



Recognizing the pattern of binary Hermitian matrices by quantum kNN and SVM methods

Joanna Wiśniewska¹ · Marek Sawerwain²

Received: 1 August 2017 / Accepted: 15 May 2018 / Published online: 30 May 2018
© The Author(s) 2018

Abstract

The article contains a description of two quantum circuits for pattern recognition. The first approach is realized with use of k nearest neighbors algorithm and the second with support vector machine. The task is to distinguish between Hermitian and non-Hermitian matrices. The quantum circuits are constructed to accumulate elements of a learning set. After this process, circuits are able to produce a quantum state which contains the information if a tested element fits to the trained pattern. To improve the efficiency of presented solutions, the matrices were uniquely labeled with feature vectors. The role of the feature vectors is to highlight some features of the objects which are crucial in the process of classification. The circuits were implemented in Python programming language and some numeric experiments were conducted to examine the capacity of presented solutions in pattern recognition.

Keywords Quantum circuits · Pattern recognition · Supervised machine learning · Hamming distance

1 Introduction

A conception of solving problems with use of k Nearest Neighbor (kNN) algorithm was created in fifties of the previous century [2] and it is still very popular and develops constantly. In addition, the concept of Support Vector Machine (SVM) for problems of classification is a known solution, which actual version was presented in 1995 [1]. As researchers working on some aspects of quantum computing, we were inspired by kNN and SVM methods to apply the idea of pattern recognition in quantum circuits [4,6,14,16].

We refer to [9,12,13], where it was shown that it is possible to build a quantum circuit which works as a classifier. The basic idea was prepared to be utilized in the field of image processing [8,17,18]. The kNN and SVM were constructed with visible inspiration flowing out of methods used in arti-

ficial neural networks: the data describing the patterns are enclosed in learning and testing sets, and the circuit obtains the learning set and is able to recognize if a tested element fits to the pattern emerging from the learning set. An aim of this article is to verify a thesis if quantum circuits based on kNN and SVM methods are able to recognize Hermitian and non-Hermitian matrices.

This work is an extended version of conference ACIIDS 2017 paper: [15]—in that article, we simulated a quantum circuit based on kNN algorithm which role was to classify the Hermitian and non-Hermitian matrices. This article is organized as follows: in Sect. 2, we present some basic definitions helpful in understanding the main idea of this work. Section 3 is dedicated to the detailed description of the quantum circuit for kNN method and Sect. 4 contains the construction of quantum solution for SVM. In Sect. 5, the results of computational experiments were analysed. A summary and conclusions are presented in Sect. 6.

✉ Joanna Wiśniewska
jwisniewska@wat.edu.pl

Marek Sawerwain
M.Sawerwain@issi.uz.zgora.pl

¹ Institute of Information Systems, Faculty of Cybernetics, Military University of Technology, Kaliskiego 2, 00-908 Warsaw, Poland

² Institute of Control and Computation Engineering, University of Zielona Góra, Licealna 9, 65-417 Zielona Góra, Poland

2 Quantum computing—basic definitions

To explain the analysed matter, we need to introduce a few definitions corresponding to quantum information processing [5]. First, notion concerns a quantum bit—so-called qubit

which is a normalized vector in a two-dimensional Hilbert space \mathcal{H}_2 .

Two orthogonal qubits constitute a computational basis. Of course, we can imagine the infinite number of such paired qubits. One of the most known basis is so-called standard basis. This basis is created by qubits:

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{1}$$

where the form $|\cdot\rangle$ is a Dirac notation. The difference between classical bit and qubit is that qubit may be a "mixture" of orthogonal vectors. This phenomenon is called superposition. Hence, the state of quantum bit $|\psi\rangle$ we can present as

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle, \text{ where } |\alpha_0|^2 + |\alpha_1|^2 = 1. \tag{2}$$

The coefficients α_0 and α_1 are complex numbers and they are termed amplitudes. A character of quantum state is non-deterministic. The probability that the state $|\psi\rangle$ equals $|0\rangle$ is $|\alpha_0|^2$ and, adequately, the probability that the state $|\psi\rangle$ is $|1\rangle$ is expressed by value $|\alpha_1|^2$. Of course, it is also possible that one of the amplitudes equals to 0 and the other to 1 (in this case state $|\psi\rangle$ is one of basic states).

If we need more than one quantum bit to perform any calculations, we can use a quantum register which is a system of qubits, joined by a tensor product (denoted by symbol \otimes) in a mathematical sense. For example, the state $|\phi\rangle$ of 3-qubit register containing qubits $|0\rangle$, $|1\rangle$ and $|1\rangle$ is

$$|\phi\rangle = |0\rangle \otimes |1\rangle \otimes |1\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix} \otimes \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{3}$$

Usually, we omit the symbols \otimes , so the above state may be also denoted as $|\phi\rangle = |011\rangle$.

In case of any n -qubit state $|\varphi\rangle$, its form can be expressed as a superposition of basic states:

$$|\varphi\rangle = \alpha_0|00\dots 000\rangle + \alpha_1|00\dots 001\rangle + \alpha_2|00\dots 010\rangle + \dots + \alpha_{(2^n-1)}|11\dots 111\rangle \tag{4}$$

where the normalization condition must be fulfilled:

$$\sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1, \quad \alpha_i \in \mathbb{C}. \tag{5}$$

To perform the calculations on quantum states, we use quantum gates. Quantum gates have to preserve a quantum state—that means the performed operation preserves the state’s normalization condition—so they have to be unitary operators to ensure this feature.

The gate which is very useful in many algorithms is a Hadamard gate H . Let $|x\rangle$ be a n -qubit state as in example Eq. (3), but with labeled qubits:

$$|x\rangle = |x_0\rangle \otimes |x_1\rangle \otimes \dots \otimes |x_{n-2}\rangle \otimes |x_{n-1}\rangle. \tag{6}$$

The impact of Hadamard gate on state $|x\rangle$ is

$$\begin{aligned} H|x\rangle &= H(|x_0\rangle \otimes |x_1\rangle \otimes \dots \otimes |x_{n-2}\rangle \otimes |x_{n-1}\rangle) \\ &= H|x_0\rangle \otimes H|x_1\rangle \otimes \dots \otimes H|x_{n-2}\rangle \otimes H|x_{n-1}\rangle \\ &= \bigotimes_{i=0}^{n-1} H|x_i\rangle = \frac{1}{\sqrt{2^n}} \left[\bigotimes_{i=0}^{n-1} (|0\rangle + (-1)^{x_i}|1\rangle) \right]. \end{aligned} \tag{7}$$

As we can see the Hadamard gate makes all absolute values of state’s amplitudes even (equal to $\frac{1}{\sqrt{2^n}}$), so it causes the phenomenon of superposition in a quantum register.

It is important to remember that quantum gates, in contrast to classical ones, always have the same number of inputs and outputs. It is so because they were designed as reversible operators the matrix form of quantum gate is unitary and Hermitian.

Another basic gate is an exclusive-or (XOR) gate, called also a controlled negation ($CNOT$) gate. This gate has two entries and, naturally, two outputs. The operation XOR is realized on the second qubit (the first qubit remains unchanged):

$$\begin{aligned} XOR|00\rangle &= |00\rangle, \quad XOR|01\rangle = |01\rangle, \\ XOR|10\rangle &= |11\rangle, \quad XOR|11\rangle = |10\rangle. \end{aligned} \tag{8}$$

The matrix form of this gate is

$$XOR = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{9}$$

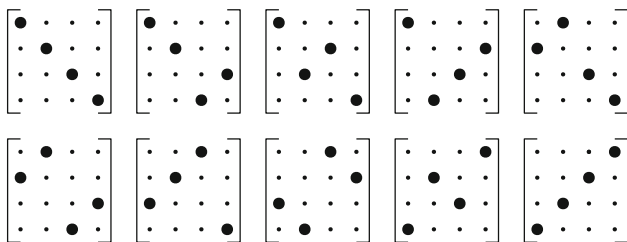
3 The k nearest neighbors algorithm

In this work, we would like to use a quantum circuit to check if some matrices are Hermitian or not. The approach presented in this section is based on the k Nearest Neighbors (kNN) algorithm for pattern classification [9]. As an example, we chose the matrices which are binary and unitary, sized 4×4 as in Fig. 1. In this case, all analysed matrices are column (or row, interchangeably) permutations of identity matrix $I_{4 \times 4}$. Let us remind that a matrix U is Hermitian when it is equal to its conjugate transpose:

$$U = U^\dagger. \tag{10}$$

Because the analysed set of matrices does not contain any matrices with complex elements, it would be some simplification to say that a Hermitian matrix is just equal to its transpose what can be observed in Fig. 1.

Hermitean matrices



Non-hermitean matrices

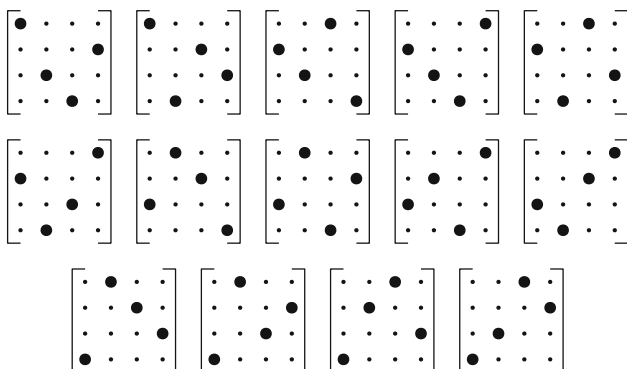


Fig. 1 Set of all binary and unitary matrices sized 4×4 , divided into Hermitian and non-Hermitian matrices. The bigger dots stand for 1 and the smaller dots for 0

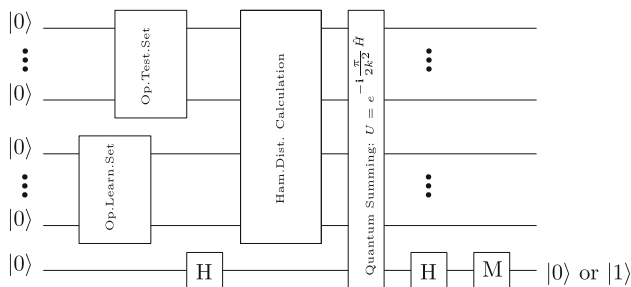


Fig. 2 Quantum circuit classifying Hermitian and non-Hermitian matrices

The quantum circuit to perform a Hermitian matrices’ recognition is shown in Fig. 2. Although in this work, we only test a circuit for matrices sized 4×4 , the presented circuit is universal in terms of matrix dimensions.

To describe the matrices, we use 16-element binary series which are built of matrix elements row by row:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \rightarrow 1000|0010|0001|0100 \rightarrow 1000001000010100. \tag{11}$$

In carried out computational experiment matrices 4×4 were taken into account, so the number of quantum circuit’s (Fig. 2) inputs/outputs is 33. In general, if the matrices are sized $k \times k$, the number of inputs/outputs is $2k^2 + 1$. The first k^2 inputs serve to enter the succeeding elements of testing set. The next k^2 inputs serve to constitute a learning set. There is also one ancilla input on which the Hadamard operation is performed.

First, we would like to describe a block labeled $Op.Learn.Set$. Its role in the circuit is to accumulate the elements of learning set. The learning set contains binary series describing Hermitian matrices. We can roughly call these series permutative—the number of ones is similar (e.g., for unitary matrices is always equal to 4), but they occupy other positions in series. It means that the block $Op.Learn.Set$ is a subsystem built of gates XOR and H , because the XOR gates ensure the permutations and H gates allow to accumulate some different series in qubits of quantum register.

The block $Op.Test.Set$ is also constructed of XOR and H gates, but it differs from the subsystem $Op.Learn.Set$. This block contains only one element of testing set, representing Hermitian or non-Hermitian matrix, in each computational experiment. It means that, in case of unitary matrices 4×4 , we have to perform 24 experiments, always with different block $Op.Test.Set$.

Summarizing this part of the circuit, we have the initial quantum state entering the system:

$$|\psi_0\rangle = |0\rangle^{\otimes 2k^2+1}. \tag{12}$$

Then, the first k^2 qubits will be affected by the block $Op.Test.Set$ and successive k^2 qubits by the block $Op.Learn.Set$ to produce the superposition of elements from a learning set:

$$\frac{1}{\sqrt{L}} \sum_{p=1}^L |l_1^p, \dots, l_{k^2}^p\rangle, \tag{13}$$

where L denotes the number of elements in learning set and l represents the successive qubits.

The first k^2 qubits describe an element from a testing set. It means that after the operations caused by blocks $Op.Learn.Set$, $Op.Test.Set$ and Hadamard gate on ancilla qubit the system’s state is

$$|\psi_1\rangle = |t_1, \dots, t_{k^2}\rangle \otimes \frac{1}{\sqrt{L}} \left(\sum_{p=1}^L |l_1^p, \dots, l_{k^2}^p\rangle \right) \otimes \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \tag{14}$$

where t stands for the qubits of element from the testing set.

The next block serves to calculate the Hamming distance between one test series and elements of learning set. The Hamming distance is a measure expressing the difference between two series. It tells how many elements, occupying the same positions, differ from each other. The block `Ham. Dist. Calculation` uses the *XOR* operations on every couple of qubits (t_i, l_i^p) and saves their results as d in the part of register which previously contained elements of a learning set:

$$XOR(t_i, l_i^p) = (t_i, d_i^p), \quad i = 1, \dots, k^2. \tag{15}$$

After this operation, the system’s state is

$$|\psi_2\rangle = |t_1, \dots, t_{k^2}\rangle \otimes \frac{1}{\sqrt{L}} \left(\sum_{p=1}^L |d_1^p, \dots, d_{k^2}^p\rangle \right) \otimes \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle). \tag{16}$$

If the differences are already computed, then they must be summed to obtain the Hamming distance. The block `Quantum Summing` stands for the operation U :

$$U = e^{-i\frac{\pi}{2k^2}\hat{H}}, \quad \hat{H} = I_{k^2 \times k^2} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}^{\otimes k^2} \otimes \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{17}$$

where i represents the imaginary unit. This results with the state $|\psi_3\rangle$:

$$|\psi_3\rangle = \frac{1}{\sqrt{2L}} \sum_{p=1}^L \left(e^{i\frac{\pi}{2k^2}d(t, l^p)} |t_1, \dots, t_{k^2}\rangle \otimes |d_1^p, \dots, d_{k^2}^p\rangle \otimes |0\rangle + e^{-i\frac{\pi}{2k^2}d(t, l^p)} |t_1, \dots, t_{k^2}\rangle \otimes |d_1^p, \dots, d_{k^2}^p\rangle \otimes |1\rangle \right). \tag{18}$$

The last step is designed to reverse the Hadamard operation on the last qubit (of course with use of reversible H gate). That will allow to measure the last qubit in the standard basis and obtain $|0\rangle$ or $|1\rangle$ with sufficiently high probability. The final state of the whole system is

$$|\psi_4\rangle = \frac{1}{\sqrt{L}} \sum_{p=1}^L \left(\cos\left(\frac{\pi}{2k^2}d(t, l^p)\right) |t_1, \dots, t_{k^2}\rangle \otimes |d_1^p, \dots, d_{k^2}^p\rangle \otimes |0\rangle + \sin\left(\frac{\pi}{2k^2}d(t, l^p)\right) |t_1, \dots, t_{k^2}\rangle \otimes |d_1^p, \dots, d_{k^2}^p\rangle \otimes |1\rangle \right). \tag{19}$$

Now, the measurement needs to be done only on the last ancilla qubit. If the result is $|0\rangle$ that means the tested matrix

is close to the pattern described in the learning set (is Hermitian), and if the result is $|1\rangle$, the matrix should not be Hermitian.

It should be emphasized that the quantum implementation of kNN method has a lower computational complexity in comparison with kNN method running on a classical computer. The first difference is connected with the phase of learning—in the classical approach, the complexity is affected by the number and length of series included in the learning set, while in quantum system, only the length k^2 of learning series influences its complexity. The second advantage of quantum approach in the field of complexity is an ability to compare a tested element with all elements from learning set simultaneously (a phenomenon of quantum parallelism takes place during the calculation of Hamming distances). Finally, the complexity of quantum kNN is: $O(k^2 + Hc)$, where Hc stands for the complexity of Hamiltonian (17) simulation. The simulation of Hamiltonian [3] may realized with no greater complexity than $O(td\|H\|_{\max} + \frac{\log(1/\epsilon)}{\log \log(1/\epsilon)})$, where t is a interval of time, d is a sparse of Hamiltonian, $\|H\|_{\max}$ is a value of Hamiltonian’s maximal element, and ϵ stands for accuracy.

4 Support vector machine

Support Vector Machine (SVM) is a method used in a field of supervised learning. Let us assume that we have a learning set containing vectors from two different classes: C_1 and C_2 . The SVM’s goal is to find a hyperplane $g(\mathbf{x})$:

$$g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0 \tag{20}$$

separating vectors from C_1 and C_2 :

$$\begin{aligned} g(\mathbf{x}) &\geq 1, \quad \forall \mathbf{x} \in C_1 \\ g(\mathbf{x}) &\leq -1, \quad \forall \mathbf{x} \in C_2 \end{aligned} \tag{21}$$

In addition, the hyperplane must be calculated in a way maximizing the margin z , which is a distance between a hyperplane and objects from C_1 and C_2 :

$$z = \frac{|g(\mathbf{x})|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|} \tag{22}$$

That means an aim is to minimize $\|\mathbf{w}\|$. This task may be formulated as finding a solution of Lagrange function $L(\lambda)$ using Karush–Kuhn–Tucker conditions:

$$L(\lambda) = \sum_{i=1}^M g_i(\mathbf{x})\lambda_i - \frac{1}{2} \sum_{i,j=1}^M \lambda_i K_{i,j}\lambda_j \tag{23}$$

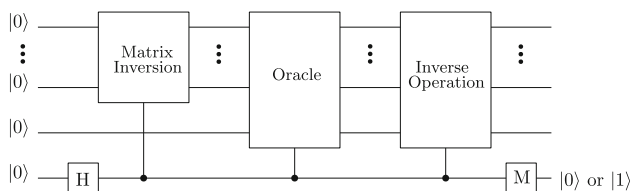


Fig. 3 Scheme of quantum circuit implementing the SVM method

under constraints:

$$\sum_{i=1}^M \lambda_i = 0 \text{ and } g_i(\mathbf{x})\lambda_i \geq 0, \tag{24}$$

where M is a number of training examples, vector λ stands for the Lagrange multipliers and K is a kernel matrix $K_{i,j} = \mathbf{x}_i^T \cdot \mathbf{x}_j$. Then, the machine should be able to classify the unknown vectors according to (21).

The computational complexity of SVM is polynomial and depends on the dimensions of classified vectors and the number of elements in the training set. Of course, using the quantum version of this method, the complexity may be reduced to logarithmic. The speedup is achieved utilizing a quantum circuit to calculate in parallel the inner products of the vectors. Then, the SVM training is converted to an approximate least-squares problem which may be solved by the quantum matrix inversion algorithm with $O(\kappa^2 \log X)$, where κ stands for a condition number of a X -equation system [7,9–11].

The process of quantum calculations may be shown as in Fig. 3 according to [18]. The presented circuit has $(N + 2)$ inputs/outputs, where N is a number of elements in classified vectors. Solving (23) leads to a least-square approximation of calculating matrix $F_{M+1 \times M+1}$, because the SVM parameters are determined by

$$(w_0, \boldsymbol{\lambda}^T)^T = F^{-1}(0, g(\mathbf{x})^T)^T, \tag{25}$$

so the first block performs the matrix F inversion to calculate the hyperplane $g(\mathbf{x})$, but it also accumulates the values of training set, because for matrix F , it is possible to calculate its eigenvectors $|u_j\rangle$ and corresponding eigenvalues α_j ($j = 1, \dots, (M + 1)$). On the first N qubits of the computational register phase estimation generates the state storing training data. After the operation performed by the block `MatrixInversion`, we will obtain the state:

$$|w_0, \boldsymbol{\lambda}\rangle = \frac{1}{\sqrt{C}} \left(w_0|0\rangle + \sum_{i=1}^M \lambda_i|i\rangle \right) \tag{26}$$

where

$$C = w_0^2 + \sum_{i=1}^M \lambda_i^2. \tag{27}$$

The training data `Oracle` causes that all eigenvectors $|\hat{u}\rangle$ of matrix F :

$$|\hat{u}\rangle = \frac{1}{\sqrt{N_{\hat{u}}}} \left(w_0|0\rangle|0\rangle + \sum_{i=1}^M \lambda_i|\mathbf{x}_i||i\rangle|\mathbf{x}_i\rangle \right) \tag{28}$$

where

$$N_{\hat{u}} = w_0^2 + \sum_{i=1}^M \lambda_i^2 |\mathbf{x}_i|^2. \tag{29}$$

The last block is to calculate the probability telling if an arbitrary vector $|\mathbf{t}\rangle$ from the training set may be classified to C_1 or C_2 :

$$|\mathbf{t}\rangle = \frac{1}{\sqrt{N_{\mathbf{t}}}} \left(|0\rangle|0\rangle + \sum_{i=1}^M |\mathbf{t}||i\rangle|\mathbf{t}\rangle \right) \tag{30}$$

with

$$N_{\mathbf{t}} = M|\mathbf{t}|^2 + 1. \tag{31}$$

The final state of the system is

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|\hat{u}\rangle|0\rangle + |\mathbf{t}\rangle|1\rangle) \tag{32}$$

and the measurement on the last qubit has to be performed. If the obtained value is $|1\rangle$, it means that the vector $|\mathbf{t}\rangle$ belongs to the class C_1 , and if the value of the last qubit is $|0\rangle$, then the vector belongs to C_2 .

5 Results of the experiments

We conducted a computational experiments to examine if a quantum system is able to recognize the pattern of Hermitian and non-Hermitian matrices sized 4×4 . The calculations for kNN were based on Eq. (19) and the calculations for SVM on (28) and (30). The results presented in this section were delivered by a script written in Python programming language.

The first experiments were conducted on sets of unitary matrices described as 16-element binary series as in (11). The results of these computations are presented in Table 1 (the numbers were rounded to four decimal places). For kNN method, the learning set contains the Hermitian matrices.

Table 1 Results of pattern recognition for quantum versions of kNN and SVM methods

Hermitian matrices in TS			non-Hermitian matrices in TS		
kNN		SVM	kNN		SVM
$P_{ 0\rangle}$	$P_{ 1\rangle}$	P	$P_{ 0\rangle}$	$P_{ 1\rangle}$	P
0.6207	0.3792	0.5000	0.6472	0.3527	0.4999
0.6972	0.3027	0.5427	0.6826	0.3173	0.4430
0.6207	0.3792	0.5569	0.6826	0.3173	0.4430
0.6972	0.3027	0.5569	0.6472	0.3527	0.5000
0.6207	0.3792	0.5569	0.6472	0.3527	0.4999
0.6972	0.3027	0.5568	0.6826	0.3173	0.4430
0.6972	0.3027	0.5569	0.6472	0.3527	0.4858
0.6972	0.3027	0.5427	0.6826	0.3173	0.4430
0.6972	0.3027	0.5568	0.6826	0.3173	0.4430
0.7621	0.2378	0.5000	0.6472	0.3527	0.4858
			0.6472	0.3527	0.5000
			0.6826	0.3173	0.4430
			0.6826	0.3173	0.4430
			0.6826	0.3173	0.5000

The learning set contains only Hermitian matrices. Abbreviation TS stands for the test set. The matrices 4×4 were written as 16-element series

As we can see the Hermitian matrices were recognized with a quite high probability. The problem is that also the non-Hermitian matrices were recognized as patterns similar to patterns in learning set—the kNN method uses the Hamming distance to distinguish between series and all used series contain 4 elements equal to one and 12 elements equal to zero, so probably, the series were treated as similar by the utilized method.

For the SVM method, we assume that Hermitian matrices belong to the class C_1 and non-Hermitian to C_2 . The algorithm for SVM was implemented to point the class with value P : if $P > 0.5$ the matrix belongs to C_1 and if $P < 0.5 - C_2$. In comparison with the SVM method allowed to obtain better results. However, there are 2 Hermitian and 3 non-Hermitian matrices with $P = 0.5000$, so the method was not able to classify these matrices unambiguously.

To improve the pattern recognition, we propose to label the matrices and run the calculations using labels. The labels should be constructed as feature vectors in a way increasing the Hamming distances between Hermitian and non-Hermitian matrices.

The labels are 16-bit binary series to provide a unique label for every binary matrix sized 4×4 . The labels are ordered: the first label contains 16 zeros, next 16 labels contain 1 figure one and 15 zeros, subsequent series contain 2 ones, 3 ones, and so on until we obtain a series of 16 ones. In every group of labels with the same number of ones, the series are sorted ascending in the meaning of the binary number

Table 2 Results of pattern recognition for quantum versions of kNN and SVM methods

Hermitian matrices in TS			non-Hermitian matrices in TS		
kNN		SVM	kNN		SVM
$P_{ 0\rangle}$	$P_{ 1\rangle}$	P	$P_{ 0\rangle}$	$P_{ 1\rangle}$	P
0.8898	0.1101	0.5634	0.7562	0.2437	0.5452
0.9223	0.0776	0.5634	0.7562	0.2437	0.5452
0.9223	0.0776	0.5872	0.8319	0.1680	0.5059
0.9223	0.0776	0.5820	0.8373	0.1626	0.4482
0.9223	0.0776	0.6097	0.6913	0.3086	0.5452
0.9440	0.0559	0.5634	0.7237	0.2762	0.5358
0.9007	0.0992	0.5634	0.6696	0.3303	0.5144
0.9332	0.0667	0.5634	0.6696	0.3303	0.5340
0.8790	0.1209	0.5905	0.7056	0.2943	0.4897
0.8790	0.1209	0.6047	0.7056	0.2943	0.5404
			0.7056	0.2943	0.4675
			0.7014	0.2943	0.5405
			0.6696	0.3303	0.5405
			0.7056	0.2943	0.5405

The learning set contains only Hermitian matrices. Abbreviation TS stands for the test set. The matrices 4×4 were encoded as 16-element feature vectors

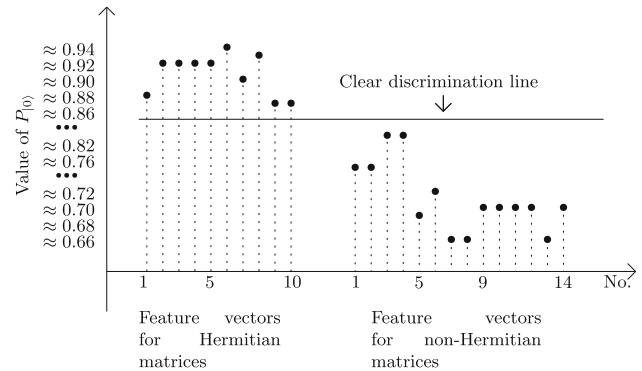


Fig. 4 Discrimination line for values of probability for the kNN method when matrices are encoded with feature vectors

written in a series. Let us assign the matrices to labels: at the beginning, the set of series presenting matrices is sorted in the same way as labels; we extract all 1024 Hermitian matrices from the set of series and we assign them to the first 1024 labels; other non-Hermitian matrices we assign to remaining labels. This approach allows to decrease the number of ones in labels corresponding to Hermitian matrices and to increase the number of ones in labels corresponding to non-Hermitian matrices. Table 2 contains the results of learning using labels.

We can see that the kNN method still tends to classify all matrices as Hermitian, but it is possible to calculate a discrimination line (see Fig. 4) between values of $P_{|0\rangle}$ for Hermitian and non-Hermitian matrices, because the lowest

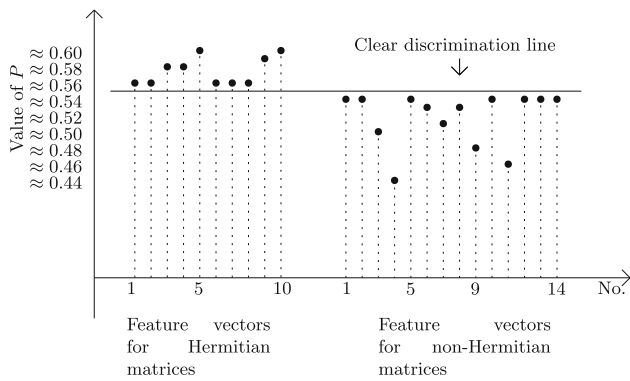


Fig. 5 Discrimination line for values of probability for the SVM method when matrices are encoded with feature vectors

value of probability for Hermitian matrices is 0.8790 and the highest value of $P_{|0\rangle}$ for non-Hermitian matrices is 0.8313. The same situation we can observe for the SVM method with a discrimination line (see Fig. 5) between values 0.5634 and 0.5452.

The last experiment was conducted with use of labels, but the learning and test sets were more numerous (28 Hermitian and 40 non-Hermitian matrices were used). The sets were enhanced with labels corresponding to non-unitary matrices. The results are presented in Table 3.

Increasing the number of elements in the learning set caused some improvement for kNN method—the lowest value of $P_{|0\rangle} = 0.8859$ for Hermitian matrices and the highest value of $P_{|0\rangle} = 0.8011$ for non-Hermitian matrices. However, this strategy did not improve the results for SVM method.

Of course, it should be emphasized that the presented approach with use of quantum versions of kNN and SVM has a probabilistic character. That implies the fact that the calculations have to be repeated to obtain the most accurate approximation of distribution shown in tables above. The probabilities of measuring $|0\rangle$ or $|1\rangle$ on the last qubit are different, so it allows to indicate the distance of analysed case from the learning set. In general, the computations illustrated by the quantum circuits should be performed $O(P(1-P)\frac{1}{\epsilon^2})$ times to obtain probability P distribution with accuracy ϵ .

6 Summary

The computational experiments showed that the quantum circuits are able to distinguish between the pattern of Hermitian and non-Hermitian matrices.

In case of kNN, the way of encoding patterns is very important. The kNN method uses Hamming distance to distinguish between the matrices so using feature vectors to

Table 3 Results of pattern recognition for quantum versions of kNN and SVM methods

Hermitian matrices in TS			non-Hermitian matrices in TS		
kNN	SVM	P	kNN	SVM	P
$P_{ 0\rangle}$	$P_{ 1\rangle}$		$P_{ 0\rangle}$	$P_{ 1\rangle}$	
0.8859	0.1140	0.5596	0.7468	0.2531	0.5425
0.9183	0.0816	0.5596	0.7865	0.2134	0.5425
0.9118	0.0881	0.5596	0.8011	0.1988	0.4696
0.9118	0.0881	0.5596	0.7856	0.2143	0.5359
0.9002	0.0997	0.5823	0.7459	0.2540	0.4942
0.9079	0.0920	0.5596	0.7451	0.2548	0.5425
0.8898	0.1101	0.5596	0.7488	0.2511	0.5380
0.9040	0.0959	0.5596	0.7488	0.2511	0.4749
0.9028	0.0971	0.5810	0.7331	0.2668	0.5380
0.8963	0.1036	0.5596	0.7331	0.2668	0.4993
0.9704	0.0295	0.8530	0.7046	0.2953	0.5376
0.9446	0.0553	0.5832	0.7016	0.2983	0.5380
0.9446	0.0553	0.5832	0.7040	0.2959	0.5373
0.9473	0.0526	0.5832	0.7046	0.2953	0.4857
0.9366	0.0633	0.5832	0.7961	0.2038	0.5425
0.9313	0.0686	0.5832	0.7509	0.2490	0.5425
0.9313	0.0686	0.5832	0.7153	0.2846	0.5380
0.9313	0.0686	0.5981	0.7205	0.2794	0.5380
0.9145	0.0854	0.5596	0.7385	0.2614	0.5380
0.9079	0.0920	0.5596	0.7385	0.2614	0.5380
0.9079	0.0920	0.5596	0.6575	0.3424	0.5347
0.9079	0.0920	0.5654	0.6823	0.3176	0.5347
0.9079	0.0920	0.5831	0.6362	0.3637	0.5347
0.8975	0.1024	0.5596	0.6376	0.3623	0.5347
0.8975	0.1024	0.5983	0.5235	0.4764	0.5322
0.9079	0.0920	0.5596	0.5502	0.4497	0.5322
0.9079	0.0920	0.5964	0.5578	0.4421	0.5322
0.8975	0.1024	0.5853	0.5230	0.4769	0.5322
			0.5710	0.4289	0.5322
			0.5641	0.4358	0.5322
			0.5443	0.4556	0.5322
			0.4556	0.5443	0.5301
			0.4145	0.5854	0.5301
			0.4423	0.5576	0.5284
			0.4352	0.5647	0.5284
			0.4765	0.5234	0.5284
			0.4502	0.5497	0.5284
			0.2317	0.7682	0.5246
			0.1990	0.8009	0.5246
			0.1115	0.8884	0.5228

Abbreviation TS stands for the test set. The binary matrices (unitary and non-unitary) were encoded as feature vectors

describe the recognized objects should comply the character of Hamming distance measure. The distances between objects from the same class should be as small as possible. On the other side, distances between objects belonging to other classes should be increased. In our experiments, we used the system of labels to describe the matrices. Hermitian matrices were described by vectors with the minimal needed number of ones and non-Hermitian matrices with maximal number of ones. In case of feature vectors for unitary Hermitian matrices, the number of ones is equal to 2 and for non-Hermitian matrices: 4 and 5. The pattern recognition was more efficient when the learning set was extended by non-unitary matrices. The conclusion is obvious: not only the number of ones is an important factor for Hamming distance, but also the position of figures in series and non-unitary matrices brought some diversity in this aspect (there may be Hermitian matrices with 0 to 4 figures equal to one and non-Hermitian with 4–16 figures equal to one). The similar results were presented in [9].

The SVM method was surprisingly efficient even for a small learning set and series generated directly from matrices as in (11). Utilizing the feature vectors allowed to introduce a clear discrimination line between the values of P pointing objects from the classes C_1 and C_2 .

It should also be emphasized that the character of quantum computations is probabilistic and conducting more experiments would let to describe the obtained values with higher accuracy.

The further work on the idea of using quantum circuit as a classifier should develop in direction of deeper analysis of Hamiltonian \hat{H} used in (19)—especially to formulate the matrix U given there. The other issue is to simulate the behaviour of quantum circuit and check if the averaged results will overlap with calculated values of probability from Tables 1, 2, 3. Finally, looking for the system of describing matrices by feature vectors to increase pattern recognition would be also very interesting.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Cortes, C., Vapnik, V.: Support-vector networks. *Mach. Learn.* **20**(3), 273–297 (1995)
2. Fix, E., Hodges, J.L.: Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties. Technical Report 4, USAF School of Aviation Medicine, Randolph Field, Texas (1951)
3. Low, G.H., Chuang, I.L.: Optimal Hamiltonian Simulation by Quantum Signal Processing. [arXiv:1606.02685v2](https://arxiv.org/abs/1606.02685v2) [quant-ph] (2016)
4. Mateus, P., Omar, Y.: Quantum Pattern Matching. [arXiv:quant-ph/0508237v1](https://arxiv.org/abs/quant-ph/0508237v1) (2005)
5. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge University Press, New York (2000)
6. Pinkse, P.W.H., Goorden, S.A., Horstmann, M., Škorić, B., Mosk, A.P.: Quantum pattern recognition. 2013 Conference on and International Quantum Electronics Conference Lasers and Electro-Optics Europe (CLEO EUROPE/IQEC), Munich, pp. 1–1 (2013)
7. Rebstrost, P., Mohseni, M., Lloyd, S.: Quantum support vector machine for big data classification. *Phys. Rev. Lett.* **113**, 130503 (2014)
8. Schaller, G., Schtzhold, R.: Quantum algorithm for optical-template recognition with noise filtering. *Phys. Rev. A* **74**, 012303 (2006)
9. Schuld, M., Sinayskiy, I., Petruccione, F.: Quantum computing for pattern classification. In: 13th Pacific Rim International Conference on Artificial Intelligence (PRICAI) and Also Appear in the Springer Lecture Notes in Computer Science 8862 (2014)
10. Schuld, M., Sinayskiy, I., Petruccione, F.: An introduction to quantum machine learning. *Contemp. Phys.* **52**, 172–185 (2014). <https://doi.org/10.1080/00107514.2014.964942>
11. Senekane, M., Taelle, B.M.: Prediction of solar irradiation using quantum support vector machine learning algorithm. *Smart Grid Renew. Energy* **7**, 293–301 (2016). <https://doi.org/10.4236/sgr.2016.712022>
12. Trugenberger, C.A.: Quantum pattern recognition. *Quantum Inf. Process.* **1**(6), 471–493 (2002)
13. Trugenberger, C.A.: Phase transitions in quantum pattern recognition. *Phys. Rev. Lett.* **89**, 277903 (2002)
14. Wiebe, N., Kapoor, A., Svore, K.M.: Quantum algorithms for nearest-neighbor methods for supervised and unsupervised learning. *Quantum Inf. Comput.* **15**(3–4), 316–356 (2015)
15. Winiewska, J., Sawerwain, M.: Recognizing the pattern of binary hermitian matrices by a quantum circuit. *Intelligent Information and Database Systems. 9th Asian Conference, ACIIDS 2017, Proceedings, Part I*, pp. 466–475 (2017)
16. Yoo, S., Bang, J., Lee, C., Lee, J.: A quantum speedup in machine learning: finding an N-bit Boolean function for a classification. *New J. Phys.* **16**(10), 103014 (2014)
17. Ruan, Y., Chen, H., Tan, J., Li, X.: Quantum computation for large-scale image classification. *Quantum Inf. Process.* **15**(10), 4049–4069 (2016)
18. Zhaokai, L., Xiaomei, L., Nanyang, X., Jiangfeng, D.: Experimental realization of quantum artificial intelligence. [arXiv: 1410.1054v1](https://arxiv.org/abs/1410.1054v1) [quant-ph] (2014)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.