




QLDT: A Decision Tree Based on Quantum Logic

Ingo Schmitt^(✉) 

Brandenburgische Technische Universität Cottbus-Senftenberg, Cottbus, Germany
schmitt@b-tu.de
<https://www.b-tu.de/fg-dbis>

Abstract. Besides a good prediction a classifier is to give an explanation how input data is related to the classification result. Decision trees are very popular classifiers and provide a good trade-off between accuracy and explainability for many scenarios. Its split decisions correspond to Boolean conditions on single attributes. In cases when for a class decision several attribute values interact gradually with each other, Boolean-logic-based decision trees are not appropriate. For such cases we propose a quantum-logic inspired decision tree (QLDT) which is based on sums and products on normalized attribute values. In contrast to decision trees based on fuzzy logic a QLDT obeys the rules of the Boolean algebra.

Keywords: Quantum logic · Decision tree · Interpretable AI

1 Introduction and Related Work

Data mining is the task of finding patterns in a large data collection. Methods of supervised learning find a mapping, called a model, between input objects and a target property. For the classification task the target property is categorical. Without loss of generality, we discuss classification tasks where the target property distinguishes between two classes denoted by the values 0 and 1, respectively. Let D be a set of objects (tuples of reals). Every object is characterized by its values for n given attributes: $o = (o_1, \dots, o_n)$. Furthermore, we assume the existence of a hidden mapping m from D to the classes, that is, $m : D \rightarrow \{0, 1\}$. We explicitly know the mapping only for a subset $O \subset D$. That is, we hold $M = \{(o, m(o)) | o \in O\}$. Let $TR \subset M$ be the set of training data and $TE = M \setminus TR$ be test data.

Solving a classification problem means to construct a mapping function $cl : D \rightarrow \{0, 1\}$, called a *classifier*, from given training data TR . The classifier should approximate m and should provide a prediction on TE with high accuracy. The *accuracy* of a classifier is quantified as the fraction of correctly classified objects of all test objects: $accuracy = |\{(o, m(o)) \in TE | m(o) = cl(o)\}| / |TE|$. In addition to a good accuracy a classifier should explain to users the connection between object attributes and the corresponding class, see [2]. A very popular

classification method is the decision tree (DT), see [1]. The DT is based on rules of Boolean logic and can be seen as a good trade-off between accuracy and power in order to explain the classifier [3]. That is, in contrast to works like [7, 15] we use logic as a means for explanation.

For finding the class of an object using a DT we navigate from the root to a leaf. Following such a path means to check conjunctively combined conditions on object attribute values. If we regard objects as points in $[0, 1]^n$ then every tree node split on an attribute corresponds to one or more hyperplanes being parallel to $n - 1$ axes.



Fig. 1. (left) Class decision lines for $(x > 0.5) \wedge (y > 0.5)$ (dashed) and $x * y > 0.5$ (solid) and (right) space decomposition by axis-parallel decisions for $x * y > 0.5$

See Fig. 1 (left) for a two-dimensional case where the dashed line refers to the class separation for $(x > 0.5) \wedge (y > 0.5)$. For that class decision the attribute values interact conjunctively on the level of Boolean truth values. But what about scenarios where the interaction takes place on object values directly? See for example the solid class separation line in Fig. 1 (left) for $x * y > 0.5$. Let, for example, $x \in [0, 1]$ encodes age and $y \in [0, 1]$ encodes continuously the BMI of a person. Furthermore, the risk of severe health damage from COVID may increase gradually in the shape of a product of age and BMI. In that and similar cases, decision trees based on axis parallel decisions can only roughly approximate non-parallel decision lines and deteriorate, see Fig. 1 (right). A tighter approximation would lead to even more deterioration.

In contrast to the traditional decision tree based on Boolean decisions we develop a quantum-logic inspired decision tree (QLDT). Instead of combining Boolean values we regard attribute values from $[0, 1]$ as results from quantum measurements and combine them directly by using negation, conjunction and disjunction following the concepts of quantum logic [9, 12]. Different from fuzzy logic, quantum logic based on mutually commuting conditions obeys the rules of a Boolean algebra. Therefore, in contrast to decision trees based on fuzzy logic [5, 6, 10], every logical formula can be represented as a set of disjunctively combined minterms which are themselves conjunctions of positive or negated

conditions (disjunctive normal form) [4]. After deriving a logic expression e in disjunctive normal form we generate a QLDT ($qldt(e)$) from it.

Referring to the solid decision line (left) in Fig. 1 we obtain the quantum logical expression $x \wedge y$ with $x, y \in [0, 1]$.

The evaluation of a traditional decision tree against an input object differs from the evaluation [$qldt(e)$] of a QLDT. Starting from the QLDT root we navigate in a parallel manner to all leaves where for each leaf we obtain a leaf-specific evaluation value from $[0, 1]$. All evaluation values of class-1-leaves are summed up to a class value from $[0, 1]$. A final threshold τ is applied to the class value for a discrete class decision. The class decision can be written as [$qldt(e)$] $>$ τ (in the example we yield $x * y > 0.5$).

For our quantum logic decision tree approach we identify the following advantages:

1. Quantum logic deals directly with continuous truth values;
2. In contrast to fuzzy logic our quantum-logic inspired approach obeys the rules of the Boolean algebra [8], for example [$e \wedge \neg e$] = 0 and [$e \wedge e$] = [e];
3. Class separation lines are not restricted to be axis-parallel.¹

In following sections we will develop the QLDT. It is based on the concepts of CQQL. The quantum-logic inspired language CQQL (commuting quantum query language) was introduced in [12, 14].

2 Commuting Quantum Query Language (CQQL)

Syntactically, a CQQL expression is an expression of propositional logic based on conjunction, disjunction, and negation. We assume n atomic, unary conditions on the n values of an object o . Such a condition expresses *gradually* whether an input value is a high value, e.g. a high BMI value. Each of the conditions returns a value from $[0, 1]$. The upper bound 1 is interpreted as *true* and the lower bound 0 as *false*. In [11] we prove that a quantum logic expression based on atomic conditions on different attributes form a Boolean (orthomodular, distributive) lattice.

A CQQL expression e in a specific syntactical normal form (CQQL normal form, see [11, 12, 14]) can be evaluated arithmetically. Each CQQL expression can be transformed into that normal form, see [12, 14].

Let the function $atoms(e)$ return the set of atomic conditions involved by a possibly nested condition e . The CQQL normal form requires that for each conjunction $e_1 \wedge e_2$ and for each disjunction $e_1 \vee e_2$ (but not for the special case of an exclusive disjunction) the atom sets are disjoint: $atoms(e_1) \cap atoms(e_2) = \emptyset$. If for $e_1 \vee e_2$ the conjunction $e_1 \wedge e_2$ is unsatisfiable in propositional logic then the disjunction is exclusive. We mark each exclusive disjunction by $\dot{\vee}$.

The evaluation of a CQQL expression e in the required normal form against an object o is written as $[\cdot]^o$. For brevity, we drop the object o and write just $[\cdot]$. In the following recursive definition of $[e]$, we distinguish five cases:

¹ Of course, there exist classification problems for which the decision tree based on Boolean logic fits perfectly.

1. Atomic condition: If e is an atomic condition then $[e] \in [0, 1]$ returns the result from applying the corresponding condition on o .
2. Negation: $[\neg e] = 1 - [e]$;
3. Conjunction: $[e_1 \wedge e_2] = [e_1] * [e_2]$;
4. Non-exclusive disjunction: $[e_1 \vee e_2] = [e_1] + [e_2] - [e_1] * [e_2]$; and
5. Exclusive disjunction: $[e_1 \dot{\vee} e_2] = [e_1] + [e_2]$.

We now extend the expressive power of a CQQL condition by introducing *weighted conjunction* ($e_1 \wedge_{\theta_1, \theta_2} e_2$) and *weighted disjunction* ($e_1 \vee_{\theta_1, \theta_2} e_2$). The work [13] develops the concept of weights in CQQL from quantum mechanics and quantum logic. Weight variables θ_1, θ_2 stand for values out of $[0, 1]$. A weight $[\theta_i] = 0$ means that the corresponding argument has no impact and a weight $[\theta_i] = 1$ equals the unweighted case (full impact). We regard every weight variable θ_i as a 0-ary atomic condition. Before we evaluate a condition with weights we map every weighted conjunction and weighted disjunction in e to an unweighted condition:

$$\begin{aligned} (e_1 \wedge_{\theta_1, \theta_2} e_2) &\rightarrow ((e_1 \vee \neg\theta_1) \wedge (e_2 \vee \neg\theta_2)) \\ (e_1 \vee_{\theta_1, \theta_2} e_2) &\rightarrow ((e_1 \wedge \theta_1) \vee (e_2 \wedge \theta_2)). \end{aligned}$$

For a certain classification problem we want to find a matching CQQL expression e together with a well-chosen output threshold value τ for cl : $cl_e^\tau(o) = th_\tau([e]^o)$

with $th_\tau(x) = \begin{cases} 1 & \text{if } x > \tau \\ 0 & \text{otherwise.} \end{cases}$

From the laws of the Boolean algebra we know that every expression e can be expressed in the complete disjunctive normal form, that is, every expression is equivalent to a subset of 2^n minterms. We implicitly assume for each of the n object attributes exactly one atomic condition c_j for $j = 1, \dots, n$ and for an object $o = (o_1, \dots, o_n)$ the equivalence $o_j = [c_j]^o \in [0, 1]$. The minterm subset relation for any logic expression can be expressed by use of minterm weights $\theta_i \in \{0, 1\}$:

$$e = \bigvee_{i=1}^{2^n} minterm_{i, \theta_i} \text{ mapped to } \bigvee_{i=1}^{2^n} minterm_i \wedge \theta_i = e \tag{1}$$

and $minterm_i = \bigwedge_{j=1}^n c_{ij}$ with $c_{ij} = \begin{cases} c_j & \text{if } (i-1) \& 2^{j-1} > 0 \\ \neg c_j & \text{otherwise} \end{cases}$.

That is, the value $i - 1$ is considered as a bitcode and identifies a minterm uniquely and $j - 1$ stands for a bit position. The symbol ‘&’ stands for the bitwise **and**.

Please note that the disjunction of any two different complete minterms is always exclusive. Thus, e is in CQQL normal form and its evaluation against object o yields

$$[e]^o = \sum_{i=1}^{2^n} \theta_i \prod_{j=1}^n [c_{ij}]^o. \tag{2}$$

3 Extraction of Minterms and Finding the Output Threshold

Next, we will extract a CQQL expression e in complete disjunctive normal form from training data. We have to find the weight θ_i for every minterm i . The starting point is the training set $TR = \{(x, y)\} = \{(o, m(o))\}$ with the input tuples (objects) $x = o \in [0, 1]^n$ and $y = m(o) \in \{0, 1\}$.

One important requirement for a classifier is high accuracy. Therefore, we maximize the accuracy of expression (1) depending on the minterm weights θ_i based on $TR = \{(x, y)\}$.

Accuracy acc for a continuous evaluation can be measured as sum over the two correct cases $(y = 1) \wedge [e]^x$ and $(y = 0) \wedge [-e]^x$ over all pairs $(x, y) \in TR$:

$$\begin{aligned} acc &= \sum_{(x,y) \in TR} (y * [e]^x + (1 - y)(1 - [e]^x)) \\ &= \sum_{i=1}^{2^n} \theta_i \sum_{(x,y) \in TR} \left((2y - 1) \cdot \prod_{j=1}^n [c_{ij}]^x \right) + \sum_{(x,y) \in TR} (1 - y). \end{aligned}$$

We see after applying Eq. (2) and some reformulations, that accuracy shows a linear dependence on the minterm weight θ_i for fixed TR -pairs. The first derivative yields a constant gradient on θ_i :

$$\frac{\partial acc}{\partial \theta_i} = \sum_{(x,y) \in TR} (2y - 1) \cdot \prod_{j=1}^n [c_{ij}]^x = \sum_{(x,1) \in TR} \prod_{j=1}^n [c_{ij}]^x - \sum_{(x,0) \in TR} \prod_{j=1}^n [c_{ij}]^x.$$

For maximizing accuracy a minterm weight θ_i should have the value 1 if $\frac{\partial acc}{\partial \theta_i} > 0$ and 0 otherwise. In other words, for the decision whether a minterm should be active or not it is sufficient to compare the impact of positive training data E_i against the impact of the negative training data N_i with

$$E_i = \sum_{(x,1) \in TR} \prod_{j=1}^n [c_{ij}]^x \text{ and } N_i = \sum_{(x,0) \in TR} \prod_{j=1}^n [c_{ij}]^x.$$

Please note that the decision depends on the relative number of positive training objects in TR . Therefore, let $\gamma_1 = |\{(x, 1) \in TR\}|$ and $\gamma_0 = |\{(x, 0) \in TR\}|$ be the number of positive and negative training objects, respectively. The fraction of negative objects is then given by $\gamma = \frac{\gamma_0}{\gamma_1 + \gamma_0}$. In the unbalanced case ($\gamma \neq 1/2$) we compensate the effect on the minterm weight decision by:

$$[\theta_i] = \begin{cases} 1 & \text{if } \gamma \cdot E_i > (1 - \gamma) \cdot N_i \\ 0 & \text{otherwise} \end{cases}. \tag{3}$$

Following that minterm decision rule we can decide for every minterm whether it is active or inactive. In case of $\gamma \cdot E_i \approx (1 - \gamma) \cdot N_i$ the decision is not clear. We

call such kind of minterms *unstable* because adding a single new training object may change the decision. Instable minterms are less expressive than stable ones and have a low impact on accuracy. We are interested in stable minterms. For measuring stability we compute the ratio ρ_i of $\gamma \cdot E_i$ to the sum of $\gamma \cdot E_1$ and $(1 - \gamma) \cdot N_i$ of a minterm i . A value for ρ_i close to 1/2 indicates an unstable minterm i , a value near 1 means a stable active minterm i , and a value near 0 means a stable inactive minterm i .

The question arises: should instable minterms be active or inactive? We propose to sort all minterms by their values ρ_i and choose a ρ -threshold θ_ρ from them that provides a good trade-off between accuracy and compactness of expression e . The modified minterm decision rule is:

$$[\theta_i] = \begin{cases} 1 & \text{if } \frac{\gamma E_i}{\gamma E_i + (1-\gamma)N_i} > \theta_\rho \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

After applying our minterm decision rule (4) we obtain the expression:

$$e = \bigvee_{i: [\theta_i]=1} \bigwedge_{j=1}^n c_{ij} \text{ with } [e]^x = \sum_{i: [\theta_i]=1} \prod_{j=1}^n [c_{ij}]^x.$$

Next we have to find the output threshold value τ for $cl_e^\tau(x) = th_\tau([e]^x)$. Let $\min_1 = \min_{(x,1) \in TR} [e]^x$ be the smallest evaluation result of the positive training objects and $\max_0 = \max_{(x,0) \in TR} [e]^x$ be the highest result of the negative training objects. In case of $\max_0 < \min_1$, positive objects and negative objects are well separated and we set τ to $(\max_0 + \min_1)/2$.

Otherwise, we have to choose a value τ from the interval $[\min_1, \max_0]$. In order to find a threshold which maximizes discrete accuracy we use the training objects from TR :

$$\tau = \underset{\substack{(x, \cdot) \in TR \\ \tau_x := [e]^x \\ \tau_x \in [\min_1, \max_0]}}{\arg \max}}{accuracy}(e, \tau_x, TR)$$

where $accuracy(e, \tau_x, TR) = |\{(x, y) \in TR | y = cl_e^{\tau_x}(x)\}| / |TR|$.

4 Quantum Logic Decision Tree

Our expression e has so far been a disjunction of active minterms. The last step is to generate a quantum-logic inspired decision tree from e . The QLDT is just a compact presentation of the CQQL expression e based on active minterms. The basic idea is to regard the derived minterm weights as training data and to use a traditional DT algorithm for constructing the QLDT. The training data for the decision tree construction are just the bit values 0 and 1 from the binary code $bitcode(i - 1) = x_1 \dots x_n$ of all minterm identifiers i : The bit values of the bitcode are regarded as attribute values and the minterm weight as target for the DT algorithm:

$$TR' = \{(bitcode(i - 1), [\theta_i]) \mid i = 1, \dots, 2^n\}.$$

For example, following minterm weights for $n = 3$ and $e = (x_1 \wedge x_2 \wedge x_3) \dot{\vee} (\neg x_1 \wedge x_2 \wedge x_3) = x_2 \wedge x_3$ with evaluation $[e]^x = x_2 * x_3$ produce the QLDT shown on the right hand side. The leaves correspond to the minterm weights:

i	x_1	x_2	x_3	$[\theta]$
1	0	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	0
7	1	1	0	0
8	1	1	1	1

But what about the evaluation of a QLDT? The generated decision tree looks like a traditional decision tree. However, its evaluation differs. The evaluation result of a QLDT should be the same as the evaluation result of e in disjunctive normal form:

$$[e]^x = \left[\dot{\vee}_{i: [\theta_i]=1} \bigwedge_{j=1}^n c_{ij} \right]^x = \sum_{i: [\theta_i]=1} \prod_{j=1}^n [c_{ij}]^x. \tag{5}$$

Please note, that the disjunction of minterms is always exclusive and, that is why, its evaluation leads to a simple sum. The same holds for a split node of a QLDT. A QLDT assigns implicitly to every leaf a set of minterms sharing the same path conditions (split attributes). Let L_A refers to the set of all active leaves where every leaf is represented by the identifier i of one of its assigned minterms and let $path_i$ be the path from the root to leaf i . Then the evaluation of a QLDT is given by:

$$[qldt(e)]^x = \sum_{i \in L_A} \prod_{j \in path_i} [c_{ij}]^x = [e]^x = \sum_{i: [\theta_i]=1} \prod_{j=1}^n [c_{ij}]^x.$$

Actually, the evaluation of a QLDT takes into account only active leaves. The inactive leaves correspond to $\neg e$ and are connected to the active leaves by $[e]^x + [\neg e]^x = 1$. Therefore, the inactive leaves are removed from the QLDT without any loss of semantics and as result the QLDT becomes more compact.

So, we end up with the QLDT classifier:

$$cl_e^\tau(x) = th_\tau([qldt(e)]^x) = \begin{cases} 1 & \text{if } \sum_{i \in L_A} \prod_{j \in path_i} [c_{ij}]^x > \tau \\ 0 & \text{otherwise.} \end{cases}$$

5 Experiment

Next, we shall apply our approach to an example scenario and compare the resulting traditional decision tree with the generated quantum-logic inspired decision tree.

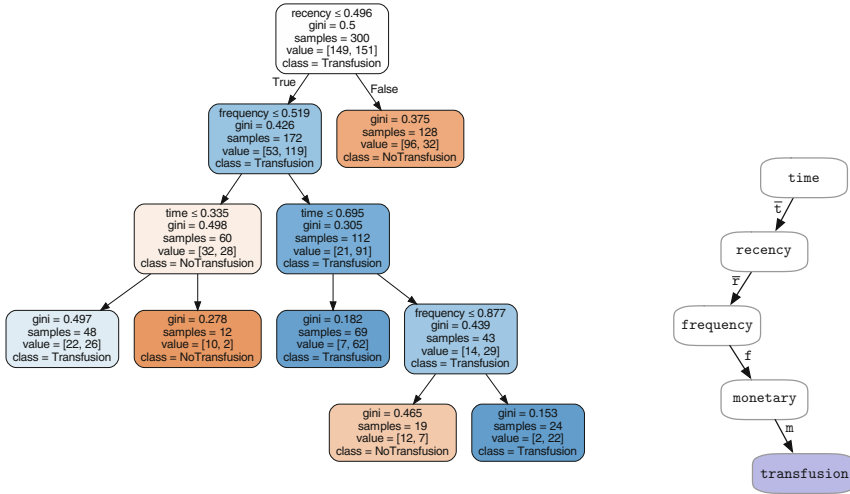


Fig. 2. Traditional decision tree for blood transfusion with 6 leaves (left) and quantum-logic inspired decision trees with best accuracy (right)

Our experimental dataset is the blood transfusion service center dataset.² A classifier needs to be found which predicts whether a person donates blood or not. To make that decision for every person we know **recency** (months since last donation), **frequency** (total number of donation), **monetary** (total blood donated in c.c.), and **time** (months since first donation). In the balanced case 178 people donated blood and 178 did not. 300 of the 356 people belong to the training set and the remaining ones to the test set. For further processing the attribute values of the four attributes are mapped to the unit cube $[0, 1]^4$ using a normalized rank position mapping. For that mapping all values of an attribute are sorted. Then, every value is mapped to its rank position divided by the total number of values. For tied values the rank maximum is taken.

We obtain a traditional decision tree with highest accuracy of 64%, 6 leaves, and 5 levels, see Fig. 2 (left). The QLDT, see Fig. 2 (right), with its highest accuracy of 70% is achieved when we choose $\theta_\rho = 0.71$ and $\tau = 0.045$. That QLDT corresponds to the expression

$$e := \neg t \wedge \neg r \wedge f \wedge m$$

and its evaluation is

$$[e] = [\neg t] \cdot [\neg r] \cdot [f] \cdot [m].$$

² <https://archive.ics.uci.edu/ml/datasets.php>.

Table 1. Differences between Boolean-logic-based decision tree and quantum-logic inspired decision tree

Criterion	DT	QLDT
Logic	Boolean logic	Quantum logic
Thresholds	On node level	On tree level
No. of classes	Many-class-classifier	One-class-classifier
No. of node children	≥ 2	≤ 2
Type of attributes	Categorical, ordinal, metric	Ordinal, metric
Class decision	Based on one single leaf	Based on all leaves
Complexity	At most exponential	Exponential

6 Conclusion

In our paper we suggest a decision tree classifier based on quantum logic. In contrast to Boolean logic, quantum logic can directly deal with continuous data, which is beneficial for many classification scenarios. Other than fuzzy logic, our quantum logic approach obeys the rules of the Boolean algebra. Thus, Boolean expressions can be transformed accordingly and a check of hypotheses against a Boolean expression becomes feasible.

In Table 1 we compare Boolean-logic-based decision trees with our quantum-logic inspired decision tree with respect to several criteria.

In a Boolean-logic-based decision tree every split decision at node level represents Boolean conditions and can be regarded as axis-parallel decision lines within the input space. For a QLDT the restriction on axis-parallel lines does not hold. A QLDT is appropriate in scenarios where the classification decision relies on sums and products rather than on a combination of Boolean values. The input for the threshold-based class decision is the evaluation result of the QLDT.

A QLDT represents syntactically a Boolean expression for one class. Thus, a QLDT classifier is a one-class classifier. Many-class classification problems can be transformed to multiple one-class classifier decisions.

Every inner QLDT node corresponds to exactly one logical expression with two outcomes. Since we drop 0-class leaves, some inner nodes have only one child. In contrast, a split rule in a traditional decision tree leads to more than or exactly two children.

For our QLDT we assume attribute values from the unit interval. The normalized rank position mapping maps ordinal and metric attribute values to the unit interval. But what about categorical attributes? We did not discuss this aspect but it can be easily solved by introducing an artificial attribute for each category.

As discussed above, the class decision in a traditional decision tree depends on exactly one leaf in contrast to the sum of all leaf scores in a QLDT.

The complexity of a Boolean-logic-based decision tree for a number n of attributes corresponds to the number of leaves. In the worst case, that number is exponential. However, in many real-case-scenarios the number of leaves is much lower. In contrast, the generation of the QLDT requires to compute the weight for all 2^n minterms. Thus, too many attributes cause a complexity problem more for the QLDT rather than for a Boolean-logic-based decision tree.

References

1. Aggarwal, C.C.: Data Mining: The Textbook. Springer, Cham (2015). <https://doi.org/10.1007/978-3-319-14142-8>
2. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint [arXiv:1702.08608](https://arxiv.org/abs/1702.08608) (2017)
3. Freitas, A.A.: Comprehensible classification models: a position paper. ACM SIGKDD Explor. Newsl. **15**(1), 1–10 (2014)
4. Hüllermeier, E., Schmitt, I.: Non-additive utility functions: Choquet integral versus weighted DNF formulas. In: Gaul, W., Geyer-Schulz, A., Baba, Y., Okada, A. (eds.) German-Japanese Interchange of Data Analysis Results. SCDAKO, pp. 115–123. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-01264-3_10
5. Janikow, C.Z.: Fuzzy decision trees: issues and methods. IEEE Trans. Syst. Man Cybern. Part B (Cybernetics) **28**(1), 1–14 (1998)
6. Jiménez, F., Martínez, C., Marzano, E., Palma, J.T., Sánchez, G., Sciavicco, G.: Multiobjective evolutionary feature selection for fuzzy classification. IEEE Trans. Fuzzy Syst. **27**(5), 1085–1099 (2019)
7. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
8. Mittelstaedt, P.: Quantum logic. In: PSA 1974, pp. 501–514. Springer, Cham (1976). https://doi.org/10.1007/978-94-010-1449-6_28
9. Mittelstaedt, P.: Quantum Logic. D. Reidel Publishing Company, Dordrecht (1978)
10. Olaru, C., Wehenkel, L.: A complete fuzzy decision tree technique. Fuzzy Sets Syst. **138**(2), 221–254 (2003)
11. Schmitt, I.: Quantum query processing: unifying database querying and information retrieval. Citeseer (2006)
12. Schmitt, I.: QQL: a DB&IR query language. VLDB J. **17**(1), 39–56 (2008)
13. Schmitt, I.: Incorporating weights into a quantum-logic-based query language. In: Aerts, D., Khrennikov, A., Melucci, M., Toni, B. (eds.) Quantum-Like Models for Information Retrieval and Decision-Making. SSTEAMH, pp. 129–143. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25913-6_7
14. Schmitt, I., Baier, D.: Logic based conjoint analysis using the commuting quantum query language. In: Algorithms from and for Nature and Life, pp. 481–489. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-00035-0_49
15. Strumbelj, E., Kononenko, I.: An efficient explanation of individual classifications using game theory. J. Mach. Learn. Res. **11**, 1–18 (2010)