

Quantum Computing Post Pruning Approaches to Combat Overfitting in Decision Trees

L. Thil
B. Rodrigues de Miranda
V. Kotolup

Department of Data Science and Knowledge Engineering, Maastricht University
Supervisors: Dr. Menica Dibenedetto, Dr. Frank Philipson
Coordinator: Dr. Gijs Schoenmakers

CONTENTS		
I	Introduction	2
II	Problem Statement	2
III	Quantum Computing Background	2
III-A	What is Quantum Computing	2
III-B	Motivation for Quantum Computing . .	3
III-C	Qubits	3
III-D	Quantum Circuits	4
III-E	Quantum Algorithms	4
III-F	Near-Term Quantum Computers	5
III-F1	Overview of Available Quantum Computers and Road Map Overview	5
III-F2	Decoherence Issues	6
III-F3	Hybrid Quantum-Classical Devices and Ramifications .	6
IV	Related Work	6
V	Approach	6
V-A	Discrete-Time Markov Chain	6
V-B	Mapping a Tree to a Markov Chain . .	7
V-C	Discrete Quantum Random Walks . . .	8
V-D	Quantum Decision Rules	8
V-D1	Rule Encoding	8
V-D2	Data Encoding	9
V-D3	Measurement	9
V-E	Quantum Representation of Decision Trees	9
V-E1	Building the Tree on Quan- tum Hardware	10
V-E2	Identifying Post Pruning Op- portunities at Run Time . . .	11
V-E3	Robustness Against Noise and Decoherence	11
VI	Experiments	12
VI-A	Precision of Quantum Decision Rule . .	12
VI-B	Precision of Quantum Decision Trees .	12
VI-C	Identification of Post Pruning Opportu- nities	12
VII	Results	12
VII-A	Tree to Markov Chain Results	12
VII-B	Building Trees with Various Depths . .	13
VIII	Discussion	13
VIII-1	Depth of Quantum Tree and Memory	13
IX	Drawbacks	13
IX-A	Information Division for Translating Decision Rules	13
IX-B	Random Walk Distributions	13
IX-C	Using Grover's Search Combined with Von Neumann's Entropy	13
X	Conclusion	14
XI	Future work	14
XI-A	Quantum Reduced Error Pruning	14
	References	15

Abstract—Decision trees are popular models for making decisions in various domains, and there are numerous algorithms for building such trees. Overfitting tends to worsen the performance of these models, so techniques have been developed to post prune decision trees to mitigate this issue. This research focuses on incorporating quantum computing into the post pruning process to improve speed and performance by the use of quantum supremacy, a term referring to the ability of certain quantum algorithms to outperform any classical algorithm. A quantum representation of a decision tree is proposed and shown to work on a classification task, and a post pruning method using quantum random walks is put forward and shown to work with classical random walks. Finally, it is suggested how readers can build on this work in combination with existing work to further explore quantum post pruning.

Experiments looking at the limitations and advantages of such approaches have been performed, notably comparing a quantum markov chains decision process against synthetic datasets and faster tree traversal to identify post pruning opportunities in this representation. The results showed comparable classifications outcomes to classical computing approaches, but highlighted significant memory gains and speed opportunities in attempting post pruning operations.

I. INTRODUCTION

Quantum computing allows for new ways of looking at classical computational problems, and while it certainly has its drawbacks, like scaling issues [1], there have been a number of significant successful applications of quantum computers to outperform classical computers in certain tasks. The classical problem which this research aims to improve with quantum computing is post pruning decision trees. Widespread use of decision trees has driven much investigation into how to reduce overfitting issues when building them, a process which quantum computing has the potential to speed up and improve in terms of accuracy. Overfitting is an important issue to tackle when building decision trees because it leads to incorrect predictions on unseen data. Depending on the domain this can have a disastrous effect, for example, if an overfit decision tree is used to make medical decisions.

This paper is organised as follows. In Section II the problem statement is discussed followed by a description of background quantum computing information in section III. Section IV outlines related work done in this domain with Section V describing the approaches taken in this research. After Section VI describes the experimental setup, Sections VII and VIII outline and break down the experimental results, after which drawbacks are mentioned in Section IX and conclusions are drawn in Section X. Finally, promising avenues for future research are presented in Section XI.

The following research questions are addressed, which were designed to investigate how quantum computing can improve the post pruning process. How can a decision tree be mapped on to a quantum circuit? Does a quantum circuit representation of a decision tree allow for faster post pruning? Can existing post pruning methods be adapted to make use of quantum computing in a beneficial way? How can we identify post-pruning opportunities in a quantum decision tree?

II. PROBLEM STATEMENT

Decision trees are great machine learning tools which are still widely used despite the huge amount of recent research into more complex models like neural networks. They have the significant advantages of being both easy to create, thanks to various well researched tree generation algorithms like ID3 or C4.5 [2], and inherently interpretable [3]. Input data is fed into a root node, which has a threshold beyond which data is passed into a child node, and before which data is passed to a different different node. These nodes again split data into new nodes until leaf nodes are reached, which contain a classifications instead of thresholds. Once such a point is reached the input is classified based on the leaf node it reached.

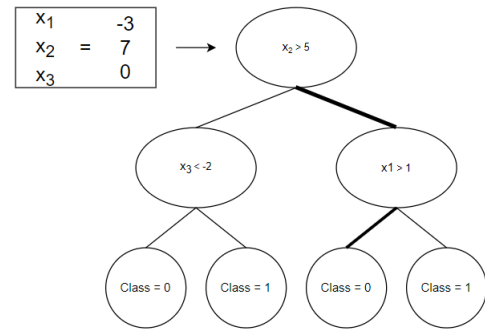


Fig. 1. a simple decision tree, which takes x_1 , x_2 , and x_3 as an input training example causing the tree traversal shown by the bold lines to a leaf node resulting in a classification of 0

As with many supervised learning models, decision trees tend to overfit to training data, which is apparent through the generation of too many decision nodes [4] which results in high accuracy on the training data but lower accuracy on unseen data. To combat this, post pruning algorithms have been developed to detect and remove these overfitting nodes after the tree structure has been learned. Quantum computing, discussed in Section III, has the potential to assist this post pruning process in two ways. Quantum algorithms have been developed which solve the same problems as classical algorithms but with a better time complexity, for example, Shor's algorithm performs quantum integer factorisation in polynomial time [1], which is a significant improvement over the current best classical approach, which is of sub-exponential time complexity [5]. This potential speedup of classical algorithms is the first way in which quantum computing can improve post pruning. The second is a possible improvement in accuracy on unseen data due to the inherent randomness of quantum bits. By forcing tree traversals to be random during training and post pruning, it may become less likely that the tree will perfectly learn the training data.

III. QUANTUM COMPUTING BACKGROUND

A. What is Quantum Computing

Quantum Computers are devices representing quantum states according to the spin of electrons being between two

defined levels: up or down. They are able to harness the properties of quantum physics in order to perform computation such as:

- **Superposition:** It is when several quantum states are combined in order to generate a new quantum state. It means that when measuring this state, information about the two original superposed states can be retrieved with their corresponding amplitudes, or proportions.
- **Entanglement:** is when a quantum state can not be described independently because it is related to at least another state even when apart with large distances. This means that representational information can be shared between these states simultaneously.

There are two main types of quantum computers:

- **Annealing:** surrounds the phenomenon of quantum fluctuation of random changes of a state's value to find the global minimum of an objective function among various potential solutions.
- **Gate-Based:** use a set of pre-defined gate operators to affect the spin of the electron. This type of quantum machine use qubits which can be seen as the equivalent of classical transistors.

In the case of this project we are using IBM QX which is gate based through the Qiskit [6] framework.

B. Motivation for Quantum Computing

Most of today's computing is performed using binary encoding of 0 or 1 using a Von Neumann's computer architecture [7]. Operations are done by encoding instructions on a tape and processing it through binary operators representing higher-level functions. The optimized use of transistors, more precise electrical components and better encoding of instructions allow to perform more computations. This process is referred as 'classical computing' and was described by Moore's law [8] stating that the amount of transistors in classical machines would double every two years. This held true for several decades since the 1970's, but lately it is observing a slower progression rate [9]. The explanation lies in the density of today's transistors, where their nano-scale dimensions makes it more difficult to control current-flow.

Therefore, as quantum computer posses lower computational complexity they could offer some speed advantages compared to classical devices. This is referred to as quantum supremacy, and opens the door to solving problems that classical running-time algorithms won't be able to solve such as breaking RSA cryptography [10]. In addition, because quantum computers represent a particle by harnessing quantum properties, quantum computers can model states that classical machines are not able to such as molecular interactions, teleportation mechanisms [11] [12] through entanglement or true-random events; and enhance certain applications such as sampling, physical dynamics, artificial intelligence, cybersecurity etc... The ability to perform superposition on the qubits also allow to represent more states than classical computers:

Even though quantum supremacy has not clearly been demonstrated on any real device [13], current development

could see the day of it becoming real more possible than ever. Disruptions in fields such as artificial intelligence, physics, engineering, chemistry, finance, communications, climate change, medicine (...), would power human progress at a faster rate. That is why it is still important to develop routines and algorithms able to be used on quantum computers for preparing the day it becomes fully usable. In the case of decision trees, tackling overfitting problems is an important step in that overall process and that is the reason of this paper.

C. Qubits

A qubit simulates the spin value of an electron and represents a quantum unitary unit of that information. Whereas a classical bit would designate its own value as being either a 0 or a 1, a qubit reflects the probability of the state as being a certain percentage '0' or '1'. As qubits represent quantum information, their representational states as designated using Dirac's notation. That way, what is seen as a classical bit being 0 is written as $|0\rangle$, and a bit being 1 is $|1\rangle$. These states $|0\rangle$ and $|1\rangle$ are called the basis states, and a qubit is a linear combination of these. A qubit $|\psi\rangle$ can be described as the superposition of these basis states such as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \quad (1)$$

Where α and β represent the probability amplitudes of these states in the complex set. These values follow the Born rule stating that the absolute squared amplitudes must sum to 1, and follow the Kolmogorov second axiom of probability theory that the sum of probability of each event equals 1, $P(\Omega) = 1$ [14][15].

The spin of a qubit is not binary either, can be seen as evolving in a three-dimensional space. This is explained in equation 1, where α and β have a total of 4 degrees of freedom as they are complex numbers, but due to the normalization one degree can be removed leading to a three dimensional space. That is why representing and measuring qubits and be seen by projecting the 4 dimensional qubit representation over a three dimensional space such as the Bloch Sphere [16][17]. Both α and β can now be represented as a coordinate system such that:

$$\alpha = \cos \frac{\theta}{2} \quad (2)$$

$$\beta = e^{i\psi} \sin \frac{\theta}{2} \quad (3)$$

To determine the state of a qubit, it needs to be measured. This operation consists of collapsing the state of the qubit to obtain the amplitudes α and β .

Additionally, a qubit can have multiple basis which are solely defined by the way the state is represented. By convention, the basis states of $|0\rangle$ and $|1\rangle$ are the one commonly used but many other can be set up as the Bloch sphere representation is three dimensional.

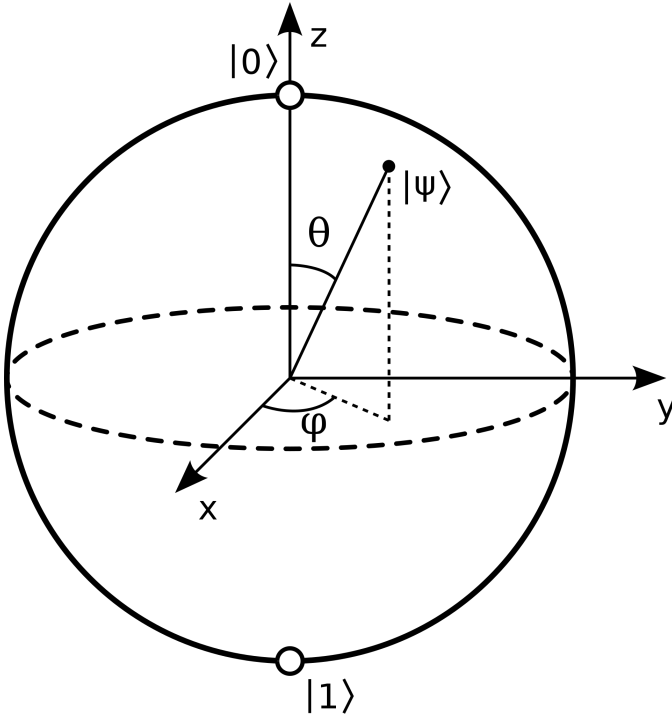


Fig. 2. Represation of a Qubit by Projecting of the Bloch Sphere.

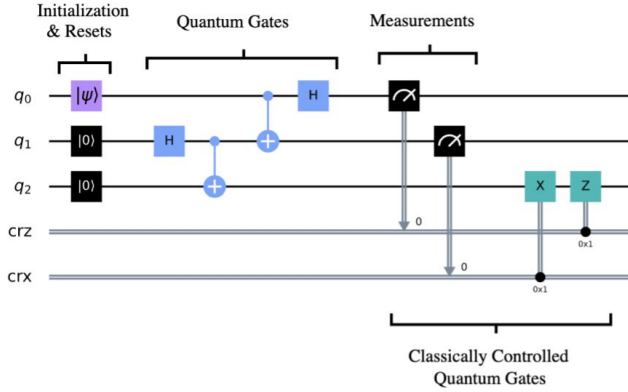


Fig. 3. Example of a Quantum Register. [18]

D. Quantum Circuits

Several qubits can be combined together to create a quantum circuit, also referred to as a quantum register. Gate operators can be applied on the qubits and the combined measurement of all states allows to represent a higher amount of possible states.

Quantum circuits can be represented using a variation of Penrose's graphical representation [19]. Each qubit state is represented as a horizontal line where quantum gate operators are applied. These gates can be linearly described by a matrix corresponding to the performed transformation. Additionally, a so-called classical register is introduced which allows to store classical measurements data from the qubits. A quantum

circuit composed of three qubits q_0, q_1, q_2 , two classical channels crz and crx as well as several gate operators be observed in figure III-D.

Some popular gates used in this work with references to their linear representation are the I or identity gate, the three X5, Y10 and Z7 also called the Pauli Gates[20][21], the Hadamard H8 gate which puts the Qubit in a state of superposition and the U gate 9 able to provide a parametrized custom transformation.

$$I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad (4)$$

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} = |0\rangle\langle 1| + |1\rangle\langle 0| \quad (5)$$

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} = -i|0\rangle\langle 1| + i|1\rangle\langle 0| \quad (6)$$

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} = |0\rangle\langle 0| - |1\rangle\langle 1| \quad (7)$$

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \quad (8)$$

$$U(\theta, \phi, \lambda) = \frac{1}{\sqrt{2}} \begin{bmatrix} \cos(\frac{\theta}{2}) & -e^{i\lambda}\sin(\frac{\theta}{2}) \\ e^{i\phi}\sin(\frac{\theta}{2}) & e^{i(\phi+\lambda)}\cos(\frac{\theta}{2}) \end{bmatrix} \quad (9)$$

The Hadamard H Gate puts the qubit in a superposition state, such that if an initial Qubit in the basis state $|0\rangle$ III-D is being transformed with a H gate, its measurement will result in a superposition state III-D such that $Q = \frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle$, giving equal probability of the state being in the basis $|0\rangle$ and $|1\rangle$. We can say that the qubit is roughly 50% in the basis state $|0\rangle$ and 50% in the basis state $|1\rangle$ III-D. The reason why it is not perfectly 50% is due to the true randomness nature of quantum computers and the phenomenon of decoherence approached in section III-F2. In order to obtain a clear result, it is highly recommended repeating the experiment and measuring many times.

E. Quantum Algorithms

Quantum Algorithms for gate-based quantum computers are a combination of linear operators referred as 'gates' on a quantum register. They harness properties of quantum mechanics to perform some computations with lower complexity than some of its classical counterparts like Shor's algorithm able to factors prime numbers in Polynomial time $O(d^3)$ compared to a classical exponential time of $O(\exp(d^{\frac{1}{3}}))$. Sometimes, these algorithms can have sub-linear time complexities such as Grover's algorithm which can find the solution of an unordered list in $O(\sqrt{N})$ whereas classical search algorithm have a computational complexity of $O(N)$.

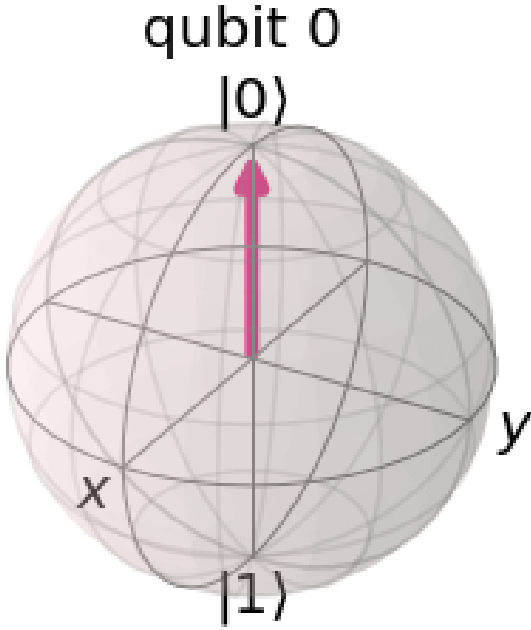


Fig. 4. Qubit in Basis State $|0\rangle$

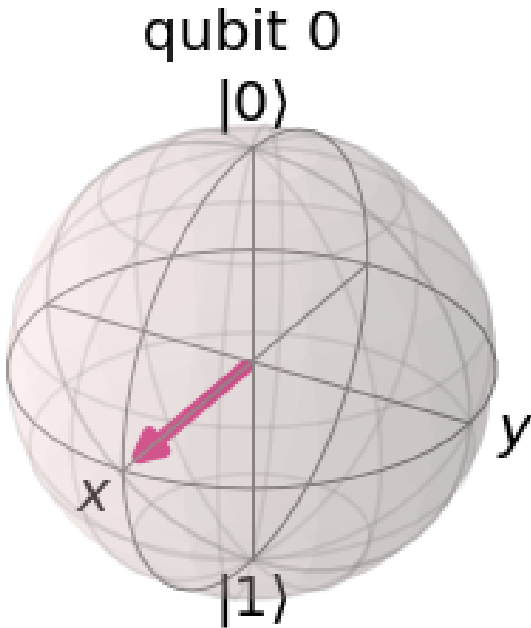


Fig. 5. Qubit in Superposition

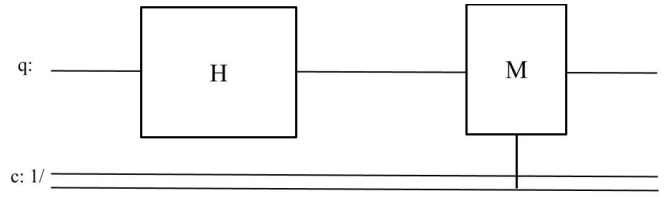


Fig. 6. Quantum Register Notation of Applying a Hadamard Gate and Measuring the Collapse of the Qubit and Storing the Result in the Classical Register

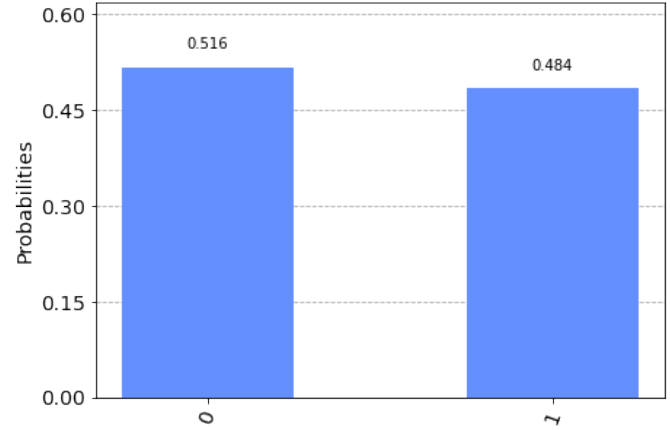


Fig. 7. Result of Measuring a Qubit in a State of Superposition After Applying a Hadamard Gate from the Basis State $|0\rangle$

F. Near-Term Quantum Computers

Most existing quantum computers do not offer any practical application due to noise issues. It is the hope that near-term devices would solve those problems, hence the term "near-term quantum computing".

1) Overview of Available Quantum Computers and Road Map Overview

Several quantum computers exist with various sizes and types. In the scope of this work, gate-based devices were used and it is through this scope that this project is about.

Their main issues lie in the noise that qubits gather, rendering many practical application not useful in the short term, but could prove real the near-term.

Many players manufacture practical quantum computers, but many other actors are actively building solutions that could one day become commercially available in the proper sense of the term. In 2019, IBM unveiled the IBM QX ecosystem of cloud-accessible quantum computers through the Qiskit API framework, in the hope of accelerating the development of quantum-based solutions. In the coming years, IBM advocates for a more 'hybrid' use of quantum computers, where classical computing works alongside quantum algorithms to use the best of both worlds as explained in their 2021 roadmap overview [22].

Other companies are working on optical computing or wave-based solutions such as D-Wave. Nonetheless, the ability to effectively deploy solutions from an outside public is a key

factor is developing quantum algorithms for a practical use. In that perspective, the approaches followed in this paper keep an lead-direction of proposing a hybrid approach to the reduce of overfitting of decision trees in hope that practicality will start mattering more in the times to come.

2) *Decoherence Issues*

A quantum state is described by a wave function representing its internal state. A qubit is coherent when its variations can be properly described through time from the phase function. As external factors come to affect the system of one or several qubits, and to entangle them, the description of the said system becomes more noisy with time by the loss of information until it becomes fully entropized. This is what is referred to as the decoherence of a quantum state.

This is why a quantum state needs to be isolated in order to preserve its coherence. A fully isolated quantum system would theoretically remain totally coherent over time, but it is never the case as particle interactions always happen at some level irrespective of the distance as described by Albert Einstein, Boris Podolsky, and Nathan Rosen in 1935 [23] and elaborated by Erwin Schrödinger in the same and following year [24][25]. That is why quantum computers need to isolate their quantum states as much as possible, namely by place the particles in a perfect vacuum at near perfect zero temperature. In the case of IBM's QX, they use a temperature of 15 millikelvin in a dilution refrigerator [26].

In other terms for our quantum register built on qiskit, it means that there is a short period of time where our qubit will keep its internal state coherent. Meaning that if we expect it to have be in a state of $\alpha|0\rangle + \beta|1\rangle$ with $\alpha=1$ and $\beta=0$, we would find that α and β will vary from their expected values. This is also shown by the superposition measurement in figure III-D where the amplitude probabilities of both states $|0\rangle$ and $|1\rangle$ are not equal as the system has noise. Thus, for a large amount of operators on a quantum register, the internal representation of the state becomes noisy with time and is one of the main drawbacks of current quantum computers being very noisy. This forces current practical applications to favor a low amount of qubits on the registers with a minimal amount of gates in order to keep the system sufficiently coherent to perform experiments.

3) *Hybrid Quantum-Classical Devices and Ramifications*

In the current state of quantum computers, decoherence of qubits is one of the major hurdles to let more practical programs perform. That is why current quantum computer reserach groups are tending to propose an approach of hybrid between classical and quantum machines. The main idea is to harness the best properties of each to let algorithms perform better. An example is using a classical device to load data and measurement from memory, and relying on the quantum machine and its quantum supremacy advantages to search through unordered sets faster, solve mathematical problems, or model chemical transformations to name a few. This approach has been recently outlined in IBM's latest report [22] regarding their quantum research roadmap.

This is particularly interesting because the proposed approach described in this paper to describe a classical tree to a quantum makes use of classical register to keep decision rules loaded into memory, before encoding them to a quantum state, perform operations and store the results into the classical register provided by the Qiskit framework.

IV. RELATED WORK

A discussion of the most popular classical post pruning algorithms presented by Esposito et al. [27]. While it does not relate to quantum computing, it is a solid starting point for understanding post pruning algorithms.

Lu and Braunstein describe a process for building quantum decision trees for classifying quantum data by using Von Neumann entropy instead of Shannon entropy as is done in the classical case [28]. The ideas presented are very promising, but differ from this work in that there is not a focus on overfitting or post pruning.

Farhi and Gutmann have outlined a method for speeding up decision tree traversal with quantum random walks [29]. Approaches in this work similarly try to speed up tree traversal with quantum random walks with the goal of using the faster traversals to speed up classical post pruning algorithms. The approaches proposed here attempt to overcome a shortcoming of the approach of Farhi and Gutman of, quantum tree traversals only being faster for certain problems.

Some years later, Farhi et al. extended this idea by using a quantum oracle and a continuous random walk [30] to speed up decision tree traversals. Unfortunately, the approaches are restricted to binary classification trees, and an attempt has been made to avoid this here. Further discrepancies lie in this paper omitting the use of a quantum oracle, and the use of a discrete quantum random walk instead of a continuous one.

In 2014, Songfeng Lu and Samuel Braunstein introduced a routine structure to generate a quantum version of classical decision trees [28]. They outlined several ways to build and chain decision rules from data, and ways to perform pruning by using Von Neumann's Entropy [1][31] through a bottom-up approach. In our case, we would be starting from an already constructed tree as we require to perform post-pruning entailing a top-down approach.

V. APPROACH

A. *Discrete-Time Markov Chain*

A discrete-time Markov chain is a sequence of random variables $X_0, X_1, \dots$ that satisfy the Markov property - the current state depends only on a finite fixed number of previous states. In our project we use first-order Markov chains, that is the current state depends only on the previous one:

$$Pr(X_n = x \mid X_1 = x_1, \dots, X_{n-1} = x_{n-1}) = Pr(X_n \mid X_{n-1}).$$

A Markov chain can be described by a transition matrix. For example the following Markov chain

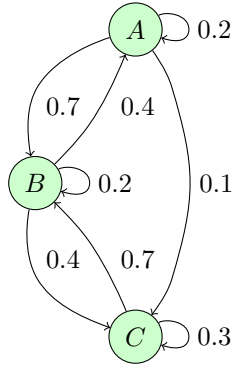


Fig. 8. Markov chain example

can be represented as a transition matrix

$$T = \begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.4 & 0.2 & 0.4 \\ 0.0 & 0.7 & 0.3 \end{bmatrix}. \quad (10)$$

Using this transition matrix we can compute with what probability we will be in a state after k steps. Assume that we can start in all states with equal probabilities $v_0 = [\frac{1}{3} \ \frac{1}{3} \ \frac{1}{3}]$ and define $v_k = v_{k-1} \cdot T^k$. Now we can compute with what probability we will be in states 1, 2 or 3.

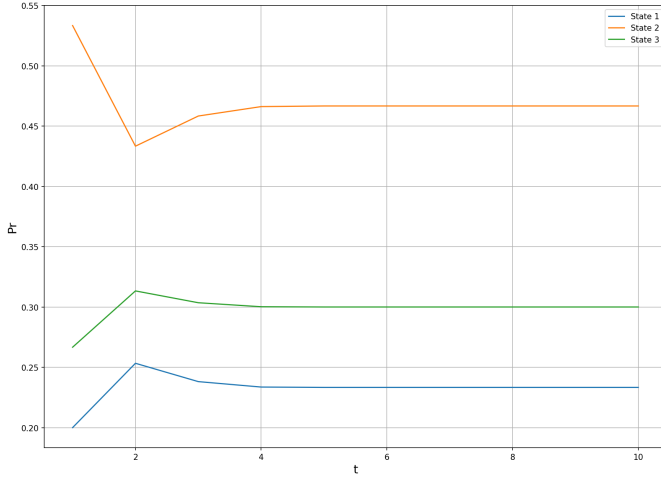


Fig. 9. States probabilities.

In this small example, we have achieved the equilibrium in a few iterations, but with larger transition matrices, it can take drastically more iterations.

B. Mapping a Tree to a Markov Chain

The motivation to map a tree to a Markov chain is to get a transition matrix T to perform random walk, assuming that at the beginning, that we can be in all nodes with equal probabilities.

To make it easier to describe this approach we only use "less than" relational operator in decision nodes. Let $X_{train}^{i,n} =$

$\{x \mid x \in X_{train}, x_i < t_n^i\}$, where x_i is the i^{th} feature from the vector x and t_n^i is the decision boundary of the node n . That is, a set $X_{train}^{i,n}$ is a collection of samples that satisfy the inequality in a decision node n . We set probabilities for as follows:

for inner nodes:

$$P_l(n) = \frac{|X_{train}^{i,n}|}{|X_{train}|},$$

$$P_r(n) = 1 - P_l(n),$$

for leaves:

$$P_{parent}(n) = 1,$$

$$P_{stay}(n) = 0,$$

where $P_l(n)$ and $P_r(n)$ are probabilities to go left or right from a node n , $P_{parent}(n)$ is the probability to go back to the parent and $P_{stay}(n)$ is the probability to stay in the node n .

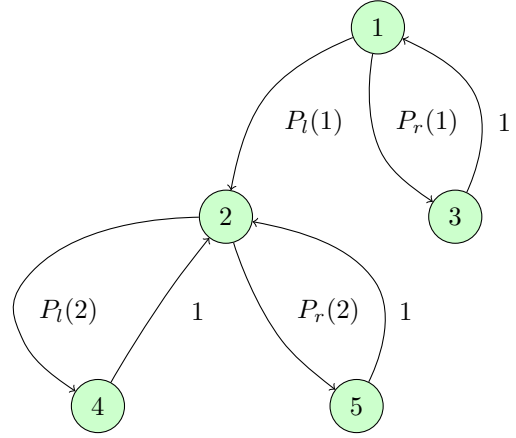


Fig. 10. Markov chain representation of a tree

After assigning probabilities for all nodes we get a transition matrix T . We set $v_{init}^T = [(1/N) \times N]$, where N is the row size of T . To get possible positions after k steps we compute $v_k = v_{k-1} \cdot T^k$. Ideally, we want $v_k = v_{k+1}$ meaning that T^k is in its equilibrium. For our experiments we set $k = 128$. Using v_k we prune a node n s.t. $p(n) < threshold$ and the value $threshold - p(n)$ is less than for other nodes. We set $threshold = \frac{1}{\text{number of active nodes}}$ which helped to force even distribution between nodes.

The main performance bottleneck in this approach is the random walk. We have to multiply the transition matrix T with itself enough times such that $T^k \sim T^{k+1}$. Even for an even k , it can be problematic to compute T^k . Suppose that a decision tree contains n nodes then it means that T consists of n^2 entries. Of course, most of these entries are zeros and this is a sparse matrix but more entries we have the bigger k should be to achieve the equilibrium. Moreover, if k is large then T^k becomes a dense matrix.

One of the possible solutions is to use the quantum walks which gives us a substantial speedup. We assume that the Szegedy quantum walk (SQW) can help to speedup this

approach in term of computational cost. The SQW makes it trivial to map a transition matrix to a quantum circuit. Moreover, it was shown that coined and Szegedy walks are equivalent [32]. That is, it is not a problem that the SQW works with probabilities on edges.

To define the SQW [33], we define the normalized states first

$$|\psi_j\rangle = \sum_{k=1}^N \sqrt{(T)_{jk}} |k, j\rangle, \quad j = 1, \dots, N.$$

We define the projection

$$\Pi = \sum_{j=1}^N |\psi_j\rangle \langle \psi_j|,$$

and the shift operator

$$S = \sum_{j,k=1}^N |j, k\rangle \langle k, j|.$$

Then the one discrete-time step is

$$U = S(2\Pi - 1).$$

C. Discrete Quantum Random Walks

The discrete classical random walk (CRW) is simply a traversal of a Markov chain where one state transition is made each time step. The discrete quantum random walk does the same thing, but takes advantage of quantum entangled Markov chain states to ensure a faster traversal than in the classical case, with the CRW walk having time complexity $O(N)$ and the QRW having time complexity $O(\sqrt{N})$ [34].

The position of the QRW is represented by a vector H_p made up of a number of basis vectors in Hilbert space which are each a single position in the Markov chain. If there are N Markov chain states, then there are N basis vectors $|i\rangle$, each being a Markov chain state, where $0 \leq i \leq N$. Additionally, a coin vector H_c , made up of a different set of basis vectors in Hilbert space, updates the superposition of the position based on its own superposition. The tensor product of the state of the position and state of the coin, shown in Equation 11, are the quantum space which is traversed by the QRW.

$$H = H_p \otimes H_c \quad (11)$$

To evolve the QRW, a unitary operator U is applied to H . U is the product of two operators. The first, S , shifts the position vector based on the coin state. In the case of decision trees, the shift operator changes the basis vector $|i\rangle$ representing a node in the tree, or alternatively, a state in the corresponding Markov chain, into $|j\rangle$, one of the children of that node depending on how the coin state collapses when measured. In the absence of measurement, the shift operator updates the superposition of the position based on the superposition of the coin. The second, C , is a coin operator which updates the coin state based on the transition matrix of the Markov chain. The superposition of the coin which results from this accommodates the simultaneous traversal of each possible path

in the Markov chain because the coin determines the position update. This operator also entangles the qubits in the system such that the collapse of one of the qubits alters the collapse probabilities of the rest of the qubits. The resulting operator U , which is shown in Equation 12 does not change, but acts on H , which does change as a result [34].

$$U = SC \quad (12)$$

Starting in state $|i\rangle$, which is the state corresponding to the root of the decision tree, U can be applied t times in succession without measurements in between applications such that H is all possible paths after t steps, which, upon measurement, will collapse into one of the possible paths. The coin reflects the transition probabilities from the current superimposed states to the superimposed states at the next time step. For a state which is not superimposed, the coin contains the transition probabilities to the states reachable from the current state, in this case, the children of a node. For superimposed states, the coin contains the transition probabilities of each base state in the superposition to each of the states they connect to.

$$H = U^d |\psi(0)\rangle \quad (13)$$

Determining which state the QRW ends up in can be done by measuring the system after a number of time steps, which forces the system to collapse to a single basis vector corresponding to a node in the tree. QRWs are fast because all paths can be traversed at the same time and just a single measurement of the system state needs to be performed to determine the resulting state of the Markov chain [34].

D. Quantum Decision Rules

Essentially, a decision tree is an ordered set of decision rules such that after being applied on an initial input, maps to a defined classification output. A numerical decision rule can be defined as a function $R(x, r)$ (14) where x is the input value in the range $[0, 1]$, r is the rule boundary with x and r being numerical. The output of R yields a binary classification, and the range of input values is normalized for that example.

Such a rule can be defined as the function

$$R(x, r) = \begin{cases} 1 & x > r \\ 0 & x \leq r \end{cases} \quad (14)$$

1) Rule Encoding

The aforementioned numerical rule can conveniently be applied to the representation of a quantum state.

In the case of a qubit, we can use the amplitudes α and β to quantify the state of the qubit as being in the basis $|0\rangle$ and $|1\rangle$. That way, it is possible to represent a binary decision rule over a qubit put in a superposition state by matching proportions of the basis when that qubit is measured. A decision rule can be encoded through the mean of a parametrized U-Gate where the angle is the desired encoding rule, putting the qubit in a superposition state with the right proportions. This implies that the desired encoded amplitude for the decision rule must

lie between 0 and 1 as it is the defined range of the summed amplitudes in equations 1, 2, 3 and matching the second axiom of probabilities [15].

The U-Gate is applied on the desired qubit by the function 15 where r is the desired decision rule to encode. The constant 0.5 represent the amplitude of a superposition with equal amplitudes α and β of the basis states. That way, the U-Gate will modify that equilibrium between the two amplitudes to increase one and decrease the other in order to favor of one of two basis states during the measurement collapse of the qubit.

$$U_{rule}(r) = \pi \times (0.5 - r) \quad (15)$$

2) Data Encoding

In the same manner, custom values can be encoded in the qubit before encoding the decision rule. That way an encoding of the input x can be performed using a U-Gate with angle from equation 16.

$$U_{data}(x) = \pi \times x \quad (16)$$

That way, it is possible to map a classical decision rule to a quantum decision one by turning it into a set of continuous values with its decisions rules. For example, if a classical decision rule is defined over a known range, it is possible to normalize it and draw from it the corresponding angle for encoding the data and the decision rule on the qubit. This is done with a limitation: that this current approach only considers two possible outputs.

3) Measurement

After the decision rule and the data value have been encoded, the classification for that quantum decision rule can be produced by measuring the qubit's state. The resulting amplitudes of α and β from the collapsing qubit give the proportions to which basis $|0\rangle$ or $|1\rangle$ the state was in. Each of these basis state correspond to one of the two outgoing branches of that node. This is found by taking the highest amplitude out of α and β as the final classification. For example, if there are two possible labels A and B respectively associated to the amplitudes of α and β , we obtain that if $\alpha > \beta$ then the classification outputs A, else B.

The binary result of this classification from the qubit can in practice be stored onto a classical register which is described in subsection V-E1.

E. Quantum Representation of Decision Trees

As the qubit decision rule represent the equivalent of a node in a decision tree, it is possible to chain several of them together to create a quantum decision tree version of its classical counterpart.

The top-down approach of this work requires to start from an existing classical decision tree. The implemented quantum decision rule requires the data to be normalized in order to encode the input value x and a decision rule for that feature into the decision qubit.

SPECTRUM

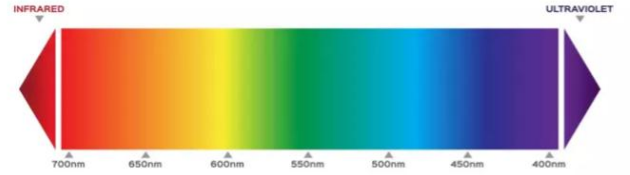


Fig. 11. Wavelength Spectrum of Colors

Furthermore, the implementation of this qubit decision rule only allows in its current state a binary output, thus a branching factor of 2 per node. That is why the conversion of a classical decision rule with a branching factor larger than 2 must be divided into several rules all chained together.

An example of feature encoding could be colors: blue, green and red. This gives a decision rules three possible output values about what is the designed color, and thus requires two qubits to encode the final output. A way to separate them is to perhaps look into what a color is: an observable wavelength where the label blue (420nm), green (570nm) and red (700nm) are attributed to as seen in figure V-E.

There is a risk of misclassification if the data is not rightfully encoded. We need to devise which decision rule should be used first. For instance, light green is located roughly at equal distance between sharp red and sharp blue with full intensity. Therefore, blue and red have the highest distance measure between them whereas they are each smaller with green. If we start applying our first decision rule to differentiate between blue and red, our decision boundary will lie exactly in the middle: on green. Thus, whatever will be our next decision rule to differentiate green from red or blue, we'll always have on average 50% of the green misclassified, except if we add a new decision rule for both red and blue.

Therefore, it is possible to design two decision rules in the following manner:

- 1) First Qubit: Classify whether the wavelength is smaller than 600nm, then the color is red and we can stop here. If it is bigger than 600nm, then the color is either green or blue and we flip the first qubit with a X gate in order to put it in the $|1\rangle$ encoding the information of this classification.
- 2) Second Qubit: Classify whether the wavelength is smaller than 520nm, if true then stop. If it is larger than 520nm then the color should be blue and we flip the second qubit to put it in the basis state $|1\rangle$ with the X gate.

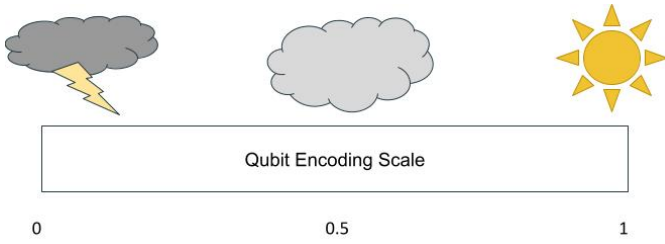


Fig. 12. Example of Continuous Representation of Weather Conditions and its Corresponding Encoding of Amplitudes in a Decision Qubit

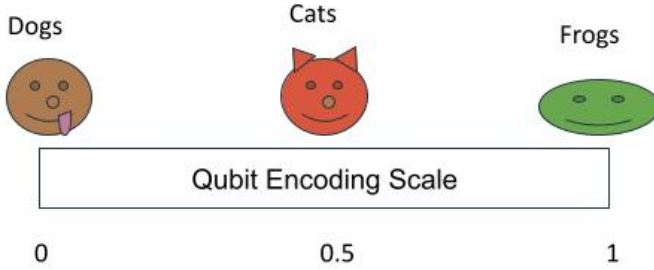


Fig. 13. Example of Continuous Representation of Discrete 'Animal Species' Feature and its Corresponding Encoding of Amplitudes in a Decision Qubit

Finally, after this classification using two qubits for three possible outcomes and two decision rules we obtain the following system measurements per color:

- $|0\rangle|0\rangle$ yields the color red
- $|1\rangle|0\rangle$ yields the color green
- $|1\rangle|1\rangle$ yields the color blue
- $|0\rangle|1\rangle$ is not attributed

A routine to build quantum decision rules from continuous numerical data is as follows:

- 1) Normalize data and decision boundaries
- 2) Encode decision rule in qubit
- 3) Encode data in qubit
- 4) Measure qubit state collapsing in $|0\rangle - |1\rangle$ basis

Regarding more discrete cases, we can have a feature describing weather conditions with three possible values: rainy, outcast and sunny. Rainy weather entails the presence of cloud, while no rain and no cloud means sunny. We can represent these values on a continuous scale which can be translated over amplitudes for our qubit encoding as seen in figure V-E.

For non numerical values, it is possible to create a numerical encoding those. In the case of an 'animal species' discrete feature having for possible values 'cat' or 'dog' can be represented as 0.0 and 1.0, with a decision rule set at 0.5 as seen in figure 13.

It is harder to properly discretize and rightfully measure the output of the measurements. If we use a branching factor of 2 in the tree, we'll have to constrain our classification at each node between two outputs. Thus for a 3-choice classification, we'll need to use 2 Qubits at different depths.

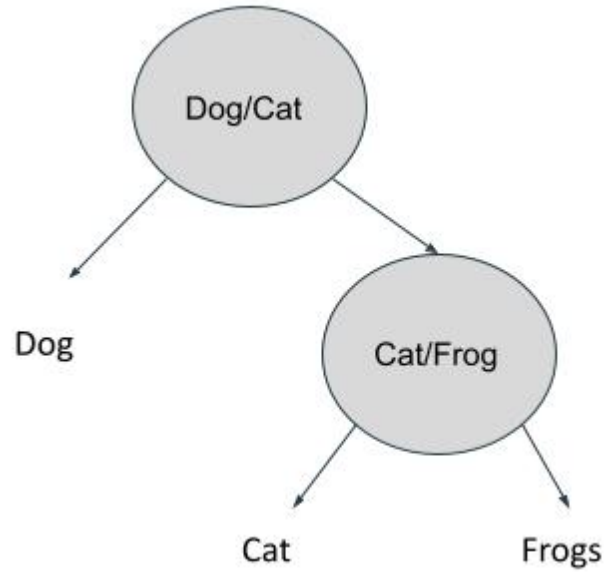


Fig. 14. Creation Example of Additional Rules from a Discrete Feature with a Decision Rule Having a Branching Factor of 3.

What is the best architecture and routine to create these additional rules? The implementation that we propose is to classify two possible outputs starting by the shortest encoding distance between them and chain the approach with the following rules. The resulting tree from the representation in figure 13 would lead to a tree with the following structure as seen in figure 14.

In short, after devising a quantum representation of the tree by transforming discrete values into continuous ones, we can use an approach like quantum random walks, quantum markov DP or even Grover's search algorithm to rearrange subtrees acting as decision rules to minimize the average running amount of nodes.

1) Building the Tree on Quantum Hardware

In order to form a quantum decision tree, several qubit decision rules described in the preceeding sections must be chained together. This comes with the constraint of data representation of the possible classification labels. Indeed with this approach, each qubit in the system can encode itself with two possible classification states due to the measurement basis in $|0\rangle|1\rangle$. This is a constraint that has been touched upon in the color and weather example in figures V-EV-E1413. This implies that the qubit decision rules must be used such that each measurement in the qubits during the tree traversal, outputs the correct classification according to the final state of the entire quantum register. This part can be modelled through the tree traversal as the information gained towards the final classification encoding according to the qubit flipped. Therefore, during the conversion from a classical to quantum tree, the division of the qubit flips must be able to properly attain the correct output encoding that is targeted. This can effectively be done by sometimes flipping qubits not being the one where our decision rule and data are being encoded

for that classification node.

As the process of performing a classification on a qubit requires it to become decoherent, it is expected to not be usable anymore to perform additional gate operations. Fortunately, in a recent Qiskit update, a new 'reset' gate has been introduced which puts the Qubit in the coherent $|0\rangle$ state where additional measures can be performed. That way, after each measurement, the final outcome is stored into the classical register and allows to chain several decision rules on the same qubit after being reset following its measurement. This allows to navigate through the tree while minimizing the amount of qubits needed to run the circuit and containing the decoherence process. This reset gate is denoted as *RESET*.

Such a circuit can be formulated as such: For n possible classification labels, use m qubits such that $m = \text{ceil}(\frac{n}{2})$ with a classical register of size m and *ceil* being the rounding up to the nearest integer function.

All of the classical decision rules can be stored in $D_i = \{R_{i,j}, R_{i,j+1}, \dots, R_{i,j+k}\}$ being the set of decision rules in the tree at depth i , containing the rules $R_{i,j}$ with j being the index of that decision rule at that depth, $k = 2 * i$, and K being the total height of the tree. Let Q_i be the qubit of our quantum register at index i with $i \leq m$ and let C_i be the memory slot at index i corresponding to the qubit of the same index. A rule $R_{i,j} = \{t, r\}$ is a set where t is the target qubit on which the rule is encoded to, and r is the rule threshold value.

A pseudo code of this procedure is described in figure V-E1.

Algorithm 1 Quantum Decision Tree Pseudo Code

Ensure: $m = \text{ceil}(\frac{n}{2})$

$x \leftarrow \text{input}$

$i \leftarrow 0$

$j \leftarrow 0$

while $i < K$ **do**

$t \leftarrow R_{i,j}.t$

$r \leftarrow R_{i,j}.r$

$Q_t \leftarrow Q_t \otimes U(U_{\text{rule}}(r), 0, 0)$

$Q_t \leftarrow Q_t \otimes U(U_{\text{data}}(x), 0, 0)$

$C_t \leftarrow \text{round}(\text{results}(M \otimes Q_i))$

$Q_t \leftarrow Q_t \otimes \text{RESET}$

$j \leftarrow C_t + i \times (2 \times i)$

$i \leftarrow i + 1$

end while

$i \leftarrow 0$

$\text{result} \leftarrow \text{Vector}(\text{size} : m)$

while $i < m$ **do**

$\text{result} \leftarrow C_i$

$i \leftarrow i + 1$

end while

return result

This approach allows for an implementation working on

a quantum computer that can be easily compared with its classical counterpart. Furthermore, this representation makes it possible to map to a Markov chain graph described in the earlier section, and more visibility in its display. With the help of Qiskit, it makes it a usable approach to implement and test against classical trees.

2) Identifying Post Pruning Opportunities at Run Time

The quantum version implementation of the classical decision tree allows to navigate through it and perform experiments in order to identify nodes to post prune. One way to do this is to follow the progressive entropy of the tree according to the input data in order to measure the information gain of the final classification while going through the decision nodes. The entropy of a classical system is defined by Von Neumann's entropy and was used by Songfeng Lu and Samuel Braunstein in their quantum tree implementation. [28]. The Von Neumann's entropy is defined in equation 17, where $\rho = \sum_{i=1}^{n_{dt}} p_i^{dt} |y_i^{dt}\rangle \langle y_i^{dt}|$ is the density operator of state Y and p_i^{dt} are the individual states of the system being involved in the tree at the decision node t .

$$S(\rho) = -\text{tr}(\rho \times \log(\rho)) \quad (17)$$

That way, while going down the tree it is possible to measure the overall entropy of the system; and if it is observed that it is not evolving according to a certain threshold, the decision can be made to cut that subtree starting at this node.

3) Robustness Against Noise and Decoherence

A motivation for this approach is the sustainability and robustness increased towards quantum decoherence of the circuit. Indeed, segmenting the continuous space into sub-instances allows for a higher data sparsity in the representational space making it less sensitive to noise when the encoding lies near the decision boundary.

Because of the sensitivity of the data points laying close to the bounds and the true randomness aspect of measuring a qubit, it happens that some state may be misclassified in those conditions. This on one hand could prevent some level of overfitting by cancelling the wished classification if a data encoding lies too close to the encoded boundary during the collapse of the state. On the other hand, it poses an issue regarding our lowest and highest bound of the normalization process of our data to create appropriate values to encode in the qubits. Indeed, data point of 0.0 with a decision boundary of 0.0 will get misclassified with equal probabilities as being in the basis state $|0\rangle$ or $|1\rangle$. To prevent this, the classical data must be normalized by adding a safety boundary a relative in aspect to the decoherence levels of the hardware used. In an experiment, we find that this value has an order of magnitude of 10^{-2} for the hardware we have used with the default Qiskit Aer simulator [?]. It is important to test before each utilization the areas of sensitivity regarding decoherence and add safety margins higher than the found values. We suggest using values with a magnitude of $a = 0.05$ for common Qiskit experiments.

VI. EXPERIMENTS

A. Precision of Quantum Decision Rule

Because of decoherence effects, normalization of the encoded classical data rules and the true randomness nature of quantum states, performing classifications on a algorithm running on a qubit will introduce some errors. These errors especially lie when the encoded data lies near the encoded decision boundary on the qubit. This cause the collapsing state to sometimes misclassify the final outcome at this node. The question seeks to see to which degree this misclassification happens. For that, several values at different bounds have been tested on the encoded qubits.

For the first case, we try with a decision boundary and encoded input data set at 0.3. The classification is run 1000 times and we find that in the lowest value in the misclassified cases yielded an amplitude level of $0.29 - 0.1 \times 10^{-14}$, and the highest one of $0.31 + 0.1 \times 10^{-14}$. This means that the encoded data has an order of error of 10^{-2} .

On boundary and data encoding of 0.0, the same results have appeared regarding the misclassified outputs. Same magnitude has been observed for 1.0. This means that when normalizing our data from classical to quantum, the normalizing bounds should account of a margin $a \leq 10^{-2}$ on each extremity.

B. Precision of Quantum Decision Trees

Various tests have been done with real classical tree. Among those were the reliability of chaining rules together over the same qubits during reset operations or verify that the states would remain coherent long enough in order to perform our classifications during the tree traversal. Figure 15 shows an example of a encoded tree with 3 different features to build rules from: a, b and c.

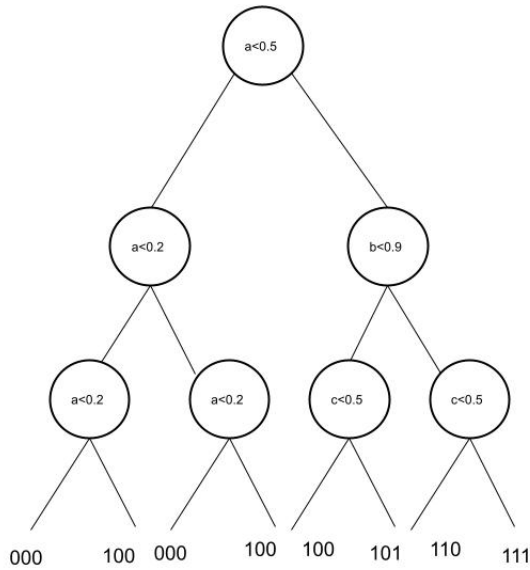


Fig. 15. Example of Encoded Depth 2 Tree

C. Identification of Post Pruning Opportunities

A way to identify post pruning opportunities was in using Von Neumann's entropy formula to measure the gained information after the performed classification at each node of the overall quantum system. The cut would be done based on a certain threshold according to how much information would be gained or lost, in an attempt to prevent overfitting.

VII. RESULTS

A. Tree to Markov Chain Results

We have found that the approach described in section V-B works well on synthetic datasets generated using **sklearn** library. The result of the algorithm are shown below.

Decision Boundaries on Test Data. Step 0.

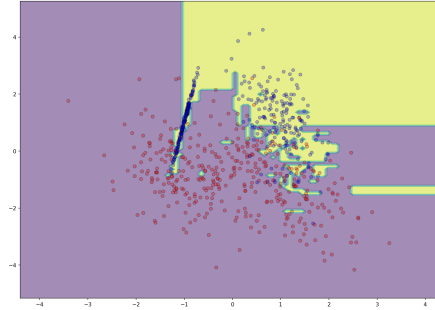


Fig. 16. Decision boundaries before pruning.

Decision Boundaries on Test Data. Step 220.

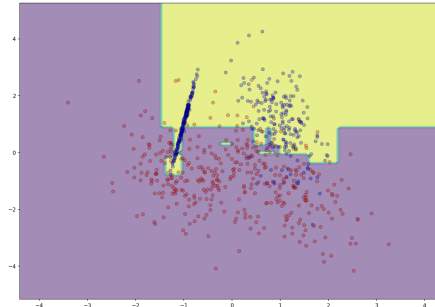


Fig. 17. Decision boundaries after pruning 87% of nodes.

Performance. Step 0.

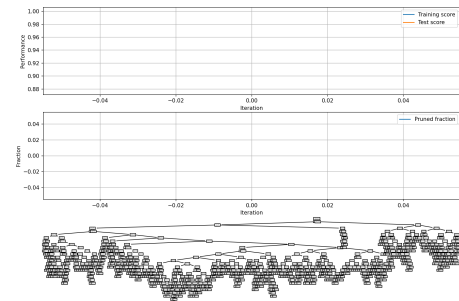


Fig. 18. The tree before pruning.

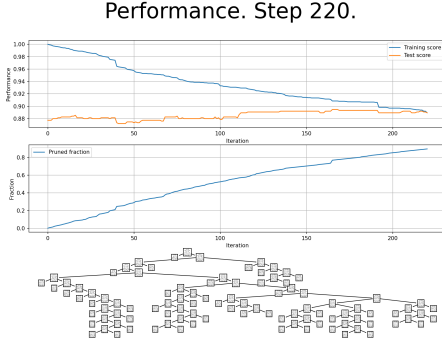


Fig. 19. The tree after pruning 87% of nodes.

B. Building Trees with Various Depths

Various trees were built and tested against according to different sets of rules, datasets, and dimensions. Figure 15 show an example done for the sake of simplicity where each letter correspond to a decision rule of a respective qubit: 0, 1 and 2. As it can be observed, each decision rule will change the encoding of the corresponding target qubit after the classification has been made on that node. The final possible outputs can be observed at the leafs. It can be noted that for three possible qubits to encode the possible states, certain combination are not represented such as '001' or '011'. Because of the precision issues that could potentially happen on a qubit, these not represented states could still be produced as a classification output in case of error. This means that the tree may produce errors that are not listed, a kind of errors that classical trees generally do not make.

VIII. DISCUSSION

1) Depth of Quantum Tree and Memory

The use of the RESET operator allows to constraint the decoherence effect of a qubit by not letting it live too long between the performed decisions. This is allowed by the amount of qubits used being equal to the amount of bits needed to encode the possible output labels. The decision rule threshold are kept in the classical part of the program in our implementation and are passed during the loop to be encoded into the quantum tree. A reason we argue that this is a valid process, is that classical tree require to read the data output by each classification node from the memory before moving the pointer to the next memory chunk relating to the following node. This is similar in the quantum tree approach, except that the data read by the pointer from the memory is fed through a quantum representation of that classical decision tree.

This RESET operator can be argued upon further, especially regarding the choice of using superposition to distinguish between two possible choices in each of the decision rule. The reasons are because it was decided to use a branching factor of 2, but we strongly believe there could exist way to encode more information in the qubit by using additional basis states in the measurement and thus allowing for a greater branching factor.

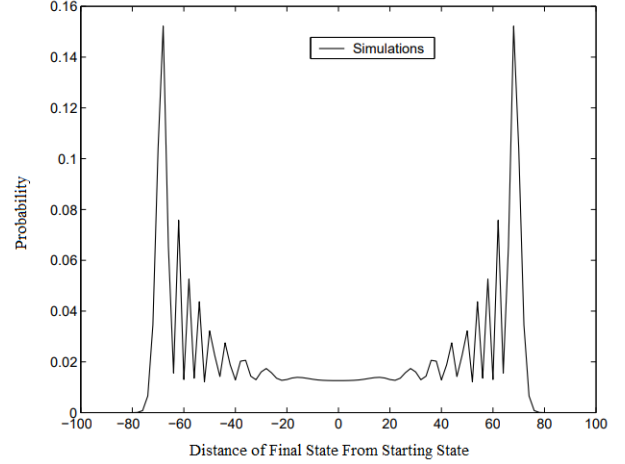


Fig. 20. The Distribution of States Reached With QRW on a Straight Line [34]

IX. DRAWBACKS

A. Information Division for Translating Decision Rules

In section V-E, we devised a routine to translate decision trees made with different types of decision rules (branching factor, data encoding...) to its quantum counterpart. An argument could be made whether this approach is sustainable in theory. An argument is favor is that everything at the smallest scale is quantum. Thus macro scale events or objects, if looked upon at small enough scales, could be turned into quantum continuous values too in order to be classified. In other words, we saw how we can convert discrete values such as colors or weather into continuous ones, but if we have some other discrete feature that seems too different to be really distinguishable, then we can look at how they are made and find their quantum representational differences.

A counter argument regards to its practical implementations, but as a first approach for building quantum decision trees in a top-down manner from an already built classical tree; there must be more elegant and robust manners to tackle this problem.

B. Random Walk Distributions

Another potential issue is the difference between the distributions of CRWs and QRWs. QRWs propagate faster to positions states further from the start, so states which are rarely reached by CRWs are reached more frequently. It is likely, therefore, that the nodes pruned in the experiments done in this paper will differ from the ones which would be pruned if a QRW was used. An example of a distribution after a simple QRW is shown in 20.

C. Using Grover's Search Combined with Von Neumann's Entropy

In order to make an effective use of Von Neumann's entropy to find post pruning opportunities, it would be convenient to

use Grover's search in order to find them in sub linear times, making an applied use of a quantum's supremacy aspect. Unfortunately, in the scope of this project, implementing this algorithm for this case appeared harder than thought, but would offer a good approach as was proposed by [28] when partitioning their data to create their decision rules. Indeed, it could be possible to revert the process regarding a top-down approach.

X. CONCLUSION

We have shown that the Markov chain approach V-B works on a classical computer but the size of a transition matrix in practical cases and the fact that T^k becomes dense when k is big enough makes this approach impractical. We expect that using the Szegedy quantum walk can greatly reduce the computational complexity.

Furthermore, we have shown a practical implementation of a quantum decision tree from a classical one by an encoding routine of normalizing the data according to a safety boundary against decoherence effect, and a method in creating a set of decision qubit from the rule nodes of a classical tree able to correctly model it.

The identification of post pruning opportunities need to make an effective use of Grover's search algorithm in order to be truly effective. Nonetheless, this implementation allows for a reformulation in order to use it through a Markov chain combined with the random walk algorithm. The hybrid implementation between classical and quantum hardware makes this approach able for near-term quantum devices, with the condition of being improved upon.

XI. FUTURE WORK

A. Quantum Reduced Error Pruning

Reduced error pruning (REP) replaces each node in a decision tree with a leaf node in turn, where the leaf node classifies data as the most frequent classification of test data which passed through that node, before measuring the accuracy on the modified tree with test data and then reinserting the removed node. If none of the accuracies measured after removing a node are better than the baseline accuracy without any node removals, then no nodes are pruned, however, if this is not the case, then the node removal which caused the best accuracy improvement is permanently removed. This process is repeated until no more nodes can be removed. REP requires a tree traversal for each temporary node removal. Where n is the number of nodes in the original decision tree before any pruning, this means n traversals for the original tree, followed by $n - 1$ traversals after the first node removal, and then $n - 2$ traversals, and so on and so fourth until no more nodes can be removed. The summation term in Equation 18 can be used to compute the number of tree traversals k necessary to perform these temporary node removals, where P is the number of nodes which are pruned, and therefore the number of iterations in the REP process.

Additional tree traversals are required to get the most popular class for each temporary node removal. For this to

happen, each data point in the test data set must pass through the tree, a process which must repeat each time a node is removed because removing nodes affects classifications. The added term on the right in Equation 18 accounts for this, where l is the length of the test data set, which is just the number of data points it contains.

$$k = \sum_{i=0}^P (n - i) + Pl \quad (18)$$

A classical tree traversal takes n steps, whereas a QRW propagates through a decision tree much faster, requiring $\sqrt{n}$ steps. Replacing classical tree traversals with tree traversals by QRW has a theoretical speedup of $nk - \sqrt{nk}$. This can be done with the method proposed by Farhi et al. as discussed in Section IV [30]. There may be a trade off which needs to be investigated between the time saved by using a QRW and the time taken to map the decision tree into the required form as described in the paper.

REFERENCES

- [1] M. A. Nielsen and I. Chuang, "Quantum computation and quantum information," 2002.
- [2] B. Hssina, A. Merbouha, H. Ezzikouri, and M. Erritali, "A comparative study of decision tree id3 and c4. 5," *International Journal of Advanced Computer Science and Applications*, vol. 4, no. 2, pp. 13–19, 2014.
- [3] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Commun. ACM*, vol. 63, no. 1, p. 68–77, dec 2019. [Online]. Available: <https://doi.org/10.1145/3359786>
- [4] *Avoiding Overfitting of Decision Trees*. London: Springer London, 2007, pp. 119–134. [Online]. Available: https://doi.org/10.1007/978-1-84628-766-4_8
- [5] J. P. Buhler, H. W. Lenstra, and C. Pomerance, "Factoring integers with the number field sieve," in *The development of the number field sieve*, A. K. Lenstra and H. W. Lenstra, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1993, pp. 50–94.
- [6] A. Abbas, S. Andersson, A. Asfaw, A. Corcoles, L. Bello, Y. Ben-Haim, M. Bozzo-Rey, S. Bravyi, N. Bronn, L. Capelluto, A. C. Vazquez, J. Ceroni, R. Chen, A. Frisch, J. Gambetta, S. Garion, L. Gil, S. D. L. P. Gonzalez, F. Harkins, T. Imamichi, P. Jayasinha, H. Kang, A. h. Karamlou, R. Lored, D. McKay, A. Maldonado, A. Macaluso, A. Mezzacapo, Z. Minev, R. Movassagh, G. Nannicini, P. Nation, A. Phan, M. Pistoia, A. Rattew, J. Schaefer, J. Shabani, J. Smolin, J. Stenger, K. Temme, M. Tod, E. Wanzambi, S. Wood, and J. Wootton. (2020) Learn quantum computation using qiskit. [Online]. Available: <http://community.qiskit.org/textbook>
- [7] F. Ogban, I. Arikpo, and I. Eteng, "Von neumann architecture and modern computers," *Global Journal of Mathematical Sciences Vol. 6, No. 2, 2007, Page 97: ISSN 1596-6208 Indexed and abstracted on AJOL (UK):* <http://www.ajol.info>, vol. Vol. 6, No. 2, 2007, Page 97, 09 2007.
- [8] E. Mollick, "Establishing moore's law," *Annals of the History of Computing, IEEE*, vol. 28, pp. 62 – 75, 08 2006.
- [9] T. N. Theis and H.-S. P. Wong, "The end of moore's law: A new beginning for information technology," *Computing in Science Engineering*, vol. 19, no. 2, pp. 41–50, 2017.
- [10] G. Brassard, "A note on the complexity of cryptography (corresp.)," *IEEE Trans. Inf. Theor.*, vol. 25, no. 2, p. 232–233, sep 2006. [Online]. Available: <https://doi.org/10.1109/TIT.1979.1056010>
- [11] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters, "Teleporting an unknown quantum state via dual classical and einstein-podolsky-rosen channels," *Phys. Rev. Lett.*, vol. 70, pp. 1895–1899, Mar 1993. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevLett.70.1895>
- [12] C. H. Bennett, G. Brassard, S. Popescu, B. Schumacher, J. A. Smolin, and W. K. Wootters, "Purification of noisy entanglement and faithful teleportation via noisy channels," *Physical Review Letters*, vol. 76, no. 5, pp. 722–725, jan 1996. [Online]. Available: <https://doi.org/10.1103%2Fphysrevlett.76.722>
- [13] J. G. . D. M. Edwin Pednault and J. Gambetta, "On quantum supremacy," <https://www.ibm.com/blogs/research/2019/10/on-quantum-supremacy/>, 10 2019.
- [14] M. H. DeGroot, "Probability and statistics." *Addison-Wesley.*, pp. 12–16, 1975. [Online]. Available: <https://archive.org/details/probabilitystati0000degr/page/12/mode/2up>
- [15] L. Bienvenu, A. Romashchenko, A. Shen, A. Tavenaux, and S. Vermeeren, "The axiomatic power of kolmogorov complexity," 2013. [Online]. Available: <https://arxiv.org/abs/1301.3392>
- [16] F. T. Arecchi, E. Courtens, R. Gilmore, and H. Thomas, "Atomic coherent states in quantum optics," *Phys. Rev. A*, vol. 6, pp. 2211–2237, Dec 1972. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.6.2211>
- [17] L. M. Narducci, C. A. Coulter, and C. M. Bowden, "Exact diffusion equation for a model for superradiant emission," *Phys. Rev. A*, vol. 9, pp. 829–845, Feb 1974. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRevA.9.829>
- [18] J. N. C. Qiskit, "Defining quantum circuits," <https://github.com/qiskit-community/qiskit-textbook/blob/main/content/ch-algorithms/defining-quantum-circuits.ipynb>, 2020.
- [19] R. Penrose, "Applications of negative dimensional tensors." *Combinatorial Mathematics and Its Applications*, pp. 221–244, 1971.
- [20] W. Pauli, *Zeitschrift für Physik*, pp. 765–783, 1925. [Online]. Available: <https://doi.org/10.1007/BF02980631>
- [21] G. Light, "Pauli matrices immersion," *Materials Science and Engineering: B*, vol. 264, p. 114910, 11 2020.
- [22] A. Einstein, B. Podolsky, and N. Rosen, "Can quantum-mechanical description of physical reality be considered complete?" *Phys. Rev.*, vol. 47, pp. 777–780, May 1935. [Online]. Available: <https://link.aps.org/doi/10.1103/PhysRev.47.777>
- [24] E. Schrödinger, "Discussion of probability relations between separated systems," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 31, no. 4, p. 555–563, 1935.
- [25] —, "Probability relations between separated systems," *Mathematical Proceedings of the Cambridge Philosophical Society*, vol. 32, no. 3, p. 446–452, 1936.
- [26] IBM, "The qubit." [Online]. Available: <https://quantum-computing.ibm.com/composer/docs/ixq/guide/the-qubit>
- [27] F. Esposito, D. Malerba, G. Semeraro, and J. Kay, "A comparative analysis of methods for pruning decision trees," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 5, pp. 476–491, 1997.
- [28] S. Lu and S. Braunstein, "Quantum decision tree classifier," *Quantum Information Processing*, vol. 13, 03 2014.
- [29] E. Farhi and S. Gutmann, "Quantum computation and decision trees," *Physical Review A*, vol. 58, no. 2, pp. 915–928, aug 1998. [Online]. Available: <https://doi.org/10.1103%2Fphysreva.58.915>
- [30] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum algorithm for the hamiltonian nand tree," *Theory of Computing*, vol. 4, pp. 169–190, 01 2008.
- [31] D. Petz, V. Neumann, and S. In, "Entropy, von neumann and the von neumann entropy," 03 2001.
- [32] T. G. Wong, "Equivalence of szegedy's and coined quantum walks," *Quantum Information Processing*, vol. 16, no. 9, Jul. 2017. [Online]. Available: <https://doi.org/10.1007/s11128-017-1667-y>
- [33] M. Szegedy, "Quantum speed-up of markov chain based algorithms," in *45th Annual IEEE Symposium on Foundations of Computer Science*, 2004, pp. 32–41.
- [34] J. Kempe, "Quantum random walks: An introductory overview," *Contemporary Physics*, vol. 44, no. 4, pp. 307–327, jul 2003. [Online]. Available: