Maastricht University

Research project

Project report

Explainable AI: Learning Arguments

Appendix A

Extended Related Work

Authors
Jonas Bei
David Pomerenke
Lukas Schreiner
Sepideh Sharbaf

Supervisors
Nico Roos
Pieter Collins

June 23, 2021

Contents

0.1 Proof with and without probability

Here we locate the approach presented in Verheij (2017) within the research field and the related literature.

In Verheij (2017), argumentation is applied in the context of forensic science. There, the training data consist of scenarios of abstract propositions. Without changing the essence of the approach too much, we can reinterpret this framing and conceive of the training data as a boolean representation.

Verheij defines three kinds of arguments, of which *(presumptively) valid arguments* are most interesting. They are probabilistically valid (or alternatively, valid based on the preference relation between cases). In this, they resemble association rules.

The primary target audience in Verheij (2017) is rather forensic science than machine learning. The conceived user story for the model may be a group of criminal investigators who hypothesize about what happened during a crime (see the case study in section 5 of that paper). In the course of their investigations, they draw up case models at their board, and revise them as new evidence arrives.

The more general target of the approach that is presented in Verheij (2017), however, is its application to machine learning: "[I]t is not a big step to view the cases in a case model as the data that can be used for learninglike in data systems—and the rules that are valid in it as the knowledge structure of knowledge systems." (Verheij, 2018, p. 66)

Verheij (2017) has been compared with another hybrid qualitative-quantitative reasoning model based on Bayesian networks in van Leeuwen and Verheij (2019). The model of Verheij (2017) has been tested in the paper itself for investigating the murder in the Alfred Hitchcock movie *To catch a thief*, with hand-made examples. Subsequently, it has been applied to a real criminal case, the *Simonshaven case* Verheij (2020). The latter analysis was undertaken for a conference[1], as part of which multiple authors have applied their formal models to investigating the Simonshaven case. The competing approaches presented at the conference are argumentation-based analysis, Bayesian networks, Bayesian thinking, scenarios, and a hybrid approach using stories and arguments. Verheij (2020) has been reviewed in Zenker (2020).

conclusive	penguin	→ ¬flies
	chicken, scared	→ flies
presumptively valid	bird	→ flies
	chicken	→ ¬flies
	scared	→ ¬flies

Figure 1: Some fictitious conclusive and (presumptively) valid arguments for the argumentation approach of Verheij (2017).

0.2 Reasoning with rules and arguments

The formal study of argumentation is only relatively recent. Historically, classical logic has often been used instead. Classical logic is monotonic, meaning that additional information cannot revise conclusions that would have been made otherwise. The computational implementation and application of classical logic is recently studied especially using *miniKanren*[2], a logic programming language embedded into other programming languages.

[1] `https://onlinelibrary.wiley.com/toc/17568765/2020/12/4`
[2] `http://minikanren.org/#implementations`

Because the satisfiability of sentences (and, hence, also entailment between them) are neither generally decidable nor efficiently computable for classical logic, logic has often been restricted to *Horn clauses* in artificial intelligence, which fare better in these respects (Russell & Norvig, 2010, ch. 7.5.3). Horn clauses are disjunctions of literals (that is, variables or predicate expressions representing atomic facts), where exactly one literal is positive. More intuitively, they are a rule from a conjunction of positive literals to another positive literal. The *Prolog* family of programming languages enables efficient reasoning with Horn clauses of predicates. [3] By means of recursive predicates, it also allows for programming. Prolog has been extended to allow for negations by a mechanism known as *negation as failure*. The ProbLog programming language extends Prolog with probability theory.[4]

Argumentation has been formalized on an abstract level by Dung (1995): An argument is the connection between some premises (the *support* of the argument) and a conclusion. Arguments can attack each other. An argument is defeated either if is *rebutted* by another argument, or if one of its premises is rebutted by another argument, in which case the argument itself is *undercut*. Abstract argumentation has been implemented in *ASPIC+* (Modgil & Prakken, 2014).

Argumentation in the context of logic has been formalized, for example, by Roos (2000). A *defeasible theory* consists of *facts*, *strict rules*, and *defeasible rules*. The defeasible rules are partially ordered by a *preference relation*, such that in the case of conflicting rules, the conflict can be resolved by ignoring the less preferred rule.

The preference relation can be derived using the criterion of *specificity* between rules (Roos, 2000, def. 4). An argument is more specific than a second argument if from the premises of the argument an argument can be made to the premises of the second argument. An argument is strictly more specific than a second argument if it is more sepcific than the second argument and the second is not more specific than the first argument. When using specificity to derive the preference relation of a defeasible theory, more specific rules will be preferred to less specific rules.

Reasoner implementations for defeasible argumentation (in various flavours) include *SPINdle*[5] in Java, *DePYsible*[6] in Python, *Oscar*[7] in Common Lisp. A variety of further argumentation libraries is maintained by the *TweetyProject*[8]. Recently, a semantic tableau reasoner has been proposed (Roos, 2020) and implemented (Barbara Futyma, Eric Nakoja, David Pomerenke, David Schimmel, & JingYang Zeng, 2021); the advantage of the semantic tableau proof method is that it is comparatively easy for humans to understand (as compared to resolution or natural deduction).

0.3 Learning arguments

In order to participate in argumentation, it is necessary to know a set of rules. Then, facts and rules can be combined to arguments, and the conclusions from the arguments can be used to construct more arguments.

Many different approaches in data mining and machine learning are concerned with the learning of rules. Here, we give a broad overview over techniques for learning rules. We do not only cover rules in the narrower sense as used within defeasible argumentation, but in the broader sense as used within explainable artificial intelligence. This is because many insights about efficient mining techniques, as

[3] For example, `https://www.swi-prolog.org/`, `http://www.gprolog.org/`, `https://ciao-lang.org/`
[4] `https://dtai.cs.kuleuven.be/problog/`
[5] `https://github.com/NICTA/SPINdle`
[6] `https://github.com/stefano-bragaglia/DePYsible`
[7] `https://web.archive.org/web/20170806211112/http://johnpollock.us/ftp/OSCAR-web-page/oscar.html`
[8] `http://tweetyproject.org/lib/index.html`

well as about the limitations of rule learning, can be transferred from this wider set of approaches to the approach here investigated.

De Raedt (2008) introduces a hierarchy of machine learning representations, from simple to complicated:

1. *Boolean* (or *binary*) representation, that is, each data point is a set of propositions.
2. *Attribute-value* representation, typical for most standard machine learning algorithms.
3. *Multi-instance* representation, where multiple rows can belong to one data point.
4. *Relational* representation, where multiple interconnected tables are used.
5. *Logic program* representation, where functors and structured terms turn the data into a turing-complete programming language such as Prolog.

This hierarchy will be useful in seeing the different (original) application areas of the presented machine learning techniques. It should also be noted that it is systematically possible to transfer a technique to work on higher-level or lower-level representations. De Raedt (2008, ch. 6) have demonstrated this by upgrading a propositional rule learner, an attribute-value decision tree learner, and a propositional frequent item set miner, all to work on relational data.

The learning of rules can be considered either *unsupervised learning*, when the rules are considered the product; or it can be considered *supervised learning*, when the rules are considered to be a tool for conducting further analysis, as in *rule-based classification*. In the latter case, the task *can* be restricted to learning only rules for directly predicting the relevant attributes.

We broadly cluster the existing approaches into the following categories. The boundaries between these categories are very fluid.

1. *Argumentation and (defeasible) logic rules.* This is most close to the approach investigated in the project.
2. *Classification rules.* These are used for rule-based classification, and do not aim to be understandable, but merely to be a tool for classification.
3. *Probabilistic and causal models.* They allow for quantitative reasoning.

0.3.1 Argumentation, logic, and association in machine learning

Kakas and Michael (2020) differentiate between multiple use cases of argumentation in machine learning: Inputs extended with arguments (most prominently, *argument-based machine learning, ABML*); inputs interpreted as arguments; hypotheses interpreted as arguments; and hypotheses expressed in argumentation. The last use case, where argumentation is the *target language* for learning (Kakas & Michael, 2020, p. 17), is the one that is relevant for this project. Kakas and Michael (2020) further distinguish two paradigms within this use case:

- In the first paradigm, arguments are monolithic rules that directly map input facts to output facts. This paradigm comprises decision lists, exception lists, exceptions in inductive logic programming, and random forest methods.
- In the second paradigm, arguments may consist of multiple chained smaller rules that are chained together, with intermediate concepts connecting the rules. Examples are the *never-ending rule discovery (NERD)* algorithm, *machine coaching*, and *simultaneous learning and prediction (SLAP)*.

0.3.1.1 Learning defeasible rules

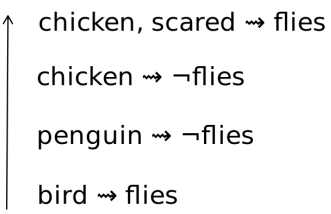

Figure 2: A fictitious defeasible theory from aviational zoology. The arrow represents the preference relation. Note that the preference relation is total here, but it can also be partial.

Two algorithms are explicitly concerned with learning defeasible rules. The first, *DefGen* uses as a basis the results of an association rule miner (see subsubsection 0.3.1.3), and then applies relevance criteria to the learned rules Governatori and Stranieri (2001).

Learning from the drawbacks of *DefGen*, *HeRo* has been developed to mine defeasible rules independently of an association rule miner Johnston and Governatori (2003). *HeRo* takes inspiration from decision tree learners, with its incremental algorithm that uses *information gain* as a core criterion (see subsubsection 0.3.2.1). The structure of the algorithm resembles decision list algorithms and covering rule algorithms (see subsubsection 0.3.2.3).

0.3.1.2 Learning logical rules and logic programs

bird, normalbird → flies

chicken, normalchicken → ¬normalbird

penguin → ¬normalbird

scared → ¬normalchicken

Figure 3: A fictitious set of monotonic rules. Here, a normality predicate is used in order to capture what would otherwise be captured by using defeasible rules. The approach of using normality predicates is found, among other places, in some examples in de Raedt 2008. It has been adapted here to a normality proposition for each originally defeasible rule.

Relational learning is similar to the learning of defeasible rules, but it typically produces monotonic rules (that is, in this context, rules which are always valid regardless of any other rules). Like association rule learning, relational learning is concerned with the discovery of patterns in the training data (De Raedt, 2008). The difference is that relational learning works on relational representations, and, in part, on logic program representations. The technique is generally applied to predicate logic. It can be downgraded to propositional logic, which would make it applicable to the task of this project. A particularly interesting discovery in relational learning is that it is possible for algorithms to create new predicates that are not present in the training data (Russell & Norvig, 2010, p. 797f.). Potentially, such predicate invention could lead to smaller, more local rules. The invention of new predicates is also considered a crucial skill for the induction of new scientific theories (Russell & Norvig, 2010, p. 797f.).

Particular research effort within relational learning is going into *inductive logic programming*, where Prolog programs are learned from examples. Such methods can successfully learn programs like fizz-buzz from very few examples. A notable example of an inductive logic program learner is the *first-order inductive learner* (*FOIL*; Quinlan (1990)), which can be viewed as an extension of covering rule algorithms

(see subsubsection 0.3.2.3) to first-order logic (Han, Pei, & Kamber, 2011, p. 362f.). Further algorithms include the *model inference system (MIS)*, and *Progol*, which improves upon FOIL and MIS.

Notably, there are inductive logic programming algorithms for learning nonmonotonic rules, for example, *XHAIL*[9], and *top-directed abdutcive learning* (*TAL*; see Corapi, Russo, and Lupu (2010)). Dimopoulos and Kakas (1995) present a theoretical framework for nonmonotonic inductive logic learning.

Kakas and Michael (2020) stress the importance of combining inductive with abductive approaches within relational learning. Such algorithms use an inductive-abductive cycle. That is, they induce rules, then create new eamples based on these rules, which in turn make it possible to induce new rules.

0.3.1.3 Learning association rules and exception rules

bird → flies
(support 100%, confidence 80%)

scared → flies
(support 5%, confidence 20%)

chicken, scared → flies
(support 1%, confidence 100%)

Figure 4: A fictitious set of association rules with important measures, from a fictitious bird database.

Association rule mining is a classical data mining task. It works on boolean representations. Its paradigmatic application is shopping basket analysis, where customers of an online shop receive recommendations based on what other customers with a similar basket have tended to buy in the past. Association rule mining is efficiently possible with the *Apriori algorithm* (Agrawal & Srikant, 1994). (And subsequently to the Apriori algorithm, many versions with additional optimizations have been developed.) The algorithm makes use of two observations about the search space:

- First, association rules with n premises can be found by mining frequent sets of $n+1$ items, then for each item in the set creating a rule with that item as a consequence of the other items. So, for rules with a single premise, we could mine frequent sets with 2 members, and when we find, for example, the set a, b with frequency p, we can create one rule $a \rightarrow b$, and one rule $b \rightarrow a$. (The *confidence* measure will need to be calculated again for all of the resulting rules; but not the *support* measure, which is handy; see Tan, Steinbach, and Kumar (2014, p. 350).

- Second, we do not need to search for frequent sets, when a subset of them is already below a threshold such that we consider it infrequent. This is due to the Apriori principle: "If an itemset is frequent, then all of its subsets must also be frequent." (Tan et al., 2014, p. 333)

  Figure 5 shows that this technique allows for pruning of the search space with dramatic effects. (De Raedt (2008) transfer this pruning technique to relational learning in general.)

Association rules typically come with two main metrics: The *support* (or *coverage*) indicates the frequency of the premises of the rule. The *confidence* (or *accuracy*) indicates for which share of these data the conclusion of the rule does apply. Additional measures are used to establish whether a rule is interesting and should be kept: The *lift* is the most important of these measures. It measures how

[9] `https://github.com/stefano-bragaglia/XHAIL`

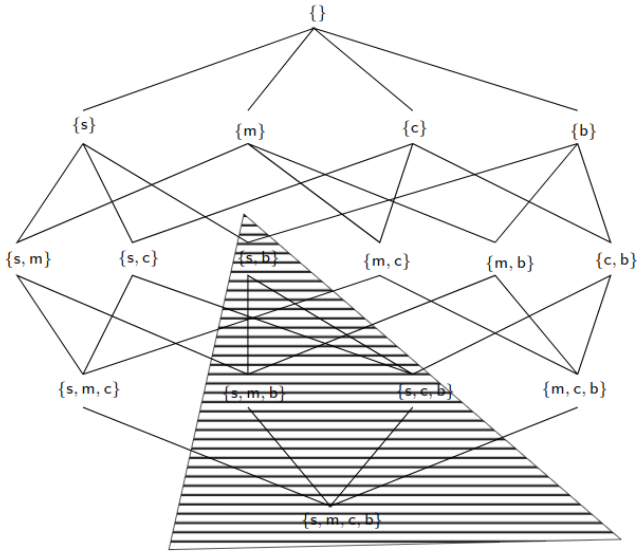

Figure 5: *Pruning specializations.* From De Raedt (2008, p. 52). All 2^n subsets of $\{s, m, c, b\}$ are systematically searched, starting from the most general set at the top. Knowing that the set $\{s, b\}$ is infrequent allows us to prune all its specializations, which is a lot.

much more likely the premises and conclusion are to occur together, in comparison to the case that the premises and the conclusion were conditionally independent.

$$\text{lift}(A \rightarrow B) = \frac{\text{support}(A \wedge B)}{\text{support}(A) \cdot \text{support}(B)}$$

Tan et al. (2014, ch. 6) describe the process of mining frequent item sets and association rules with various optimizations in detail. They also give a broader overview over evaluation measures for association rules. (Tan et al., 2014, ch. 7) considers advanced techniques, for dealing with categorical and continuous data.

The topic of *exception rules* has been investigated within the context of association rule mining (see, for example, Taniar, Rahayu, Lee, and Daly (2008)). Exception rules are typically ordered in a hierarchy of rules, exceptions, exceptions to the exceptions, and so forth. This hierarchical representation is similar to a representation using a preference relation on rules, where a rule receives a lower preference than the exceptions to it (Witten, 2017, p. 81f.). As far as we see, it seems that the research fields of *exception rules within association rule mining* and of *defeasible rules within the context of logic and argumentation* are not much mutually aware of each other.

0.3.2 Learning classification rules

Classification rules aim exclusively at supervised learning, where they are used as a mere tool; the rules are optimized for accuracy rather than for explainability. (Within the domain of supervised learning, they are still more explainable than most alternative approaches.) Classification rules are interesting mainly because of the pruning techniques developed for them, and for two special cases of them, regression trees and decision lists. These three points may be transferable to the problems faced in this project.

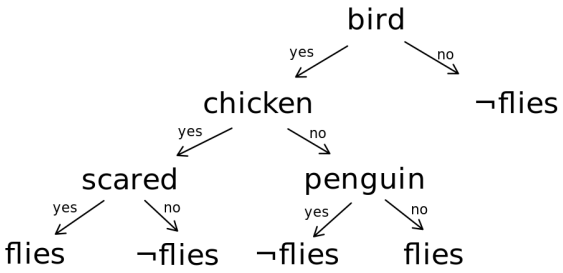

Figure 6: A fictitious decision tree.

0.3.2.1 Decision trees

Decision trees are one of the most popular methods in machine learning, and –within the domain of supervised learning– especially suited for explainability, since they can easily be executed by a human (Russell & Norvig, 2010, ch. 18.3). They work on attribute-value representations with binary as well as (by simple extensions) with categorical and continuous data.

However, Breidenbach (2021) notes that decision trees go the wrong way for explainability: Decision trees start with the assumptions and end with the conclusions. The same conclusion may be derived at the end of many different branches. Human reasoning, in contrast, starts with the conclusion and ends with all the different nested assumptions (that is, it uses argumentation).

Decision trees are mined by iteratively applying the criterion of information gain. The criterion of information gain tells which attribute should be used next (see Russell and Norvig (2010, ch. 18.3); Witten (2017, ch. 4.3)). Decision trees can be pruned, which improves accuracy, because it avoids overfitting, and in turn also makes the rules simpler.

A decision tree can be regarded as a set of *classification rules* that can be used together (Witten (2017, ch. 3.4); Han et al. (2011, p. 358)). Each classification rule involves all the criteria along a branch from the root node to the leaf node with the decision. This has the disadvantage that the rules may be very long, and the advantages that each rule is always very accurate, and that there are no conflicts between the rules.

0.3.2.2 Regression trees

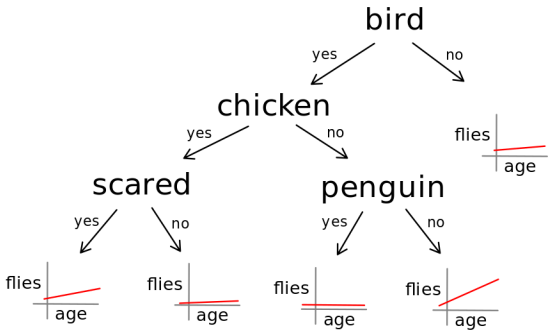

Figure 7: A fictitious regression tree. Unlike in this example, where the regression models in the leafs node are about an additional attribute, the regression models may as well be about continuous attributes that are already involved (in discretized form) in the decision structure of the tree.

Regression trees extend decision trees to work on continuous data without discretizing the predicted attribute (Witten, 2017, p. 72ff.)).

0.3.2.3 Decision lists

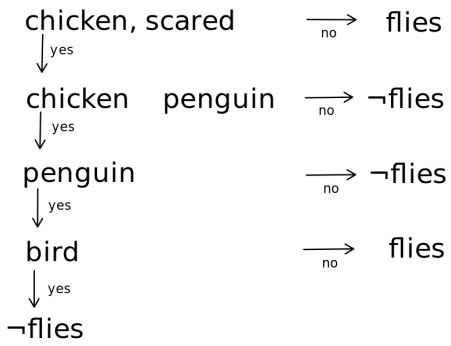

Figure 8: A fictitious decision list.

A decision list is a special kind of decision tree, where all questions are arranged a single branch (Russell & Norvig, 2010, ch. 18.5.1). It can be mined similarly to normal decision trees on attribute-value data, and has a similar efficiency. Similar to decision list miners are *covering rule* algorithms. They create the same output as a decision list, but use the criterion of *coverage* instead of the criterion of information gain. Covering rule algorithms are also used explicitly for mining defeasible rules, especially the *HeRo* algorithm (see subsubsection 0.3.1.1).

A decision list is very similar to a list of rules with with a preference, and thus to a defeasible theory.

0.3.3 Learning probabilistic and causal models

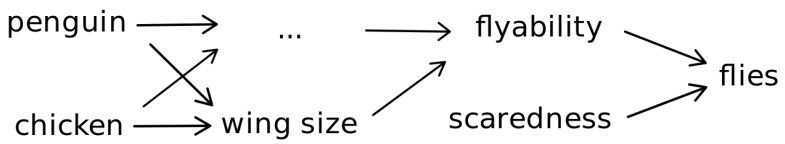

Figure 9: A fictitious causal probability network. To construct a proper causal network, variables that are hidden in the figures of the other approaches had to be introduced. The probability of each variable is a function (usually, a weighted sum or a logic table) of the probability of its parents; the functions are not further specified in this figure.

Probability networks (Bayesian networks) are the most common tool for reasoning with uncertainty (Russell & Norvig, 2010, ch. 14.2). They are used for attribute-value representations. In their general version, they are unsuitable for explainable artificial intelligence for three reasons, all of which have been addressed.

1. Probability networks contain numbers, and a lot of them. This is resolved by ignoring the numbers and instead just using positive and negative associations between variables. This is done in *qualitative Bayesian networks*. Another approach is to replace probabilities with qualitative ranks, as is done in ranking theory. Ranking theory[10](Spohn, 1990, 2012) is a qualitative alternative to probability theory. It uses a preference relation on beliefs, and allows for basic reasoning. It has been applied to probability networks and especially to causal networks.

2. Probability networks may contain too many connections. A network will achieve the highest accuracy if it is fully connected. This makes reasoning hard both for the computer (addressable by

[10] `https://github.com/tjitze/RankPL`, `https://github.com/tjitze/ranked-programming`

sampling techniques) and, more importantly for us, for the human who tries to understand. This issue is resolved by training techniques for *sparse Bayesian networks*. There, a regularization term penalizes the creation of connections, leading to networks with only a few connections. Reasonable networks with few connections are possible if the attributes of the training data describe a *locally structured system* (Russell & Norvig, 2010, p. 516).

3. Probability networks — as other rule-based techniques — are prone to causal misinterpretation. (That they are not causal is underlined by the fact that for the same data completely different valid networks can be constructed, depending on the order in which the nodes are processed during the construction of the network; see Russell and Norvig (2010, fig. 14.3)) This is resolved by using *causal networks*, which actually allow for causal reasoning (Pearl, Glymour, & Jewell, 2016). The construction of causal networks requires knowledge about the causal connections between the variables. This can be done in three ways:

    - First, if the causal connections in the domain are fully understood, then humans can determine between which variables there are connections, and they can make sure that all confounding variables are included in the model.
    - Second, if the causal theory of the domain is not fully understood, interventions can be undertaken and modelled using do-calculus to find out more about it. Pearl et al. (2016) emphasizes that such interventions are more flexible than randomized controlled trials.
    - A third approach tries to learn causal connections from the observational training data alone; see, for example, Wallace, Korb, and Dai (1996).

The links in causal networks can be regarded as causal rules. When we want to determine which action we want to take, these rules are the only kind of rules that can help us. All the other approaches that are presented here only take into consideration statistical associations, which cannot in general be used to reason about the consequences of interventions.

0.4 Propositionalization

As mentioned in section 0.1, we will try to transfer the approach from Verheij (2017) to work on propositional data. This requires that continuous attributes be converted to boolean (or, propositional values). On a theoretical level, this idea is explored in De Raedt (2008). Here we present practical approaches for propositionalization. Propositionalization of multiple attributes at once is known as *clustering*, and may also be relevant for our approach.

0.4.1 Equal-Width Partitioning

This is the simplest clustering approach considered. Here, the range of the variable is divided into k bins, each of size $\frac{Max-Min}{k}$ (where Max and Min refer to the maximum and minimum value of the variable, respectively).

This approach is easy to implement and not computationally expensive. However, there are some notable flaws: This algorithm assumes that each cluster has the same diameter, and it is prone to outliers.

0.4.2 Equal-Depth Partitioning

Another simple algorithm is known as equal-depth partitioning. Here, the clusters are built in such a way that each of them hold the same number of instances.

This could prove to be problematic as not all clusters necessarily have this property.

0.4.3 K-Means

K-Means(Lloyd, 1982) is based on the idea of centroids, which are points in the centre of the cluster. Here, k centroids are initialized randomly, and the instances are assigned to the cluster whose centroid is closest. Then, the centroids are moved to the mean of the cluster, and the instances are assigned to their new cluster. This pattern is repeated until the algorithm converges.

One flaw of this algorithm is that it may be sensitive to outliers. To make it more robust to noise, the centroids can take on the median value. Alternatively, the instance closest to the average of the cluster can be referred to as the centroid of the cluster.

Another flaw of this algorithm is that it is highly dependant on the random initialization. Therefore, this algorithm must run multiple times to ensure the best clusters are found. (To evaluate the clusters, the 'silhouette score' explained later in this section can be used.)

Lastly, the algorithm is built on the assumption that every cluster has a similar diameter, which may not be true for some datasets.

0.4.4 DBSCAN

This algorithm considers clusters to be regions of high density. To find the clusters, the algorithm counts the number of instances within a distance defined by ϵ, also called the instance's ϵ-neighbourhood. If this number of neighbours of an instance surpasses a predefined threshold, the instance is considered to be a *core instance*, an instance within a dense region. The neighbours of this core instance are considered to be in the same cluster, where some neighbours may also be core instances themselves. Therefore, a cluster consists of a multitude of core instances.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB* (Vol. 1215, pp. 487–499). Citeseer.

Barbara Futyma, Eric Nakoja, David Pomerenke, David Schimmel, & JingYang Zeng. (2021). *Explainable AI: From Arguments to Decisions* (Tech. Rep.). Maastricht: Maastricht University.

Breidenbach, S. (2021, February). *Von Text zu Code.* LegalOS.

Corapi, D., Russo, A., & Lupu, E. (2010). Inductive logic programming as abductive search. In *Technical communications of the 26th international conference on logic programming.* Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

De Raedt, L. (2008). *Logical and relational learning.* Springer Science & Business Media.

Dimopoulos, Y., & Kakas, A. (1995). Learning non-monotonic logic programs: Learning exceptions. In J. G. Carbonell et al. (Eds.), *Machine Learning: ECML-95* (Vol. 912, pp. 122–137). Berlin, Heidelberg: Springer Berlin Heidelberg. doi: 10.1007/3-540-59286-5_53

Dung, P. M. (1995, September). 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

Governatori, G., & Stranieri, A. (2001). Towards the application of association rules for defeasible rules discovery. *Jurix 2001*, 63–75.

Han, J., Pei, J., & Kamber, M. (2011). *Data Mining: Concepts and Techniques, 3rd Edition.* Morgan Kaufmann.

Johnston, B., & Governatori, G. (2003). An algorithm for the induction of defeasible logic theories from databases. In *Proceedings of the 14th Australasian database conference-Volume 17* (pp. 75–83).

Kakas, A., & Michael, L. (2020). Abduction and Argumentation for Explainable Machine Learning: A Position Survey. *arXiv preprint arXiv:2010.12896*.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, *28*(2), 129–137.

Modgil, S., & Prakken, H. (2014, January). The ASPIC + framework for structured argumentation: A tutorial. *Argument & Computation*, *5*(1), 31–62. doi: 10.1080/19462166.2013.869766

Pearl, J., Glymour, M., & Jewell, N. P. (2016). *Causal inference in statistics: A primer.* John Wiley & Sons.

Quinlan, J. R. (1990, August). Learning logical definitions from relations. *Machine Learning*, *5*(3), 239–266. doi: 10.1007/BF00117105

Roos, N. (2000). On Resolving Conflicts Between Arguments. *Computational Intelligence*, *16*(3), 469–497.

Roos, N. (2020). A Semantic Tableau Method for Argument Construction. *BeNeLux AI Conference (BNAIC).*

Russell, S. J., & Norvig, P. (2010). *Artificial intelligence: A modern approach.* Upper Saddle River: Prentice Hall.

Spohn, W. (1990). A General Non-Probabilistic Theory of Inductive Reasoning. In *Machine Intelligence*

and Pattern Recognition (Vol. 9, pp. 149–158). Elsevier. doi: 10.1016/B978-0-444-88650-7.50017-2

Spohn, W. (2012). *The laws of belief: Ranking theory and its philosophical applications*. Oxford: Oxford University Press.

Tan, P.-N., Steinbach, M., & Kumar, V. (2014). *Introduction to data mining*. Harlow: Pearson.

Taniar, D., Rahayu, W., Lee, V., & Daly, O. (2008). Exception rules in association rule mining. *Applied Mathematics and Computation*, *205*(2), 735–750.

van Leeuwen, L., & Verheij, B. (2019). A Comparison of Two Hybrid Methods for Analyzing Evidential Reasoning. In *JURIX* (pp. 53–62).

Verheij, B. (2017). Proof with and without probabilities. *Artificial Intelligence and Law*, *25*(1), 127–154.

Verheij, B. (2018). *Arguments for good artificial intelligence* [Inaugural Lecture]. Groningen: University of Groningen.

Verheij, B. (2020). Analyzing the Simonshaven case with and without probabilities. *Topics in cognitive science*, *12*(4), 1175–1199.

Wallace, C., Korb, K. B., & Dai, H. (1996). Causal discovery via MML. In *ICML* (Vol. 96, pp. 516–524).

Witten, I. H. (Ed.). (2017). *Data mining: Practical machine learning tools and techniques*. Amsterdam: Elsevier.

Zenker, F. (2020). From Stories-via Arguments, Scenarios, and Cases-to Probabilities: Commentary on Floris J. Bex's" The Hybrid Theory of Stories and Arguments Applied to the Simonshaven Case" and Bart Verheij's" Analyzing the Simonshaven Case With and Without Probabilities". *Topics in cognitive science*, *12*(4), 1219–1223.