Implementation of Classical Decision Trees in a Quantum Computing paradigm*

M.P. Cuellar[1][0000−0002−9736−1608], L.G.B. Ruiz[2][0000−0001−6716−5115], and M.C. Pegalajar[1][0000−0002−2795−7502]

[1] University of Granada, Department of Computer Science and Artificial Intelligence, Granada, 18014, Spain {manupc,mcarmen}@decsai.ugr.es
http://decsai.ugr.es

[2] University of Granada, Department of Computer and Software Engineering, Granada, 18014, Spain
bacaruiz@ugr.es

Abstract. Decision trees are widely known models in Supervised Machine Learning with efficient inference mechanisms and outstanding interpretability. In this article, we design the implementation of classical Inductive Decision Trees under a quantum computing paradigm, and explore the advantages of Quantum Decision Trees designed in the presence of missing and uncertain data. Our findings extend to quantum ensembles analogous to Decision Forests as a Quantum Machine Learning method to improve the interpretability of a type of variational quantum circuits. Our approach provides an improvement in efficiency in the case of probabilistic inference with respect to the classical counterpart, and a general methodology is designed to address multiple classification tasks with Quantum Machine Learning tools, with a focus on the interpretability of quantum models. The theoretical results are supported by experimental simulations using different data sets and state-of-the-art examples.

Keywords: Quantum Decision Trees and Quantum Decision Forests and Quantum Machine Learning.

1 Introduction

Quantum Machine Learning [4] (QML) is a research area at the intersection of Artificial Intelligence (AI) and Quantum Computing (QC). Its goal is to either use classical AI to address QC challenges, or to incorporate AI into QC to solve classical or quantum problems. Different pure quantum and hybrid classical/quantum QML models have been proposed in the literature to solve supervised, unsupervised and reinforcement learning tasks [7][8], and they can be

* This article was funded by the project QUANERGY (Ref. TED2021-129360B-I00), Ecological and Digital Transition R&D projects call 2022 by MCIN/AEI/10.13039/501100011033 and European Union NextGeneration EU/PRTR.

classified into two categories: Some of them are inspired by well-known classical methods [8] such as Quantum Support Vector Machines (qSVMs), Quantum Neural Networks (qNNs), quantum Generative Adversarial Networks [3] (qGANs), or quantum Policy Iteration [2] to mention a few. Others are pure quantum algorithms such as the Quantum Approximate Optimization Algorithm (QAOA) or the Grover Adaptive Search(GAS) [3]. In any case, all proposals aim to use QC features such as superposition, entanglement or quantum parallelism to produce a significant speedup in terms of algorithmic complexity, or to take advantage of quantum measurement nonlinearity to improve state-of-the-art results by reducing model complexity. This work belongs to the first category. It combines a hybrid approach that explores how to implement classical Inductive Decision Trees (IDT) [5] in a quantum algorithm to solve classification problems, and the advantages that a quantum nature can provide to process imperfect and/or missing data. While the Quantum Decision Trees obtained regard to the quantum implementation of classical IDTs, the inference is performed in a pure quantum state evolution setting.

The proposals presented in this manuscript are based on the design of a procedure to perform a direct implementation of IDTs as a quantum circuit. We design this procedure taking into account three requirements: Simplicity, utilization of the minimum resources needed to represent the tree (measured as the minimum number of qubits), and maintaining the same interpretability as the original IDT. The result provided by the method is called Quantum Decision Tree (QDT), and offers a number of advantages: First, the interpretation of any quantum circuit with the same designed structure as a classical IDT, allowing the explanation and interpretability of a quantum program in a natural human way; Second, a base methodology for the design of quantum circuits that solve different tasks beyond classification, e.g. oracles [6]; and third, a natural way to perform probabilistic inference on the tree with a substantial improvement in algorithmic complexity. Having presented the method, we also explore how the designed QDT can be used to deal with imperfect knowledge. In our case, we take advantage of quantum superposition to represent multiple input feature values simultaneously, and quantum parallelism to evaluate all these values in superposition simultaneously. Our final contribution in this paper extends the approach to construct Quantum Decision Forests (QDFs) together with their aggregation procedure as QDTs superimposed on a quantum circuit, while maintaining the same expressivity as classical Decision Forests (DFs).

A background in Quantum Computing

In Quantum Computing [6], an arbitrary qubit $|\psi\rangle$ is modeled as an element of a vector subspace in the complex plane $\mathbb{C}^2$. A qubit is always in superposition (linear combination) of the elements of the computational basis vectors $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle ; \alpha_i \in \mathbb{C}$ with the constraint $\sum_i |\alpha_i|^2 = 1$. Here, $|0\rangle, |1\rangle$ uses Dirac notation to represent the column basis vectors $[1,0]^t, [0,1]^t \in \mathbb{C}^2$, respectively. The computational space of a system containing n qubits is a vector subspace in

$\mathbb{C}^{2^n}$ obtained as the tensor product ($\otimes$) of the basis vectors of each qubit. As an example, the computational basis of a system containing two qubits is $\{|0\rangle\otimes|0\rangle = |00\rangle = [1,0,0,0]^t, |0\rangle\otimes|1\rangle = |01\rangle = [0,1,0,0]^t, |1\rangle\otimes|0\rangle = |10\rangle = [0,0,1,0]^t, |1\rangle\otimes|1\rangle = |11\rangle = [0,0,0,1]^t\}$. A quantum state of n qubits is represented in its computational space as $|\psi\rangle_n = \sum_{i=0}^{2^n-1} \alpha_i |i\rangle, \alpha_i \in \mathbb{C}, \sum_{i=0}^{2^n-1} |\alpha_i|^2 = 1$, where $|i\rangle$ represents the i-th basis vector in binary representation.

Operations on quantum states with n qubits are modeled as complex unitary linear transformations in matrix form $U_{2^n,2^n}$, and their application on a set of qubits is written as $U|\psi\rangle_n$ and computed as a classical matrix multiplication. Each transformation must be reversible, except the *measurement* of the quantum state which makes the quantum state collapse to any of the basis vectors $|i\rangle$ of the computational space with probability $|\alpha_i|^2$.

In this work, we use single-qubit operations to create quantum states representing classical data, and to construct QDTs and QDFs. Of particular relevance is the X/NOT gate (Equation 1) similar to a classical NOT gate, and the rotation about the x-axis of the Bloch sphere $R_x(\theta)$ (Equation 2). In this manuscript, multi-qubit operations are limited to controlled gates and, in particular, controlled NOT gates and controlled $R_x(\theta)$ gates. A controlled operation implements a single-qubit gate U on a target qubit when other qubits (*controls*) are in a control state $|0\rangle$ or $|1\rangle$. The mathematical model for a controlled gate operation acting on n qubits, where the n-th qubit is the target and the remaining $\{1...n-1\}$ are controls with control states $|11...1\rangle_{n-1}$, is a block matrix constructed as in Equation 3 for an arbitrary single-qubit operation U, where $I_{2^{n-1},2^{n-1}}$ represents the identity matrix of size 2^{n-1} and $0_{2^{n-1},2}$ is a zero matrix of size 2^{n-1} rows and 2 columns. An example of a controlled operation is the CNOT gate (Equation 4). The CNOT operation maps a quantum state $|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle\ \alpha_3|11\rangle$ to $CNOT|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_3|10\rangle\ \alpha_2|11\rangle$, i.e., the value of the second qubit is negated iif the value of the first qubit is $|1\rangle$. Assuming an initial quantum state $|q_0q_1\rangle = |q_00\rangle$, the CNOT gate is a key operation in our proposed interpretation of quantum decision trees, since it can be read directly as *"If $|q_0\rangle = |1\rangle$, then $|q_1\rangle = |1\rangle$; otherwise $|q_1\rangle = |0\rangle$"*, in the non-probabilistic case. However, under uncertain data, the result of a CNOT can also be read in terms of the measured state amplitudes as $p(|q_1\rangle = |1\rangle\,|\,|q_0\rangle = |1\rangle) \cdot p(|q_0\rangle = |1\rangle)$, allowing probabilistic inference.

$$X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \tag{1}$$

$$R_x(\theta) = \begin{pmatrix} cos(\theta/2) & -isin(\theta/2) \\ -isin(\theta/2) & cos(\theta/2) \end{pmatrix} \tag{2}$$

$$CU = \begin{pmatrix} I_{2^{n-1},2^{n-1}} & 0_{2^{n-1},2} \\ 0_{2,2^{n-1}} & U \end{pmatrix} \tag{3}$$

$$CNOT = \begin{pmatrix} 1\,0\,0\,0 \\ 0\,1\,0\,0 \\ 0\,0\,0\,1 \\ 0\,0\,1\,0 \end{pmatrix} \tag{4}$$

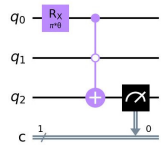


Fig. 1: Example of a quantum circuit with a $R_x(\theta\pi)$ gate acting on qubit q_0 followed by a Toffoli (CCNOT) over target q_2 controlled by $|q_0 q_1\rangle = |10\rangle$

A quantum program can be formulated as a sequence of quantum gates applied on qubits in parallel. A quantum circuit is a visual model that facilitates the evaluation of a quantum program. In it, each qubit is assigned a horizontal line, and the qubits are usually grouped into quantum registers. The sequence of operations is read from left to right, and marked with symbols over the qubits involved in the operation. As an example, Figure 1 shows a circuit with three qubits that implements the program $U = (I \otimes X \otimes I)CCNOT(R_x(\theta)\,|q_0\rangle \otimes X\,|q_1\rangle \otimes |q_2\rangle)$. If the initial quantum state is $|000\rangle$ then the resulting quantum state of this circuit is written as $U\,|000\rangle$ and shown in Equation 5. Measuring $|q_2\rangle$ in the classical bit c would provide bit 1 or 0 depending on the value of θ, and its expectation using the observable σ_z is represented as $\langle 000|\,U^\dagger(I^{\otimes 2} \otimes \sigma_z)U\,|000\rangle$, where $U^\dagger$ represents the complex conjugate transpose of U.

$$U\,|000\rangle = cos(\theta\pi/2)\,|000\rangle - isin(\theta\pi/2)\,|101\rangle \tag{5}$$

Another relevant feature of Quantum Computing in this manuscript is Quantum Parallelism. As we can see in Figure 1, the parallel structure of qubit lines allows multiple operations to be performed simultaneously on different sets of qubits. This structure has no added value over classical parallelization on multiple CPUs or GPUs; however, when combined with additional features such as superposition and entanglement, it could serve as a powerful tool to evaluate multiple input values simultaneously in a single circuit execution and, in some cases, obtain quantum supremacy. This is the case for well-known quantum algorithms such as Shor's algorithm for integer factorization, Simon's algorithm or Deutsch-Jotza's [6] algorithm. In our work, the advantages of quantum parallelism are combined with the superposition of input feature values as a way to deal with imperfect input classification patterns, and helped to achieve a significant speedup in QDT inference under partial observability with respect to classical IDTs. For example, setting $\theta = 0.5$ in the circuit in Figure 1 moves $|q_0\rangle$ from $|0\rangle$ to an equiprobable superposition $R_x(0.5\pi)\,|0\rangle = cos(\pi/4)\,|0\rangle - isin(\pi/4)\,|1\rangle$,

which means that both values $|q_0\rangle = |0\rangle$ and $|q_0\rangle = |1\rangle$ are processed simultaneously by the circuit. As a result, measuring $|q_2\rangle$ with the observable σ_z provides an expectation value equal to 0, since the probability of $|q_2\rangle = |1\rangle$ results in 0.5.

The final tool from Quantum Computing that we have used in this manuscript is entanglement [6], as a possible avenue for developing quantum-inspired missing value imputation methods. Two qubits are said to be entangled if their values are highly correlated, as for example in the resulting unitary state in Equation 5. In the possible results after measurement of all qubits $|q_0 q_1 q_2\rangle \in \{|000\rangle, |101\rangle\}$, we can verify a high correlation between qubits $|q_0\rangle, |q_2\rangle$, since once we know the value of one qubit, then we already know the value of the other without measurement. In our work, the entanglement is performed using $R_x(\theta\pi)$ gates along with $NOT/R_x(\theta\pi)$ multiple control gates. In fact, the construction of a QDT in our approach aims at finding entanglements between class label qubits (target) and input feature values (controls) that serve to predict correct classifications. Thus, we argue that a QDT model could also be used in some situations to perform missing value imputation thanks to the entanglement between input features and the target class.

2 From classical to quantum decision trees

Our method for encoding a classical IDT as a QDT consists of two steps: 1) a state preparation procedure to map the input feature data to a quantum state; and 2) a quantum circuit containing the IDT logic (antsatz). If there is no uncertainty in the input data, a basis encoding mechanism [8] can be used for state preparation. Otherwise, the uncertainty is encoded in the quantum state amplitudes as the probability that the input feature contains one value or another. Depending on the uncertainty, this can be achieved using angle encoding [8] or more sophisticated methods such as amplitude embedding [1].

Given a dataset containing n input features $\{F_1, F_2, ..., F_n\}$ and a target class C, a QDT must implement a unitary transformation U such that, for a given set of feature values $\{V_1, V_2, ..., V_n\}$ mapped to quantum states $\{|Q_1\rangle, |Q_2\rangle, ..., |Q_n\rangle\}$ and a class label c_l assigned to $|c_l\rangle$, the circuit computes the target quantum state $|Q_1\rangle |Q_2\rangle ... |Q_n\rangle |c_l\rangle = U |Q_1\rangle |Q_2\rangle ... |Q_n\rangle |0\rangle$. Our approach constructs this unitary operation as a sequence of multiple controlled NOT gates (MCX), where each gate corresponds to a path from the root of the classical IDT to a leaf node. Thus, each MCX operation implements a rule for class value prediction.

The complete procedure for creating a quantum circuit that implements QDT logic from a classical IDT is shown in Algorithm 1. The classical IDT T is an input as well as the interpretation I where $I_{F,v}$ represents the encoding of the binary representation basis for the v value of the F feature. In lines 1-4, the quantum registers for all input and output features are created. Next, line 6 finds the *paths* of T. This operation can be easily performed with a preorder traversal of the tree, and we assume that a path contains a set of (F_i, v_i) pairs along with the target class label. Next, each *path* is implemented as a MCX operation in lines 8-17, and added to the output QDT. The variables *controls*

and *states* store the quantum states involved in the *MCX* and its control state, respectively.

The first thing to remark here is that reaching a particular leaf node in classical IDT is performed in $log(n)$ node evaluations, assuming a balanced tree with n leaf nodes (the worst case for QDT). However, reaching the same leaf node in QDT is realized in $\mathcal{O}(1)$, since it requires a constant number of operations depending on the instruction set of a quantum computer to implement MCX. The main limitation of QDT is that it requires the evaluation of all MCX operations in the circuit to return a result, which is n. However, if we consider a scenario where the inference is probabilistic, all nodes of the classical IDT could be evaluated, while the number of QDT operations remains n. This is a significant efficiency improvement over classical IDT inference.

Algorithm 1: Algorithm to encode a classical Inductive Decision Tree into a Quantum Decision Tree

T: Classical Decision Tree
I : Interpretation/mapping from classical feature values to quantum register basis states

foreach *non-terminal feature* $F_i \in T$ **do**
  Create quantum register $|Q_{F_i}\rangle$ for F_i
end
Create quantum register $|Q_{label}\rangle$ for target class
QDT = $\emptyset$
paths= set of paths from root to leaf nodes in T
foreach *path* $\in$ *paths* **do**
  states = {}
  controls = {}
  c_l = get class label value from path
  q_c = qubit of $|Q_{label}\rangle$ for which I_{C,c_l} is 1
  if q_c *is not None* **then**
    foreach *non-terminal pair* F_i, v_i *in path* **do**
      append $|Q_{F_i}\rangle$ to controls
      append I_{F_i,v_i} to states
    end
    rule = MCX(target=q_c, controlled_by=controls, controlled_states=states)
    QDT = QDT $\bigcup$ {rule}
  end
end
return QDT

The second aspect we want to highlight is that the interpretability of the QDT remains unchanged with respect to the original IDT, since each *path* can be easily located in the QDT circuit as a *MCX* operation. Moreover, we can observe that all QDTs obtained with our procedure contain the same structure:

a set of input quantum registers, an output register, and a sequence of *MCX* operations. Thus, our approach allows the interpretability of any quantum circuit with the same structure as a classical IDT. We believe that this is a relevant contribution to the interpretability of quantum circuits in the field of Quantum Machine Learning.

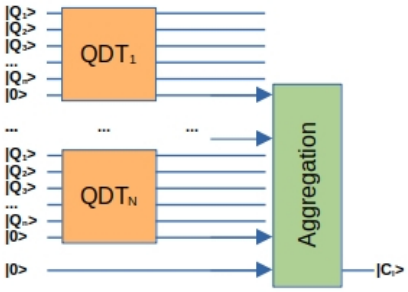
Finally, the inference in a QDT is straightforward and equivalent to DT inference: First, the feature values must be encoded by the state preparation procedure. After that, a single *MCX* will move the corresponding output qubit of the quantum register $|Q_{label}\rangle$ from $|0\rangle$ to $|1\rangle$, and the output register is measured.

Beyond its high interpretability, the main practical advantage of a QDT concerns the natural processing of unknown or imperfect input data. We take advantage of quantum superposition and parallelism to encode different values of an unknown input feature and evaluate all these values simultaneously in the QDT. The main idea is to encode the probability of each feature value in the amplitudes of a quantum register. The output register will then provide the probability of the target class labels considering all established superpositions. As it was mentioned above, our supporting examples use $R_x(\theta\pi)$ rotation gates in the input state preparation step with angle encoding due to its simplicity, although amplitude embedding could also be used. A rotation $R_x(\theta\pi)$ applied as state preparation with $\theta = 0$ leaves a qubit value unchanged in $|0\rangle$ state, while a $\theta = 1$ value negates the qubit as using a NOT gate to $|1\rangle$ state. Any value of $\theta \in (0, 1)$ alters the probability that the qubit is in a nontrivial superposition of $|0\rangle$ and $|1\rangle$. Since a QDT is a direct implementation of a classical DT, its prediction accuracy is also not altered, although it retains the same training limitations in terms of overtraining risk.

2.1 Quantum Decision Forests

We devise two strategies to create a Quantum Decision Forest from a set of existing QDTs in Figure 2. In either of them, each IDT is implemented in its corresponding Quantum Decision Tree module from QDT_1 to QDT_N using Algorithm 1. In addition, a new module *Aggregation* must be created to return the output register $|C_l\rangle$ from the partial outputs of each tree. In the model in Figure 2a, all trees are evaluated in parallel, while the model in Figure 2b executes the QDTs sequentially. The advantage of the parallelized strategy in Figure 2a is a substantial speedup, since the only bottleneck concerns the evaluation of the QDT with the largest circuit depth; however, its main limitation is that many input feature registers must be replicated to feed each QDT with its inputs, so the number of qubits required from this strategy is substantially larger and therefore not suitable for simulation on contemporary classical computers. On the other hand, the sequential strategy in Figure 2b requires fewer resources in space, although the time complexity depends on the number of QDTs in the forest.

We have implemented the sequential strategy of Figure 2b in our example, since it has been tested under simulation on a classical computer. In addition, we have implemented the criterion *Weighted Majority Voting* as an aggregation



(a) Parallel Quantum Decision Forest.

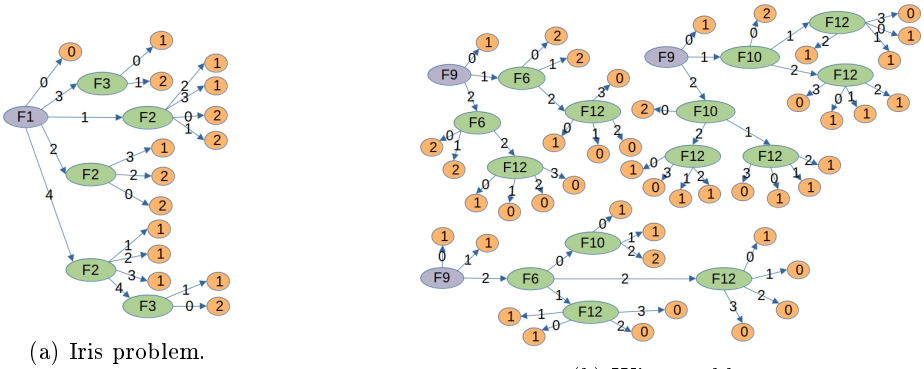
(b) Sequential Quantum Decision Forest.

Fig. 2: Strategies to create Quantum Decision Forests.

module for illustrative purposes due to its simplicity. This aggregation procedure returns the most frequent result among the weighted partial results of QDTs, and can be easily implemented in a QDF by replacing the MCX lines with Controlled Multiple X-Rotations ($MCR_x(\theta)$) parameterized by θ. Here, θ controls the contribution of the result of each QDT to the final output register $|C_l\rangle$, so that no additional registers are required to store the partial output of each QDT.

3 Experimental results

We design quantum decision trees using a quantum gating framework under simulation to solve multiclass classification problems. IBM's Python Qiskit library is used to demonstrate our examples and inference tests. All experiments were performed on an Intel(R) Core(TM) i5-9600K CPU desktop computer at 3.70GHz with 32GB RAM.



(a) Iris problem.

(b) Wine problem.

Fig. 3: Decision Tree and Decision Forest representation of the datasets used.

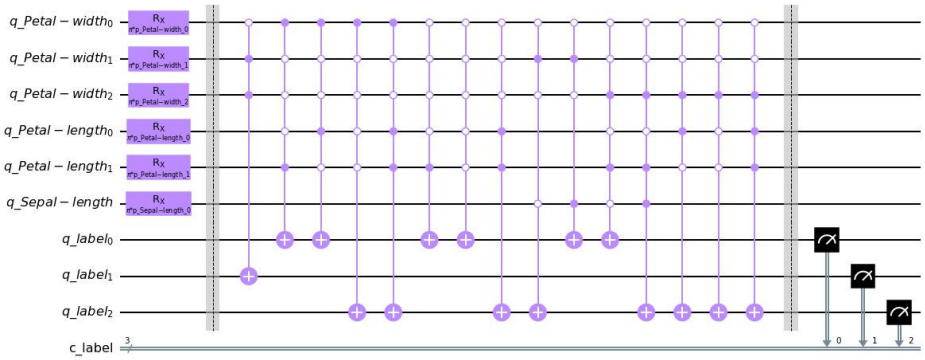


Fig. 4: Quantum Decision Tree for the Iris problem

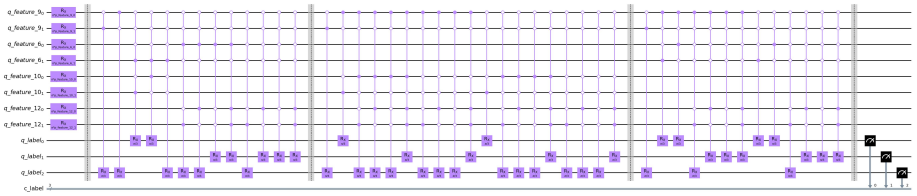


Fig. 5: Quantum Decision Forest for the Wine problem overlapped sequentially

We used classical reference data sets to experimentally support our findings: The *Iris* classification problem with four features (sepal length, sepal width, petal length, petal width) of three Iris flower types (target class: Setosa, Versicolor, Virginica), and the *Wine* classification dataset containing 13 features and 3 target class labels. The *Iris* problem is used as an example for multiclass QDT coding, with the classical tree depicted in Figure 3a. On the other hand, *Wine* is used to show Quantum Decision Forest coding, for the classical decision forest containing three trees in Figure 3b. The tree was designed to provide 100% accuracy on the *Iris* data set. However, the decision trees in the forest in Figure 3b provide between 95% and 98% accuracy separately for the *Wine* dataset, and 100% accuracy with the majority voting output aggregation. Our experiments are limited to features with discrete values, so that preprocessing should be applied to discretize features with continuous data, such as the *Iris* and *Wine* datasets. In addition, a mapping from classical data to quantum states is required to achieve full interpretability of the generated QDTs and to devise a mechanism for quantum preparation of the input state. Tables 1 and 2 describe the discretization limits along with an example interpretation that maps each feature value to a quantum ground state for the trees in Figure 3. The number of quantum registers used to translate an IDT into a QDT is equal to the num-

Feature	Q. Register	Range	Discretization	Basis Encoding
F_1: Petal width	Q_{pw}	$[0, 0.8]$	0	$\|011\rangle$
		$(0.8, 1.55]$	1	$\|100\rangle$
		$(1.55, 1.65]$	2	$\|000\rangle$
		$1.65, 1.75]$	3	$\|010\rangle$
		$(1.75, 2.5]$	4	$\|001\rangle$
F_2: Petal length	Q_{pl}	$[0, 4.85]$	0	$\|01\rangle$
		$(4.85, 4.95]$	1	$\|10\rangle$
		$(4.95, 5.45]$	2	$\|00\rangle$
		$(5.45, 6.9]$	3	$\|11\rangle$
F_3: Sepal length	Q_{sl}	$[0, 5.95]$	0	$\|0\rangle$
		$(5.95, 7.90]$	1	$\|1\rangle$
Target: Label	Q_{label}	Setosa	0	$\|010\rangle$
		Virginica	1	$\|001\rangle$
		Versicolor	2	$\|100\rangle$

Table 1: Discretization bounds and quantum encoding for the *Iris* dataset.

Feature	Q. Register	Range	Discretization	Basis Encoding
F_6: Total phenols	Q_{f6}	$[0, 1.58]$	0	$\|01\rangle$
		$(1.58, 1.71]$	1	$\|10\rangle$
		$(1, 71, \infty)$	2	$\|00\rangle$
F_9: Proanthocyanins	Q_{f9}	$[0, 3.49]$	0	$\|01\rangle$
		$(3.49, 3.92]$	1	$\|10\rangle$
		$(3.92, \infty]$	2	$\|00\rangle$
F_{10}: Color intensity	Q_{f10}	$[0, 0.83]$	0	$\|01\rangle$
		$(0.83, 0.97]$	1	$\|10\rangle$
		$(0.97, 1]$	2	$\|00\rangle$
F_{12}:diluted wines	Q_{f12}	$[0, 679]$	0	$\|01\rangle$
		$(679, 706.5]$	1	$\|10\rangle$
		$(706.5, 724.5]$	2	$\|00\rangle$
		$(724.5, \infty)$	3	$\|11\rangle$
Target: Label	Q_{label}	Cultivar 1	0	$\|010\rangle$
		Cultivar 2	1	$\|001\rangle$
		Cultivar 3	2	$\|100\rangle$

Table 2: Discretization bounds and quantum encoding for the *Wine* dataset.

ber of features in a tree plus one output register. If a feature F_i containing n_i values is paired with quantum register Q_i, the number of qubits in the register is $\lceil log_2 n_i \rceil$. The resulting implementation of DT and DF are shown in Figures 4 and 5, respectively. Here, we can distinguish each path from the root of each tree to their respective leaf nodes implemented as MCX. Experimental simulations concluded that the accuracy of the quantum circuits on the datasets equals the accuracy of the initial classical DTs and reaches an accuracy of 100% in both cases.

The main practical advantage of a QDT concerns the natural processing of unknown or imperfect input data. We take advantage of quantum superposition and parallelism to encode different values of an unknown input feature and evaluate all these values simultaneously in the QDT. The main idea is to encode the probability of each feature value in the amplitudes of a quantum register. The output register will then provide the probability of the target class labels considering all established superpositions. As mentioned above, our supporting examples use $R_x(\theta\pi)$ rotation gates in the input state preparation step with angle encoding due to their simplicity, although amplitude embedding could also be used. A rotation $R_x(\theta\pi)$ applied as state preparation with $\theta = 0$ leaves a qubit value unchanged in the $|0\rangle$ state, while a value $\theta = 1$ negates the qubit as using a NOT gate and moves it to the $|1\rangle$ state. Any value of $\theta \in (0, 1)$ alters the probability that the qubit is in a nontrivial superposition of $|0\rangle$ and $|1\rangle$. As an example, consider the pattern *(p(Sepal-length=0/1)=0.5, p(Petal-length=0)=1, p(Petal-width=4)=1)* for the data set *Iris*. This pattern corresponds to $|Q_{pw}\rangle |Q_{pl}\rangle |Q_{sl}\rangle = |001\rangle_{pw} |01\rangle_{pl} R_x(0.5\pi) |0\rangle_{sl}$. The rules that are triggered in this case correspond to the 11th and 12th *MCXs* in figure 4, which compute $p(label = Versicolor|Petal - width = 4, Petal - length = 0, Sepal - length = 0) \cdot p(Sepal - length = 0)$ and $p(label = Virginica|Petal - width = 4, Petal - length = 0, Sepal - length = 1) \cdot p(Sepal - length = 1)$. The result is thus $p(label = Versicolor) \approx 0.5, p(label = Virginica) \approx 0.5$. Since a QDT is a direct implementation of a classical DT, its prediction accuracy is also not altered, although it retains the same training limitations regarding possible overtraining.

Another example in the *Iris* data set concerns possible applications for missing value imputation. Consider the pattern *(p(Petal width=2)=1)* to predict *label=Versicolor*, while the other features are supposed to be unknown. Let us assume that the goal is to perform missing value imputation of the feature *Petal length*. An input state could be conceived as $|Q_{pw}\rangle |Q_{pl}\rangle |Q_{sl}\rangle = |000\rangle_{pw} (R_x(0.5\pi) \bigotimes R_x(0.5\pi)) |00\rangle_{pl_1} R_x(0.5\pi) |0\rangle_{sl}$. The possible rules that can be triggered are the 6th, 7th, 8th *MCXs* in Figure 4. The measurement on the $|Q_{pl}\rangle$ and $|Q_{label}\rangle$ registers is performed on the output quantum state described as $|Q_{pl}\rangle |Q_{label}\rangle = \alpha_0 |00\rangle_{pl} |100\rangle_{label} + \alpha_1 |01\rangle_{pl} |100\rangle_{label} + \alpha_2 |11\rangle_{pl} |001\rangle_{label}$, which means that the value $|Q_{pl}\rangle = |11\rangle$ would lead to a classification of *label=Virginica* and the remaining values to *label=Versicolor* with the same probability. Since the pattern contains *label=Versicolor*, we could replace the missing value with $|Q_{pl}\rangle = |00\rangle$ or $|Q_{pl}\rangle = |01\rangle$.

4 Conclusions

This work addresses the representation and learning of Decision Trees and their extension to Decision Forests under a Quantum Computing paradigm. In the results, we have shown how well-known and widely used quantum gates can be used to translate a classical DT into a quantum circuit in a simple way, without loss of interpretability. In our opinion, this is an important step forward for the analysis and explanation of the behaviour of certain types of quantum programs. Moreover, the use of quantum mechanisms such as superposition and quantum parallelism provide relevant advantages to deal with imperfect knowledge regarding the output inference process for a given input pattern. We have found that an important speed-up could be achieved under a massively probabilistic inference mechanism in a QDT with respect to the classical counterpart, and also that possible applications to develop new techniques for missing value imputation could be a focus of future research using quantum entanglement. Besides, extending QDTs to Quantum Decision Forests was also found an easy task that can be performed with minimum modifications to each QDT in the forest and the implementation of the output aggregation procedure.

Acknowledgements This work was performed in cooperation with the IEEE Computer Society Task Force in Quantum Computational Intelligence (URL at https://cmte.ieee.org/quantum-computational-intelligence/).

References

1. Araujo, I.F., Park, D.K., Petruccione, F., da Silva, A.J.: A divide-and-conquer algorithm for quantum state preparation. Scientific Reports **11**, 6329–6341 (2021). https://doi.org/10.1038/s41598-021-85474-1
2. Cherrat, E.A., Kerenidis, I., Prakash, A.: Quantum reinforcement learning via policy iteration. Quantum Machine Intelligence **5**, 1–18 (2023). https://doi.org/10.1007/s42484-023-00116-1
3. Combarro, E.F., Gonzalez-Castillo, S.: A practical guide to Quantum Machine Learning and Quantum Optimization. Packt, Birmingham, United Kingdom (2023)
4. Ganguly, S.: Quantum Machine Learning: An Applied Approach. Apress, New York (2021)
5. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (mar 1986). https://doi.org/10.1023/A:1022643204877, https://doi.org/10.1023/A:1022643204877
6. Sutor, R.: Dancing with Qubits. Packt, Birmingham, UK (11 2019)
7. Umer, M.J., Sharif, M.I.: A comprehensive survey on quantum machine learning and possible applications. International Journal of E-Health and Medical Communications **13**(5), 1–17 (dec 2022). https://doi.org/10.4018/IJEHMC.315730, https://doi.org/10.4018/IJEHMC.315730
8. Wittek, P.: Quantum Machine Learning: What Quantum Computing means to data mining. Elsevier, Amsterdam, The Netherlands (2014)