the trend is similar, with (*extension*, *avg*) being the best configuration. However, in this case, the *extension* encoding with the *diff* distance estimate has performed the worst, which means that among the k nearest neighbors selected by this configuration, the correct ones are placed in the last positions. In addition, few differences are statistically significant as reported in Table 13. Among these, it is worth mentioning the (*extension*, *avg*) configuration statistically outperforming all the others in terms of median and (*extension*, *diff*) also in terms of mean.

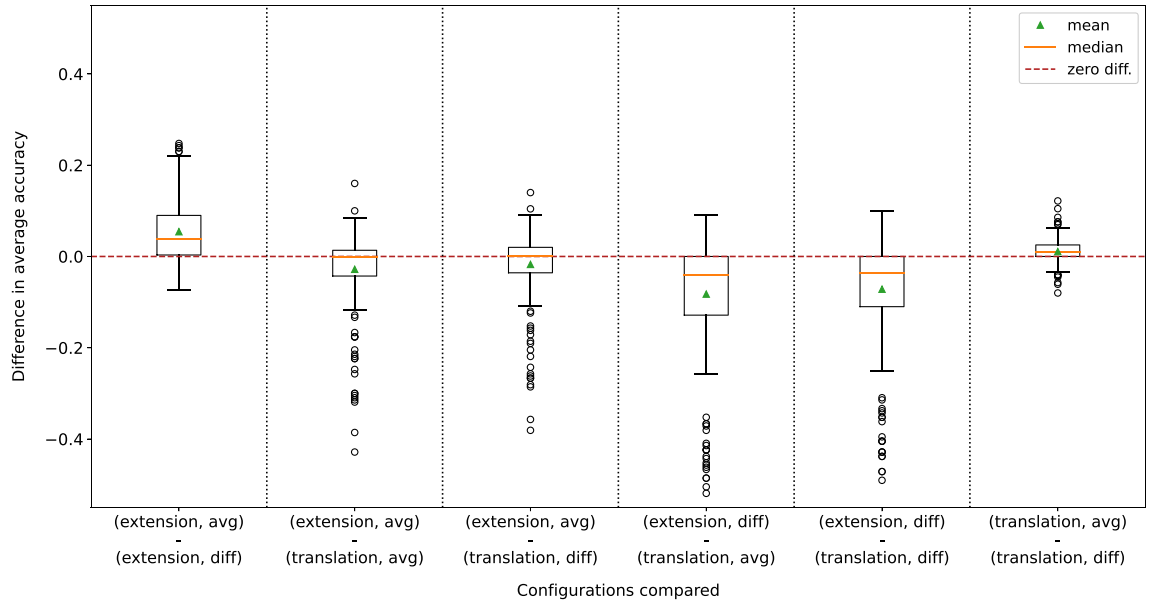
Table 6 Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the distributions shown in Fig. 3

	k=3	k=5	k=7	k=9
Figure 3a	1.624E−10	1.622E−10	5.140E−10	1.608E−09
Figure 3b	1.626E−11	1.630E−11	1.629E−11	1.630E−11
Figure 3c	1.629E−11	1.630E−11	1.630E−11	1.630E−11

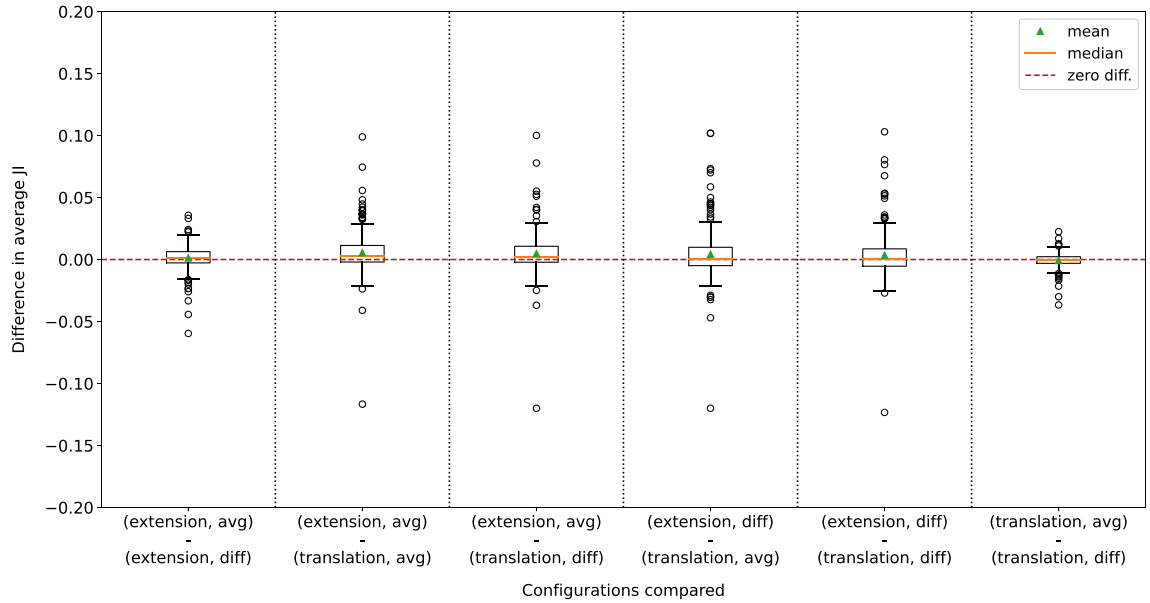
The values reported in the table are the p-values obtained

5.4.3 Comparison with baseline methods

Some classical baseline methods have been chosen for comparison in terms of classification accuracy. Figure 5 shows the comparisons with the *statevector* execution modality in the (*translation*, *avg*) configuration; actually, the configuration used is irrelevant for *statevector*, as explained in the previous sections. In practice, in the ideal case, the quantum k-NN based on the Euclidean distance metric statistically



(a)



(b)

Fig. 4 Comparison of (*encoding*, *distance estimate*) configurations in terms of accuracy (**a**) and Jaccard index (**b**) for the *simulation* execution modality. The number of shots is 1024, and each data point corresponds to the difference for a (*dataset fold*, *k value*) pair

outperforms both the classical k-NN with the cosine distance metric (Fig. 5a) and the SVM with the linear kernel (Fig. 5d), as confirmed by Table 8. Instead, it is outperformed by both the random forest (Fig. 5b) and the SVM with the Gaussian kernel (Fig. 5c), although the differences are almost never statistically significant (only the difference with respect to the SVM with the Gaussian kernel, for $k = 3$, is significant).

Analogous comparison plots for the *simulation* execution modality in the (*translation*, *avg*) configuration, namely, the configuration that has achieved the best results in classification accuracy, are available in Appendix C.3 (Fig. 11). Specifically, all baseline methods considered have statistically outperformed the quantum k-NN in the *simulation* execution modality, as confirmed by Table 14.

Table 7 Wilcoxon signed-rank test and one-sample T-test applied to the distributions shown in Fig. 4a (a) and 4b (b)

	EA-ED	EA-TA	EA-TD	ED-TA	ED-TD	TA-TD
(a)						
Wilcoxon	9.378E−26	3.019E−05	0.069	1.771E−24	1.539E−21	1.084E−09
T-test	1.236E−27	3.706E−07	0.001	1.404E−20	2.037E−18	2.492E−09
(b)						
Wilcoxon	0.003	8.888E−09	1.917E−07	0.010	0.043	0.104
T-test	0.054	6.775E−07	1.102E−05	0.001	0.005	0.054

Each column corresponds to a different comparison, and the first letter identifies the encoding (E=*extension*, T=*translation*), while the second letter identifies the distance estimate (A=*avg*, D=*diff*). The values reported in the tables are the p-values obtained ($\alpha = 0.05$)

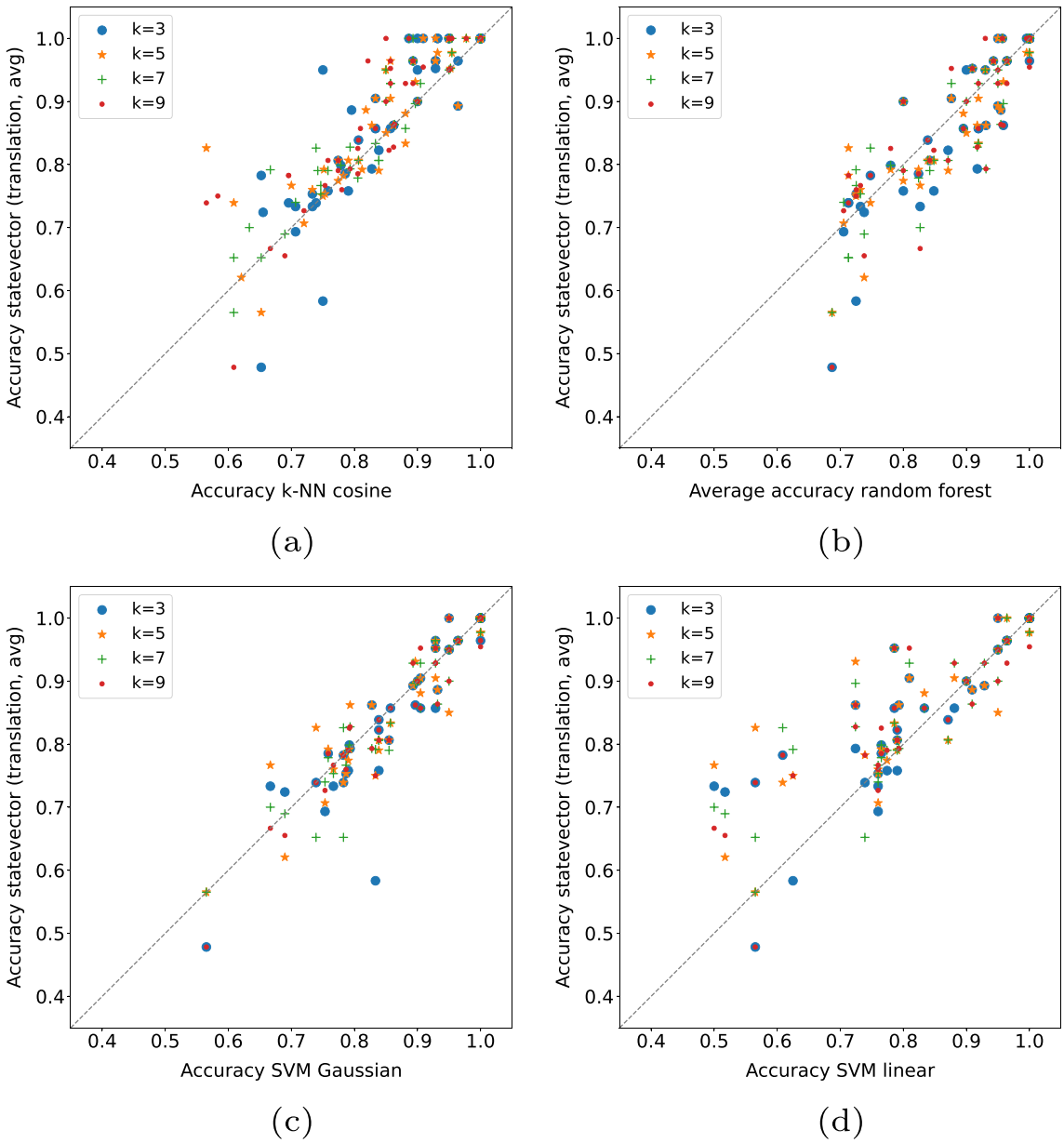


Fig. 5 Comparison between some classical baseline methods and *statevector* in terms of accuracy. The configuration used for *statevector* is (*translation*, *avg*), but the results are the same for all configurations. Each point is related to a dataset fold

Table 8 Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the distributions shown in Fig. 5

	k=3	k=5	k=7	k=9
Figure 5a	0.003	0.001	6.502E−05	8.149E−05
Figure 5b	0.074	0.165	0.095	0.258
Figure 5c	0.046	0.407	0.103	0.062
Figure 5d	0.045	0.005	0.012	0.007

The values reported in the table are the p-values obtained

5.4.4 Number of shots analysis

The last analysis is devoted to the relationship between number of shots (measurements) and performance for the *simulation* execution modality. In particular, for this investigation, only the best quantum k-NN configuration has been considered. Since the primary goal of the quantum k-NN is to correctly find the k nearest neighbors, the (*extension*, *avg*) configuration has been used. Indeed, it has achieved the best Jaccard index and Average Jaccard score, as shown in Section 5.4.2. The results are presented in Fig. 6 and 12 (the latter is available in Appendix C.4) by means of difference boxplots in which 512 is employed as the baseline number of shots.

In practice, the performance tends to improve by increasing the number of shots, and the trend is more evident for both the Jaccard index (Fig. 6b) and the Average Jaccard score (Fig. 12), although the differences in absolute value are smaller when compared to the ones for the accuracy (Fig. 6a). Furthermore, almost all performance differences are statistically significant in terms of both median (Wilcoxon signed-rank test) and mean (one-sample T-test), as reported in Table 9 and 15 (available in Appendix C.4); the only exception is represented by the 1024 − 512 comparison in terms of mean for the accuracy. Eventually, it is worth highlighting that, the larger the dataset, the higher the number of shots required to estimate the joint probability values.

6 Conclusion

In this article, a novel quantum k-NN algorithm based on the Euclidean distance metric has been introduced. In detail, two new encodings of the input data into the quantum states amplitudes, with different properties and low qubit requirements, have been presented (these encodings do not require the unit-norm normalization of the input data). The quantum circuit employed, which does not involve oracles, performs a SWAP-test-like procedure characterized by a fixed number of elementary gates; in this way, quantities related to the pairwise Euclidean distances are computed in parallel. Eventually, given the measurements results (the measurements must be repeated several times), two different ways of estimating the Euclidean distance values have been illustrated; the final training data sorting and classification are classical.

In addition to the theoretical formulation and some complexity observations, an implementation of the algorithm in Python and an extensive empirical evaluation have been provided. First of all, the experimental results have confirmed the correctness of the formulation, with the *statevector* execution modality (ideal execution with an infinite number of shots) achieving the same performance as the *classical* one;

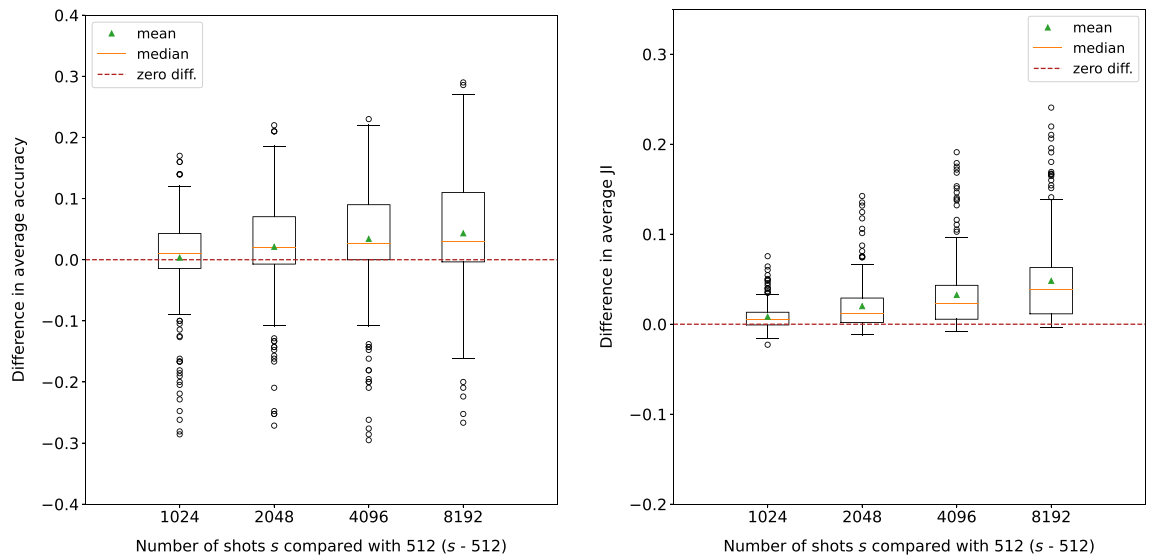


Fig. 6 Comparison of different numbers of shots in terms of accuracy (**a**) and Jaccard index (**b**) for the *simulation* execution modality in the (*extension*, *avg*) configuration. Each data point corresponds to the difference for a (*dataset fold*, *k value*) pair

Table 9 Wilcoxon signed-rank test and one-sample T-test applied to the distributions shown in Fig. 6a (a) and 6b (b)

	1024-512	2048-512	4096-512	8192-512
(a)				
Wilcoxon	0.001	3.131E−07	2.432E−11	6.830E−11
T-test	0.473	5.648E−05	3.016E−08	7.904E−11
(b)				
Wilcoxon	1.151E−17	5.672E−33	4.761E−38	1.393E−40
T-test	4.330E−17	5.712E−26	1.955E−30	1.561E−38

The values reported in the tables are the p-values obtained ($\alpha = 0.05$)

it is worth remarking that the advantage over the classical counterpart lies in the execution time. As expected, *statevector* has outperformed *simulation*, for which the number of measurements is limited, in both classification accuracy and correctness of the nearest neighbors found (the difference is more marked for the latter). Among the (*encoding*, *distance estimate*) configurations tested, (*translation*, *avg*) has achieved the best results in terms of classification accuracy, whereas (*extension*, *avg*) has found the best nearest neighbors. It is worth highlighting that, in the two configurations just mentioned, the encoding is different, while the distance estimate is the same; however, since the primary goal of the algorithm is to find the correct nearest neighbors, (*extension*, *avg*) can be considered the best configuration overall. Concerning the classical baseline methods considered, half of them have achieved better results than the quantum k-NN in the *statevector* modality, whereas the quantum k-NN in the *simulation* modality has been always outperformed. Eventually, the analysis on the number of shots has certified that the performance of the algorithm in *simulation* can be improved by increasing the number of measurements.

Possible future work includes testing the model presented here on different datasets, with a higher number of shots, and on real quantum machines, which are characterized by the presence of noise.

Appendix A: Derivations

Some derivations related to the output state $|\gamma\rangle$ (Eq. 3) are provided in this appendix.

A.1 Probability of measuring 1 on the first qubit

The probability of measuring 1 on the first qubit of the state $|\gamma\rangle$ is

$$
\begin{aligned}
P(1) &= \| |1\rangle\langle 1|\gamma\rangle \|^2 = \\
&= \left\| \frac{1}{2}|1\rangle \otimes \left(\frac{1}{\sqrt{2}}(|0\rangle|\alpha\rangle - |0\rangle|\beta\rangle + |1\rangle|\beta\rangle - |1\rangle|\alpha\rangle) \right) \right\|^2 = \\
&= \frac{1}{8}(\langle 0|\langle\alpha| - \langle 0|\langle\beta| + \langle 1|\langle\beta| - \langle 1|\langle\alpha|) \times \\
&\quad \times (|0\rangle|\alpha\rangle - |0\rangle|\beta\rangle + |1\rangle|\beta\rangle - |1\rangle|\alpha\rangle) = \\
&= \frac{1}{8}(1 - \langle\alpha|\beta\rangle - \langle\beta|\alpha\rangle + 1 + 1 - \langle\beta|\alpha\rangle - \langle\alpha|\beta\rangle + 1) = \\
&= \frac{1}{8}(4 - 2\langle\alpha|\beta\rangle - 2\langle\beta|\alpha\rangle) = \\
&\quad |\alpha\rangle \text{and} |\beta\rangle \text{have real coefficients} \\
&= \frac{1}{8}(4 - 4\langle\alpha|\beta\rangle) = \\
&= \frac{1}{2}(1 - \langle\alpha|\beta\rangle).
\end{aligned}
$$

A.2 Reduced final state

Given $|\gamma\rangle$, let us first pull out the summation on the index register $|j\rangle$ inside $|\alpha\rangle$ and $|\beta\rangle$. In this way, we obtain a state in the form

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} [...]\,|j\rangle,$$

where [...] includes all circuit qubits except those belonging to the index register. Then, let us trace out, namely, discard, the second qubit in the circuit and the features register $|i\rangle$; from the mathematical viewpoint, this corresponds to compute the partial trace over these qubits of the density operator describing the system. By doing this, we obtain a reduced version of the final state including only the first qubit in the circuit and the index register, which can be written as

$$\frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \left[\sqrt{P(0 \mid j)}|0\rangle + \sqrt{P(1 \mid j)}|1\rangle \right] |j\rangle.$$

Eventually, let us exploit the derivations presented in Appendix A.1. In practice, as shown in Eq. (5), $P(1 \mid j)$ turns out to be equal to $\frac{1}{2}(1 - \langle \mathbf{x}_j, \mathbf{x}'_j \rangle)$, since the summation on the index register (together with its coefficient) has been pulled out. In addition, $P(0 \mid j)$ must be equal to $1 - P(1 \mid j)$ due to

the law of total probability. This leads to the definition of the reduced final state $|\delta\rangle$ provided in Eq. (4).

Appendix B: Distance estimates

In this appendix, some additional information about the *avg* and *diff* distance estimates is provided. In particular, let us consider two preprocessed training instances $\mathbf{v}_{j_1}$ and $\mathbf{v}_{j_2}$, with $j_1, j_2 \in \{0, ..., N-1\}$, and a preprocessed test instance $\mathbf{v}'$. Let $d_0(\mathbf{v}_{j_1}, \mathbf{v}')$ and $d_1(\mathbf{v}_{j_1}, \mathbf{v}')$ be the Euclidean distances from $\mathbf{v}'$ estimated from the joint probabilities $P(0, j_1)$ and $P(1, j_1)$ for $\mathbf{v}_{j_1}$ (analogously for $\mathbf{v}_{j_2}$). Then, the following relationships hold:

$$avg = \frac{d_0(\mathbf{v}_{j_1}, \mathbf{v}') + d_1(\mathbf{v}_{j_1}, \mathbf{v}')}{2},$$

$$diff = \sqrt{\frac{d_0(\mathbf{v}_{j_1}, \mathbf{v}')^2 + d_1(\mathbf{v}_{j_1}, \mathbf{v}')^2}{2}}.$$

The former is just the definition of the *avg* distance estimate, whereas the latter can be easily verified using Eq. (8) (or 9, depending on the encoding selected) together with Eq. (6) and (7).

B.1 Instance sorting

It is possible that $\mathbf{v}_{j_1}$ and $\mathbf{v}_{j_2}$ are sorted differently according to the *avg* and *diff* distance estimates. Indeed, let us consider the following scenario:

$$d_0(\mathbf{v}_{j_1}, \mathbf{v}') = 0.5 \qquad d_0(\mathbf{v}_{j_2}, \mathbf{v}') = 0.4$$
$$d_1(\mathbf{v}_{j_1}, \mathbf{v}') = 0.29 \qquad d_1(\mathbf{v}_{j_2}, \mathbf{v}') = 0.4\,.$$

In this case, the *avg* distance estimates for $\mathbf{v}_{j_1}$ and $\mathbf{v}_{j_2}$ are 0.395 and 0.4, respectively, while the *diff* distance estimates are 0.409 and 0.4, respectively. Hence, $\mathbf{v}_{j_1}$ turns out to be closer than $\mathbf{v}_{j_2}$ (to $\mathbf{v}'$) according to *avg* and further than it according to *diff*.

B.2 Magnitude

Let us assume that $d_0(\mathbf{v}_{j_1}, \mathbf{v}')$ and $d_1(\mathbf{v}_{j_1}, \mathbf{v}')$ can be mathematically computed, namely, the arguments of the square root in Eq. (8) (or 9, depending on the encoding selected) are non-negative. Then, the *avg* distance estimate is always lower than or equal to the corresponding *diff* estimate. In fact, the opposite would be true if and only if $(d_0(\mathbf{v}_{j_1}, \mathbf{v}') - d_1(\mathbf{v}_{j_1}, \mathbf{v}'))^2 < 0$, which is not possible.

If the assumption about the square root arguments does not apply due to the state count distribution obtained, the *avg* distance estimate might turn out to be higher than the corresponding *diff* estimate. Indeed, in the implementation provided here, square roots of negative values are approximated to 0, as explained in Section 4.

Appendix C: Additional plots

Additional results plots and related statistical significance tables are provided in this appendix.

C.1 Execution modalities comparison

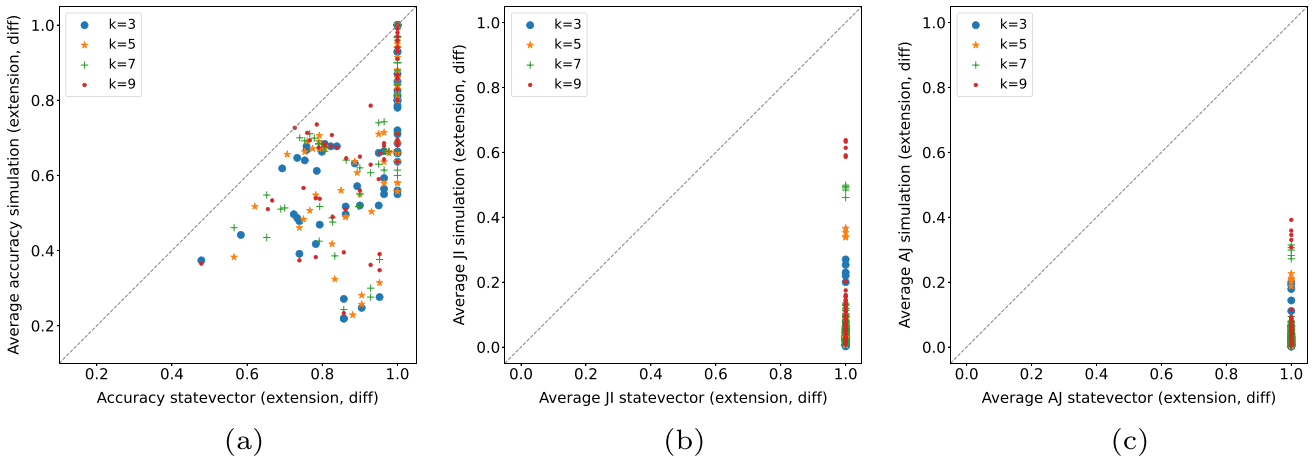


Fig. 7 Comparison between *statevector* (*extension*, *diff*) and *simulation* (*extension*, *diff*) in terms of accuracy (**a**), Jaccard index (**b**), and Average Jaccard score (**c**). The number of shots for *simulation* is 1024, and each point is related to a dataset fold

Table 10 Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the distributions shown in Fig. 7

	k=3	k=5	k=7	k=9
Figure 7a	1.106E−10	1.106E−10	7.530E−11	1.719E−10
Figure 7b	1.628E−11	1.629E−11	1.630E−11	1.630E−11
Figure 7c	1.630E−11	1.630E−11	1.630E−11	1.630E−11

The values reported in the table are the p-values obtained

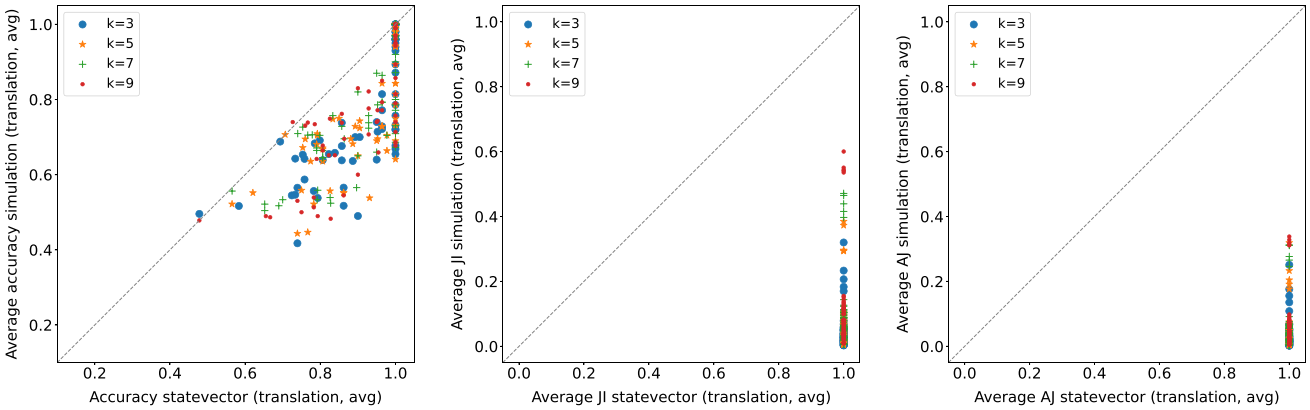


Fig. 8 Comparison between *statevector* (*translation*, *avg*) and *simulation* (*translation*, *avg*) in terms of accuracy (**a**), Jaccard index (**b**), and Average Jaccard score (**c**). The number of shots for *simulation* is 1024, and each point is related to a dataset fold

Table 11 Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the distributions shown in Fig. 8

	k=3	k=5	k=7	k=9
Figure 8a	1.234E−10	3.492E−10	3.492E−10	6.149E−10
Figure 8b	1.628E−11	1.630E−11	1.630E−11	1.630E−11
Figure 8c	1.630E−11	1.630E−11	1.630E−11	1.630E−11

The values reported in the table are the p-values obtained

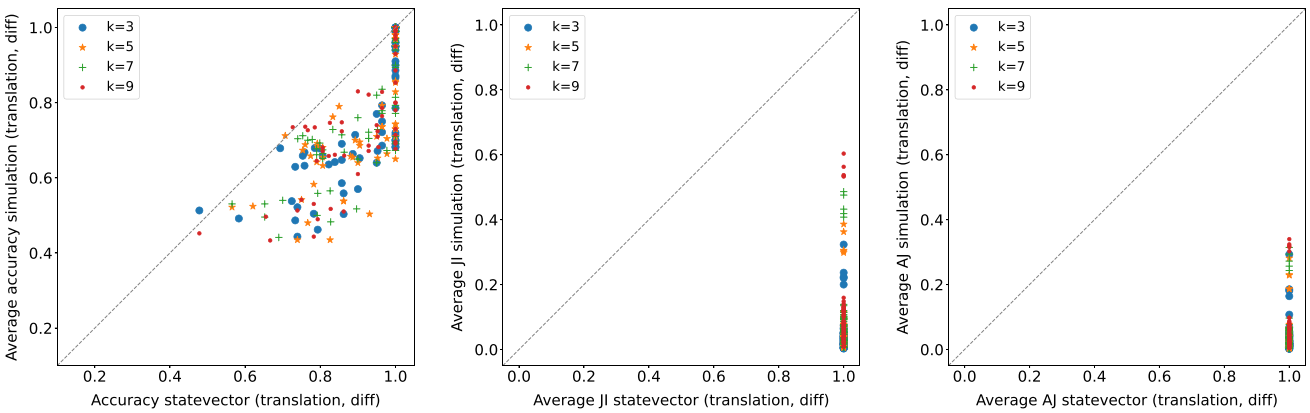


Fig. 9 Comparison between *statevector* (*translation*, *diff*) and *simulation* (*translation*, *diff*) in terms of accuracy (**a**), Jaccard index (**b**), and Average Jaccard score (**c**). The number of shots for *simulation* is 1024, and each point is related to a dataset fold

Table 12 Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the distributions shown in Fig. 9

	k=3	k=5	k=7	k=9
Figure 9a	1.379E−10	1.712E−10	7.516E−11	2.523E−10
Figure 9b	1.627E−11	1.629E−11	1.630E−11	1.630E−11
Figure 9c	1.629E−11	1.630E−11	1.630E−11	1.630E−11

The values reported in the table are the p-values obtained

C.2 Encodings and distance estimates comparison

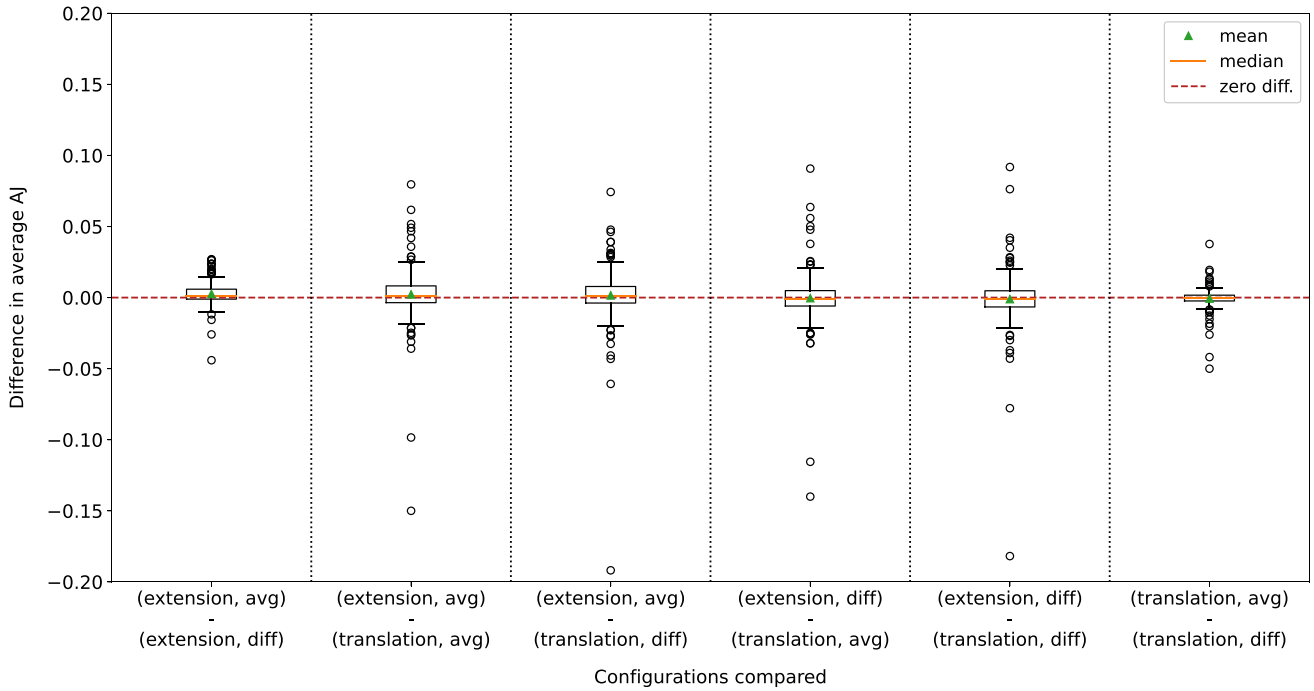


Fig. 10 Comparison of (*encoding*, *distance estimate*) configurations in terms of Average Jaccard score for the *simulation* execution modality. The number of shots is 1024, and each data point corresponds to the difference for a (*dataset fold*, *k value*) pair

Table 13 Wilcoxon signed-rank test and one-sample T-test applied to the distributions shown in Fig. 10

	EA-ED	EA-TA	EA-TD	ED-TA	ED-TD	TA-TD
Wilcoxon	1.556E−07	0.002	0.008	0.315	0.111	0.043
T-test	6.693E−07	0.057	0.202	0.735	0.374	0.123

Each column corresponds to a different comparison, and the first letter identifies the encoding (E=*extension*, T=*translation*), while the second letter identifies the distance estimate (A=*avg*, D=*diff*). The values reported in the table are the p-values obtained ($\alpha = 0.05$)

C.3 Comparison with baseline methods

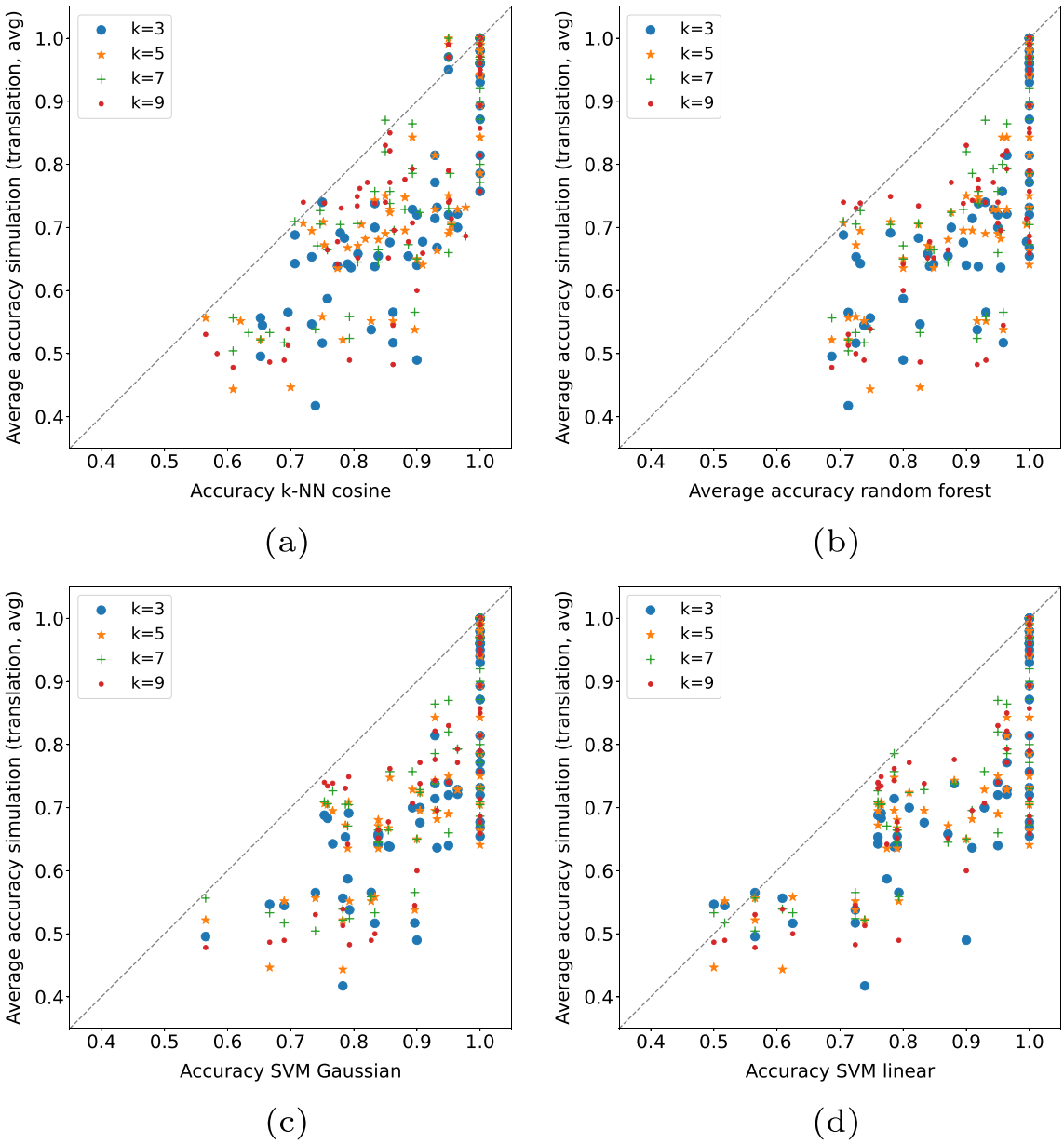


Fig. 11 Comparison between some classical baseline methods and *simulation* (*translation*, *avg*) in terms of accuracy. The number of shots for *simulation* is 1024, and each point is related to a dataset fold

Table 14 Wilcoxon signed-rank test ($\alpha = 0.05$) applied to the distributions shown in Fig. 11

	k=3	k=5	k=7	k=9
Figure 11a	2.147E−10	5.660E−10	8.949E−10	1.210E−09
Figure 11b	1.105E−10	2.521E−10	3.705E−10	6.632E−10
Figure 11c	1.105E−10	2.380E−10	3.494E−10	3.500E−10
Figure 11d	3.172E−10	3.976E−10	1.299E−09	3.502E−10

The values reported in the table are the p-values obtained

C.4 Number of shots analysis

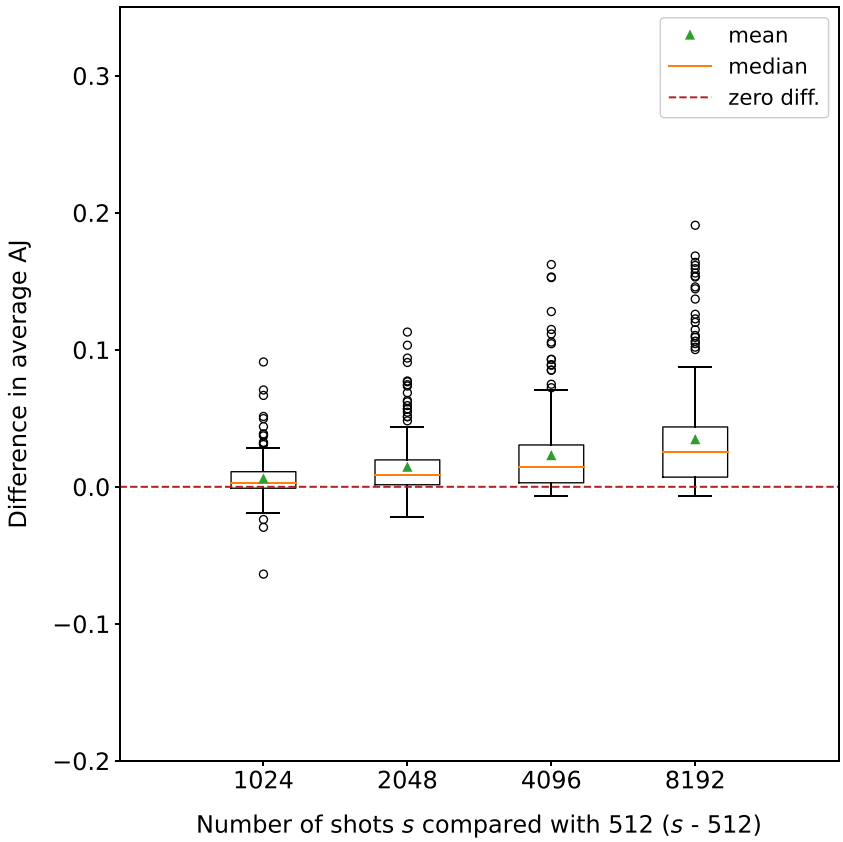


Fig. 12 Comparison of different numbers of shots in terms of Average Jaccard score for the *simulation* execution modality in the (*extension*, *avg*) configuration. Each data point corresponds to the difference for a (*dataset fold*, *k value*) pair

Table 15 Wilcoxon signed-rank test and one-sample T-test applied to the distributions shown in Fig. 12

	1024-512	2048-512	4096-512	8192-512
Wilcoxon	1.977E−11	8.804E−30	2.327E−37	1.030E−39
T-test	3.037E−09	6.719E−23	1.862E−27	3.622E−33

The values reported in the table are the p-values obtained ($\alpha = 0.05$)

Acknowledgements The authors gratefully acknowledge the Italian Ministry of University and Research (MUR), which, under the initiative "Dipartimenti di Eccellenza 2018–2022 (Legge 232/2016)", has provided the computational resources used in the experiments.

Author Contributions Conceptualization: E.Z., E.B., D.P. Data curation: E.Z. Formal analysis: E.Z., E.B., D.P. Investigation: E.Z. Methodology: E.Z., E.B., D.P. Software: E.Z. Supervision: E.B., D.P. Validation: E.Z. Visualization: E.Z. Original draft preparation: E.Z. Review and editing: E.Z., E.B., D.P.

Funding Open access funding provided by Universitá degli Studi di Trento within the CRUI-CARE Agreement. This work was supported by Q@TN, the joint lab between University of Trento, FBK-Fondazione Bruno Kessler, INFN-National Institute for Nuclear Physics and CNR-National Research Council. In addition, this work was partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

Availability of data and materials The data that support the findings of this study (datasets, information needed to reproduce the experiments, and collected results data) are available in the Figshare repository, https://doi.org/10.6084/m9.figshare.22598455.v1. In addition, the same data, together with the raw results data, are reachable from the GitHub repository, https://github.com/ZarHenry96/euclidean-quantum-k-nn.

Code availability The code is available in the GitHub repository, https://github.com/ZarHenry96/euclidean-quantum-k-nn.

Declarations

Conflict of interest The authors declare no competing interests.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Abbas A, Sutter D, Zoufal C et al (2021) The power of quantum neural networks. Nat Comput Sci 1:403–409. https://doi.org/10.1038/s43588-021-00084-1, https://www.nature.com/articles/s43588-021-00084-1#citeas

Afham A, Basheer A, Goyal SK (2020) Quantum k-nearest neighbor machine learning algorithm. https://arxiv.org/abs/2003.09187v1

Anis MS, Abraham H, AduOffei, et al (2021) Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2573505

Basheer A, Afham A, Goyal SK (2021) Quantum k-nearest neighbors algorithm. https://arxiv.org/abs/2003.09187

Biamonte J, Wittek P, Pancotti N et al (2017) Quantum machine learning. Nature 549:195–202. https://doi.org/10.1038/nature23474

Brassard G, Hoyer P, Mosca M et al (2002) Quantum amplitude amplification and estimation. Contemporary Mathematics 305:53–74

Buhrman H, Cleve R, Watrous J et al (2001) Quantum Fingerprinting. Phys Rev Lett 87(167):902. https://doi.org/10.1103/PhysRevLett.87.167902

Cleve R, Ekert A, Macchiavello C et al (1998) Quantum algorithms revisited. Proceedings of the Royal Society of London Series A: Mathematical, Physical and Engineering Sciences 454(1969):339–354. https://doi.org/10.1098/rspa.1998.0164

Dang Y, Jiang N, Hu H et al (2018) Image classification based on quantum K-Nearest-Neighbor algorithm. Quantum Information Processing 17(9):239. https://doi.org/10.1007/s11128-018-2004-9

Dua D, Graff C (2017) UCI Machine Learning Repository. http://archive.ics.uci.edu/ml

Dürr C, Høyer P (1999) A Quantum Algorithm for Finding the Minimum. https://arxiv.org/abs/quant-ph/9607014

Fastovets DV, Bogdanov YI, Bantysh BI, et al (2019) Machine learning methods in quantum computing theory. In: International Conference on Micro- and Nano-Electronics 2018, International Society for Optics and Photonics, vol 11022. SPIE, Zvenigorod, Russia, pp 752 – 761, https://doi.org/10.1117/12.2522427

Feng C, Zhao B, Zhou X, et al (2023) An Enhanced Quantum K-Nearest Neighbor Classification Algorithm Based on Polar Distance. Entropy 25(1). https://doi.org/10.3390/e25010127, https://www.mdpi.com/1099-4300/25/1/127

Fix E, Hodges JL (1951) Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties. Tech. Rep. 4, USAF School of Aviation Medicine, Randolph Field

Gao LZ, Lu CY, Guo GD et al (2022) Quantum K-nearest neighbors classification algorithm based on Mahalanobis distance. Front Phys 10. https://doi.org/10.3389/fphy.2022.1047466

Getachew AT (2020) Quantum K-medians Algorithm Using Parallel Euclidean Distance Estimator. https://doi.org/10.48550/ARXIV.2012.11139. https://arxiv.org/abs/2012.11139

Giovannetti V, Lloyd S, Maccone L (2008) Quantum Random Access Memory. Phys Rev Lett 100(160):501. https://doi.org/10.1103/PhysRevLett.100.160501

Gosset WS (1908) The Probable Error of a Mean, originally published under the pseudonym "Student". Biometrika 6(1):1–25. https://doi.org/10.2307/2331554

Greene D, O'Callaghan D, Cunningham P (2014) How many topics? stability analysis for topic models. In: Calders T, Esposito F, Hüllermeier E et al (eds) Machine Learning and Knowledge Discovery in Databases. Springer, Berlin Heidelberg, Berlin, Heidelberg, pp 498–513

Grover LK (1996) A fast quantum mechanical algorithm for database search. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing. Association for Computing Machinery, New York, NY, USA, STOC '96, p 212-219. https://doi.org/10.1145/237814.237866

Kaye P (2004) Reversible addition circuit using one ancillary bit with application to quantum computing. https://doi.org/10.48550/ARXIV.QUANT-PH/0408173, https://arxiv.org/abs/quant-ph/0408173

Li J, Lin S, Yu K et al (2021) Quantum K-nearest neighbor classification algorithm based on Hamming distance. Quantum Information

Processing 21(1):18. https://doi.org/10.1007/s11128-021-03361-0

Lloyd S, Mohseni M, Rebentrost P (2013) Quantum algorithms for supervised and unsupervised machine learning. https://doi.org/10.48550/ARXIV.1307.0411, https://arxiv.org/abs/1307.0411

Ma Yz, Song Hf, Zhang J (2021) Quantum Algorithm for K-Nearest Neighbors Classification Based on the Categorical Tensor Network States. International Journal of Theoretical Physics 60(3):1164–1174. https://doi.org/10.1007/s10773-021-04742-y

Mitarai K, Kitagawa M, Fujii K (2019) Quantum analog-digital conversion. Phys Rev A 99(012):301. https://doi.org/10.1103/PhysRevA.99.012301

Miyamoto K, Iwamura M, Kise K (2019) A Quantum Algorithm for Finding *k*-Minima. https://doi.org/10.48550/ARXIV.1907.03315, https://arxiv.org/abs/1907.03315

Patrício M, Pereira J, Crisóstomo J et al (2018) Using Resistin, glucose, age and BMI to predict the presence of breast cancer. BMC Cancer 18(1):29. https://doi.org/10.1186/s12885-017-3877-1

Quezada LF, Sun GH, Dong SH (2022) Quantum Version of the k-NN Classifier Based on a Quantum Sorting Algorithm. Annalen der Physik 534(5):2100,449. https://doi.org/10.1002/andp.202100449

Rebentrost P, Mohseni M, Lloyd S (2014) Quantum Support Vector Machine for Big Data Classification. Phys Rev Lett 113(130):503. https://doi.org/10.1103/PhysRevLett.113.130503

Rebentrost P, Steffens A, Marvian I et al (2018) Quantum singular-value decomposition of nonsparse low-rank matrices. Phys Rev A 97(012):327. https://doi.org/10.1103/PhysRevA.97.012327

Ruan Y, Xue X, Liu H et al (2017) Quantum Algorithm for K-Nearest Neighbors Classification Based on the Metric of Hamming Distance. International Journal of Theoretical Physics 56(11):3496–3507. https://doi.org/10.1007/s10773-017-3514-4

Sarma A, Chatterjee R, Gili K, et al (2020) Quantum unsupervised and supervised learning on superconducting processors. Quantum Information and Computation 20(7–8):541–552. https://doi.org/10.26421/QIC20.7-8-1

Schuld M, Sinayskiy I, Petruccione F (2014) Quantum Computing for Pattern Classification. In: Pham DN, Park SB (eds) PRICAI 2014: Trends in Artificial Intelligence. Springer International Publishing, Cham, pp 208–220

Schuld M, Fingerhuth M, Petruccione F (2017) Implementing a distance-based classifier with a quantum interference circuit. Europhysics Letters 119(6):60,002. https://doi.org/10.1209/0295-5075/119/60002

Silva PFB, Marçal ARS, da Silva RMA (2013) Evaluation of Features for Leaf Discrimination. Springer Lecture Notes in Computer Science 7950:197–204

Trugenberger CA (2002) Quantum Pattern Recognition. Quantum Information Processing 1(6):471–493. https://doi.org/10.1023/A:1024022632303

Wang Y, Wang R, Li D et al (2019) Improved Handwritten Digit Recognition using Quantum K-Nearest Neighbor Algorithm. International Journal of Theoretical Physics 58(7):2331–2340. https://doi.org/10.1007/s10773-019-04124-5

Wiebe N, Kapoor A, Svore KM (2015) Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. Quantum Information and Computation 15(3–4):316–356. https://doi.org/10.26421/QIC15.3-4-7

Wilcoxon F (1945) Individual Comparisons by Ranking Methods. Biometrics Bulletin 1(6):80–83. http://www.jstor.org/stable/3001968

Wilson EB (1927) Probable Inference, the Law of Succession, and Statistical Inference. Journal of the American Statistical Association 22(158):209–212. https://doi.org/10.1080/01621459.1927.10502953

Wiśniewska J, Sawerwain M (2018) Recognizing the pattern of binary Hermitian matrices by quantum kNN and SVM methods. Vietnam Journal of Computer Science 5(3):197–204. https://doi.org/10.1007/s40595-018-0115-y

Yeh IC, Yang KJ, Ting TM (2009) Knowledge Discovery on RFM Model Using Bernoulli Sequence. Expert Syst Appl 36(3):5866–5871. https://doi.org/10.1016/j.eswa.2008.07.018

Yu K, Guo GD, Li J et al (2020) Quantum Algorithms for Similarity Measurement Based on Euclidean Distance. International Journal of Theoretical Physics 59(10):3134–3144. https://doi.org/10.1007/s10773-020-04567-1

Zardini E (2023a). QML Pipeline Datasets. https://doi.org/10.6084/m9.figshare.22333102.v1, https://www.figshare.com/articles/dataset/QML_Pipeline_Datasets/22333102

Zardini E (2023b) QML Pipeline Raw Results. Figshare https://doi.org/10.6084/m9.figshare.22333147.v1, https://figshare.com/articles/dataset/QML_Pipeline_Raw_Results/22333147

Zardini E, Blanzieri E, Pastorello D (2023) Implementation and empirical evaluation of a quantum machine learning pipeline for local classification. PLOS ONE 18(11):1–28. https://doi.org/10.1371/journal.pone.0287869

Zhou NR, Liu XX, Chen YL et al (2021) Quantum K-Nearest-Neighbor Image Classification Algorithm Based on K-L Transform. International Journal of Theoretical Physics 60(3):1209–1224. https://doi.org/10.1007/s10773-021-04747-7