Priscila Gabriele Marques dos Santos

QUANTUM ENHANCEMENTS FOR MACHINE LEARNING BASED ON A PROBABILISTIC QUANTUM MEMORY

Dissertação de Mestrado

Universidade Federal Rural de Pernambuco
secretaria@preg.ufrpe.br
http://www.ufrpe.br/br/graduacao

RECIFE
2019

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema Integrado de Bibliotecas da UFRPE
Biblioteca Central, Recife-PE, Brasil

S237q Santos, Priscila Gabriele Marques dos
Quantum enhancements for machine learning based on a probabilistic quantum memory / Priscila Gabriele Marques dos Santos. – 2019.
55 f. : il.

Orientador: Adenilton José da Silva.
Dissertação (Mestrado) – Universidade Federal Rural de Pernambuco, Programa de Pós-Graduação em Informática Aplicada, Recife, BR-PE, 2019.
Inclui referências.

1. Computação quântica 2. Aprendizado do computador 3. Redes neurais (Computação) 4. Algorítmos computacionais I. Silva, Adenilton José da, orient. II. Título

CDD 004

Universidade Federal Rural de Pernambuco
Departamento de Estatística e Informática
Programa de Pós Graduação em Informática Aplicada

Priscila Gabriele Marques dos Santos

QUANTUM ENHANCEMENTS FOR MACHINE LEARNING BASED ON A PROBABILISTIC QUANTUM MEMORY

Dissertação de Mestrado apresentada ao Programa de Pós Graduação em Informática Aplicada do Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco como requisito parcial para obtenção do grau de Mestre em Informática Aplicada.

Orientador: *Adenilton José da Silva*

RECIFE
2019

Dissertação de Mestrado apresentada por **Priscila Gabriele Marques dos Santos** ao Programa de Pós Graduação em Informática Aplicada do Departamento de Estatística e Informática da Universidade Federal Rural de Pernambuco sob o título **Quantum enhancements for Machine Learning based on a Probabilistic Quantum Memory**, orientada pelo **Prof. Adenilton José da Silva** e aprovada pela banca examinadora formada pelos professores:

RECIFE
2019

Agradecimentos

Em especial, aos meus pais, para os quais não há agradecimentos suficientes que expressem o tamanho da minha gratidão. Também, não poderia deixar de mencionar o companheirismo de Izabella de Moraes e Lara Cardoso. Agradeço enormemente a Rodrigo Sousa, namorado, cujo apoio e suporte me impulsionaram sempre adiante.

Ao professores da UFRPE. Sobretudo, ao professor Adenilton Silva, que me orientou nesse trabalho, sempre incentivando minha autonomia nas pesquisas, receptivo a seguir novos rumos e mudanças de escopo. Agradeço todas as sugestões e conselhos.

Aos colegas de mestrado e do grupo de computação quântica da UFRPE. Em especial a Rodrigo Sousa e Ismael Cesar, agradeço as colaborações nas pesquisas, as discussões e debates.

Por fim, à FACEPE (Fundação de Amparo à Ciência e Tecnologia de Pernambuco), que me concedeu apoio financeiro durante o mestrado. Agradeço também ao instituto Serrapilheira, que financiou as pesquisas do nosso grupo de computação quântica.

No man ever steps in the same river twice, for it's not the same river and he's not the same man.

—HERACLITUS

Resumo

A aprendizagem de máquina quântica surge a partir da interação das áreas de aprendizagem de máquina e computação quântica. Aprendizagem de máquina é um ramo da inteligência artificial de impacto em diversas áreas que provê aos computadores a habilidade de aprender de maneira autônoma a partir de experiências. A computação quântica, por outro lado, é um diferente paradigma computacional. O processamento de informação e comunicação em um computador quântico faz uso de princípios e propriedades da mecânica quântica, obtendo efeitos computacionais que não podem ser realizados eficientemente em computadores clássicos. A computação quântica levanta novas possibilidades a partir de abordagens promissoras que fazem uso desses efeitos. De fato, propostas de algoritmos quânticos demonstram seu potencial em superar a eficiência dos algoritmos clássicos em algumas tarefas.

O presente trabalho busca contribuir com o campo de aprendizagem de máquina quântica. Para tanto, foi investigado o uso e as aplicações de uma memória probabilística quântica como ferramenta para propor algoritmos de aprendizagem de máquina melhorados. Aqui, a memória quântica foi utilizada para desenvolver procedimentos melhorados para as tarefas de validação cruzada, seleção e avaliação de arquiteturas de redes neurais artificiais. Além disso, um modelo de rede neural sem peso que utiliza a memória quântica foi avaliado e melhorado.

Palavras-chave: computação quântica, aprendizagem de máquina, redes neurais, aprendizagem de máquina quântica, memórias quânticas, memórias quânticas probabilísticas

Abstract

Quantum machine learning arises from the interaction of fields of machine learning and quantum computing. Machine learning is a branch of artificial intelligence relevant in many areas. It provides computers the ability to learn autonomously from experience. Quantum computing, on the other hand, is a different computational paradigm. The processing of information and communication in a quantum computer makes use of the principles and properties of quantum mechanics. With this, it is possible to achieve computational effects that cannot be efficiently reached classically. Quantum computing raises new possibilities through promising approaches that make use of these effects. In fact, proposed quantum algorithms demonstrate their potential in outperforming classical algorithms in some tasks.

The present work aims to contribute with the field of quantum machine learning. In order to do so, the use and applications of a quantum probabilistic memory as a tool to propose improved machine learning algorithms is investigated. Here, the quantum memory is used to develop improved procedures for tasks such as cross-validation, and the selection and evaluation of artificial neural network architectures. In addition, a weightless neural network model using the probabilistic quantum memory was evaluated and improved.

Keywords: quantum computing, machine learning, neural networks, quantum machine learning, quantum memory, probabilistic quantum memory

Sumário

1 Introduction

This work delves into the capabilities and applications of a Probabilistic Quantum Memory (TRUGENBERGER, 2001) for machine learning methods. Machine Learning, as a field of study, investigates and develops automated methods for data analysis with the objective of building learning models capable of self-improvement from processing new data with none or very little human interference (NASRABADI, 2007).

The field of Machine Learning (ML) (BISHOP, 2006) can be divided into three main branches, each representing a different learning method. Supervised learning, which deals with labeled data and is used in classification and regression tasks; unsupervised learning, works with unlabeled data and is used for data clustering; and reinforcement learning, a method where the learning is guided by a reward and punishment system. Machine Learning nowadays is seeing increasingly more applications in industry, business, security, medical diagnostics, consumer products, etc. o This work focus on supervised learning tasks, in which all data samples used are labeled with their correct classification. A common practice for evaluating a model involves dividing the available data into disjoint training and test datasets. The model learns only from the samples in the training dataset and, afterwards, its performance can be accessed by using it to classify the test samples and calculate classification accuracy and other metrics. The method known as k-fold cross-validation (STONE, 1974; GEISSER, 1975) is an evaluation method commonly used in machine learning. In this method, the data is divided into k disjoint groups or folds, where each fold is used as a test dataset to evaluate a classifier trained by the remaining $k-1$ folds.

A Probabilistic Quantum Memory (PQM) is an associative information storage model proposed in TRUGENBERGER (2002, 2001). The PQM implements a quantum memory capable of storing binary patterns on qubits in a uniform quantum superposition. This quantum superposition can hold all possible patterns for a n-qubits quantum memory register. In order to retrieve some pattern from the memory an input pattern is passed to the PQM retrieval algorithm which calculates the Hamming distance from the given input to all the stored patterns. This procedure works by rotating the memory state based on the given input which results in increasing the amplitudes of the patterns most close to it, consequently increasing the memory probability of outputting the desired pattern as well.

The PQM model presents certain advantages over classical memory models. Storing and easily retrieving information is a critical process in light of the huge quantities of data being generated nowadays (PROVOST; FAWCETT, 2013). Traditional memory architectures retrieve information by knowing the memory address, where the data is stored. Therefore, to store in this address-oriented systems it is necessary to worry about the stored information location and corrupted or incomplete inputs cannot be retrieved (MÜLLER; REINHARDT; STRICKLAND, 2012). An associative memory, or content-addressable memory, is a model that retrieves stored information based on its content instead of its storage location. It is possible to retrieve information from these memories even with incomplete inputs. On the other hand, associative memories cannot store nearly as many patterns as the address-oriented systems due to the interference stored patterns exert to each other. The PQM is an associative memory model and, as its classical counterparts, is able to retrieve information even if the input is noisy or incomplete. Additionally, the PQM as a quantum model does not have the same constraints regarding its storage capacity, as it can store all possible n bit patterns.

The PQM as a memory model presents a severe drawback. While a quantum superposition can be used in order to achieve an optimal storage capability, it also implies that all the content stored on the memory will be lost after an input is given to the retrieval procedure. This occurs due to the quantum measurement that has to be performed on the memory in order to obtain the output, which is obtained by operating on the memory state. The PQM is said to suffer from fundamental issues (DUNJKO; BRIEGEL, 2018), which reduces its value to that of just a creative idea and of one of the first proposals involving amplitude encoding of classical data, rather than being a complete proposal. In BRUN et al. (2001) it is argued that the PQM model has poor performance and presents no storage advantage over the classical models since the same scheme could be devised classically. However, these claims are contested by TRUGENBERGER (2003), where it is pointed out that the classical probabilistic database search scheme proposed in order to simulate the PQM does not accurately represents how the PQM actually works. In SCHULD; SINAYSKIY; PETRUCCIONE (2014) the model is further examined, the authors conclude that it can be seen as an asset for pattern classification tasks but the Hamming distance constitutes as a rather imprecise approach.

In short, the PQM as a memory model has its advantages but also presents limitations. In this work, the main idea consists in using the PQM as a data structure to manipulate information in superposition. Similar approach was first seen in MENNEER; NARAYANAN (1995), where a classical model inspired on quantum computing is proposed, in which patterns are presented to a set of superposed neural networks. Quantum approaches that follow the idea of training neural networks in superposition are devised in PANELLA; MARTINELLI (2011); RICKS; VENTURA (2004). In SILVA; OLIVEIRA (2017), the PQM is used to evaluate neural network architectures. In DUNJKO; TAYLOR; BRIEGEL (2017) quantum computation and reinforcement learning are used to evolve the parameters of a classifier in superposition. In SILVA; LUDERMIR;

OLIVEIRA (2016) neural networks architectures are trained and evaluated in superposition and a nonlinear quantum operator is used to choose the best architecture.

The remainder of this chapter is organized as follows: Section 1.1 presents the motivations behind the topics researched. The objectives are displayed in Section 1.2. In Section 1.3, the papers that form this work are summarized. Finally, Section 1.4 concludes this work with the main contributions obtained and discusses possible future works.

1.1 Motivation

Nowadays, machine learning techniques are a strong component of new technologies applications and products. Several aspects of today's society are incorporating digital solutions powered by machine learning and AI techniques in what is being called the AI revolution (MAKRIDAKIS, 2017). Its great potential in solving everyday problems makes this a highly applicable and researched field which is gathering increasingly attention from governments and industry that want to gain the upper hand in this revolution. Moreover, machine learning provides ways of analyzing the relations between multiple features even in large and complex data sets. Machine learning techniques have been successfully applied to a wide range of problems, from accurate medical diagnosis and disease assessment (SHEN; WU; SUK, 2017) to mastering complex games (SILVER et al., 2016) and data mining tasks (WITTEN et al., 2016).

Quantum Computing is another field of study that is considered to be very pertinent. It consists in a computing paradigm that uses quantum mechanics as a base theory for information processing and communication (NIELSEN; CHUANG, 2002). Researches in the field are motivated by an increase in computational power that could be achieved trough this paradigm that provides a different computational approach FEYNMAN (1982). Using features such as quantum parallelism and quantum interference, quantum algorithms able to outperform the best known classical ones have been devised in DEUTSCH; JOZSA (1992); GROVER (1996); SHOR (1999). These proposals suggest that quantum computation may solve hard problems which conventional computers are simply not capable of. It is important to consider, for that matter, that quantum computing is an abstract paradigm which does not have a direct implementation in technology.

The biggest technology corporations such as IBM, Intel, and Google, are in the race to build the first universal quantum computer. Even though these devices are still out of reach, some small-scale quantum computers are currently available. These devices have a few qubits and gates at disposal. For operations involving 2 qubits or more, the device architecture and qubits connections must be taken in consideration. It is important to note that these are noisy devices and it is still a big challenge to deal with the decoherence of quantum states. Bigger quantum devices, having from 50 to 100 qubits, are expected in the next years (PRESKILL, 2018). These noisy intermediate-scale computers may outperform conventional classical ones for certain tasks, in what is being called Quantum Supremacy (HARROW; MONTANARO, 2017). Some quantum applications have been devised in BEHERA; BANERJEE; PANIGRAHI (2017); FIGGATT et al. (2017), even for the small-scale devices available nowadays.

These two fields of study, Machine Learning and Quantum Computing (or Quantum Physics in general), can greatly contribute with each other. There is a pursuit of applying machine learning to even more difficult tasks, which can demand great computational power. In fact, ML can be applied to predict or extract information about quantum systems. In MAVADIA et al. (2017), machine learning is used as a tool to predict quantum state evolution. At the same time, quantum computing holds the powerful potential of speedup over classical computation. Various works use quantum computing techniques to propose or enhance ML techniques LLOYD; MOHSENI; REBENTROST (2013); LLOYD; GARNERONE; ZANARDI (2016); WIEBE; BRAUN; LLOYD (2012).

Quantum Machine Learning (QML) studies the usage of quantum computation to enhance machine learning. Several proposals in this field exhibit quantum speedups, meaning they are able to outperform the best known classical algorithms for that specific task (BIAMONTE et al., 2017). One of the immediate applications of QML involves quantum data. In similar fashion to classical machine learning, QML finds patterns in quantum states initialized from classical data by manipulating these states through quantum operations. The Probabilistic Quantum memory used in this work is a quantum system which not only holds quantum data but is also capable of manipulating it in order to search for patterns. The PQM is used as an asset in order to develop and evaluate quantum enhancements for machine learning.

1.2 Objectives

This section states the general and specific objectives of this work.

1.2.1 General

Achieve machine learning enhancements through the usage of probabilistic quantum memories.

1.2.2 Specific

- Propose a quantum enhancement for the k-fold cross validation method;
- Devise a method based on the PQM to perform neural network architecture selection;
- Evaluate the quantum memory performance as a classifier and investigate modifications to improve its accuracy.

1.3 Outline

The remainder of this work is divided in three chapters composed by papers. The first work (DOS SANTOS et al., 2018) named Quantum enhanced k-fold cross-validation, was published on the 7th Brazilian Conference on Intelligent Systems. The second work (DOS SANTOS et al., 2018) named Quantum enhanced cross-validation for near-optimal neural networks architecture selection, was published on the International Journal of Quantum Information. The last one, named Evaluation and improvement of a Probabilistic Quantum Memory Weightless classifier, is an extended version of a recently accepted paper on the 27th European Symposium on Artificial Neural Network. All the papers composing this work follow the main objective previously stated in Section 1.2, that is, to achieve machine learning enhancements through the use of probabilistic quantum memories. This section summarizes each of the three papers which compose this work.

Chapter 2 presents a quantum method for a k-fold cross-validation with linear-speedup. The PQM is used to store and evaluate the dataset patterns in superposition. To perform the validation in all the folds, two additional quantum registers are used in order to distinguish between folds used for training and validation. With that, the learning is conducted only when the state of the registers differ. This method performs the validation with only one training execution, instead of k times. A reduced experimental analysis is conducted to evaluate the method.

Chapter 3 describes an enhanced architecture selection and evaluation method for neural networks. It consists in a hybrid quantum-classical algorithm that operates by performing an enhanced k-fold cross-validation on the dataset and training the neural networks with all the patterns while storing the results in superposition on the PQM. It has an exponential-speedup over the classical approach. A classical description of the method with a reduced number of networks in superposition is used to evaluate the algorithm in a conventional computer.

Chapter 4 evaluates a Weightless Neural Node model based on a PQM. It also proposes a modification of the model by adding a parameter to the PQM retrieval algorithm. The evaluation is performed through experiments conducted on a classical computer using the PQM algorithm description. The models performances were compared against the K-Nearest Neighbors (KNN) algorithm using public benchmark datasets. The proposed modification has surpassed the unmodified method over all the datasets and the Weightless Neural Node model has shown satisfactory results in general when compared to KNN.

1.4 Conclusion

The main contributions of this work are related to the field of quantum machine learning through the production of quantum enhanced ML techniques. More specifically, the Probabilistic Quantum Memory capabilities necessary for achieving this result were investigated and explored here. The PQM allows the manipulation of binary patterns in superposition, which can be used to improve classical machine learning methods.

In supervised learning, the k-fold cross-validation method is used to estimate a classifier performance, the probability of correctly classifying new data. Each evaluated model must be trained and tested for each fold, that is, k times. Therefore, the cost of performing a k-fold cross-validation is $O(k \times m)$, where m is the classifier cost. Using the PQM, this method was improved by performing the validation with all folds in superposition. Since all the folds are evaluated in one single step, the total cost becomes $O(m)$, achieving a linear speed-up for the k-fold cross-validation method.

The PQM can also be used to approach an NP complete task like the evaluation and selection of neural networks architectures. There is no known efficient classical method to evaluate whether a neural network architecture can learn a given task. This problem has been approached before with a polynomial algorithm using a non-linear quantum operator. Here, the problem was tackled by storing and training the neural networks in superposition and also performing the quantum enhanced k-fold cross-validation. That way, through the use of the PQM an exponential quantum speed-up for the task was achieved.

Another PQM application investigated here concerns quantum classifiers models. Here, a quantum weightless classifier model based on the PQM was evaluated using public benchmark datasets. Also, a modification to this model resulting in a considerable performance improvement was proposed and evaluated. Evaluation performed on a classical computer showed that the modified quantum weightless classifier performed better than its unmodified version in all the tested datasets.

In conclusion, despite the PQM having some drawbacks as a memory model, this work successfully contributed with new QML techniques and approaches based on it. Nevertheless, several aspects of the PQM can still be further investigated in future works. For instance, a study can be made on the influence of different distance functions in comparison to the Hamming distance. Another possible work is to investigate the application of the proposed methods to parameter selection tasks for other machine learning models. Regarding the PQM based weightless classifier, an open investigation path is the construction of the model based in different architectures. Finally, an important future work could be the realization and evaluation of the PQM in future quantum computers with enough qubits available.

Referências

BEHERA, B. K.; BANERJEE, A.; PANIGRAHI, P. K. Experimental realization of quantum cheque using a five-qubit quantum computer. **Quantum Information Processing**, [S.l.], v.16, n.12, p.312, 2017.

BIAMONTE, J. et al. Quantum machine learning. **Nature**, [S.l.], v.549, n.7671, p.195, 2017.

BISHOP, C. M. **Pattern recognition and machine learning**. [S.l.]: springer, 2006.

BRUN, T. et al. Comment on “Probabilistic Quantum Memories”. **Phys. Rev. Lett**, [S.l.], v.87, n.6, p.067901, 2001.

DEUTSCH, D.; JOZSA, R. Rapid solution of problems by quantum computation. **Proc. R. Soc. Lond. A**, [S.l.], v.439, n.1907, p.553–558, 1992.

DOS SANTOS, P. et al. Quantum enhanced k-fold cross-validation. In: BRAZILIAN CONFERENCE ON INTELLIGENT SYSTEMS (BRACIS), 2018. **Anais...** [S.l.: s.n.], 2018. p.194–199.

DOS SANTOS, P. G. et al. Quantum enhanced cross-validation for near-optimal neural networks architecture selection. **International Journal of Quantum Information**, [S.l.], v.16, n.08, p.1840005, 2018.

DUNJKO, V.; BRIEGEL, H. J. Machine learning & artificial intelligence in the quantum domain: a review of recent progress. **Reports on Progress in Physics**, [S.l.], v.81, n.7, p.074001, 2018.

DUNJKO, V.; TAYLOR, J. M.; BRIEGEL, H. J. Advances in quantum reinforcement learning. In: SYSTEMS, MAN, AND CYBERNETICS (SMC), 2017 IEEE INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2017. p.282–287.

FEYNMAN, R. P. Simulating physics with computers. **International journal of theoretical physics**, [S.l.], v.21, n.6-7, p.467–488, 1982.

FIGGATT, C. et al. Complete 3-qubit Grover search on a programmable quantum computer. **Nature communications**, [S.l.], v.8, n.1, p.1918, 2017.

GEISSER, S. The predictive sample reuse method with applications. **Journal of the American statistical Association**, [S.l.], v.70, n.350, p.320–328, 1975.

GROVER, L. K. A fast quantum mechanical algorithm for database search. In: ACM SYMPOSIUM ON THEORY OF COMPUTING. **Proceedings...** [S.l.: s.n.], 1996. p.212–219.

HARROW, A. W.; MONTANARO, A. Quantum computational supremacy. **Nature**, [S.l.], v.549, n.7671, p.203, 2017.

LLOYD, S.; GARNERONE, S.; ZANARDI, P. Quantum algorithms for topological and geometric analysis of data. **Nature communications**, [S.l.], v.7, p.10138, 2016.

LLOYD, S.; MOHSENI, M.; REBENTROST, P. Quantum algorithms for supervised and unsupervised machine learning. **arXiv preprint arXiv:1307.0411**, [S.l.], 2013.

MAKRIDAKIS, S. The forthcoming Artificial Intelligence (AI) revolution: its impact on society and firms. **Futures**, [S.l.], v.90, p.46–60, 2017.

MAVADIA, S. et al. Prediction and real-time compensation of qubit decoherence via machine learning. **Nature communications**, [S.l.], v.8, p.14106, 2017.

MENNEER, T.; NARAYANAN, A. Quantum-inspired neural networks. **Tech. Rep. R329**, [S.l.], 1995.

MÜLLER, B.; REINHARDT, J.; STRICKLAND, M. T. **Neural networks**: an introduction. [S.l.]: Springer Science & Business Media, 2012.

NASRABADI, N. M. Pattern recognition and machine learning. **Journal of electronic imaging**, [S.l.], v.16, n.4, p.049901, 2007.

NIELSEN, M. A.; CHUANG, I. **Quantum computation and quantum information**. [S.l.]: AAPT, 2002.

PANELLA, M.; MARTINELLI, G. Neural networks with quantum architecture and quantum learning. **International Journal of Circuit Theory and Applications**, [S.l.], v.39, n.1, p.61–77, 2011.

PRESKILL, J. Quantum Computing in the NISQ era and beyond. **arXiv preprint arXiv:1801.00862**, [S.l.], 2018.

PROVOST, F.; FAWCETT, T. Data science and its relationship to big data and data-driven decision making. **Big data**, [S.l.], v.1, n.1, p.51–59, 2013.

RICKS, B.; VENTURA, D. Training a quantum neural network. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS. **Anais...** [S.l.: s.n.], 2004. p.1019–1026.

SCHULD, M.; SINAYSKIY, I.; PETRUCCIONE, F. Quantum computing for pattern classification. In: PACIFIC RIM INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE. **Anais...** [S.l.: s.n.], 2014. p.208–220.

SHEN, D.; WU, G.; SUK, H.-I. Deep learning in medical image analysis. **Annual review of biomedical engineering**, [S.l.], v.19, p.221–248, 2017.

SHOR, P. W. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. **SIAM review**, [S.l.], v.41, n.2, p.303–332, 1999.

SILVA, A. J. da; LUDERMIR, T. B.; OLIVEIRA, W. R. de. Quantum perceptron over a field and neural network architecture selection in a quantum computer. **Neural Networks**, [S.l.], v.76, p.55–64, 2016.

SILVA, A. J. da; OLIVEIRA, R. L. F. de. Neural Networks Architecture Evaluation in a Quantum Computer. In: INTELLIGENT SYSTEMS (BRACIS), 2017 BRAZILIAN CONFERENCE ON. **Anais...** [S.l.: s.n.], 2017. p.163–168.

SILVER, D. et al. Mastering the game of Go with deep neural networks and tree search. **nature**, [S.l.], v.529, n.7587, p.484, 2016.

STONE, M. Cross-validatory choice and assessment of statistical predictions. **Journal of the Royal Statistical Society: Series B (Methodological)**, [S.l.], v.36, n.2, p.111–133, 1974.

TRUGENBERGER, C. Probabilistic quantum memories. **Physical Review Letters**, [S.l.], v.87, n.6, p.067901, 2001.

TRUGENBERGER, C. Quantum pattern recognition. **Quantum Information Processing**, [S.l.], v.1, n.6, p.471–493, 2002.

TRUGENBERGER, C. **Comment on"Probabilistic quantum memories-Reply**. [S.l.]: AMER PHYSICAL SOC ONE PHYSICS ELLIPSE, COLLEGE PK, MD 20740-3844 USA, 2003.

WIEBE, N.; BRAUN, D.; LLOYD, S. Quantum algorithm for data fitting. **Physical review letters**, [S.l.], v.109, n.5, p.050505, 2012.

WITTEN, I. H. et al. **Data Mining**: practical machine learning tools and techniques. [S.l.]: Morgan Kaufmann, 2016.

2 First Contribution

Quantum enhanced k-fold cross-validation

Quantum enhanced k-fold cross-validation

Priscila G. M. dos Santos, Ismael C. S. Araujo, Rodrigo S. Sousa and Adenilton J. da Silva
Departamento de Computação
Universidade Federal Rural de Pernambuco
Recife, Pernambuco, Brazil
{priscila.marques, ismael.cesar, rodrigo.silvasouza, adenilton.silva}@ufrpe.br

***Abstract*—In this work, we propose a quantum-classical algorithm able to perform a k-fold cross-validation with linear speedup. The proposed method creates a quantum superposition with patterns from a dataset and a classifier can evaluate all patterns at once. We used a probabilistic quantum memory in order to conduct the performance evaluation. The proposed method was verified through a reduced experimental analysis conducted classically.**

***Index Terms*—Quantum computing, quantum cross-validation, quantum machine learning**

I. Introduction

The advantages of quantum computing with respect to time speed-up over known classical algorithms are granted by the possibility to perform an exponential number of computations through quantum parallelism [1], [2]. There are also quantum algorithms exponentially faster than any possible deterministic classical algorithm [3]. By using quantum computation, one can search in an unordered database quadratically faster [4] or factorize integer numbers exponentially faster [5] than any known classical algorithm.

Machine learning is among the fields that can benefit from the advantages provided by quantum computing [6], [7]. Several quantum machine learning algorithms have been proposed in the literature. For instance, there are quantum models of neural networks [8], [9], support vector machines [10] and gradient descent [11].

A classifier is a function $f : \mathcal{V} \to \mathcal{C}$ that maps unlabeled $v \in \mathcal{V}$ data to a label $c \in \mathcal{C}$. A learning algorithm $\mathcal{L}$ builds a classifier $\mathcal{L}(D)$ using a labeled dataset D [12]. The classifier probability of correctly classifying a random instance is its accuracy. Given a dataset, one needs to choose the best model and algorithm to obtain a configuration with high accuracy. It is necessary to perform empirical tests to achieve a machine learning model suitable to the problem at hand.

One method used in machine learning model evaluation is the k-fold cross-validation [12]. Cross-validation is a method used in machine learning to estimate a classifier's accuracy. In this work, we explore the idea of using quantum parallelism to create a quantum enhanced cross-validation with a linear speed-up advantage over classical cross-validation.

This work was supported by CNPq, FACEPE and Serrapilheira Institute (grant number Serra-1709-22626) (Brazilian research agencies).

Given a data set D. A k-fold cross-validation splits the data set into k disjoint sets $D_1, D_2, \ldots, D_k$ with n samples each. A learning algorithm $\mathcal{L}$ is used to build k classifiers Y_i, where the i-th classifier is trained using the subset $D - D_i$ and tested with D_i. An estimative of the classifier $\mathcal{L}(D)$ accuracy is calculated using equation (1).

$$Accuracy(\mathcal{L}(D)) = \frac{1}{nk}\sum_{i=0}^{k}\sum_{d_j \in D_i}\delta(Y_i(d_j), c_j) \tag{1}$$

Where $\delta(a, b) = 1$ if $a = b$ and 0 otherwise. $Y_i(d_j)$ is the label returned by the classifier Y_i with the instance d_j as input. And c_j is the label of the instance d_j of the test set D_i.

The computational cost to perform a k-fold cross validation is $O(k \cdot m)$, in which the evaluated classifier has cost $O(m)$. In most cases, m is a function of the number of patterns in the dataset. The cost to perform cross-validation can be a limitation when the classification procedure has a high cost or when the classifier is working with a large amount of data.

In this paper, we propose a quantum enhanced k-fold cross-validation, in which the classifier needs to be trained only once instead of $O(k)$. The proposed method is based on the idea that any classical classifier can be lifted to the quantum domain. Any function implemented in a digital computer is a binary function. For instance, neural networks, support vector machines, nearest neighbor classifiers are functions implementable in classical computers, and thus binary.

All efficient computable binary functions can be efficiently simulated in a quantum computer [1]. Classical learning algorithms are also binary functions and theoretically can also be lifted to quantum domain. For instance, the back propagation learning algorithm receives an input, desired output, actual weights and then produces the weights of the next iteration. Therefore, classifiers can be trained with all folds in superposition and then we use a quantum associative memory to estimate the classifier accuracy.

The remainder of this work is divided into six sections. Section II introduces some quantum computing concepts. Section III presents the quantum associative memory used. Section IV is the main contribution of this work, containing the quantum enhanced cross-validation method proposed. The experiments are presented in section V. Section VI contains some discussions concerning the quantum enhanced cross-validation and quantum machine learning. Finally, section VII is the conclusion.

II. Quantum computing

The basic unit of information in quantum computing is the quantum bit (qubit). Unlike a classical bit, which can only assume one value at a given time, a quantum bit can be described in a superposition of states. A qubit is presented in equation (2), where α and β are the probabilistic amplitudes associated with the states.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{2}$$

The probabilistic amplitudes are represented by complex numbers, obeying the normalization rule described in equation (3). Thus, the modulus squared of each amplitude represents a probability, and the probabilistic amplitudes describe the qubit's behavior.

$$|\alpha|^2 + |\beta|^2 = 1 \tag{3}$$

The quantum systems are on a Hilbert space in which complex vectors are used to represent the states. A system with several qubits is composed through the tensor product, represented by the symbol $\otimes$. For instance, the tensor product of $|\psi_0\rangle = \alpha_0|0\rangle + \beta_0|1\rangle$ and $|\psi_1\rangle = \alpha_1|0\rangle + \beta_1|1\rangle$ results in $|\psi_0\psi_1\rangle = \alpha_0\alpha_1|00\rangle + \alpha_0\beta_1|01\rangle + \beta_0\alpha_1|10\rangle + \beta_0\beta_1|11\rangle$. Some quantum states representing multiple qubits cannot be decomposed into tensor products of smaller systems, these states are known as entangled states.

The operations under a quantum system are described by $2^n \times 2^n$ sized unitary matrices, where n is the number of qubits the gate acts on. A matrix U is unitary if its inverse is its conjugate transpose, as presented in equation (4), where I represents the identity matrix and $U^\dagger$ is the conjugate transpose of U.

$$U^\dagger U = UU^\dagger = I \tag{4}$$

An important quantum operator is the Hadamard operator, as it is usually used to generate a quantum state that contains a superposition of the basic states with the same amplitude. The matrix that represents the Hadamard operator can be seen in equation (5), where $H|0\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $H|1\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{5}$$

Other quantum operators such as X and the $CNOT$ are used to switch the state of a qubit. The operator X is equivalent to the classical NOT gate, switching the state of the qubit. And $CNOT$ is a gate that acts in two qubits, the first qubit being the control qubit and the second being the target. The $CNOT$ is a controlled version of the X gate, conditionally switching the state of the target qubit if the control qubit is set to 1. The matrix representation of the gates X and $CNOT$ can be seen in equation (6).

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{6}$$

It is necessary to perform a measurement to extract any information from a quantum state. The measurement is not unitary. After the measurement, the quantum states collapse to one of its values in superposition. For instance, given the quantum state described in equation (7), the probability of finding $|i\rangle$ after the measurement is $p_i = |\alpha_i|^2$ and the state would collapse to $|i\rangle$.

$$|\psi\rangle = \sum_i \alpha_i |i\rangle \tag{7}$$

III. Probabilistic quantum memory

A content-addressable memory is a memory in which content can be retrieved even with noisy input or when only a partial knowledge of the input is known. These memories are also called associative memories and its classical models have a storage capacity shortage.

A probabilistic quantum memory (PQM) is the quantum analog of an associative memory. It is able to overcome the storage limitation of its classical counterpart with an exponential storage capacity with respect to the number of qubits [13], [14].

Let $\mathcal{T}$ be a set of N patterns with n bits used as input to the system $\{p^1, ..., p^N\}$. The probabilistic quantum memory requires three quantum registers to perform the pattern's storage. The first is the input register $|i\rangle$ and the second is the memory register $|m\rangle$, both registers have the size equal to the size of the patterns in the set $\mathcal{T}$. The third register is a two-qubit register $|u_1, u_2\rangle$ prepared in the state $|01\rangle$, which is used as an auxiliary register in the storage algorithm [13].

For each pattern of the input set $\mathcal{T}$ the state of the system is prepared and each pattern is stored on the memory. The storage algorithm for the PQM works by storing the patterns of qubits in superposition as in equation (8). The storage algorithm works as in Fig. 1.

$$|M\rangle = \frac{1}{\sqrt{N}} \sum_{j=1}^{N} |p^j\rangle \tag{8}$$

The content stored on the memory is retrieved probabilistically by the Hamming distance between the input and the superposed states on the memory register. The retrieval algorithm also requires quantum registers for input, memory and control.

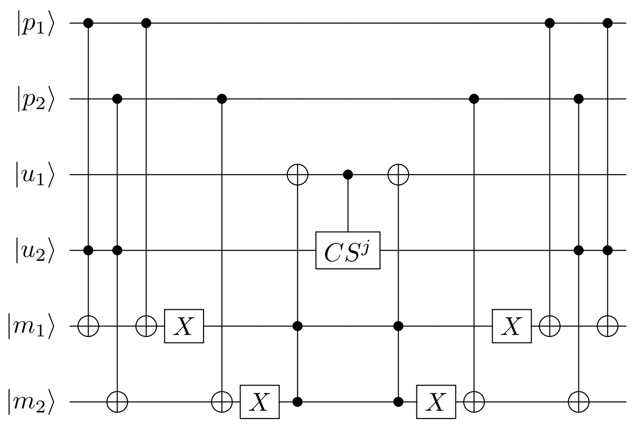


Fig. 1. A quantum circuit for storing patterns of two qubits

The control register is a b-qubit register prepared in the state $|0\rangle_b$. First, a Hadamard gate is applied to the first qubit of the control register, leaving the state as in equation (9)

$$|\psi_0\rangle = \frac{1}{\sqrt{2N}} \sum_{j=1}^{n} \left|i; p^j; 1_1 0_2 \cdots 0_b\right\rangle + \sum_{j=1}^{n} \left|i; p^j; 0_1 0_2 \cdots 0_b\right\rangle \tag{9}$$

Where $|\psi_0\rangle$ is the resulting state of the first step of the retrieval algorithm. In the second step, the state of the memory is processed by applying a $CNOT$ gate to each qubit of the input register as control and each qubit of the memory as target, then a X gate is applied to each qubit of the memory.

$$|\psi_1\rangle = \prod_{j=1}^{N} X_{m_j} CNOT_{i_j,m_j} |\psi_0\rangle \tag{10}$$

Where X_{m_j} is the NOT gate being applied to the j-th qubit of the memory register and $CNOT_{i_j,m_j}$ is the $CNOT$ gate being applied to the system using the j-th qubit of the input as control qubit and j-th qubit of the memory register as target. If there is a pattern stored in the memory that is equal to the input, all the memory qubits will be set to 1.

In the next step the operators U and CU^{-2} are applied to each qubit of the memory, in order to compute the Hamming distance between memory and input. The operators CU^{-2} and U are applied to the state as in equation (11).

$$|\psi_2\rangle = \prod_{j=1}^{N} CU^{-2}_{i_j,m_j} \prod_{j=1}^{N} U_{m_j} |\psi_1\rangle \tag{11}$$

Where CU^{-2} is a controlled version of the U^{-2} operator.

$$U = \begin{bmatrix} e^{(i\frac{\pi}{2N})} & 0 \\ 0 & 1 \end{bmatrix}$$

Then, the inverse transformation is applied to the state in equation (10) in order to restore the memory register to its original state so the process can start again. The state of the system is to be prepared for each qubit of the control register $|c\rangle_b$.

After the state preparation, the control register is measured. If the input pattern is very distant from the patterns stored in the memory, we would obtain a large number of 1s as result. Otherwise, a large number of 0s would be obtained.

Therefore, the quantum cross-validation algorithm is designed to consider the quantity of 0s in the control register. The probability of the quantum register $|c\rangle$ equals to $\mathcal{B}$ is given by equation (12).

$$P(c = \mathcal{B}) = \frac{1}{N} \sum_{j=1}^{N} \binom{d}{\mathcal{B}} cos^{2(d-\mathcal{B})} \left(\frac{\pi}{2N} d_h \left(i, p^j\right)\right) \cdot sin^{2\mathcal{B}} \left(\frac{\pi}{2N} d_h \left(i, p^j\right)\right) \tag{12}$$

IV. Quantum enhanced cross-validation

Given a classifier $\mathcal{L}$, a data set with n patterns stored in superposition can be used to perform a quantum enhanced cross-validation.

Extra quantum registers, $|fold\rangle$ and $|testFold\rangle$, are required to represent the folds used during training and validation. The learning algorithm $\mathcal{L}$ is performed only when the $|fold\rangle$ and $|testFold\rangle$ states are different.

The performance of the classifier $\mathcal{L}$ is stored on register $|performance\rangle$ by applying a X gate to the i-th qubit of the register only if $\mathcal{L}$ correctly classifies the i-th pattern of the k-th fold. Thus, if the classifier $\mathcal{L}$ has a good performance, the register $|performance\rangle$ will have many qubits set to 1.

$$\sum_{testFold \in \{1,\ldots,k\}} |testFold\rangle |\mathcal{L}\rangle |performance_{\mathcal{L},testFold}\rangle \tag{13}$$

With our method the classifier can be trained and evaluated over the entire dataset only once with cost $O(C)$ instead of $O(k \cdot \mathcal{C})$, where k is the number of folds and $\mathcal{C}$ is the cost of training the classifier $\mathcal{L}$.

At the end of the cross-validation process, the registers $|performance_{\mathcal{L},testFold}\rangle$ will store the calculated performances of the evaluated classifier $\mathcal{L}$. This performance register can be used as the memory register for the PQM. In order to access the resulting performance of $\mathcal{L}$ an input state representing 100% performance, $|1\rangle_n$, is used as input to the PQM retrieval algorithm applied to $|performance_{\mathcal{L},testFold}\rangle$.

The algorithm is a classical-quantum procedure, which means that some steps can be executed classically whereas others have to be run in a quantum processor. In this work, it is assumed the existence of a quantum processor in a host classical computer.

Algorithm 1 describes the steps for the quantum enhanced cross-validation. Lines 1 to 3 classically divide the dataset into k folds. Each fold is stored in superposition using the PQM storage algorithm.

The training and initialization take place in steps 5 to 7. The classifier $\mathcal{L}$ is trained in superposition with the folds $|fold\rangle$ that are different from $|testFold\rangle$. In step 7 the classifier has already been trained. Therefore, the $|testFold\rangle$ is used to perform the validation. The register $|performance\rangle$ is initialized in state $|0\rangle_n$. The resulting state after the training and the initialization steps is as in equation (14)

$$\sum_{testFold \in \{1,\dots,k\}} |testFold\rangle|\mathcal{L}\rangle|0\rangle_n \tag{14}$$

The **for** loop in steps 8 to 13 performs the validation of the classifier $\mathcal{L}$ with the data in superposition according with the folds $|testFold\rangle$. In line 9, U is a unitary operator used to calculate the output of classifier $\mathcal{L}$ according to the pattern p_j. The performance of the classifier is stored on the register $|performance\rangle$. The resulting state after step 13 is as described in equation (13).

In step 15 the PQM retrieval algorithm is applied to the performance register. At the end, the number of 1s is retrieved and stored in $n_{\mathcal{L}}$.

Algorithm 1: Quantum Enhanced Cross-Validation (QECV)

```
1  Divide the dataset into $k$ folds
2  for each fold do
3      Store fold in a superposition
4  end
5  Add the parameters of classifier $\mathcal{L}$ into the superposition
6  Train the classifier in superposition with the folds
     different from $|testFold\rangle$
7  Initialize the performance register with state $|0\rangle_n$
8  for each pattern $p_j$ and desired output $t_j$ in $|testFold\rangle$
     do
9      Apply $U|p_j\rangle$ to calculate the output $|o\rangle$
10     if $|o\rangle = |t_j\rangle$ then
11         Set $|performance\rangle_j$ to 1
12     end
13     Apply $U^{-1}|p_j\rangle$
14 end
15 Apply the retrieval algorithm from the PQM with input
     $|1\rangle_n$, memory as $|performance\rangle$ and $b$ qubits in the
     control register $|c\rangle$
16 Measure $|c\rangle$ and store the number of 1s in $n_{\mathcal{L}}$
17 Return $n_{\mathcal{L}}$
```

V. Experiments

Current quantum computers still do not have a suffient number of qubits to perform big scale experiments. Quantum computers with a sufficient number of qubits to perform an experiment able to evaluate the proposed method are still to be developed. Therefore, in order to verify our algorithm we designed an experiment using classical classifiers and a classical description of the PQM retrieval algorithm.

All quantum operations can be put into classical domain. However, due to how costly computationally-wise it is to simulate quantum algorithms on classical computers, we had to perform a reduction on the scope of the algorithm.

We do not perform a simulation of quantum computers. The experiments were conducted in a classical computer by applying the description of the QECV algorithm to evaluate neural network architectures. The architectures were represented by the number of neurons in the hidden layer. As stated before, it was necessary to limit the number of architectures considering they were to be trained classically.

The datasets used were from the PROBEN1 repository [15], which consists in a collection of real world problems for neural network learning. Details of these datasets can be seen in table I. The datasets were divided into train and test sets: the train set contains 10% of the samples while the test set contains the remaining 90%.

TABLE I
Datasets

Dataset	Features	Classes	Examples
Cancer	9	2	699
Gene	120	3	3175

We followed the steps of the proposed algorithm to achieve a classical version of the proposed method. In the first step of the experiment, 1000 neural network instances were trained using scikit-learn [16] version 0.19.1. Then, we performed a 10-fold cross-validation for each architecture, storing all the architectures' performances and output vectors.

The performance vectors obtained after the training process contain the information regarding the classification of each sample of the validation set for all the neural networks. We then stored these vectors on the PQM memory register.

The next step was to measure the control registers. We simulated the retrieval algorithm from the PQM by using equation (12). In this experiment the number of control qubits on the PQM was set to 100.

Finally, for every neural network architecture we calculated the expected value $E(X)$ of the number of 1s on the output from the PQM retrieval algorithm. The results of this experiment can be seen in Table II, in which the expected value $E(X)$ grows inversely proportional to an increase in the performance. Therefore, it suggests a correlation between the architectures performances and the PQM output.

VI. Discussions

The QECV procedure can be used to find optimal parameters, which lead to global minima in the error space, for learning algorithms. As it is shown in Section V. This might not be the case for specific classifiers which have parameters updated during the training, such as *naive Bayes*. However, since the classical cross-validation is used not only for model selection but also for accuracy estimation [12], [17], we can

TABLE II
DATASET RESULTS

Neurons	Performance	$E(X)$
1	0.5269	46.3695
2	0.7148	25.9858
3	0.8423	12.7966
4	0.8930	7.8277
5	0.9336	3.9903
6	0.9566	1.8340
7	0.9647	1.1213
8	0.9730	0.4200
9	0.9733	0.4945
10	0.9753	0.3059
11	0.9763	0.1581
12	0.9764	0.1571
13	0.9766	0.1543
14	0.9773	0.1461
15	0.9765	0.1543
16	0.9769	0.1510
17	0.9770	0.1488
18	0.9774	0.1426
19	0.9775	0.1423
20	0.9772	0.1449

(a) Cancer dataset

Neurons	Performance	$E(X)$
1	0.5778	38.3819
2	0.7862	13.2412
3	0.8619	5.1163
4	0.8781	3.7319
5	0.8809	3.5152
6	0.8829	3.3877
7	0.8827	3.3940
8	0.8814	3.4649
9	0.8808	3.4963
10	0.8796	3.5675
11	0.8790	3.6003
12	0.8787	3.6177
13	0.8777	3.6754
14	0.8777	3.6798
15	0.8781	3.6539
16	0.8768	3.7269
17	0.8775	3.6924
18	0.8772	3.7104
19	0.8772	3.7047
20	0.8769	3.7227

(b) Gene dataset

safely assume that QECV can be used to perform the same tasks as well.

The method proposed in this work can perform a k-fold cross-validation with a linear quantum speedup. Classically, the classifier needs to be evaluated to all the folds. In the proposed method, it is only necessary to evaluate the classifier once for all the folds since we have access to all the data through quantum superposition. One can train and evaluate the classifiers with all the folds simultaneously.

We proposed an algorithm with a linear reduction in the number of steps required to perform a k-fold evaluation. The idea is to provide to the classifier all the folds and parameters in a quantum superposition, to train and test it for all the inputs and parameters with just one execution. In this way, we tested the classifier with different parameters in superposition and evaluated multiple configurations at the same time, unlike one configuration at a time as it is done classically.

Quantum computing can contribute to machine learning algorithms as it provides a way to process all information simultaneously. By making use of entanglement and quantum superposition it is possible to deal with large amounts of data. Quantum computation can provide time complexity advantages as we may prepare the data to be processed in superposition and evaluate it all at once.

Not only time complexity advantages can be obtained over classical computing. For instance, the quantum associative memory used in this work has exponential storage capacity in the number of qubits. There are several machine learning models and methods that may somehow benefit from quantum computation. Moreover, the quantum approaches towards information data representation can also vary. Thus, there are still many points to explore in this field of study.

VII. Conclusion

The QECV allows evaluating classifiers with a linear speedup in computational cost. This result may allow a faster analysis of machine learning algorithms for big data classification and also a faster evaluation of deep learning models.

To actually run the QECV algorithm a quantum processor with thousands of qubits would be required. However, as it is shown in Section V, it is possible to perform experiments in a classical computer, using a limited number of folds without loss of generality to show the viability of QECV.

Representing data in a quantum state is one of the key challenges in quantum machine learning [6]. The approach presented in this work assumes the data are represented by qubits in the computational basis state.

In a future work, we might explore the possibility of using QECV and the representation of data in the amplitudes of the qubits as presented in [18], using the states as indices. Another future work might consider expanding the power of QECV, allowing us to perform a k-fold or a complete k-fold cross-validation. Finally, with small-scale quantum computers becoming a reality, another possible work can explore how to implement this method in such computers using simple classifiers.

References

[1] M Nielsen and I Chuang. *Quantum computation and Quantum information*. Cambridge University Press India, 2000.

[2] NS Yanofsky and MA Mannucci. *Quantum computing for computer scientists*, volume 20. Cambridge University Press Cambridge, 2008.

[3] D Deutsch and R Jozsa. Rapid solution of problems by quantum computation. *Proc. R. Soc. Lond. A*, 439(1907):553–558, 1992.

[4] LK Grover. Quantum mechanics helps in searching for a needle in a haystack. *Physical review letters*, 79(2):325, 1997.

[5] PW Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

[6] M Schuld, I Sinayskiy, and F Petruccione. An introduction to quantum machine learning. *Contemporary Physics*, 56(2):172–185, 2015.

[7] J Biamonte, P Wittek, N Pancotti, P Rebentrost, N Wiebe, and S Lloyd. Quantum machine learning. *Nature*, 549(7671):195, 2017.

[8] AJ da Silva and RLF de Oliveira. Neural networks architecture evaluation in a quantum computer. In *Intelligent Systems (BRACIS), 2017 Brazilian Conference on*, pages 163–168. IEEE, 2017.

[9] AJ da Silva, TB Ludermir, and WR de Oliveira. Quantum perceptron over a field and neural network architecture selection in a quantum computer. *Neural Networks*, 76:55–64, 2016.

[10] P Rebentrost, M Mohseni, and S Lloyd. Quantum support vector machine for big data classification. *Physical review letters*, 113(13):130503, 2014.

[11] P Rebentrost, M Schuld, L Wossnig, F Petruccione, and S Lloyd. Quantum gradient descent and newton's method for constrained polynomial optimization. *arXiv preprint arXiv:1612.01789*, 2016.

[12] R Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145, 1995.

[13] CA Trugenberger. Probabilistic quantum memories. *Physical Review Letters*, 87(6):067901, 2001.

[14] CA Trugenberger. Quantum pattern recognition. *Quantum Information Processing*, 1(6):471–493, 2002.

[15] Lutz Prechelt. Proben1: A set of neural network benchmark problems and benchmarking rules, 1994.

[16] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[17] S Arlot and A Celisse. A survey of cross-validation procedures for model selection. *Statistics surveys*, 4:40–79, 2010.

[18] M Schuld, M Fingerhuth, and F Petruccione. Implementing a distance-based classifier with a quantum interference circuit. *EPL (Europhysics Letters)*, 119(6):60002, 2017.

3 Second Contribution

Quantum enhanced cross-validation for near-optimal neural networks architecture selection

QUANTUM ENHANCED CROSS-VALIDATION FOR NEAR-OPTIMAL NEURAL NETWORKS ARCHITECTURE SELECTION

PRISCILA G. M. DOS SANTOS, RODRIGO S. SOUSA, ISMAEL C. S. ARAUJO AND ADENILTON J. DA SILVA

Abstract. This paper proposes a quantum-classical algorithm to evaluate and select classical artificial neural networks architectures. The proposed algorithm is based on a probabilistic quantum memory and the possibility to train artificial neural networks in superposition. We obtain an exponential quantum speedup in the evaluation of neural networks. We also verify experimentally through a reduced experimental analysis that the proposed algorithm can be used to select near-optimal neural networks.

1. Introduction

Artificial neural networks (ANN) are computational models inspired by the human brain and with learning capacities. The first artificial neuron was proposed in the 1940s,[1] a learning rule is proposed by Hebb[2] and the backpropagation algorithm based on gradient descent was proposed in 1980s.[3] ANNs have several applications in industry and research. For instance, in pattern recognition,[4] clustering,[5] image[6] and speech processing[7] and other applications.

An artificial neuron with m real inputs $x_1, \ldots, x_m$ has m weights $w_1, \ldots, w_m$, a bias b and its output is described in Eq. (1), where f is a nonlinear activation function.

$$f\left(\sum_{k=1}^{m} w_k \cdot x_k + b\right) \tag{1}$$

A Feedforward Neural Network (FNN) is composed of layers of neurons and each layer receives its input signal from the previous layer. FNN optimization has received much attention in the last 20 years.[8] Metaheuristics as meta-learning,[9] differential evolution,[10] genetic algorithms,[11] evolutionary programming, simulated annealing, tabu search,[12] particle swarm optimization,[13] etc.[8] have been used to optimize neural networks architecture.

The number of neurons in the hidden layers and the number of hidden layers are some of the most important elements of a feedforward ANN because there is a relation between them and the ANN performance.[14] The optimization of neural networks weights with backpropagation or other techniques based on gradient descend leads to local minima in the error space. To evaluate a neural network

Departamento de Computação, Universidade Federal Rural de Pernambuco, Rua Dom Manoel de Medeiros, s/n. Campus Dois Irmãos, 52171-900, Recife, Pernambuco, Brazil

E-mail address: {priscila.marques, rodrigo.silvasouza, ismael.cesar, adenilton.silva}@ufrpe.br.

architecture, it is necessary to perform an empirical evaluation that involves a tedious trial and error process with several random weights initializations. This trial and error process can involve a procedure to estimate the accuracy of candidate classifiers. The κ-fold cross-validation[15] is an accuracy estimation method used, for instance, to perform model evaluation and model selection. A dataset T is split in κ disjoint folds or subsets $T_1, \ldots, T_\kappa$ and a classifier is trained κ times in which each iteration $t \in [1, \kappa]$ the model is trained with dataset $T - T_t$ and tested with fold T_t.

To determine if a neural network architecture can learn a given task is an NP-complete problem named the loading problem.[16] If $P \neq NP$ then developing a function that maps neural networks architectures to their best performance over a given data set is an intractable problem. The objective of this work is to investigate the possibility to use quantum computation for selecting a near-optimal classical neural network architecture for a given learning task. In previous works on neural network architecture evaluation[17] or architecture selection on a quantum computer,[18] a nonlinear quantum operator was used to propose a polynomial algorithm that solves the loading problem. As it is not known whether nonlinear quantum operators are physically realizable or not, in this paper we take the safer road by obeying the principles of quantum mechanics by using unitary quantum operators. We have already followed this track by performing an *evaluation* of neural networks performances using unitary quantum operators,[19] here we address the problem of unitarily performing an *architecture selection* of neural networks.

Several quantum machine learning models[20] and quantum neural networks[21] have been proposed, but the non-existence[1] of quantum computers does not allow an empirical comparison between classical and quantum neural networks models. We cannot evaluate numerically the quantum proposed models to verify if they present advantages when compared with classical models. This technical limitation is named the benchmark problem.[20] The algorithm proposed in this work is a quantum algorithm and requires a universal quantum computer. It is also designed to allow a (reduced) simulation in a classical computer and we show that the proposed method can choose a near optimal neural network architecture without the necessity of random weights initializations and with a single training of each neural network architecture. This result has two main implications: i) we can use a quantum enhanced cross-validation to perform neural network parameter evaluation/selection with an exponential quantum speedup and ii) the proposed method can be evaluated numerically and presents advantages over classical strategies using real benchmark problems.

The remainder of this work is organized into 5 sections. Section 2 presents the probabilistic quantum memory used in this work. Section 3 is the main section and presents a quantum algorithm that evaluates classical neural networks architectures and is used to perform neural networks architecture selection. Section 4 presents experiments, that have been performed in a classical computer (and can be executed exponentially faster in a quantum computer). Section 5 presents a discussion of the results. Section 6 presents the conclusion.

[1]Actual quantum computers do not have enough quantum bits, "remain coherent for a limited time[22]" or are designed for specific tasks. The quantum computer necessary to perform the tasks described in this work should be universal and have thousands of qubits

2. Probabilistic quantum memories

A content-addressable memory is called associative memory because of the possibility to retrieve information from it even with partial knowledge of the desired content. Models of associative memories, like the Hopfield network, suffer from a capacity shortage.[23] The quantum counterpart of an associative memory has the advantage of having an exponential capacity because the patterns stored in the memory are kept in superposition. Given a dataset of n patterns with k qubits $T = \{p^1, p^2, ..., p^n\}$, the quantum memory creates the state described in Eq. (2), where $|M\rangle$ is the quantum register that will store the patterns.

$$|M\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} |p^j\rangle \tag{2}$$

In this work, we use the Probabilistic Quantum Memory[24] (PQM). The storage algorithm of the PQM creates a superposition of binary patterns as described in Eq. (2). The retrieval algorithm of the PQM is probabilistic and depends on the Hamming distance between the input pattern and stored patterns.

It is necessary to reload the memory after each execution of the recovering algorithm of the probabilistic quantum memory. This problem is pointed out as a fundamental limitation of the PQM[25] and decreases the speedup for tasks as machine learning.[26] In this work, we propose an application of the PQM that requires a single execution of the PQM recovering algorithm for a given input and this limitation does not affect the method proposed in this paper.

One second limitation of the PQM is its inability to deal with continuous inputs. In the actual small-scale quantum computers this is a strong limitation, but if quantum computers with thousands or millions of qubits are built the binary representation can be used to represent continuous inputs with some precision. In this work, we assume the existence of such quantum computers and continuous inputs can be approximately represented by using binary numbers.

2.1. The storage algorithm.

The states during the PQM storage algorithm are divided into three quantum registers $|p_1p_2...p_k; u_1u_2; m_1m_2...m_k\rangle$. Where $|p_j\rangle$ is the j-th qubit of the input register, u_1u_2 are control qubits prepared in a state $|01\rangle$ and $|\mathbf{m}\rangle = |m_1, \ldots, m_k\rangle$ is the memory register, where the patterns are to be stored. To build a coherent superposition of the patterns to be stored it is necessary to make use of $Toffoli$, X and CS^j gates. The CS^j gate is described in Eq. 3.

$$CS^j = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \sqrt{\frac{j-1}{j}} & \frac{1}{\sqrt{j}} \\ 0 & 0 & \frac{-1}{\sqrt{j}} & \sqrt{\frac{j-1}{j}} \end{bmatrix} \tag{3}$$

A circuit representing a 2 qubit probabilistic quantum memory storing procedure is described in Fig. 1. In the first iteration the quantum registers $|p, u, m\rangle$ are initialized as described in the Eq. 4 and run the circuit described in Fig. 1. For each other pattern p^j in the dataset, we initialize the quantum register input with p^j and execute the circuit described in Fig. 1 again.

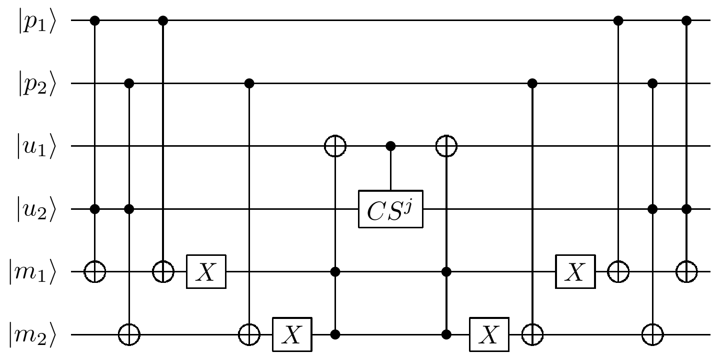


FIGURE 1. A quantum circuit for storing patterns of two qubits

$$|\psi_0\rangle = \left|p_1^1 p_2^1 ... p_k^1, 01, 00...0\right\rangle . \tag{4}$$

In order to store the n patterns of the set P, it is necessary to perform n iterations. In the application of the CS^j gate, the parameter j is $j = p + 1 - iter$ where p is the number of patterns and $iter$ is the iteration number.

2.2. **The retrieval algorithm.** The retrieval algorithm also requires three registers[23, 24] $|i_1...i_k; m_1...m_k; c_1...c_d\rangle$. Where $|i_j\rangle$ is the j-th qubit of the input register, $|m_j\rangle$ is the j-th qubit of the memory register and c_j is the j-th qubit of the control register,[24] The control qubits start at the state $|0...0\rangle$. The retrieval of the patterns is made probabilisticaly.[23] For each c_l in $|c\rangle$, the following steps are performed. Step 1 apply the Hadamard gate to $|c_l\rangle$ giving us, at the first iteration, the state described in Eq.(5).

$$|\psi_0\rangle = \frac{1}{\sqrt{2n}} \sum_{j=1}^{n} \left| i_1...i_k; p_1^j...p_k^j; 0_1 0_2 ... 0_b \right\rangle + \frac{1}{\sqrt{2n}} \sum_{j=1}^{n} \left| i_1...i_k; p_1^j...p_k^j; 1_1 0_2 ... 0_d \right\rangle \tag{5}$$

In the second step, for each bit i_j in the input pattern, we apply the CNOT gate with the j-th qubit of the input as the control and the j-th qubit of the memory as the target and apply the NOT gate to the j-th bit of the memory to obtain the state

$$|\psi_1\rangle = \prod_{j=1}^{n} CNOT(i_j, m_j) X(m_j) |\psi_0\rangle .$$

After the second step, if there is a pattern stored in the superposition equal to the input all its qubits will be set to $|1\rangle$.[23]

Step 3 applies the quantum operator described in Eq. 6.

$$|\psi_2\rangle = \prod_{j=1}^{k} (CV^{-2})(c_l, m_j) \prod_{j=1}^{k} U(m_j) |\psi_1\rangle \tag{6}$$

Where the operator V is a unitary matrix and CV^{-2} is a controlled version of the V^{-2} operator.

$$V = \begin{bmatrix} e^{(i\frac{\pi}{2n})} & 0 \\ 0 & 1 \end{bmatrix}$$

The inverse of steps 1 and 2 are applied to the quantum state $|\psi_2\rangle$ to restore the memory quantum register to its original state. This is the last deterministic step of the retrieval algorithm and the resulting state is described in Eq. (7).

$$(7) \qquad |\psi\rangle = \frac{1}{\sqrt{n}} \sum_{j=1}^{n} \sum_{l=0}^{d} cos^{d-l}\left(\frac{\pi}{2k} d_h\left(i, p^j\right)\right) \cdot sen^l\left(\frac{\pi}{2k} d_h\left(i, p^j\right)\right) \sum_{\{J^l\}} \left|i; p^k; J^l\right\rangle$$

Where $\left\{J^l\right\}$ is the set of all binary strings with exactly l bits set to 1 and $(d-l)$ bits set to 0.[24] After processing the state, it is necessary to perform a measurement to the control qubits.

The result of the measurement is a large number of control qubits in the state $|0\rangle$ if all stored patterns are similar to the input, and a large number of control qubits in the state $|1\rangle$ if all stored patterns are very distant of the input. In this work, we consider that the number of 1s obtained after the measurement of quantum register $|c\rangle$ is the output y of the PQM. From Eq. 7 we can easily verify that the probability to obtain $y = \mathcal{K}$ is given by

$$(8) \qquad P(y = \mathcal{K}) = \frac{1}{p} \sum_{j=1}^{p} \binom{d}{\mathcal{K}} cos^{2(d-\mathcal{K})}\left(\frac{\pi}{2k} d_h\left(i, p^j\right)\right) \cdot sin^{2\mathcal{K}}\left(\frac{\pi}{2k} d_h\left(i, p^j\right)\right)$$

3. Selection of neural networks architecture in a quantum computer

An artificial neural network for classification is defined as a function $N : \mathbb{R}^m \to \{c_1, \ldots, c_k\}$. Despite being a real function the implementation of a neural network in a classical digital computer is a binary function. All binary functions can be simulated in a quantum computer, then theoretically a classical neural network can be represented by a quantum circuit where the weights, inputs and outputs are strings of qubits. One backpropagation iteration is also a real function that receives inputs $x(t)$ and weights $w(t)$, and outputs weights $w(t+1)$. In a digital computer one backpropagation iteration is also a binary function and theoretically can be represented with a quantum operator.

Figure 2 represents the idea of training neural networks in superposition. The first quantum register receives patterns from the training set. The load function loads a pattern in the quantum register. This loading function can be accomplished because the state of input quantum register is always a basis state. The BP operator represents a backpropagation step. The first BP operator receives input $|x_t\rangle$ and weights $|w_t\rangle$ and produces $|w_{t+1}\rangle$. The second BP operator receives $|x_{t+1}\rangle$ and $|w_{t+1}\rangle$ to produce $|w_{t+2}\rangle$. The quantum operator $BP^\dagger$ inverts the action of the first BP operator and prepares the third quantum register to receive the next weight vector. If $|w_t\rangle$ is a superposition of weights, a sequence of *load*, BP and $BP^\dagger$ operators will train all the networks in the superposition simultaneously. We can obtain all the neural networks with a given architecture in superposition just by applying the Hadamard operator in all qubits in the quantum register $|w\rangle$.

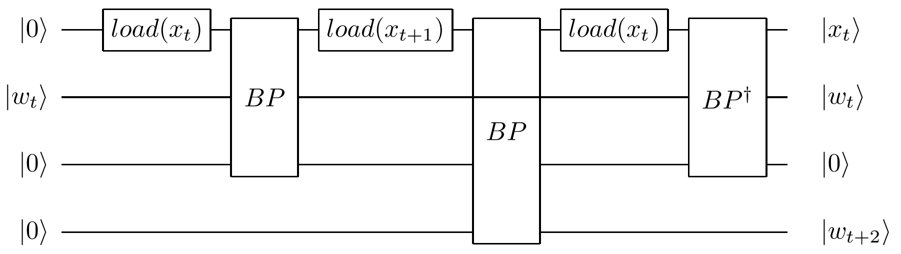


FIGURE 2. Theoretical quantum circuit implementing two back-propagation iterations

The idea to present an input pattern to neural networks in superposition is presented by Meener,[27] he named this strategy Strongly Inspired Neural Network. Instead of developing the strongly inspired neural network the authors developed a weakly inspired neural network, where for each pattern in a dataset a neural network is trained and later all neural networks are stored in superposition. We develop the strategy of strong inspired neural networks and we use the name superposition based learning[28] to avoid confusion with works that deal with classical neural networks only based on ideas from quantum computing.

In addition to creating a superposition of neural networks, we also manage the dataset to perform a κ-fold cross-validation with the neural networks in superposition. To perform the cross-validation using quantum superposition, we use two additional quantum registers $|fold\rangle$ representing the fold used as test and $|input\_fold\rangle$ which contains information about the fold of the actual input sample. The learning algorithm iteration is applied only when $|fold\rangle$ and $|input\_fold\rangle$ are different. If the $|fold\rangle$ quantum register receives a superposition representing all possible folds, the cross-validation can be performed in superposition with the cost of only one learning algorithm execution.

The effect to train all neural networks in superposition is to obtain a superposition of weight vectors at local minima of the error surface. Classically one could just choose the neural network with the best accuracy over a validation dataset. But this information cannot be accessed directly from the state in quantum superposition. To obtain a useful measure of the architecture performance for the dataset we use a quantum procedure to calculate the distance between the neural networks accuracy and 100% of accuracy in the validation set.

We calculate the performance of the neural networks in superposition by presenting patterns in fold l ($l = 1, \ldots, \kappa$) to the neural networks in superposition and applying an X gate in the ith qubit of the performance quantum register if the network correctly classifies the ith pattern and $testFold$ is equal to l. After this procedure the performance and weights quantum registers will be entangled and their state is described in Eq. (9).

$$\sum_{w,testFold} |testFold\rangle\,|w\rangle\,|performance_{w,testFold}\rangle \tag{9}$$

It is necessary to present the dataset only once to calculate the performance of all neural networks in all folds.

The quantum register performance is in state

$$\sum_{w,testFold} |performance_{w,testFold}\rangle \,,$$

we use this state as the memory of a probabilistic quantum memory with input $|1\rangle_n$ with k control qubits. The auxiliary quantum register $|c\rangle$ is measured and the output is the number of 1s obtained. The algorithm is repeated one time for each architecture to be evaluated.

Algorithm 1 presents the quantum algorithm to select neural networks architectures. We suppose that the quantum device is controlled by a host classical computer. Heterogeneous computer architectures with different processors specialized for different tasks are increasingly common. The first quantum computers with such architecture are in development. Algorithm 1 uses this hybrid architecture to perform a heuristic search over classical neural networks architectures using a quantum device.

Step 1 of Algorithm 1 creates the folds of the cross-validation and is performed in the classical computer. Each fold created in the cross-validation procedure has the same size and if necessary some patterns are removed from the dataset.

The for loop starting in step 2 performs an enhanced cross-validation of all neural networks with a given architecture. Steps 3 to 5 are initialization steps and are executed in the theoretical quantum device. Step 3 initializes neural network weights in a superposition of all possible weights. Step 4 initializes the fold quantum register with the quantum state $\sum_{l=1}^{\kappa} \frac{1}{\sqrt{\kappa}} |l\rangle$. Step 5 initializes the quantum register performance with the quantum state $|0\rangle_n$. After step 5, the state of quantum registers testFold, weights and performance will be described by Eq. (10), where W is the set of all possible weights with a given precision.

$$|testFold\rangle\,|weights\rangle\,|performance\rangle = \sum_{\substack{w\in W,\\ testFold\in\{1,\cdots,\kappa\}}} |testFold\rangle\,|w\rangle\,|0\rangle_n \tag{10}$$

Step 6 trains the neural networks in superposition. The training procedure is a quantum-classical algorithm described in Fig. 2. At each iteration, the classical device selects a pattern from the dataset and loads the pattern in the input quantum register (that is always in a basis state) and the fold of the pattern in the inputFold quantum register. Then the learning iteration is performed in the parcels in the superposition where the testFold and inputFold are different. After this step the weights quantum register is in a superposition of trained neural networks and the testFold, weights and performance quantum register will be in the state described in Eq. (11) where $\tilde{W}_{testFold}$ is the set of neural networks weights trained with data $T - T_{testFold}$.

$$\sum_{\substack{w_{testFold}\in\tilde{W}_{testFold},\\ testFold\in\{1,\cdots,\kappa\}}} |testFold\rangle\,|w_{testFold}\rangle\,|0\rangle \tag{11}$$

The for loop starting in step 7 calculates the performance of each neural network in superposition in the validation set. Each test fold of the cross-validation is presented and the neural networks accuracy correspondent to this fold is evaluated. The evaluation is performed in superposition and the test set needs to be presented only once. After this for loop, the state of quantum registers testFold, weights and performance are described in Eq. (12).

$$\sum_{\substack{w_{testFold}\in\tilde{W}_{testFold},\\ testFold\in\{1,\cdots,\kappa\}}} |testFold\rangle\,|w_{testFold}\rangle\,|performance_{w_{testFold}}\rangle \tag{12}$$

Step 15 runs the recovering algorithm of the probabilistic quantum memory with input $|1\rangle_n$ representing a performance of 100% and the state in quantum register performance as memory.

Step 16 measure the output of the quantum probabilistic memory and store the number of 1s in a classical variable n_N for each architecture N. At the end of the algorithm the simplest network N that minimizes the value of n_N is indicated by the algorithm.

Algorithm 1: Architecture selection

```
1  Divide the dataset $\mathcal{T}$ in $\kappa$ folds
2  for each neural network architecture $N$ do
3      Initialize all weights qubits with $H|0\rangle$
4      Create a superposition with the values 1 to $\kappa$ in quantum register testFold
5      Initialize quantum register $|performance\rangle$ with the quantum register $|0\rangle_n$
6      Train the neural networks in superposition with the folds with label
        different of testFold
7      for each pattern $p_j$ and desired output $d_j$ in $testFold_j$ do
8          Initialize the quantum registers input, calculatedOutput and
            desiredOutput with the basis quantum state $|p_j, 0, d_j\rangle$
9          Calculate $N|p_k\rangle$ to calculate network output in quantum register
            calculatedOutput $|o\rangle$
10         if $|o\rangle = |d\rangle$ and $|testFold\rangle = |inputFold(p_j)\rangle$ then
11             Set $|performance\rangle_j$ to 1
12         end
13         Calculate $N^{-1}$ to restore $|o\rangle$
14     end
15     Apply the quantum associative recovering algorithm with input $|1\rangle_n$,
        memory $|performance\rangle$ and $b$ qubits in the output
16     Measure quantum register $|c\rangle$ and stores the number of 1s in $n_N$
17 end
18 Return the simplest neural network architecture $N$ that minimize $n_N$.
```

4. Experiments

Since there are no quantum computers with sufficient qubits to run the proposed algorithm, it was necessary to perform some changes in the quantum algorithm in order to simulate it on a classical computer. Therefore, we reduced (without loss of generality) the number of neural network instances in the quantum parallelism. Besides this change, we simply followed the algorithm description in order to make a classical version of Algorithm 1.

Table 1. Datasets.

Dataset	features	classes	examples	description
cancer	9	2	699	diagnosis of breast cancer
gene	120	3	3175	detect intron/exon boundaries in nucleotide sequences
diabetes	8	2	768	diagnose diabetes of Pima indians
card	51	2	690	predict the approval of a credit card to a customer
glass	9	6	214	classify glass types
heart	35	2	920	predict heart disease
horse	58	3	364	predict the fate of a horse that has a colic
mushroom	125	2	8124	discriminate edible from poisonous mushrooms

To perform the experiments, we use the Multilayer Perceptron (MLP). The training and evaluation were performed using the scikit-learn[30] version 0.19.1. After training the neural networks we evaluated the performance of every neural network instance and stored the performance vectors which have the size of the validation set and the i-th position is set to 1 if the trained network correctly classifies the i-th vector in the validation set and is set to 0 otherwise. The performance vectors are used as the memory of the probabilistic quantum memory and the output of the probabilistic quantum memory was calculated using Eq. (7).

The datasets used in this work were from the PROBEN1 repository, which consists in a collection of problems for neural network learning in the realm of pattern classification and function approximation.[31] PROBEN1 contains 15 datasets from real-world problems and from 12 different domains. We used 8 datasets to perform the experiments: cancer, gene, diabetes, card, glass, heart, horse and mushroom. The details about the datasets used can be seen in Table 1. The datasets were divided into 10 folds and the train set contains 9 folds while the test set contains the remaining fold. The number of output qubits in the probabilistic quantum memory was set to 100.

We consider the number of neurons in the hidden layer as the architecture to be evaluated. Thus, the number of neurons in the hidden layer was varied between 1 and 20. All neural network architectures were trained and tested for every dataset and for 1000 different initialization weights (100 for each fold). The alpha parameter used avoids overfitting by constraining the size of the weights. The learning algorithm is ‘adam’ which refers to the stochastic gradient-based optimizer proposed by Kingma, Diederik, and Jimmy Ba.[32] The parameters used can be seen in Table 2.

5. Results and discussion

Let X be a random variable representing the number of 1s in the output of the probabilistic quantum memory. Table 3 shows the results of the experiment for cancer and gene datasets. We can verify that the expected value $E(X)$ is related to neural network mean performance. An increase in performance corresponds to a reduction in $E(X)$.

In Fig. 3 we plot the mean performance of each architecture versus the expected value of the X for cancer, card, diabetes, gene, glass, heart, horse and mushroom datasets. We can easily see that there is an approximately linear relation between the neural network mean performance and $E(X)$. In this way, we can

Table 2. MLP Parameters.

Parameter	Value	Description
solver	adam	stochastic gradient-based optimizer
alpha	1e-4	regularization term parameter
beta_1	0.9	decay rate for estimates of first moment vector
beta_2	0.999	decay rate for estimates of second moment vector
epsilon	1e-8	value for numerical stability in adam
max_iter	100	maximum number of iterations
activation	relu	rectified linear unit function
learning_rate_init	1e-3	controls the step-size in updating the weights
number of hidden neurons	[1,20]	number of hidden neurons

use Algorithm 1 to select a near-optimal neural network and also to estimate the mean-performance of the neural network architecture over a dataset.

One limitation of classical neural networks is the absence of an algorithm to determine the best neural network architecture for a given dataset. Classical strategies to evaluate neural network architectures requires a costly process that can last from minutes to days.[14] How to select a neural network architecture is yet an open problem and the use of more complex neural networks with deep architectures increases the complexity to determine a neural network architecture with optimal performance.

In this work, we explore the principles of quantum computing to create a hybrid classical and quantum algorithm to perform classical neural network architecture selection. If C is the cost to train one neural network, given n neural networks architectures the proposed method has cost $O(n \cdot C)$ and determine a near-optimal neural network architecture in the given set of architectures. A classical algorithm to evaluate n neural networks architectures over all possible initial weights will have

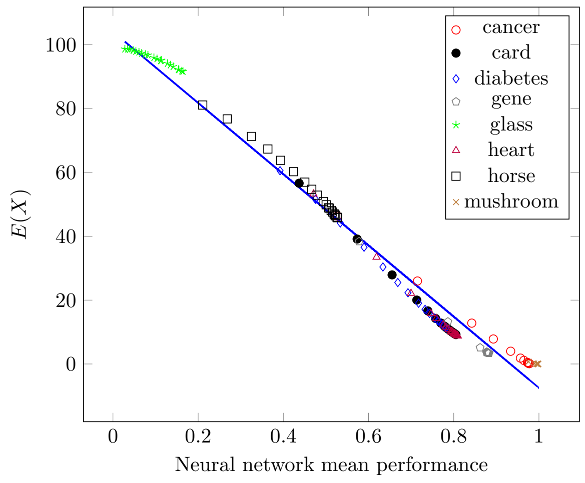


Figure 3. Cancer, card, diabetes, gene, glass, heart, horse and mushroom datasets mean performance versus $E(X)$

Neurons	Performance	$E(X)$
1	0.5269	46.3695
2	0.7148	25.9858
3	0.8423	12.7966
4	0.8930	7.8277
5	0.9336	3.9903
6	0.9566	1.8340
7	0.9647	1.1213
8	0.9730	0.4200
9	0.9733	0.4945
10	0.9753	0.3059
11	0.9763	0.1581
12	0.9764	0.1571
13	0.9766	0.1543
14	0.9773	0.1461
15	0.9765	0.1543
16	0.9769	0.1510
17	0.9770	0.1488
18	0.9774	0.1426
19	0.9775	0.1423
20	0.9772	0.1449

(A) Cancer dataset

Neurons	Performance	$E(X)$
1	0.5778	38.3819
2	0.7862	13.2412
3	0.8619	5.1163
4	0.8781	3.7319
5	0.8809	3.5152
6	0.8829	3.3877
7	0.8827	3.3940
8	0.8814	3.4649
9	0.8808	3.4963
10	0.8796	3.5675
11	0.8790	3.6003
12	0.8787	3.6177
13	0.8777	3.6754
14	0.8777	3.6798
15	0.8781	3.6539
16	0.8768	3.7269
17	0.8775	3.6924
18	0.8772	3.7104
19	0.8772	3.7047
20	0.8769	3.7227

(B) Gene dataset

TABLE 3. Results cancer dataset (left) and gene datset (right)

at least cost $O(n \cdot 2^{|W|} \cdot C)$, where W is the set of all possible weights with a given precision. The proposed method has an exponential speed up when compared to its classical version.

One limitation of the proposed method is its inability to evaluate neural networks architectures with very close performance. This limitation occurs because of the use of Hamming distance. Then the method should be used to select a set of near-optimal neural networks and then a classical experimentation could be performed to finish the search for the neural network with the best performance.

With a promise that the neural networks to be evaluated have a significant difference in their accuracy over the test set, the proposed method will perform an optimal selection (instead of near-optimal) with high probability. For instance, with 1 neuron in the hidden layer we obtain a neural network mean accuracy of 0.52 and with 19 neurons in the hidden layer, the neural network obtains mean accuracy of 0.97. With the 19 hidden neurons neural network the proposed method will have 0 or 1 ones in the output with probability 0.9874 and the neural network with 1 hidden neuron will have 0 or 1 ones in the output with probability 0.1548. With the objective to illustrate the behavior of the proposed method, Fig. 4 presents the number of ones in the output of the probabilistic memory and the probability of each output.

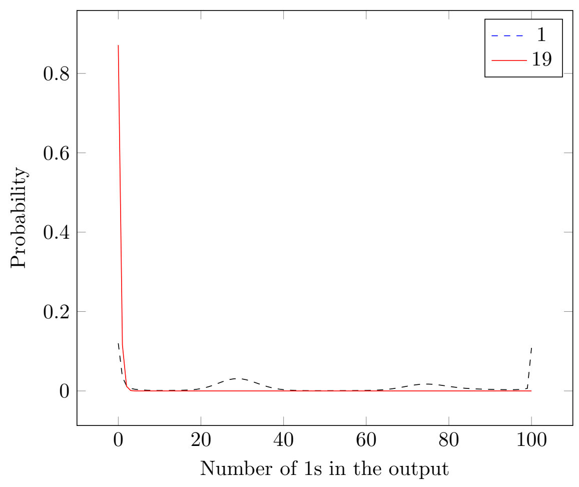


Figure 4. Probability output for cancer dataset with 1 hidden neuron and 19 hidden neurons

The neural network architecture selection uses quantum registers inputs, weights, desired output, calculated output and performance. Algorithm 1 creates a superposition of neural networks with all possible weights and evaluate the performance of each neural network in superposition. This evaluation in a quantum superposition is based on a neural network quantum learning algorithm,[17] one of the authors generalizes this strategy to perform a selection[18][2] and evaluation[19] of neural networks architectures. We notice a related work where a quantum superposition of classifiers is used to perform parameter selection.[33]

The main idea of the quantum cross-validation is to explore the quantum parallelism to execute a κ-fold cross-validation, training the model only once. This strategy leads to a constant speedup in the cross-validation process. The exponential speedup obtained in this paper came from the superposition of neural networks. In this way, using cross-validation we can evaluate an exponential number of neural networks with the cost to train and run a single neural network.

6. Conclusion

In this work, we proposed a classical-quantum algorithm to select neural networks architectures (number of neurons in the hidden layer). We evaluated the proposed method using its classical description and reducing the number of artificial neural networks in superposition.

Our main result is the ability to evaluate one neural network architecture through a κ-fold cross-validation with the cost to train only one neural network instance. The proposed method evaluates an exponential number of neural networks weights simultaneously with an exponential improvement in computational cost when compared with known classical alternatives. The fast neural network evaluation allows the selection of near-optimal neural network architectures by repeating the cross-validation for each neural network architecture.

[2]Where we supposed the viability of nonlinear quantum operators

Quantum computation can be used to evaluate neural networks models with an exponential speedup. The lack of experimentation is one problem in quantum machine learning because of the non-existence of quantum computers with enough quantum bits. To allow experimentation, we use benchmark problems to evaluate (without loss of generalization) a classical simplified version of the proposed method.

One possible future work is to extend the proposed method to deal with models with close performance. One suggestion to accomplish this improvement is to change the probabilistic associative memory to deal with others distance functions. We also can use other kinds of quantum memories and machine learning models.

Acknowledgement

This work was supported by the Serrapilheira Institute (grant number Serra-1709-22626), CNPq and FACEPE (Brazilian research agencies).

References

[1] W. S. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[2] D. O. Hebb. *The organization of behavior: A neuropsychological theory.* Psychology Press, 2005.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[4] S. Samarasinghe. *Neural networks for applied sciences and engineering: from fundamentals to complex pattern recognition.* CRC Press, 2016.

[5] J. Xu, B. Xu, P. Wang, S. Zheng, G. Tian, and J. Zhao. Self-taught convolutional neural networks for short text clustering. *Neural Networks*, 88:22–31, 2017.

[6] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.

[7] W. Chan, N. Jaitly, Q. Le, and O. Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE, 2016.

[8] V. K. Ojha, A. Abraham, and V. Snášel. Metaheuristic design of feedforward neural networks: A review of two decades of research. *Engineering Applications of Artificial Intelligence*, 60:97–116, 2017.

[9] A. Abraham. Meta learning evolutionary artificial neural networks. *Neurocomputing*, 56:1–38, 2004.

[10] J. Ilonen, J. Kamarainen, and J. Lampinen. Differential evolution training algorithm for feed-forward neural networks. *Neural Processing Letters*, 17(1):93–105, 2003.

[11] D. J. Montana and L. Davis. Training feedforward neural networks using genetic algorithms. In *Proc. 11th Int. Joint Conf. Artificial Intelligence*, volume 89, pages 762–767, San Mateo, CA, 1989. Morgan Kaufmann.

[12] D. Pham and D. Karaboga. *Intelligent optimisation techniques: genetic algorithms, tabu search, simulated annealing and neural networks.* Springer, 2012.

[13] J. Zhang, J. Zhang, T. Lok, and M. R. Lyu. A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training. *Applied mathematics and computation*, 185(2):1026–1037, 2007.

[14] P. G. Benardos and G. Vosniakos. Optimizing feedforward artificial neural network architecture. *Engineering Applications of Artificial Intelligence*, 20(3):365–382, 2007.

[15] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *International Joint Conference on Artificial Intelligence*, volume 14, pages 1137–1145. Stanford, CA, 1995.

[16] J. S. Judd. *Neural network design and the complexity of learning.* MIT press, 1990.

[17] M. Panella and G. Martinelli. Neural networks with quantum architecture and quantum learning. *International Journal of Circuit Theory and Applications*, 39(1):61–77, 2011.

[18] A. J. da Silva, T. B. Ludermir, and W. R. de Oliveira. Quantum perceptron over a field and neural network architecture selection in a quantum computer. *Neural Networks*, 76:55–64, 2016.

[19] A. J. da Silva and R. L. de Oliveira. Neural networks architecture evaluation in a quantum computer. In *6th Brazilian Conference on Intelligent System*, pages 163–168, Uberlândia, MG, 2017. IEEE.

[20] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[21] M. Schuld, I. Sinayskiy, and F. Petruccione. The quest for a quantum neural network. *Quantum Information Processing*, 13(11):2567–2586, 2014.

[22] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta. Open quantum assembly language. *arXiv preprint arXiv:1707.03429*, 2017.

[23] C. A. Trugenberger. Probabilistic quantum memories. *Physical Review Letters*, 87(6):067901, 2001.

[24] C. A. Trugenberger. Quantum pattern recognition. *Quantum Information Processing*, 1(6):471–493, 2002.

[25] T. Brun, H. Klauck, A. Nayak, M. Rötteler, and Ch. Zalka. Comment on "probabilistic quantum memories". *Phys. Rev. Lett.*, 91:209801, 2003.

[26] M. Schuld, I. Sinayskiy, and F. Petruccione. Quantum computing for pattern classification. In *Pacific Rim International Conference on Artificial Intelligence*, pages 208–220. Springer, 2014.

[27] T. Menneer and A. Narayanan. Quantum-inspired neural networks. *Tech. Rep. R329*, 1995.

[28] B. Ricks and D. Ventura. Training a quantum neural network. In *Advances in neural information processing systems*, pages 1019–1026, 2004.

[29] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[30] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnayothers. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[31] L. Prechelt. Proben1: A set of neural network benchmark problems and benchmarking rules. 1994.

[32] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[33] V. Dunjko, J. M. Taylor, and H. J. Briegel. Advances in quantum reinforcement learning. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 282–287, 2017.

4
Third Contribution

Evaluation and improvement of a
Probabilistic Quantum Memory Weightless
classifier

Evaluation and improvement of a Probabilistic Quantum Memory Weightless classifier

Priscila G. M. dos Santos, Rodrigo S. Sousa and Adenilton J. da Silva[*]

Universidade Federal Rural de Pernambuco
Departamento de Computação.
Recife, Brazil

Abstract

In this work, we evaluate a Quantum Weightless Classifier based on a Probabilistic Quantum Memory. We also propose a modified version of the model by adding a scale parameter to the memory retrieval algorithm. The evaluation of both models was conducted through classical experiments using an equivalent classical description of the Probabilistic Quantum Memory algorithm. We present the first evaluation of a quantum weightless neural network on public benchmark datasets and propose a modification to better adjust the model in pattern classification tasks. The original classifier presented satisfactory accuracy while its modified version obtained improved results in all datasets.

Keywords: quantum computing, quantum machine learning, probabilistic quantum memory

1. Introduction

Quantum Computing is a paradigm that has been gathering increasingly attention several decades now. The current advances in the field are bringing us to a new era in quantum technology: the era of Noisy Intermediate-Scale Quantum Computers (NISQ) [1]. The quest for quantum supremacy, which

[*]Corresponding author
Email address: priscila.marques@ufrpe.br, rodrigo.silvasouza@ufrpe.br, adenilton.silva@ufrpe.br (Priscila G. M. dos Santos, Rodrigo S. Sousa and Adenilton J. da Silva)

will be achieved when a quantum computer outperforms a classical one in a given task, is currently a heated discussion topic in the field. Given the current state of the art, it is expected that quantum supremacy can be achieved in the next few years. [2]. One of the approaches for achieving it is through quantum machine learning.

Machine learning is a widely applicable and relevant field. It focus on developing automated ways for computers to learn some specific task from a given set of data samples. Quantum machine learning studies the use of quantum concepts in order to build quantum enhanced machine learning models able to outperform the classical ones [3]. Several works have been conducted in this area. A quantum generalisation of a neural network is proposed in [4]. In [5] a quantum model of a distance-based classifier is proposed and implemented in a small-scale quantum computing device.

The aim of this work is to contribute with the field by investigating a quantum machine learning application, proposing a classical experimentation set-up to evaluate the model and proposing an improvement for it. This work is centered on a quantum weightless neural node classifier. We perform the first experimental evaluation of the model and propose a modification that improves its classification accuracy.

The weightless neural node evaluated in this work is a quantum model. It would be required a large scale quantum device in order to experimentally test the models classification capabilities. Quantum computing is currently on the rise and quantum devices with increasing numbers of qubits are being developed to supply this demand. However, the publicly accessible quantum devices currently at disposal can only perform experiments which require a small amount of qubits. High scale experiments cannot be conducted on such small devices. Considering this, we take the classical approach. Since quantum circuits can be calculated classically with a polynomial amount of memory, we conducted the model evaluation on a conventional computer through classical reduced versions of the models algorithms.

2. Quantum Computing

Quantum computing is a field that has been gathering increasingly attention due to its current advances [1]. It touches upon ideas of quantum mechanics and information theory. A quantum computer is the concept for a computational device capable of representing information by making use of microscopic quantum level effects to perform computational tasks [6]. In quantum computing, the quantum bit (qubit) represents the basic unit of information in a quantum system. Analogously to the behavior of a subatomic particle, the qubit can be in more than one state at a given time. Equation (1) describes one qubit in superposition, where α and β are the probabilistic amplitudes associated with the states $|0\rangle$ and $|1\rangle$, respectively.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

The probabilistic amplitudes are represented by complex numbers, obeying the normalization rule described in equation (2). The probability of a qubit being found in any of the possible states is given by the modulus squared of its amplitudes before a measurement is made.

$$|\alpha|^2 + |\beta|^2 = 1 \tag{2}$$

An important quantum characteristic is the necessity to measure in order to extract information from a quantum state. After a measurement the system collapses to one of its possible states in the superposition. For instance, given the quantum state described in equation (3), the probability of finding $|i\rangle$ after a measurement is $p_i = |\alpha_i|^2$.

$$|\psi\rangle = \sum_i \alpha_i |i\rangle \tag{3}$$

Due to the capacity of dealing with states in superposition and other incorporated quantum effects, quantum computers provide a different way of approaching computational tasks, which, could be used to solve problems that are hard for conventional classical computers.

3. Probabilistic Quantum Memories

In this section, we present the quantum memory model used to build the weightless network classifier. The Probabilistic Quantum Memory (PQM) [7, 8] is a content-addressable quantum memory. It outputs the probability of a given input pattern being stored on the memory by calculating the Hamming distance between the input pattern and all the patterns stored on the memory. It is a probabilistic model designed to recognize even incomplete or noisy information. Despite being an associative model, the PQM possess a highly scalable storage capability, being able to store all the possible 2^n binary patterns of n bits. The storage and retrieval PQM procedures are explained in the following subsections.

3.1. Storage Procedure

The PQM stores information in an uniform quantum superposition. The quantum state after the storage mechanism execution is described in Eq. (4), where p is the number of patterns in the dataset and p^i are the stored patterns.

$$|M\rangle = \frac{1}{\sqrt{p}} \sum_{i=1}^{p} |p^i\rangle \tag{4}$$

3.2. Retrieval Procedure

The retrieval procedure computes the Hamming distance between an input and all the patterns superposed on the memory quantum state. It probabilistically indicates the chance of a given input pattern being on the memory based on the results of its distance distribution to the stored patterns in superposition. If the input pattern is very distant from the patterns stored on the memory, one will obtain 1 as a result with a large probability. Otherwise, 0 would be obtained. Since the memory state is prepared in a superposition, the retrieval procedure can calculate the distances from input to all the patterns at once.

The PQM retrieval algorithm is described in Algorithm 1. It uses three quantum registers: $|i\rangle$, $|m\rangle$ and $|c\rangle$. The size of the first two registers is given by the patterns size and $|c\rangle$ is a single qubit register. Step 1 of the algorithm loads the pattern to be retrieved into the first register. The second register $|m\rangle$, is the memory, it contains all the patterns stored. And $|c\rangle$ is a control qubit initialized with an uniform superposition of $|0\rangle$ and $|1\rangle$. The quantum state after the first step of the algorithm can be seen in (5), where p is the number of stored patterns

Algorithm 1: Probabilistic quantum memory retrieval algorithm

```
1 Load the input $|p\rangle$ in the quantum register $|i\rangle$
2 $|\psi_1\rangle = \prod_{j=1}^{n} X_{m_j} XOR_{i_j,m_j} |\psi_0\rangle$
3 $|\psi_2\rangle = \prod_{i=1}^{n} \left(CU^{-2}\right)_{c,m_i} \prod_{j=1}^{n} U_{m_j} |\psi_1\rangle$
4 $|\psi_3\rangle = H_c \prod_{j=n}^{1} XOR_{i_j,m_j} X_{m_j} |\psi_2\rangle$
5 Measure qbit $|c\rangle$
6 if $c == 0$ then
7     Measure the memory to obtain the desired state.
8 end
```

$$|\psi_0\rangle = \frac{1}{\sqrt{2p}} \sum_{k=1}^{n} \left|i; p^k; 0\right\rangle + \frac{1}{\sqrt{2p}} \sum_{k=1}^{n} \left|i; p^k; 1\right\rangle \tag{5}$$

On step 2, the memory register qubits are set to $|1\rangle$, if they are identical on the input and memory registers; and set to $|0\rangle$, if they differ. That is, if a pattern stored on the memory is equal to the input pattern, step 2 would set its memory state to $|1\rangle_n$, where n is the pattern size. In step 3 the operators U and a controlled U in $|c\rangle$ are applied to the memory registers.

$$U = \begin{bmatrix} e^{(i\frac{\pi}{2k})} & 0 \\ 0 & 1 \end{bmatrix}$$

This step is responsible for computing the Hamming distance between the input pattern and the patterns on the memory. It computes the number of

0s in the memory register (the qubits that differ between memory and input). When $|c\rangle$ is $|0\rangle$, step 3 computes the amount of 0s on the memory state with a positive sign and with a negative sign when $|c\rangle$ is $|1\rangle$. Step 4 simply reverts the memory register to its original state by computing the inverse of Step 2. After the state preparation, the control register is measured. An input pattern similar to the stored patterns increases the probability of measuring $|c\rangle = 0$ and a input that is very distant to the stored patterns increases the probability of measuring $|c\rangle = 1$. The measurement probabilities can be seen in (6), where p is the number of stored patterns and $d_H(i, p^k)$ denotes the Hamming distance between input and the k-th stored pattern.

$$\begin{aligned} P(|c\rangle = |0\rangle) &= \sum_{k=1}^{p} \frac{1}{p} cos^2(\frac{\pi}{2n} d_H(i, p^k)) \\ P(|c\rangle = |1\rangle) &= \sum_{k=1}^{p} \frac{1}{p} sin^2(\frac{\pi}{2n} d_H(i, p^k)) \end{aligned} \tag{6}$$

4. Quantum Weightless Classifier

The Quantum Weightless Neural Network Classifier [9] (QWC) is composed of Probabilistic Quantum Memories acting as the network neurons. The model is devised by using an array of PQM instances capable of distance based classification. Each PQM instance, by itself, works as a single class classifier, being responsible for the classification of one of the classes in the dataset. The model does not demand any training in a sense that the neurons do not have to be iteratively adjusted to learn from the training patterns. The model classification procedure and the necessary set-up are detailed bellow.

4.1. Set-up Procedure

Despite not demanding any training, the Quantum Weightless Classifier requires an initial set-up procedure in order to perform classification tasks. For a given dataset with n classes, the model is constructed with n PQMs acting as neurons. The training samples must be divided and grouped by class. For each group, a new PQM is created and used to store all the samples belonging to that group, making in total n PQM instances, one for each class.

The set-up procedure consists in storing the training samples on their respective class PQM. The n PQMs together define a single classifier. The described setup process can be seen in Alg. 2. Once all the training samples are correctly stored, the model can perform the classification task by calling the PQM retrieval algorithm. Hence, the Quantum Weightless Classifier does not require any previous training in order to classify for all the classes present in a dataset.

Algorithm 2: Probabilistic Quantum Memory Classifier Setup

```
1 Initialize a PQM Classifier
2 for each class in dataset do
3     Create a new PQM and assign the class label to it
4     Store the class training samples on the PQM
5     Add the PQM to the PQM Classifier
6 end
7 Return the PQM Classifier
```

4.2. Classification Procedure

Once the training samples are stored the model is ready to classify new patterns. The classification procedure can be seen in Alg. 3. In order to classify a new sample, the Quantum Weightless Classifier must present it to all the PQM neurons which constitute the its network. Each PQM neuron performs its retrieval algorithm using the presented sample as input. Since each PQM hold the patterns of a specific class, each output will be the probability of the sample having similar features to the patterns of that specific class. Therefore, the PQM neuron which outputs the smallest expected value, $E(X)$, is assumed to be the one that correctly classifies the sample.

Algorithm 3: Probabilistic Quantum Memory Classifier Classification

```
1 for each PQM in PQM Classifier do
2     Run the PQM retrieval algorithm with input testPattern
3     Calculate the expected value $E(X)$ from the retrieval algorithm
        output
4 end
5 Return the label from the PQM Classifier with the smallest $E(X)$
```

5. Parametric Quantum Weightless Classifier

The PQM retrieval algorithm output is based on the calculated Hamming distances between the input and the stored patterns. This distance metric is not very reliable in every context as patterns too close to patterns from another class would likely be misclassified. In order to improve the memory output probabilities, we propose a modified version of the PQM.

A Parametric Probabilistic Quantum Memory (P-PQM) operates exactly in the same way as the PQM but with the addition of a scale parameter in the recover algorithm. This scale parameter is used when the Hamming distance between the input pattern and the patterns stored on the memory is calculated. That way, the amplitude of each vector can be adjusted according to the chosen parameter.

The modified U operator is given bellow, in which *param* is the given scale parameter and k is the number of patterns stored on the memory.

$$U = \begin{bmatrix} e^{\left(i\frac{\pi}{2\times k\times param}\right)} & 0 \\ 0 & 1 \end{bmatrix}$$

The QWC model demonstrates how the PQM can be used to perform pattern classification tasks. The proposed parametric version, P-PQM, allows the necessary adjustments according to a given application. Hence, by adjusting the scale parameter we can improve classification accuracy for each class in a given dataset.

We propose a modified version of the quantum classifier, the Parametric Quantum Weightless Classifier (P-QWC) model. It has the same basic construction as of the previously described QWC but is based on the P-PQM instead of the PQM. In the next section we evaluate both models performances.

6. Model evaluation set-up

We presented in this section the experiments conducted without loss of generality on a conventional computer through classical reduced versions of Algorithm 1. First, it is necessary to simulate the Probabilistic Quantum Memory classically. To do so, we simply followed the description of its recover algorithm as presented in Section 3.2. As for the storage mechanism, it is not needed outside the quantum context and could be highly simplified by just storing the patterns directly on the memory. Once obtained the PQM classical representation, the QWC can be evaluated by following the set-up and the classification procedures described in Section 4. The same procedure applies to the P-QWC model, with the only modification being the addition of a parameter to the Hamming distance calculation step.

To perform the experiments we used categorical and numerical datasets from the UCI Machine Learning Repository [10]. Details of the selected datasets can be seen in Table 1. All datasets were preprocessed in order to binarize feature values and deal with any missing values. The binarization process is required in order to simplify the PQM usage. It was done by transforming each possible value a feature could assume in its own separate feature in the sample vector. Datasets containing real numerical values were not considered in order to further simplify the process. Sample vectors containing missing feature values were not removed from the datasets. All the missing feature values were replaced by the value with highest occurrence for the corresponding feature in all the other samples of the considered dataset.

Following the QWC set-up algorithm, we stored the training samples in specific PQMs according to the classes they belong to. Then, we followed the

Dataset	Classes	Instances	Attributes	Missing Values
Balance scale	3	625	4	No
Breast cancer	2	286	9	Yes
Lymphography	4	148	18	No
Mushroom	2	8124	22	Yes
SPECT Heart	2	267	22	No
Tic-tac-toe	2	958	9	No
Voting records	2	435	16	Yes

Table 1: Datasets characteristics

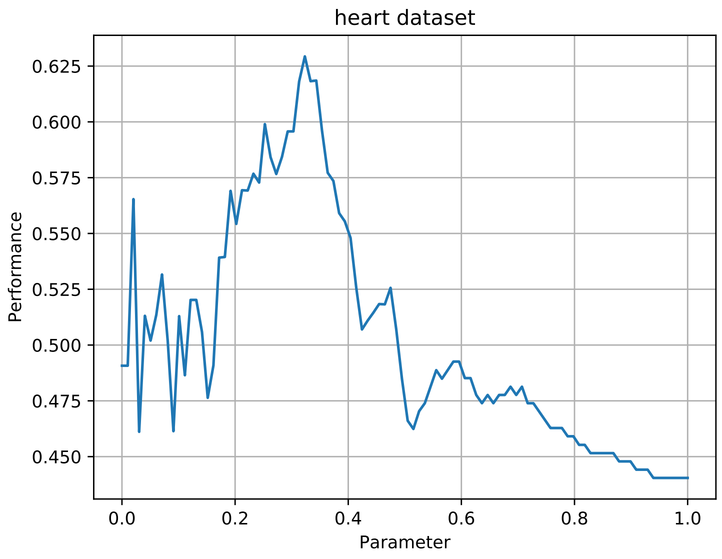
QWC classification algorithm. The model classification accuracy was evaluated by passing the patterns in the test set as input to each of the PQMs. The class of the PQM which output the lowest expected value was set as the evaluated pattern class. This procedure was done for each of the evaluated datasets. For the P-QWC model, we optimize and select the parameters which achieve the best test set accuracy for each P-PQM in the classifier. We tested 15 parameter values in the range $(0, 1]$ for each P-PQM.

7. Results

We verified that the P-QWC model performed better than the QWC over all the datasets. The parameter influence on the heart dataset performance can be seen in Fig. 1. For simplification purposes, the curve shows the performance obtained by using the same parameter value for all P-PQMs constituting the classifier. Considering the original PQM performance is equivalent to P-PQM with parameter 1.0, a considerable increase in classification performance was observed through parameter variation and even better performances are possible by choosing different parameters for each P-PQM.

The results obtained with the experimental set-up described in Section 6 can be seen in Table 2, where the accuracy of the QWC and P-QWC models can be compared against the results obtained using the k-nearest neighbors algorithm

Figure 1: Parameter impact on Heart dataset



(KNN). The accuracy values shown are the average obtained from a 10-fold cross-validation. The respective values for the standard deviation are included between parentheses. We choose KNN as a baseline comparison because, as well as our evaluated model, it is a non-generalizing learning model and does not require training. The KNN model was set to use uniform weights for all its points and the k nearest neighbors value was optimized and selected from values between 1 and 50.

To perform an appropriate comparison of the models, a nonparametric statistical test was employed. We used the Wilcoxon paired signed-rank test [11] with $\alpha = 0.05$ to verify whether there exist significant differences between the compared classifiers performances over the chosen datasets. Obtained results are statistically equivalent in Balance scale, Breast cancer, SPECT Heart and Tic-tac-toe datasets. KNN has better accuracy on Mushroom and Voting records datasets. P-QWC performed better on Lymphography dataset. The significant results are highlighted in the table.

The P-QWC has a performance equivalent to KNN in four out of the seven tested datasets and outperforms it in one dataset. The main advantage of the QWC is its memory requirements. While a RAM node memory grows exponentially with input size, the QWC memory size grows linearly. This memory advantage can allow the implementation of new weightless neural networks architectures.

Dataset	QWC	P-QWC	KNN
Balance scale	0.8111 (0.0666)	0.87 (0.2512)	0.8834 (0.0488)
Breast cancer	0.7309 (0.2639)	0.7380 (0.2604)	0.6970 (0.3797)
Lymphography	0.7829 (0.0815)	**0.8442** (0.1118)	0.7695 (0.0931)
Mushroom	0.886 (0.0919)	0.929 (0.073)	**1.0** (0.0)
SPECT Heart	0.4405 (0.2694)	0.8157 (0.1113)	0.7921 (0.181)
Tic-tac-toe	0.4542 (0.1199)	0.8309 (0.0678)	0.6714 (0.2989)
Voting records	0.892 (0.0575)	0.8966 (0.0545)	**0.9332** (0.0377)

Table 2: 10-fold cross-validation average accuracy per dataset

8. Conclusion

In this work we evaluated a Quantum Weightless Classifier on public benchmark datasets through classical computation. We also proposed a modification of the model by including a scale parameter to the PQM. The models are based on probabilistic quantum memories capable of distance based classification. In order to compare the performance of both models, we conducted experiments through the direct simulation of the quantum memory retrieval algorithm.

Both quantum models achieved an average accuracy similar to the performance obtained by the KNN algorithm in seven datasets. The proposed parametric quantum model performed better than its unmodified version in all datasets. The QWC model worst performance was in heart dataset with 37% loss in accuracy compared to P-QWC and 35% to KNN. P-QWC best performance shows an approximate accuracy gain of 7% over KNN in lymphography dataset while its worst result was in voting records dataset, an approximate 3% loss in accuracy compared to KNN.

The quantum model has shown satisfactory classification performance and was further improved by the parametrization of the PQM. This work performed the first empirical evaluation of a quantum weightless neural network and proposed a modification able to achieve a considerable improvement in the classification capabilities of the model. As future works we want to verify the model with different distance functions for the PQM and different architectures for the classifier construction.

Acknowledgements

This work was supported by the Brazilian Research Agencies: Serrapilheira Institute (grant number Serra-1709-22626), CNPq (grant number 420319/2016-6) and FACEPE (grant number IBPG-1578-1.03/16).

References

[1] J. Preskill, Quantum computing in the nisq era and beyond, arXiv preprint arXiv:1801.00862.

[2] A. W. Harrow, A. Montanaro, Quantum computational supremacy, Nature 549 (7671) (2017) 203.

[3] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, S. Lloyd, Quantum machine learning, Nature 549 (7671) (2017) 195.

[4] K. H. Wan, O. Dahlsten, H. Kristjánsson, R. Gardner, M. Kim, Quantum generalisation of feedforward neural networks, npj Quantum Information 3 (1) (2017) 36.

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

[6] M. A. Nielsen, I. Chuang, Quantum computation and quantum information (2002).

[7] C. A. Trugenberger, Probabilistic quantum memories, Physical Review Letters 87 (6) (2001) 067901.

[8] C. A. Trugenberger, Quantum pattern recognition, Quantum Information Processing 1 (6) (2002) 471–493.

[9] A. Silva, W. de Oliveira, T. Ludermir, A weightless neural node based on a probabilistic quantum memory, in: 2010 Eleventh Brazilian Symposium on Neural Networks, IEEE, 2010, pp. 259–264.

[10] D. Dheeru, E. Karra Taniskidou, UCI machine learning repository (2017). URL `http://archive.ics.uci.edu/ml`

[11] J. Demšar, Statistical comparisons of classifiers over multiple data sets, Journal of Machine learning research 7 (Jan) (2006) 1–30.