university of groningen / faculty of science and engineering / artificial intelligence

The effect of dependencies in data on the accelerated argumentation-based learning algorithm

Bachelor's Project Thesis

Jonas Scholz, s4287797, j.scholz.1@student.rug.nl,
Supervisor: Prof Dr H.B. Verheij

Abstract: With the increased usage of artificial intelligence(AI), there is an increase in the misuse of AI, as users do not understand how AI comes to its decisions. This means that there is a need for an AI that is explainable, meaning that humans can understand how the AI came to its decision. There are some explainable AI systems that are able to both learn as well as stay human-understandable, one such system is the accelerated argumentation-based learning(AABL) algorithm. This is a new learning algorithm that uses argumentation to determine the correct output. As the AABL algorithm is still new the limits of this algorithm are still unknown, as this algorithm seems promising it is important to determine where the limits lie. In this regard, this paper will explore the effects of dependencies in the data on the accuracy of the model in comparison to a decision tree and neural network. The data shows that the AABL algorithm outperforms the other machine learning algorithms eventually for higher dependencies, though this trend does not seem to continue for 21 dependencies. Furthermore, the AABL algorithm slows down considerably for higher dependencies compared to the other two algorithms. This, therefore, highlights that while this approach towards explainable AI works also for other scenarios compared to the original scenario it was tested on, it also shows that the algorithm still has some limitations that need to be addressed by derived algorithms.

1 Introduction

,,Texas professor misuses ChatGPT and fails most of class for AI plagiarism"*, this is a headline from the Independent on 18th of May 2023. It is about a professor that uses ChatGTP, a large language model created by OpenAI†, to determine whether his students used ChatGTP or a similar language model to write their assignment. This demonstrates a problem in artificial intelligence (AI), as AI gets more popular more and more people are coming into contact with AI systems and are using it while not understanding how these systems work. This is made even more complicated due to the fact that modern AI systems often make use of deep neural networks, which are hard to understand, as the system does not inherently make it clear how it came to the decision. This also extends to the designers of the system, which makes it more problematic to ensure that the system behaves as intended. This can be problematic as can be seen by the example above, but this also has consequences on other parts of society where it is vital that there is a clear reasoning behind a decision. One such sector is law, where it is important that an AI system not only gives the decision if a person is guilty or not, but also gives a good foundation on which it based its decision on in order for humans to be able to understand and make sure that the decision is correct and in this case just.

This is a reason why there is an increasing movement for the development of explainable artificial intelligence (XAI), which is putting more emphasis on the ability of AI systems to indicate how they

*https://www.independent.co.uk/news/world/americas/chatgpt-ai-plagiarism-texas-a-and-m-b2341238.html, Last retrieved on 01.06.2023

†https://openai.com/blog/chatgpt, Last Retrieved 01.06.2023

came to the result in a human-understandable manner. According to Adadi & Berrada (2018) there are four main reasons on why it is important to work on XAI, the first being that an explanation is useful to justify the decision was made. This then also ties into the next reason, which is the ability to understand the system better and therefore have greater control over the system as it can be easier to identify flaws in the system. This then also means that better explainability makes it easier to improve such systems and lastly understanding the reasoning of AI systems might allow us to discover new insights, as AI is good at discovering patterns in data, which if the AI is explainable could help us understand why the data has these patterns.

Given all of these reasons, it is clear that it is important to look into XAI systems. One approach to make XAI is to an architecture for the AI that is inherently explainable to humans, which means that it is trivial to see how a decision was reached by the AI. A possible type of architecture for that are argumentation systems, which as the name implies use arguments to determine the output. They could be an effective approach for XAI, as humans are prone to use argumentation in our daily life and are therefore accustomed to using them (Baroni et al., 2020), in fact, we often expect that other people can provide us with arguments to support their opinion, even kids use arguments to persuade their parents to not do for example homework.

One of these argumentation systems is the accelerated argumentation base-learning algorithm(AABL) by Ayoobi et al. (2021), which is capable of creating an attack graph in an online-incremental manner, which means that the system learns how to predict the output by using one data point at a time, which allows it to smoothly integrate new data. This algorithm outperformed other similar online learning algorithms, for the decision of the appropriate recovery behaviour for a robot.

As this algorithm is still new the limits of this algorithm are not yet known, therefore it is prudent to see how this algorithm will perform in other conditions. As the algorithm is currently designed to only take into account data dependencies of up to two variables it is especially important to see how a modified algorithm which allows for higher dependencies will perform if the data contains higher dependencies. This then leads to the research question: Does the number of dependencies in the data affect the accuracy or learning rate of the accelerated argumentation-based learning algorithm?

In order to answer this question, the paper will explain the argumentation framework that the learning algorithm uses in Section 1.1, which is then followed by the explanation of the learning algorithm in section 1.2, which is then followed by two sections describing the other two machine learning approaches used to compare to the AABL algorithm. After that section, a description on the problems of the original algorithm is given followed by a description of how it was modified to fit the experiment follows. This is then followed by a description of the data generation and the experimental setup. Section 3 holds the results of the experiment and is followed by the last section, which discusses the results and concludes this paper.

1.1 Argumentation

Argumentation is an integral part of human communication (Baroni et al., 2020), as it allows to support opinions and therefore convince others of their views, furthermore, it allows others to understand why certain decisions were made and can therefore better evaluate whether they agree with the reasoning.

An influential formalization of argumentation was created by Dung (1995), in his paper he defines the abstract argumentation framework which allows for a formal description of acceptance for sets of arguments. In order to achieve this he abstracts from the actual content of the arguments to attack relations between arguments which allows for the evaluation of for example admissible sets of arguments. This can then be visualized as a directed graph where the individual nodes are arguments and the connections are the attack relations, between them. An admissible set of arguments is a subset of all arguments that do not attack each other and also defend each other from attacks from arguments that do not belong to that set.

1.1.1 Bipolar Argumentation Frameworks

The bipolar argumentation framework is an extension of the abstract argumentation framework and was formalized by Cayrol & Lagasquie-Schiex

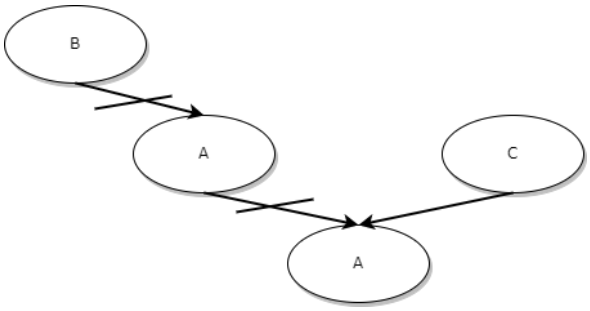

Figure 1.1: A directed graph representing arguments using the BAF, there are two attack relations (crossed lines) and one support relation.

(2005). The bipolar argumentation framework also abstracts away from the actual argument to only include the relation between the argument, but in comparison to the abstract argumentation framework, it allows for both attacking as well as supporting relations between the arguments, which allows for more complex relations. Similarly, to the abstract argumentation framework, a set of arguments with their relations can be displayed as a directed graph were the nodes represent the arguments and arrows represent the relationship between them. A connection with a line crossing indicates an attack relation, while a connection without the line indicates a support relation. An example of an argumentation graph using the bipolar argumentation framework can be seen in Figure 1.1.

To determine an admissible set in the bipolar argumentation framework a set must, similarly to an admissible set under the abstract argumentation framework, be conflict-free and defend itself against attacks from outside of the set. Though the addition of supporting relations complicates it a bit, as a support relation to an argument outside of the set which attacks an argument in the set means that the set is not conflict-free, as there is a supported attack on a member of the set. This definition of admissibility is one of three different admissibilities defined within the paper, this admissibility is the closest to the definition for an admissible set in the abstract argumentation framework and is the least stringent one.

1.2 Argumentation-based learning algorithm

This understanding of the bipolar argumentation framework allows us to go to a machine learning algorithm developed using this argumentation framework as a basis. In his PhD thesis Ayoobi et al. (2021) introduces two machine learning algorithms that learn by constructing an argumentation graph using the data that is provided to them, they are called Argumentation-based learning (ABL) and Accelerated Argumentation-based learning (AABL), which is an improvement on the first algorithm, as it is faster than the ABL algorithm and has higher accuracy.

AABL works by using a bipolar argumentation graph that represents both the output as well the input features and their relations with each other. The graph starts out as an empty graph without nodes or connections, then whenever a new output value is encountered, then a bidirectional attack relation between these output nodes is created in order to encourage exclusivity between the different output values. Other nodes representing input features are then used to support different output nodes based on the encountered data instances.

The prediction of the AABL algorithm works by first extracting all the feature value combinations of the input with a length of the current limit, which starts as 1. Then all the non-output nodes in the graph that match one of the combinations created using the input are considered and the supported output node is stored in a list. Then after all supporting nodes have been checked if exactly one output node was supported then the value of that node is the output of the model, if multiple output nodes match, then a random output node among the matching ones is used. If none match, then a random output value from the known output values is chosen.

To update the bipolar argumentation graph the correct output needs to be known, as well as the current bipolar argumentation graph and all combinations of feature values given the input of a size given the current limit. As already mentioned above whenever a new output value is encountered when updating the argumentation graph a bidirectional attack relation is added between all previously known output values and the new output value. After potentially updating the output val-

ues, the supporting input nodes are updated using the new data instance and the correct output value.

To achieve this all combinations of features and values of the input are checked on whether a supporting node exists that covers that combination, if such a node exists and the supported output of that node does not correspond to the correct output, then the supporting node is removed and the combination is marked as a non-unique combination and therefore irrelevant for the purpose of determining the output. If the combination was not already encountered, then a new supporting node is added which supports the correct output value. If all combinations were already encountered and were not unique, then the limit for the number of features in the combinations is increased by one. Then the new combinations of feature values given the input is computed and the new combinations are used to update the graph until the limit is increased to two or the update algorithm indicates that the limit should not be increased anymore.

1.3 Decision Trees

A decision tree is a machine learning algorithm that recursively divides the data points into subsets in order to determine the correct output (Loh, 2011). For this project, only classification decision trees will be considered, though regression decision trees also exist. As already shortly explained above classifying decision trees, from here on out called decision trees(DT), split the data based on learned rules in order to be able to classify the input. Then each of the subsets can be split further according to a newly learned rule if necessary, which allows the decision tree to build a tree that can be traversed to get to a classification label which should correspond to the given input.

1.4 Multi-layer feed-forward neural networks

A multi-layer feed-forward neural network is a machine learning algorithm that consists of artificial neurons which are divided into layers, with the first layer being the input layer and the last layer being the output layer(Svozil et al., 1997). All other layers in between are called hidden layers. The layers of a network are connected with each other with directed connections which allows for the transfer of values from one layer to the next, in a feed-forward neural network all connections go forward meaning that no connections are directed towards the same or a previous layer. Each connection is between two neurons of different layers and will transfer the output of one neuron as the input of the next neuron, though these connections are weighted which means that the input will be the weighted output of the previous node. Every neuron is connected to all neurons in the previous layer.

Neurons can have multiple inputs which are summed up and then the sum is used as an input for the activation function of the neuron, the activation function will then transform the summed input into the output value of the neuron (Svozil et al., 1997). These activation functions are non-linear, as this allows for a more flexible model compared to a model using linear activation functions.

The multi-layer feed-forward neural network will learn in a supervised manner, which means that it knows the correct output and can use that with the predicted output of the model as well as an optimization function to modify the weights in the network so that the model can learn to produce a better output.

2 Methods

2.1 Modified AABL Algorithm

In order to test the influence of dependencies on the accuracy of the Accelerated Argumentation Based Learning algorithm by Ayoobi et al. (2021) I made some modifications to the original algorithm, in order to allow it to function in a broader set of circumstances.

2.1.1 Problems with the AABL Algorithm Implementation

The original implementation of the AABL algorithm suffers from three big problems, namely:

1. High coupling of data generation and the algorithm.
2. Non-uniform representation of the data.
3. Hard-Limit of two dependencies.

The first issue is a problem as it significantly hinders the usability of the algorithm. This is due to the fact that each new dataset now needs to come with the same helper functions in order for the algorithm to work. This means that it is harder to use the algorithm for other problems.

The second issue further contributes to the problem by requiring that the data has to both have a string and a numeric representation, as the algorithm does not only use one representation but both, which makes it harder to adapt the algorithm for other problems.

Lastly, the third issue means that the algorithm can only be applied to data with low dependencies. This also reduces the applicability of the algorithm.

2.1.2 Changes

In order to fix the first problem of the high coupling of the AABL algorithm and the data generation, necessary helper functions from the data generation were moved to the algorithm. These functions transform the array into different forms in order for the algorithm to be able to find proper combinations. Some of these functions were removed when fixing the second problem. This reduced but not completely eliminated the high-coupling of the classes.

In order to completely eliminate the coupling between data generation and the learning algorithm the second problem about the non-uniform representation needs to be addressed. This is due to the fact that functions of the data generation class were used to associate the string representation of the data to a numeric representation. Therefore I decided to only use numbers to represent the feature values, as it is more in line with other machine learning algorithm implementations and therefore allows for easier usage with them.

This switch to a purely numeric representation for the modified algorithm means that there are some new restrictions on how the numerical representation works, as the original AABL algorithm used strings to differentiate between feature values as they were completely unique both within a single feature and also between features. This is not the case for the number representation of the original AABL is only unique within a feature. As the modified AABL algorithm does not use a string representation, but only a numeric representation of the data, the feature values need to have a unique numerical number not only within the feature but also between different features to be able to differentiate between them. To illustrate this assume that there are four features such that each feature has a binary value, so 0 or 1. Then this could be a potential input vector: [0,1,0,1]. The problem with that vector is that the numbers aren't unique and therefore such an input vector is illegal. A way to transform such an array such that it fulfils the requirements is to take the index position of the feature into account. This means that in the previous example, each feature is binary then we can take the index number of the feature and multiply it by two and finally add the feature value to obtain a unique number. This means that the array [0,1,0,1] would be transformed into the array [0,3,4,7], which makes the new array guaranteed to comply with the unique feature requirement.

The third problem in the original implementation is that it only considers a combination with a maximum size of two features, which means that it will at most only consider 2 dependencies in the data, which severely limits the usability of the code, as well as making this project trivial to answer, so this limit was removed which theoretically should allow for an unlimited amount of features being taken into account if the algorithm has unlimited time and memory. The fixing of the problem was aided by switching to a numerical approach, as this allowed for simpler looping over the features and therefore allowed the algorithm to more easily handle a variety of feature lengths. To facilitate more than two features the modified AABL algorithm needs to be initialized with an array which has the size of the number of features and in each position has the number of possible feature values at that location. For example, the array [3, 4, 2] indicates that the algorithm will receive scenarios with three different features, with the first feature having 3 possible values, the second feature having 4 possible features and the last feature having 2 possible values.

The modified code can be accessed on GitHub[‡].

[‡] `https://github.com/jojoscholzgmailcom/Accelerated_ABL`

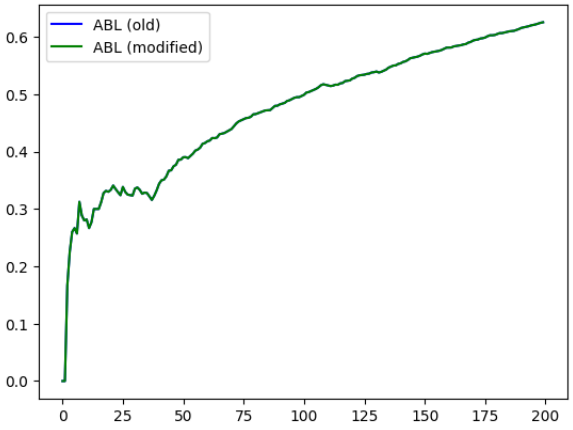

Figure 2.1: The blue line indicates the accuracy of the original algorithm, while the green line indicates the accuracy of the modified algorithm.

2.1.3 Testing

After these changes were added to the algorithm, the newly modified algorithm was tested in order to verify that the changes to the implementation did not modify the behaviour of the algorithm. In order to facilitate this test both the old implementation and the new implementation were given the same inputs, with the only adjustment being made to fit the input into the correct format for both algorithms, given that the learning algorithm does not rely on random chance both the modified and unmodified version should have the exact same accuracy the whole time. When tested this was the result, as can be seen in Figure 2.1, where the two lines overlap perfectly, further inspection of the underlying values confirmed the graph.

2.2 Data Generation

To generate data with a variety of dependencies a Bayesian network is used. A Bayesian network consists of nodes that are possibly connected using a directional connection which indicates a dependency. This means that if for example there is a connection from Node A to Node B then this means that the probability of Node B having a specific value is dependent on the value of Node A. This example shows clearly why a Bayesian network is an appropriate choice to generate data with a variety of different dependencies. The implementation of the Bayesian networks used for data generation can be found in the Github repository[§].

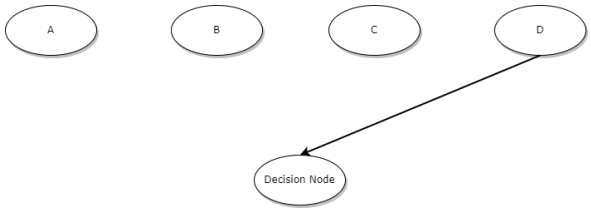

Figure 2.2: A Bayesian Network with one decision node and four other nodes one of which is a parent of the decision node.

2.2.1 General Structure

For the data generation of this project, there are some facets of the structure of the Bayesian Network that are constant. Namely, each Bayesian network will consist of two types of nodes first normal nodes and second decision nodes which are the output nodes and are as such hidden from the learning algorithms as the value of that node is what the learning algorithms will have to predict.

For the experiment, there are two different types of Bayesian Networks that will be used for the experiment. The first type of Bayesian network only increases the number of nodes in the network but keeps the number of dependencies to one, which means that only one node influences the value of another node, furthermore, the dependent node is the decision node an example of such a Bayesian Network can be seen in Figure 2.2.

The second type of Bayesian network has the number of dependencies which is the number of nodes minus 1 meaning that the decision node depends on all other nodes. An example of such a network can be seen in Figure 2.3.

2.2.2 Normal Node

In order to reduce the number of possible combinations that can be generated every node is a boolean node, which means that it can only be either 0 or 1. Upon the creation of a new node the parents of

[§] `https://github.com/jojoscholzgmailcom/Accelerated_ABL`

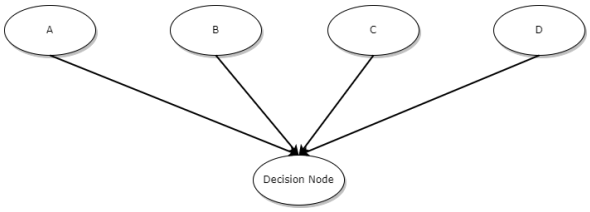

Figure 2.3: A Bayesian Network with one decision node and four parents.

that node need to be provided, which allows the determining of conditional probabilities for the value of this node. For this, every possible value combination of the parents is calculated. Then for each combination, a random number between 0 and 1 is chosen which indicates the conditional probability that this node will have a value of 1 given that combination of values of the parent nodes.

2.2.3 Decision Node

The decision node is in essence like any other node with the only exception being that depending on the combination of parent values there is either a 0% or 100% chance that the value of this node is 1, which means that this node behaves deterministically given the combination of values from parent nodes. This was done in order to allow for better comparability of different runs as this ensures that hypothetically the algorithm has 100% accuracy if it learns the correct rule.

The first approach of only having one value combination that gives this node a value of 1 does not work out due to the exponential nature of how many combinations arise when the number of parents increases. For example, with two parents there exist $2^2 = 4$ combinations which means that if the parents have a probability of 0.5 there is a one in four chance that the combination occurs, while for 20 parents there are $2^{20} = 1,048,576$ combinations which means that the is less than 1 in a million chance to sample a combination that results in a 1 for this node. This leads to very unbalanced datasets which reward the classification of everything as zero, because if 99% of the combinations result in a zero for the decision node then a model that always predicts zero will have an accuracy of 99%.

In order to prevent this, the solution is to make more combinations that result in the decision node having a value of one. It is important to make sure that when deciding on these combinations that they will not lead to them making one of the parents irrelevant by making the decision node independent from the value of that parent. This could happen if, for example, only combinations where the last parent has a value of one will make the decision node one. In order to circumvent this issue the parity of the sum of parents is used to determine the value of the decision if the sum of parents is even then the decision node will have a value of one.

2.2.4 Sampling

To generate data for the AABL algorithm the Bayesian network will first start randomly assigning a value to nodes without any parents according to the probability of each node. Then the values of these nodes can be used to assign a value to nodes that are dependent on these nodes. As the value of their parent is known and therefore the probabilities for that node can be easily resolved, a value for that node can be randomly selected. All these values together then make up one sample of the Bayesian network. As the values are assigned pseudo-randomly using Pythons random generator multiple sampling of the Bayesian Network can result in different samples.

2.3 Experimental Setup

In order to test the effect of dependencies on the accuracy of machine-learning approaches the two types of Bayesian networks will be used in order to sample data that contains a variety of dependencies. To thoroughly test the effect of dependencies in data on the AABL algorithm two different hyperparameters will be manipulated, namely the Bayesian network type and the number of nodes in the Bayesian network. There are two different types of Bayesian networks as described in Section 2.2.1. For the number of nodes the values 3 and every even value up to and including 22 were tested, which means that there are 11 different values for that parameter. This means that there are a total of 22 configurations of the two hyperparameters that will be tested.

For each of the configurations, a random Bayesian network will be generated based on the

configuration. From the Bayesian network then individual samples can be sampled as described in Section 2.2.4. A single run consists of 200 samples that were sampled one at a time and were used to train the algorithm. Each of these training runs was repeated 25 times each time with an untrained model to reduce the effect of chance of for example the sampling on the accuracy of the model.

To be able to determine the performance of the AABL algorithm compared to other machine-learning approaches the same experimental setup will be used to train the decision tree and the neural network. For both machine learning approaches the implementation of the Python library scikit-learn with version 1.2.2 was used.

2.3.1 Decision Tree

The decision tree used for this experiment uses Gini impurity in order to determine which split of the data is the best and then the rule that produced the best split is used to build the decision tree. This is repeated until only two samples are unaccounted for, after which the tree is completed. Contrary to the AABL algorithm which stays the same per run, a new decision tree is generated and trained on all available data after a new sample is created, this is done as a decision tree is not an online-incremental learning algorithm.

2.3.2 Neural network

The neural network used in this experiment is a multi-layer feed-forward neural network with three layers. As described for the decision tree a new neural network is trained at each timestep with the data from the previous timestep and then afterwards the trained model is used to predict the answer, which can then be used to determine the accuracy of the model as the number of timesteps and therefore data increase.

The first layer of the neural network is the input layer and therefore corresponds to the number of features that the scenarios have. The second layer is a hidden layer and has 8 nodes and finally, the third layer is the output layer and therefore has two nodes, one for each of the possible answer values. The update function for the neural network is Limited-memory BFGS, which was chosen due to its ability to converge faster and perform better with fewer data compared to other algorithms. As an activation function for the nodes in the hidden layer, the RELU function was chosen, as it is a non-linear function.

The neural network uses the Average Cross-Entropy Loss function as well as the L2 regularization with a factor of 0.0001 to calculate the loss of the neural network. The model was trained for 200 timesteps or until convergence whichever condition occurred sooner.

Lastly, similar to the decision tree a new neural network is trained after a new sample was generated using all generated data, as this kind of neural network is not an online-incremental learning algorithm.

3 Results

In Figure 3.1 it can be seen that the mean accuracy of the AABL algorithm decreases when the number of nodes in the Bayesian network that was used to generate the data increases, as this also increases the number of dependencies. The figure shows that at the 199th attempt with 3 nodes, the algorithm has an accuracy of 97.42%, while it only has an accuracy of 45.84% at the 199th attempt with 22 nodes. Furthermore, it can be seen that a higher number of dependencies leads to a less steep learning curve.

In Figure 3.2 it can be seen that the mean accuracy of the AABL algorithm does not decrease drastically with an increase in the number of nodes used in the Bayesian network, this is due to the fact that these Bayesian networks only have a single dependency. The figure, therefore, shows that the mean accuracy for the three nodes is 98.44%, while the mean accuracy for the 22 nodes is 97.82%.

In Figures 3.3 to 3.6 the mean accuracy of the AABL algorithm is compared to the mean accuracies of the neural network (NN) and the decision tree (DT) algorithm for a variety of dependencies in the data. These figures were selected from the 11 different figures, as they were the most interesting the other figures can be found in the Github repository[¶].

In Figure 3.3, the mean accuracies of the algorithms are shown for two dependencies. In this Fig-

[¶] `https://github.com/jojoscholzgmailcom/Accelerated_ABL`

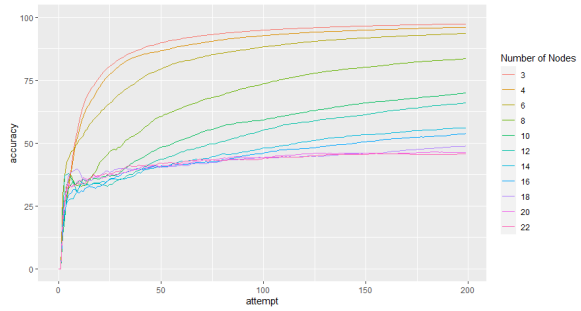

Figure 3.1: The mean accuracy of the AABL algorithm when the data has dependencies equal to the number of nodes minus 1.

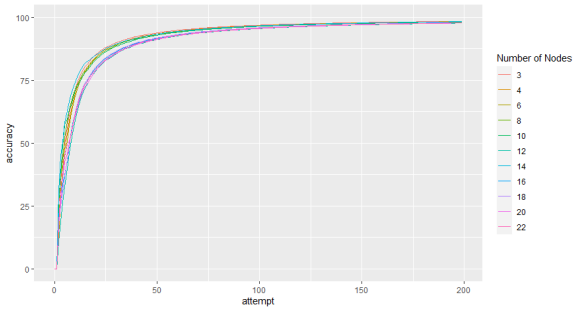

Figure 3.2: The mean accuracy of the AABL algorithm when the data has only one dependency.

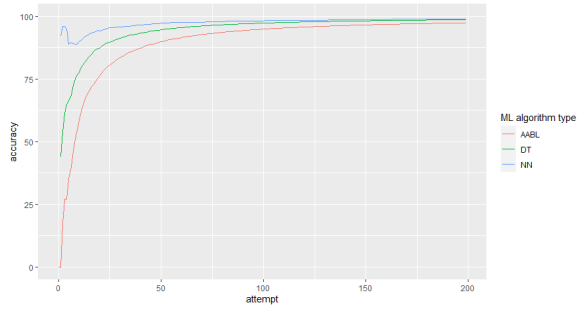

Figure 3.3: The mean accuracy of the AABL algorithm, neural network and decision tree when the data has 2 dependencies.

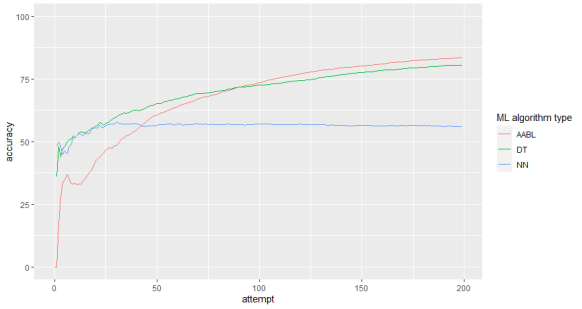

Figure 3.4: The mean accuracy of the AABL algorithm, neural network and decision tree when the data has 7 dependencies.

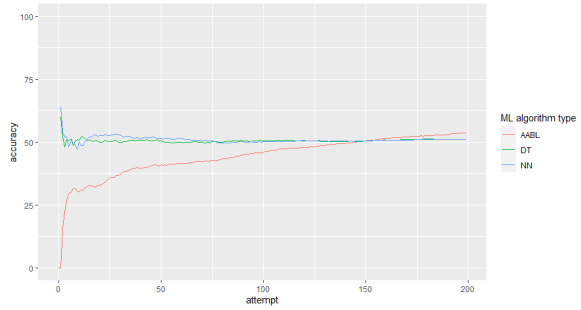

Figure 3.5: The mean accuracy of the AABL algorithm, neural network and decision tree when the data has 15 dependencies.

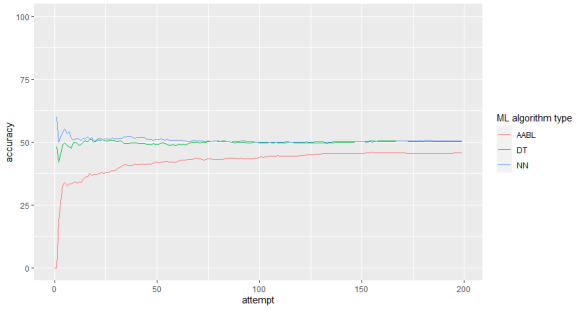

Figure 3.6: The mean accuracy of the AABL algorithm, neural network and decision tree when the data has 21 dependencies.

ure, it can be seen that all algorithms converge to a high accuracy of over 95%. Furthermore, it can be seen that both the decision tree and AABL algorithm have a similar learning curve, with initially a low accuracy and then a major improvement before slowly approaching 100% accuracy. The neural network mean accuracy meanwhile starts high at around 90% accuracy and then first drops a bit down to around 85% before recovering and then slowly approaching the 100% as more data instances become available.

In Figure 3.4, the mean accuracies of the algorithms are shown for seven dependencies. In this Figure, it can be seen that the AABL which has

initially the lowest accuracy of the algorithms outperforms the other two algorithms after 200 data samples. The mean accuracy for both the DT and AABL algorithm has not levelled of after 200 data samples. The neural network has levelled of at a mean accuracy of below 60%.

In Figure 3.5, the mean accuracies of the algorithms are shown for 15 dependencies. In this Figure, it can be seen that the AABL which has initially the lowest accuracy of the algorithms outperforms the other two algorithms after 200 data samples, furthermore, it can be seen that the accuracy of the AABL has not levelled off after 200 samples. Both the decision tree and the neural network have stabilized at around 50%.

In Figure 3.6, the mean accuracies of the algorithms are shown for 21 dependencies. In this Figure, it can be seen that similar to Figure 3.5 both the DT and the NN algorithm stabilize at an accuracy of around 50%, while the AABL algorithm seems to stabilize at an accuracy of around 45%.

Lastly, in figure 3.7 the mean runtime of all three algorithms in seconds given the number of dependencies in the data can be seen. It can be seen that both the neural network, as well as the decision tree, have a relatively stable runtime as the number of dependencies in the data increases. The runtime of the AABL algorithm on the other hand does increase significantly when the number of dependencies in the data increases.

4 Discussion

In conclusion, it can be seen in Figure 3.1 that the number of dependencies in the data does have an effect on the accuracy of the AABL algorithm, as the mean accuracy of the algorithm decreases as the number of dependencies in the number increases, furthermore this effect does not appear in Figure 3.2, as that one only increases the number of nodes without increasing the number of dependencies. This seems to indicate that the effect on the accuracy is due to the increased number of dependencies in the data and not due to the increased number of features.

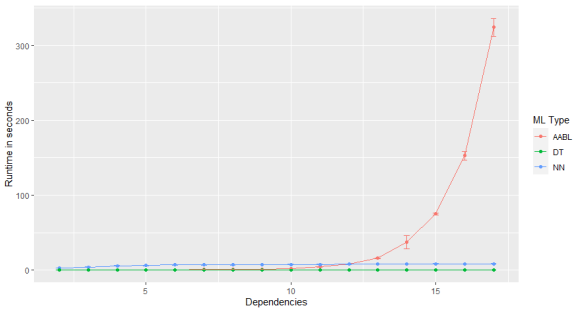

Figure 3.7: The mean runtime of the AABL algorithm given the number of dependencies.

Furthermore, it can be seen in Figures 3.4 and 3.5 that the AABL algorithm while initially having a lower accuracy seems to be able to increase faster than the other two machine learning algorithms, which could indicate that the AABL algorithm might outperform the other two approaches given more data also for higher dependencies. Though this is not completely in line with what can be seen in Figure 3.6, which seems to indicate that the AABL algorithm levels off at around 45%, though it is possible that the learning rate is relatively small and given more the data that the mean accuracy for the AABL algorithm will still increase such that it outperforms the other two approaches.

In Figure 3.1 it can be further seen that especially for the lines representing data generated by a Bayesian network with 6 to 18 nodes they are still having a positive growth rate, which could indicate that the mean accuracy of the AABL algorithm for these Bayesian networks might still drastically increase given more samples. This, in turn, could suggest that the AABL algorithm is able to perfectly track statistics produced by a Bayesian Network given enough data samples. Though this does not seem likely given that the algorithm is pruning all non-unique supporting nodes, which will mean that for example if a decision node in a Bayesian Network given a specific parent node value combination will have a 70% chance of having a value of 1, then if the same parent combination is sampled multiple times it is possible that the decision node has different values. This would mean that the parent combination supports multiple values, which would lead to the incorrect pruning of this combination from the argumentation graph. This then shows one of the limitations of the AABL algorithm, as the algorithm is likely only able to learn deterministic dependencies, where the same parent value combination will always result in the same value and will perform worse for dependencies

where the same parent value combination can lead to different values.

Another limitation of the algorithm can be seen in Figure 3.7, which shows the time it takes for the algorithms to run for 200 data samples given the number of actual dependencies in the data. As it can be seen there is a major increase in the runtime with higher dependencies for the AABL algorithm, but not for the other two algorithms. This shows that the AABL algorithm does not scale that well with the number of dependencies when the considered dependencies are not capped as they were in the original AABL algorithm. This is likely due to the fact that with an increase in the length of the feature value combinations, the number of possible combinations increases exponentially, which then contributes to the exponential runtime increase.

4.1 Conclusion

All in all, it can be said that the initial research question on whether the number of dependencies in the data has an effect on the accuracy and learning rate of the AABL algorithm can be answered, as there was a decrease in the learning rate as well as the mean accuracy of the AABL algorithm, furthermore a few other limitations of the algorithm were outlined such as the poor time scaling. This then suggests that while this is a good step towards an explainable machine learning algorithm, there are still issues with this algorithm that prevent it from being able to be used in more varied environments, as it is limited to deterministic relations. Though this algorithm could be a possible starting point from which other argumentation-based learning algorithms could be developed, those could then potentially take into account probabilistic relations, which could vastly improve the applicability of such an algorithm, furthermore, revisions of the code and potential parallelization of the code could potentially reduce the time needed to run the algorithm and might improve the time scaling of the algorithm. This or maybe other argumentation-based learning algorithms might then possibly lead to more explainable AI in a variety of fields. This could be especially useful for end-user that in that scenario interact with AI which gives proper reasoning for its decision, which could reduce the chances that people would accidentally misuse that AI, as they would understand it better.

References

Adadi, A., & Berrada, M. (2018). Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access*, *6*, 52138–52160. doi: 10.1109/ACCESS.2018.2870052

Ayoobi, H., Cao, M., Verbrugge, R., & Verheij, B. (2021). Argue to Learn: Accelerated Argumentation-Based Learning. In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 1118–1123). doi: 10.1109/ICMLA52953.2021.00183

Baroni, P., Toni, F., & Verheij, B. (2020). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games: 25 years later. *Argument & Computation*, *11*(1-2), 1–14. doi: 10.3233/AAC-200901

Cayrol, C., & Lagasquie-Schiex, M. C. (2005). On the Acceptability of Arguments in Bipolar Argumentation Frameworks. In L. Godo (Ed.), *Symbolic and Quantitative Approaches to Reasoning with Uncertainty* (pp. 378–389). Berlin, Heidelberg: Springer. doi: 10.1007/11518655_33

Dung, P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, *77*(2), 321-357. doi: 10.1016/0004-3702(94)00041-X

Loh, W.-Y. (2011). Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *1*(1), 14–23. doi: 10.1002/widm.8

Svozil, D., Kvasnicka, V., & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, *39*(1), 43–62. doi: 10.1016/S0169-7439(97)00061-0