The XAI Paradox: Systems that Perform Well for the Wrong Reasons

Cor Steging, Lambert Schomaker, Bart Verheij
Department of Artificial Intelligence, Bernoulli Institute
University of Groningen

Abstract

Successful machine learning approaches are often regarded as "black box" systems: they perform well, but are unable to explain the reasoning behind their decisions. The emerging sub-field of Explainable Artificial Intelligence (XAI) aims to create systems that can provide explanations underlying their internal reasoning. In this study, we investigate to what extent systems learn explanatory structures that make sense in a domain. Using artificial datasets whose internal structure is determined beforehand, we are able to show that the reasoning of systems that perform well is not necessarily sound. We generate datasets whose outputs are based on predetermined formal input conditions defined in terms of symmetric Boolean functions. When the output of the generated datasets are dependent on a conjunctions of input conditions, we find that trained systems perform well on the conjunction problem, but—paradoxically—not on the individual conditions separately. This implies that a confounding structure within the data is learnt that still allows making the correct decisions. In other words, the systems perform well, but for the wrong reasons, providing a puzzle for the XAI movement.

Introduction

In recent years, the notion of artificial intelligence has reached the masses with the introduction of smart devices, digital assistants, self-driving cars and many other inventions. This booming interest in AI can largely be attributed to the success of machine learning in combination with the growing availability of big data. Commonly employed machine learning techniques are inherently "black box" systems, lacking a possibility to provide an explanation underlying their reasononing.

In certain domains, such as the law and medicine, there is a need for more explainability and transparency (cf. also the discussion about a 'right to an explanation' in the EU General Data Protection Act; see e.g. (Edwards and Veale 2017)). Increasing explainability and transparency is studied in the emerging field of Explainable Artificial Intelligence (XAI) (Gunning 2017). Examples of such XAI systems are rule extraction algorithms, such as (Lu, Setiono, and Liu 1996), that create sets of rules for the users that describe the reasoning of "black box" systems, or glass-box systems, like (Holzinger et al. 2017), in which the users are able to influence what the systems learn. Yet another approach is used in a recent deep learning system for diagnosing retinal diseases (outperforming medical experts). The system is divided into a framework of smaller systems; one for each stage of the diagnostic process (De Fauw et al. 2018), making it easier for the clinicians to investigate and understand the modular reasoning of the system. In these and different ways, XAI systems aim to yield accurate and meaningful explanations of their reasoning, without sacrificing performance.

A question that must be asked in XAI research is whether or not the explanation that a XAI system gives indeed makes sense in the domain. That this is not a trivial issue is in particular suggested by the phenomenon of adversarial examples in image classification, where input images are slightly altered using a perturbation (Yuan et al. 2017). To the human eye, there is no apparent difference between the original input image and its altered version, while a machine learning system makes a completely different classification after the image is altered. Another example is the classic double-spiral problem (Lang and Witbrock 1989), in which neural networks are tasked with discriminating between two interlocking spirals given an x and y coordinate. Rather than actually learning the mathematical functions that define the spirals, the networks create a hierarchical structure of regions that divide the image into a number sectors, each belonging to one of the two spirals. This suggests that the understanding that the systems have differs strongly from that of humans, despite the fact that the system usually makes the correct classifications. In other words, systems perform well, but apparently for the wrong reasons. In and of itself, this is still impressive, as the machine learning system is able to extrapolate structures from the data that we as humans are not able to perceive, and consequently achieve a high performance. However, explanations that are generated using XAI techniques may not make sense to the users, countering the goals of transparancy and explainability.

The aim of this study is to investigate this question: to what extent do systems learn structures that make sense in a domain? In order to operationalize this idea, we generate structured datasets, where the embedded structure is used as a model of what makes sense in a domain. We investigate machine learning techniques in terms of how well they are able to learn the structure that defines datasets, rather

than only in terms of a general performance accuracy. More specifically, artificial datasets will be generated from a set of predetermined formal conditions in terms of symmetric Boolean functions, on which neural networks and decision tree systems will train. In this way, we can study how well the systems are able to learn the correct conditions from the data. By using conjunctions of conditions when defining datasets, we can investigate whether the system learns each of the individual conditions correctly, or instead learns a different confounding structure that still accurately maps the input to the output.

Our method of using artificial datasets with predetermined structure was inspired by neural network research in a legal setting (Bench-Capon 1993), a 1990s study of XAI themes in the field of AI & Law. In that field, XAI issues arose already then, since explainability and transparency are especially pertinent in the law, where decisions must be made for the right reasons. That work used a specific legal example. In this study, we use an abstract, parameterized set of problems (defined using families of symmetric Boolean functions) in order to investigate the task of learning structure that makes sense in a domain.

Previous Research: the Welfare Benefit Dataset

(Bench-Capon 1993) uses a particular case study in the field of law investigating whether neural networks can achieve a high performance on open texture law problems. He examines whether or not the rationale that the neural networks use in their classification is acceptable. To this end, a fictional dataset is generated that describes the personal information of elderly people and whether they are eligible for a particular welfare benefit. The eligibility depends on the six conditions shown below, and the personal information of the individual. If and only if all of the six conditions apply to an individual, her or she is eligible for a benefit.

1. The person should be of pensionable age (60 for a woman, 65 for a man).
2. The person should have paid contributions in four out of the last five relevant contribution years.
3. The person should be a spouse of the patient.
4. The person should not be absent from the UK.
5. The person should have capital resources not amounting to more than £3,000.
6. If the relative is an in-patient the hospital should be within a certain distance: if an out-patient, beyond that distance.

Training neural networks on this dataset for predicting the eligibility of new individuals leads to high classification accuracies, with the number of correct predictions ranging from 98.75% to 99.25%.

In order to determine how well the conditions that define eligibility are learned by networks, special test datasets are generated for each condition, in which the personal information is generated such that all other conditions are satisfied. For instance, the first condition states that the person should be of pensionable age, which is 60 for women and 65 for men. In order to measure how well this condition is learned by a network, a test dataset is generated with values for the age and gender variables across the full possible range of values. The values of the remaining variables are generated such that they satisfy the other five conditions. In this way, a high performance on the resulting test dataset can only be achieved if the network has learned the condition 'the person should be of pensionable age'.

The research shows that the networks are unable to learn all of the conditions that define eligibility, despite the high classification accuracy. The condition of pensionable age, for instance, is not learned correctly by the networks. Bench-Capon shows that a high classification accuracy can be obtained in this example by only having learned a few of the conditions. By altering the statistical distribution of the training dataset it was possible to improve upon how well certain conditions are learned, but accurately learning all of them was not achieved. The actual conditions and structures that define this dataset are therefore not learned by the networks. In this particular example, a high performance does therefore not necessarily guarantee a sound rationale.

In followup research, a comparison was made between the performance of the neural networks (Bench-Capon 1993), defeasible logic (Johnston and Governatori 2003) and an adaptation of the CN2 algorithm using argumentation (Možina et al. 2005) on the the same fictional welfare benefit dataset. All systems produced a high classification accuracy, but none of them were able to exactly reproduce all of the six conditions of the dataset. Additionally, it was shown that conditions 3 and 4, the simple boolean conditions, are easily identified by all of the systems. Similarly, the systems seem to have no difficulty in identifying condition 5, in which a specific threshold value needs to be exceeded. Conditions 1 and 6 on the other hand, in which specific combinations of two variables are required, are difficult for the systems to learn. Additionally, so-called condition 2, in which at least 4 out of the 5 variables need to be true, are not easily identified by the system either.

This research showed that for the welfare benefit dataset good performance could be achieved without completely learning the structure embedded in the data. In the present study we aim to gain more insight in this phenomenon. For this, we construct a series of structured data sets in terms of symmetric Boolean functions.

Method: Symmetric Boolean Functions

We saw that in the welfare benefit dataset systems were unable to learn all of the conditions defining the data, despite achieving a high classification accuracy. Individually, the conditions were quite simple and all modern machine learning systems should be able learn them. Combined, however, they become more difficult to learn. This could indicate that there is an interaction effect between conditions, that causes a condition to be learned less successfully when other conditions are present. This study sets out to investigate how well machine learning systems are able to internalize such conditions that define a training dataset. Furthermore, the possible interaction effect between conditions will be examined.

For defining conditions that generate datasets, we use symmetrical Boolean functions. The output of these functions is only based on the number of 1's in the input vector, hence the location of the 1's in the input vector is irrelevant. An example of a symmetrical Boolean function with two variables is the XOR function (well-known for its role in the history of neural networks), which provides an output of 1 if and only if the number of 1's in the input vector is equal to 1. With more than two variables, the XOR function can be generalized to the parity function, which is a function that returns an output of 1 if and only if there is an odd number of 1's in the input vector. We use three families of symmetrical Boolean functions (Wegener 1987):

- Parity function: $f^n(x) = 1 \leftrightarrow |x| \equiv 1 \bmod 2$.
- Exact value function: $f^n_m(x) = 1 \leftrightarrow |x| = m$.
- M-out-of-N function: $f^n_m(x) = 1 \leftrightarrow |x| \geq m$.

Here $|x|$ denotes the number of 1's in vector x. All functions have integer parameter n for the length of the Boolean input vector x with n values. The exact value and M-out-of-N functions have an additional integer parameter $m \leq n$. These three families of symmetrical Boolean functions form the basis of the conditions that define datasets as they are used in our experiments.

Datasets

For our experiments, a parametrized series of datasets is generated based on conditions defined in terms of symmetrical Boolean functions. Each instance in the dataset has an output value, which is true if and only if all conditions are true as well. This set of conditions can include the M-out-of-N function, the exact value function and the parity function, with parameters n and m as above. Parameter n denotes the number of variables of the condition and parameter m is used in M-out-of-N functions and exact value functions to determine the value of the output. The total number of variables of each instance in the datasets is based on the number of variables of each of the conditions, as for instance in the examples in Table 1. The output variable is set to true if and only if two conditions both hold: the parity condition with 2 variables A and B (in red) and the M-out-of-N condition with 3 variables C, D and E (in blue), where parameter m is set to 2. In the first instance, the parity condition is false and the M-out-of-N condition is true; hence output is false. In the second instance, the output variable is true, as both the parity condition and the M-out-of-N condition are true.

For any pair of conditions (Parity, Exact; Parity, M-out-of-N; M-out-of-N, Exact) and all possible combinations of parameters m and n, the following datasets are generated: a training set, a general test set and a specific test set for each condition. First of all, a training set consists of 150,000 instances where half of the instances are generated randomly such that their output value is true, and the other half are generated randomly such that their output value is false. In the latter half, the instances are generated such that they fail on each of its conditions equally. With two conditions for example, this means that the latter half (the half where the output is false) would consists for 50% out of instances where the first condition is false and for 50% out of instances where the second condition is false. If a condition is not specifically set up to fail, its variables are provided with random values. The general test set consists of 150,000 instances, and is generated in the exact same fashion.

Table 1: Two example instances of a dataset with 2 parity variables A and B (red) and 3 M-out-of-N variables C, D and E (blue), where m=2.

A	B	C	D	E	Output
1	1	0	1	1	0
1	0	0	1	1	1

For each condition, a specific test set is also generated, which is used to determine how well a system is able to learn that specific condition. Given such a condition, its test set is constructed by generating all of the possible input values that the condition can have. For example, the XOR condition (or parity function with n=2) has $2^2 = 4$ possible values and would therefore have a specific test set with 4 instances. The variables of any additional conditions are given random values such that these conditions are satisfied. This ensures that the output is dependent on only the single condition for which the test set is generated.

Machine Learning Algorithms

Both a decision tree algorithm and a neural network are trained on the training sets. The resulting systems are tasked with classifying the general test sets and the specific test sets. The decision tree system uses the CART algorithm (Breiman et al. 1984) with the Gini impurity and it does not make use of early stopping or any other form of pruning. The neural networks used consist of three hidden layers, with 25, 10 and 3 hidden nodes respectively. The learning rate is set to 0.05 and the sigmoid is used as the activation function. The networks use a mini-batch approach with a batch size of 50. The maximum number of times that the entire training dataset will be presented to the network during the training phase is set to 2000 rounds. Training stops early if there is no change in training errors over the past 10 rounds.

Experiments

Both machine learning algorithms are first trained and tested using datasets with only a single condition; either the parity, M-out-of-N or exact value function. This will determine whether the systems are actually able to learn each of the conditions individually. The systems will then be trained and tested on datasets for each pair of conditions, in order to investigate both how well the systems are able to learn the combined problem and how well they are able to learn the individual conditions that make up the combined problem.

To show the ability of the systems perform under noise, additional tests will be performed. In these experiments, a particular percentage of the instances of the training datasets are given a "noisy variable". These noisy variables have their values inverted, such that a 1 becomes a 0 and a 0 becomes a 1. After training on the datasets with noise, the systems are tasked with classifying the same test sets as before.

Results

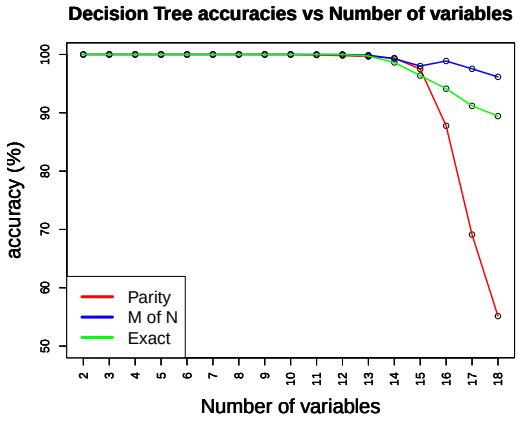
Single Condition Datasets

First of all, datasets are generated with only one condition with different numbers of variables n; from 2 to 18. For the M-out-of-N and exact value condition, the value of m varies as well: between 1 and n. For each condition and each possible combination of m and n, both the decision tree algorithm and the neural network are trained on the training dataset and then tested on the general test set and the specific condition test set. In Figure 1 the accuracy of the decision tree and the neural network on the general test set is shown versus the number of variables for all three conditions. For the M-out-of-N and exact value condition, the mean accuracies across all possible values values of m are displayed.

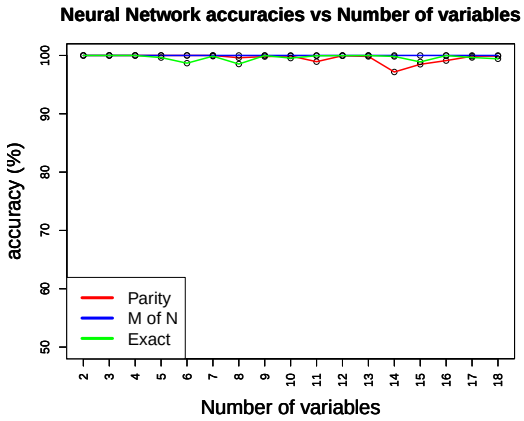
The accuracies of these systems on the specific test sets were almost identical, and a repeated measures anova showed no significant difference between the performance on the general test set and on the specific test set for all of the conditions. Figure 1 shows that the accuracy of the decision tree system starts to decrease as the number of variables of the condition increases. The neural network does not appear to have difficulties with any of the conditions, as the mean accuracy for each number of variables never reaches lower than 98%. Earlier research has shown that with sufficient training and enough hidden nodes, a neural network should theoretically be able to learn any parity function (Wilamowski, Hunter, and Malinowski 2003). If we assume that a similar rule applies to the other types of symmetrical Boolean functions, the 150,000 training instances appear to be sufficient for a network of this format to learn the conditions. Clearly, this is not sufficient for the decision tree algorithm for higher values of variables. The average accuracy of the neural network is relatively stable near 100% for each condition and each number of variables. The accuracy of the decision tree decreases at around 14 variables for all three conditions. The accuracy on the parity condition drops rapidly to 55% with 18 variables, whereas the accuracies of the exact value and M-out-of-N conditions decrease to 89% and 96% respectively at 18 variables.

Varying the percentage of instances with a noisy variable between 0 and 100% yields the graphs shown in Figure 2. These graphs display the mean accuracies of the decision tree (red) and the neural network (blue) on the specific test set for each of the three symmetric Boolean functions. The accuracies are average accuracies across values of n ranging from 2 to 10 and across all possible values of m. It should be noted that inverting the value of a variable does not always make the outcome of the instance incorrect, and 50% instances with a noisy variable need not mean 50% incorrect instances.

The results show that even under extreme levels of noise, the systems are able to learn the M-out-of-N and exact value function quite effectively. The M-out-of-N function is most robust to noise, as seen in Figure 2b. The accuracies on the exact function drop faster with more noise, as seen in Figure 2c.



(a) Decision tree accuracies

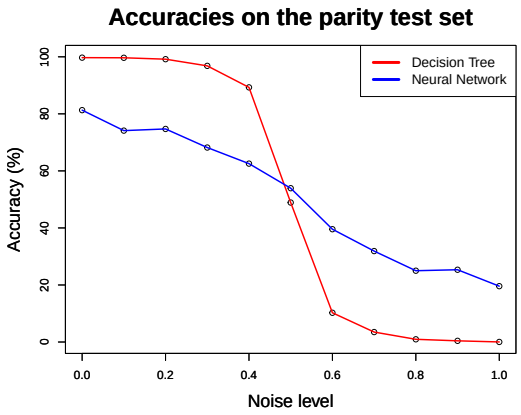


(b) Neural network accuracies

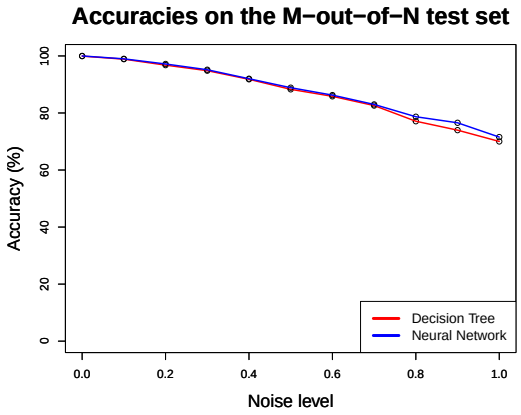
Figure 1: The accuracies of the decision tree (A) and neural network (B) on the parity condition (red), M-out-of-N condition (blue) and exact value condition (green) for varying number of variables.

Multiple Condition Datasets

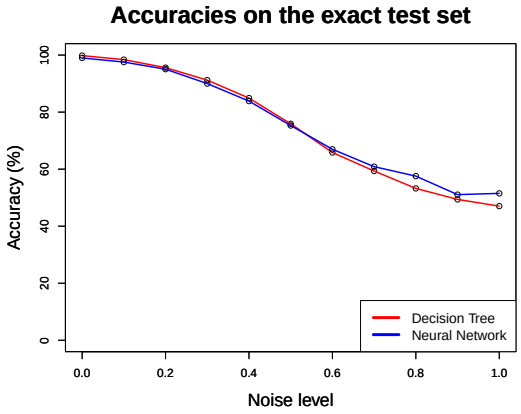
As mentioned earlier, systems that are able to learn how to solve a combined problem, may not necessarily have learned all of the conditions that make up the combined problems; there can be an interaction effect between the conditions. To investigate this, neural networks and decision trees were trained and tested on datasets that include two of the three symmetrical Boolean functions. First of all, the interaction between the parity function and the exact value function is examined, followed by the interaction between the parity function and the M-out-of-N function, and lastly the interaction between the M-out-of-N function and the exact function will be explored. For all three combinations, the number of variables per function will be varied between 2 and 10, thus creating instances with 4 to 20 variables when combining the two conditions. Just as in the results of the single conditions, the accuracies of the M-out-of-N function and the exact function are averaged over all possible values of m.



(a)



(b)



(c)

Figure 2: The effect of varying levels of noise on the accuracies of the decision tree algorithm (red) and the neural network (blue) after training on the parity function (a), the M-out-of-N function (b) and the Exact value function (c).

For each interaction, three accuracies will be examined per system: the accuracy on the general test set, the accuracy on a specific test set of the first condition and a specific test set of the second condition.

An overview of the average accuracies of the decision trees and neural networks is given in Tables 2 and 3. These show the mean accuracies for each interaction on each of the different test sets: the general test set, and one test specific test set for each of the two functions of the interaction. What is clear from these tables, is that the mean accuracy of both the neural network and the decision tree on the general test set is quite high for each interaction. However, the accuracies on the specific test sets, which indicate how well the functions themselves are learned, are almost always a lot lower. With the parity-exact interaction, for instance, the decision tree algorithm preforms with an accuracy of 98.5% on the general test set, but only 17.5% on the parity test set and 78.8% on the exact test set. This shows that a system can perform with a high accuracy, even if it has not learned all of the conditions correctly. The accuracies of the neural network and the decision tree system are also quite similar to each other, which shows that the trends in accuracies are not limited to a single machine learning algorithm. The relationships between the number of variables for each function and the accuracies of each of the tests sets are recorded as well, an example of which can be seen in the heatmaps of Figure 3 and Figure 4. These display the mean accuracies of the decision tree on the three different test sets for the M-out-of-n and exact value function interaction versus the number of variables of both functions on the x- and y-axis.

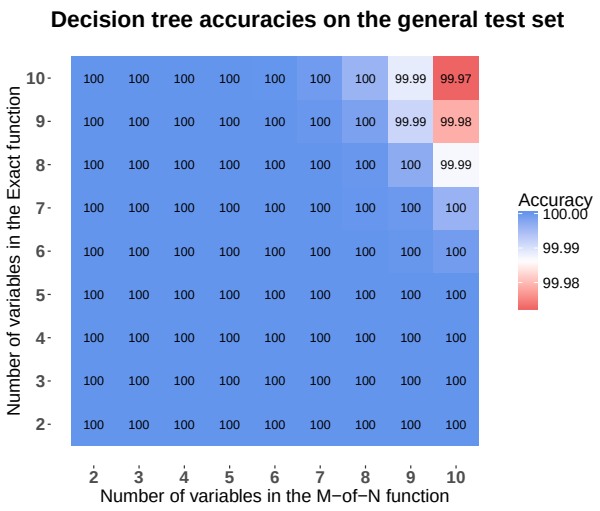
The accuracies on the general test set (Figure 3a) show the overall performance of the decision tree system on the combined problem. Unsurprisingly, more M-out-of-N and exact value function variables lead to a decrease in accuracy; as the problem becomes more complex, it is more difficult for the system to learn. Figure 3b displays how well the M-out-of-N function is learned by the system, and a similar decrease in accuracy can be observed when more variables are used in the data. Interestingly, the number of M-out-of-N variables has a bigger influence on the accuracy on this test set

Table 2: The mean accuracies of the decision tree on each test set after training on each interaction.

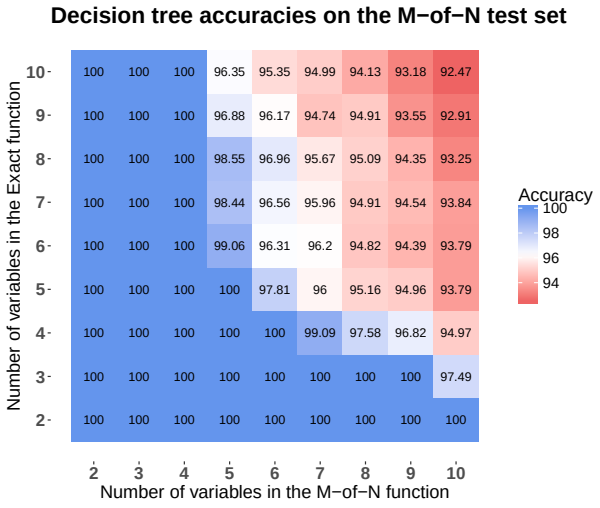
	General accuracy	Function 1 accuracy	Function 2 accuracy
Parity, Exact	98.50	17.48	78.75
Parity, M-out-of-N	99.99	30.36	81.75
M-out-of-N, Exact	99.99	96.00	32.53

Table 3: The mean accuracies of the neural network on each test set after training on each interaction.

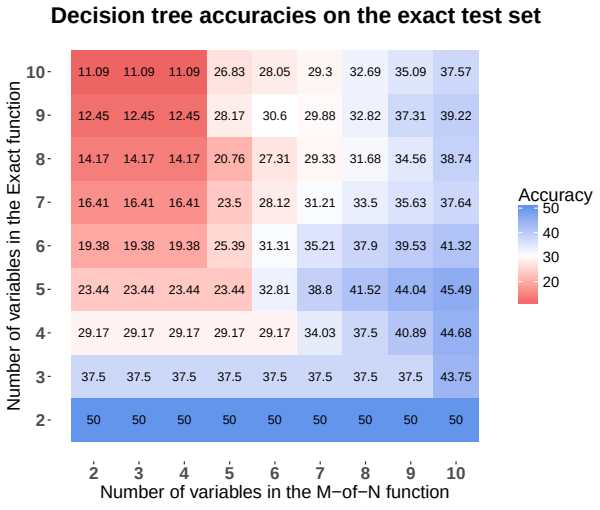
	General accuracy	Function 1 accuracy	Function 2 accuracy
Parity, Exact	99.65	14.59	78.31
Parity, M-out-of-N	100.00	30.59	81.65
M-out-of-N, Exact	99.99	99.88	26.14



(a)

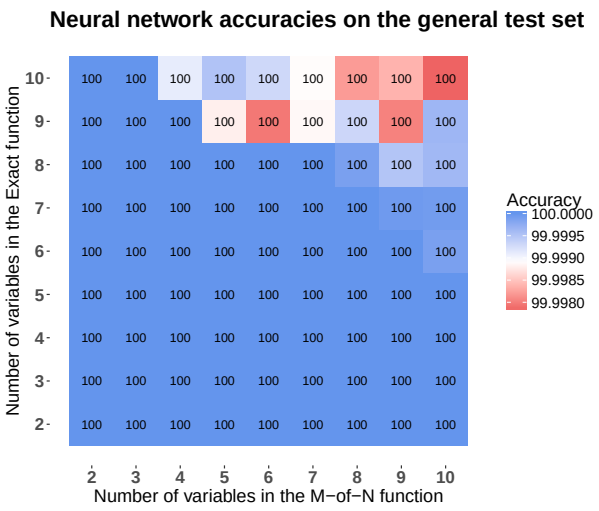


(b)

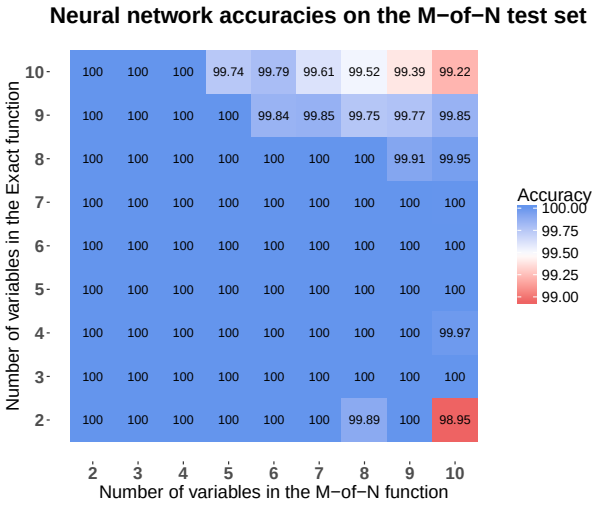


(c)

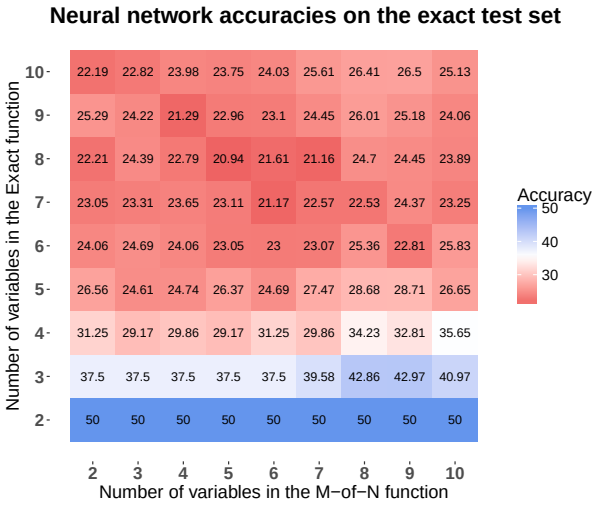
Figure 3: The mean accuracies of the decision tree system on the different test sets after training on the training set with a M-of-N function (x-axis) and an exact function (y-axis) with varying numbers of variables in each function.



(a)



(b)



(c)

Figure 4: The mean accuracies of the decision tree system on the different test sets after training on the training set with a M-of-N function (x-axis) and an exact function (y-axis) with varying numbers of variables in each function.

than the number of exact value variables. This could be because functions with more variables are more difficult for the systems to learn. In Figure 3c, which displays how well the exact value function is learned, a different effect can be observed. The accuracy on this test set decreases with more exact value variables, but increases with more M-out-of-N variables. So a larger number of exact value variables decreases how well the M-out-of-N function is learned, while a larger number of M-out-of-N variables increases how well the exact function is learned. The accuracies of the neural network (Figure 4 show similar, but less pronounced trends.

We have investigated these effects for all pairs of conditions. A summarized overview of the effects that the conditions have in the other interactions is shown in Table 4.

Discussion

Our findings show that in our generalized setting with symmetric Boolean functions a high performance in terms of classification accuracy does not guarantee that the system has learned the conditions that define the dataset, as evident by Tables 2 and 3. Also we saw that when dealing with interactions, the number of variables of a function can either increase or decrease how well a system learns the function, depending on the other function that is in the dataset. The accuracy on the parity test set increases with a low number of parity variables in both the interaction with the exact value function and the M-out-of-N function. The accuracy on the exact value test set increases with a high number of exact variables in the interaction with the parity function and with a low number of exact variables in the interaction with the M-out-of-N function. Similarly, the accuracy on the M-out-of-N test set increases with a high number of M-out-of-N variables in the interaction with the parity function, and with a low number of M-out-of-N variables in the interaction with the exact value function. Table 4 shows that higher numbers of parity variables always have a negative effect on the how well the parity function is learned, regardless of other the functions that define the dataset. This is also true for the other two functions, except when interacting with the parity function; when interacting with the parity function, a high number of exact/M-out-of-N variables has a positive effect. This suggests that the interference of the parity function has a greater negative impact on how well the the exact and M-out-of-N function are learned than the number of variables of said function. This could explain why the increase in variables creates a increase in learning the exact and M-out-of-N functions in combination with the parity function.

Table 4: This table shows the effect that a high number of variables of a function has on the performance of a system (in terms of learning the function) for all functions that it interacts with.

Performance on ↓	Parity	Exact	M-out-of-N
Parity	-	decreases	decreases
Exact	increases	-	decreases
M-out-of-N	increases	decreases	-

The experiment with the symmetrical Boolean functions has confirmed that the neural network and the decision tree algorithm can learn the XOR condition and its generalized form, the parity function, if there is no interference from other variables. Compared to the M-out-of-N and exact value function, the parity function seems to be the most difficult function for the machine learning systems to learn. Both neural networks and decision trees, however, can learn them with sufficient training (cf. the fact that neural networks are universal approximators (Hornik 1991)). Once multiple conditions define the output of a dataset, however, the systems are unable to learn the individual conditions as well. The accuracies on the general test sets are still quite high for all possible interactions between the functions, but the systems do not learn the functions themselves as well. This supports the claim that a high classification accuracy is no guarantee that the right rationale is used in the classification process. The accuracies on the general test set and on each of the functions individually can be seen in Table 2 and Table 3. It is interesting to see how systems can perform with an accuracy of almost 100% on the general test set, yet fail dramatically on the test sets of the individual functions. This indicates that they do not actually learn these functions, but rather a different function that somehow accurately maps the input to the output. There must therefore be a confounding structure within the data that the systems learn instead of the original, intended structure. The systems could potentially learn the intended structure if additional constraints are added during the learning phase (Schwab and Link 2012). However, this would not prove that the systems themselves are able to internalize the intended structure, but rather display a form of intelligence by proxy.

In most of the experiments, the performances of the neural network and the decision trees were quite similar. However, the experiments in this study used relatively simple functions and conditions, which both connectionist and symbolic machine learning techniques can learn. When dealing with high-dimensional image or speech data, for instance, (deep) neural networks will generally outperform the symbolic learning techniques.

Conclusion

In the future, machine learning algorithms that simply execute a task with a high performance will not always suffice; for certain domains and certain applications, an explanation of their decision making will be required (Gunning 2017). This explanation, however, is dependent on the reasoning of the systems; if the reasoning is unsound, the explanation will be unsound as well. This study has shown that machine learning algorithms do not always internalize the structure of their training data as we would expect. With regards to the emerging XAI techniques and the explainable models that they generate, such unsound rationales underlying machine learning systems can form a hindrance in creating an understandable explanation.

References

Bench-Capon, T. 1993. Neural networks and open texture. In *Proceedings of the 4th International Conference on Artificial Intelligence and Law (ICAIL 1993)*. New York (New York): ACM. 292–297.

Breiman, L.; Friedman, J.; Stone, C.; and Olshen, R. 1984. *Classification and Regression Trees*. Boca Raton (Florida): CRC press.

De Fauw, J.; Ledsam, J.; Romera-Paredes, B.; Nikolov, S.; Tomasev, N.; Blackwell, S.; Askham, H.; Glorot, X.; ODonoghue, B.; Visentin, D.; et al. 2018. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature Medicine* 1.

Edwards, L., and Veale, M. 2017. Slave to the algorithm? why a right to explanation is probably not the remedy you are looking for. *Duke Law & Technology Review* 16:18–84.

Gunning, D. 2017. Explainable artificial intelligence (XAI). https://www.darpa.mil/attachments/XAIProgramUpdate.pdf. Accessed: 2018-06-15.

Holzinger, A.; Plass, M.; Holzinger, K.; Crisan, G.; Pintea, C.; and Palade, V. 2017. A glass-box interactive machine learning approach for solving np-hard problems with the human-in-the-loop.

Hornik, K. 1991. Approximation capabilities of multilayer feedforward networks. *Neural Networks* 4(2):251–257.

Johnston, B., and Governatori, G. 2003. Induction of defeasible logic theories in the legal domain. In *Proceedings of the 9th International Conference on Artificial Intelligence and Law (ICAIL 2003)*, 204–213. ACM, New York.

Lang, K., and Witbrock, M. 1989. Learning to tell two spirals apart. *Proceedings of 1988 Connectionists Models Summer School* (04):52–59.

Lu, H.; Setiono, R.; and Liu, H. 1996. Neurorule: A connectionist approach to data mining. In *Proceedings of the 21st VLDB Conference*.

Možina, M.; Žabkar, J.; Bench-Capon, T.; and Bratko, I. 2005. Argument based machine learning applied to law. *Artificial Intelligence and Law* 13(1):53–73.

Schwab, I., and Link, N. 2012. Learn more about your data: a symbolic regression knowledge representation framework. *International Journal of Intelligence Science* 2(04):135.

Wegener, I. 1987. The complexity of symmetric boolean functions. In *Computation Theory and Logic*. New York (New York): Springer. 433–442.

Wilamowski, B.; Hunter, D.; and Malinowski, A. 2003. Solving parity-n problems with feedforward neural networks. In *Proceedings of the International Joint Conference on Neural Networks*, volume 4, 2546–2551. Piscataway (New Jersey): IEEE.

Yuan, X.; He, P.; Zhu, Q.; Bhat, R.; and Li, X. 2017. Adversarial examples: Attacks and defenses for deep learning. *CoRR* abs/1712.07107.