DATA-DRIVEN MODEL-THEORY BASED CLASSIFICATION

by

TANGRUI LI

(Under the Direction of O. Bradley Bassler)

ABSTRACT

I have briefly analyzed how neural networks and inductive logic programming obtain knowledge from data items. And I found that the knowledge found by those methods on one dataset may cause problems when combined with the knowledge found on the other datasets, which means the knowledge is not robust enough. To mitigate this problem, I have proposed obtaining robust knowledge according to the definition of the global model in the model theory. Specifically, it finds mathematical logic rules from data items. The rules found from different datasets can be mixed and used correctly. Using the method proposed, I have further proposed methods to solve classification problems in three situations. The experiments show that the mathematical logical rules found have good performance even on small datasets, and they will not cause problems when combined with rules found on the other datasets.

INDEX WORDS: [Artificial Intelligence, Rule Extraction, Data-driven, Model Theory, Neural Networks, Inductive Logic Programming]

DATA-DRIVEN MODEL-THEORY BASED CLASSIFICATION

by

TANGRUI LI

B.E., University of Science & Technology Beijing, China, 2019

A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

MASTER OF SCIENCE

ATHENS, GEORGIA

2021

DATA-DRIVEN MODEL-THEORY BASED CLASSIFICATION

by

TANGRUI LI

DEDICATION

For the beautiful world. And for my efforts of making it more beautiful.

ACKNOWLEDGEMENTS

Thank you, my parents, and my teacher.

Actually, I originally had a lot to say, but now when everything is done, I just want to rest quietly and have a cup of tea.

For my feelings about the days, when I had been working on my research and thesis, I will leave them to myself when I see these words again.

I still hope that more blessings will be left to tomorrow.

I hope that when I come back many years after, I will still be that ambitious and hardworking boy.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

After thousands of years of accumulation, humans have a lot of knowledge that can be recorded, such as mathematics. However, there is still much knowledge that humans do not understand yet. Fortunately, related data could be collected with the development of information technology. To extend human knowledge, methods of obtaining knowledge directly from data play a critical role.

However, the data collected is always limited and targeted. Therefore, I hope that the knowledge obtained from the data is not specific to the data itself and can be mixed with knowledge obtained from other data without conflict. Such knowledge is called robust knowledge in this paper.

Neural network (NN) is a method of obtaining knowledge from data, but the knowledge acquired has considerable pertinence, which is the reason for overfitting. In addition, combining the knowledge from NNs may yield conflicting results. Although NN's using of deep features is very enlightening, it is not the best method for gaining robust knowledge.

Except for NN, inductive logic programming (ILP) is also a possible approach. The knowledge acquired is in logic programs, which are Boolean expressions that can be explained easier than NN parameters. Because of the Boolean formation, the auxiliary predicates proposed by some ILP methods can conclude some problem-independent knowledge in solving specific tasks, which shows the robustness to some extent. However, the logic programs still cannot be mixed arbitrarily, just like NNs. Therefore, though ILP is better than NN, it is still not the desired method for getting robust knowledge.

To propose such a method, this paper proposes a method of finding mathematical logic rules based on model theory. Rules are found from data directly, and it can also be mixed without conflicts. To show the method can really work on toy and practical cases, experiments using the method to deal with classification problems in three situations are given and analyzed in detail, and sound results are also provided.

CHAPTER 2

RELATED WORKS

The purpose of finding robust knowledge is to allow an intelligent method to accumulate knowledge in processing different data. This requires that the knowledge found on old data can be used safely on new data. It is not to say that robust knowledge is omnipotent, but it is required to make correct judgments as much as possible. And when the knowledge is not suitable for making judgments, such knowledge just should not be used.

This part will analyze the way NN and ILP acquire and utilize their knowledge, discover their disadvantages, and then suggest a way for amelioration. The related knowledge of model theory is closely related to the method proposed, and it is complicated, so I will make a more systematic introduction next as a separate part.

2.1 – Knowledge In NN Models

NNs store the knowledge in NN models (NN models only refer to the trained NNs), but they are not the minimal unit of knowledge, which means there is sub-knowledge in them. [2] analyzes the relation between the hidden nodes in convolution neural networks (CNNs) and input images. It finds that the hidden nodes in the second layer are activated largely by the parts which are consistent with our understanding of edges and corners in input images. However, recognizing edges and corners is clearly not only for a specific image recognition task, but a kind of general knowledge in all image-related tasks, so this NN model must contain sub-knowledge that could be used to solve other problems. Corresponding research of transfer learning [7] and knowledge distillation [8, 9] also support this viewpoint.

Experiments have shown that common sub-knowledge exists indeed, and it is helpful in training other NNs. Compared with not using such techniques, training NNs is slower, and the performance of trained NNs is worse. However, no matter whether using transfer learning or knowledge distillation, the knowledge in one NN model does not directly constitute the knowledge of another NN model, but is adjusted through optimization algorithms. The parameters of the transferred model are different from the transferring model, and current research is not able to show which knowledge has been transferred and which has not. From this perspective, although NN models do contain sub-knowledge, this sub-knowledge only supports the model-level transferring. If we want to mix the knowledge in NN models, we have no choice but to mix different NN models directly, and collect them in a set. Even so, since NN models are trained for highly specific tasks, we cannot perform the endless accumulation of every tiny task. [3, 7] also give some supports, they point out that with the deepening of NNs, hidden layer nodes become closer and closer to the training data. In addition, the input space of NNs is often very large, for example, ResNet-18 [17] accepts all inputs with the size 224×224. If the input is an RGB image, there are $256^{224\times224\times3}$ possibilities in total, but at present, it is hard that some datasets have such many data items. What's more, according to the explanation in [3, 7], at the last few layers of a NN, the meaning of the hidden nodes is very close to the training data, so for other possible inputs in the input space, even though the size of them meets the requirements, the activated results of hidden nodes are not meaningful. As a result, even we are able to collect different NN models, if they have the same input size (which is common), we still need to make it clear to which NN model an input should be forwarded. This is simple for human intelligence, but for artificial intelligence, it is much harder.

Additionally, in the research on the input sensitivity of NNs [4], some specific input content may have a greater impact on the optimization. [5] analyzes two NN models for recognizing emails, the mainly focused content of one model is consistent with human understanding, but the other model is apparently using unrelated parts, though these two models have close performance. Here, I take the degree of human's understanding of the knowledge into account, since if we want to separate sub-knowledge in NN models when such knowledge cannot be understood by people, we cannot find a way to use them. However, in the analysis of [2], although humans can understand the meaning of the hidden nodes in the first few layers of a CNN model, we cannot take them off the NN models as independent sub-knowledge. Since [6] also points out that there are lots of focused input parts that are not consistent with human understanding heavily affects the performance. [6] shows, in an example of an image classification task, by modifying an image that cannot be distinguished by the human eyes on those parts, the judgment of a NN model will be distorted or even manipulated. Therefore, even if we think that some hidden nodes are consistent with our cognition, we do not know what role they actually play in making judgments.

Since we cannot obtain sub-knowledge by separating certain parameters, some studies hope to summarize logic rules from NN models [15]. Compared with parameters, logic rules apparently have higher interpretability, and rules from different data can be naturally mixed, just like combining rules from different knowledge bases in classical symbolic artificial intelligence approaches. However, compared with training NN models first and then extracting rules, ILP [1] is a more direct method.

2.2 – Knowledge Obtained By ILP Methods

ILP is also a symbolic artificial intelligence method. After providing background knowledge, positive and negative examples in a formal language, it can draw logic programs that entail all positive examples while entailing no negative examples.

Logic programs are usually represented in horn clauses with the form $H{:}-B_1, B_2, \dots, B_n$, in which H is called the head, and $B_i, i \in [1,2,\dots,n]$ are in a conjunctive relationship, which is called the body. The whole horn clause can also be translated into a well-formed formula (wff) with $\rightarrow$ as the main connective like $B_1 \& B_2 \& \dots \& B_n \rightarrow H$. A logic program represents that if the background knowledge given entails all $B_i, i \in [1,2,\dots,n]$ in the body, then the background knowledge should also entail H in the head, which is just like an "if … then …" conditional description.

When there are multiple logic programs found, bodies of them may overlap. To make programs look concise, some ILP approaches propose the auxiliary predicates to represent these overlapping parts [10, 11, 12]. These predicates are exactly the sub-knowledge mentioned before. But such sub-knowledge is not robust enough. In the research of model theory [14], mathematical logic rules are introduced. Examining their formation will help find the reason why they are not robust.

(**Argument**) An argument is expressed as $H \vdash C$, where H stands for the hypothesis and C stands for the conclusion. H may contain multiple wffs (well-formed formulas), and C contains only one wff.

(**Mathematical logic rule**) A rule is expressed as $R = I/O$, where I represents the premise of the rule, also called the input; O represents the result of the rule, also called the output, both of which are arguments. The premise can contain multiple arguments, but the result has only one argument.

The most general mathematical logic rule has the following form.

$$\frac{\begin{array}{c} H_1 \vdash I_1 \\ \vdots \\ H_n \vdash I_n \end{array}}{H_{n+1} \vdash O}$$

In which $H_i, i \in [1,2, \dots, n, n+1]$ contains any number of wffs, but $I, i \in [1,2, \dots, n]$ and O can only contain one wff. $H_i \vdash I_i, i \in [1,2, \dots, n]$ and $H_{n+1} \vdash O$ are called arguments, in which H_i stands for the hypothesis and I_i stands for the conclusion.

When an argument is satisfied, **<u>if</u>** all wffs in the hypothesis are entailed, **<u>then</u>** the wff in the conclusion should also be entailed. And when a rule can be used, **<u>if</u>** all arguments in the premise are satisfied, **<u>then</u>** the argument in the result should also be satisfied. We can see there are two "if … then …" descriptions. Compared with logic programs, the additional "if … then …" description is for arguments, describing whether a rule should be used. For example, when we talk about red apples, we may say "when if an apple is a kind of fruit, then it is red if red is a kind of color." The "color" and "fruit" here are for the additional "if … then …" descriptions of "red" and "apple." But usually, we just say "a red apple is red" since we all agree that apple is a kind of fruit and red is a kind of color. Since these additional conditionals are all satisfied by default, they will not affect using the rules, and so we could just ignore them. In this way, mathematical logic rules could be viewed as logic programs, if $I_i, i = 1,2, \dots, n$ and O are all positive literals. But even so, finding mathematical logical rules is still different from ILP since only data items (not the expert data) will be used in finding these rules. Except for that, these data items only work as the "positive examples" in ILP, and no "negative examples" will be referred to.

$$\frac{\begin{array}{c} H_1 \vdash I_1 \\ \vdots \\ H_n \vdash I_n \end{array}}{H_{n+1} \vdash O} \quad \Rightarrow \quad O{:}-I_1, I_2, \ldots, I_n$$

As said before, auxiliary predicates conclude the sub-knowledge in data. Naturally, we may use auxiliary predicates found in one task to similar tasks. It seems reasonable, but when discussed seriously in the domain of mathematical logic rules, problems arise.

For example, a logic program is found in one dataset when the hypothesis H_1 is satisfied, but this logic program is going to be used in another dataset when the hypothesis H_2 is satisfied.

$$\frac{\begin{array}{c} H_1 \vdash B_1 \& B_2 \& \cdots \& B_n \\ H_2 \vdash B_1 \& B_2 \& \cdots \& B_n \to H \end{array}}{? \vdash H}$$

Though it is still possible H_1 and H_2 could be satisfied at the same time, clearly it needs further explanation, since this does not happen by default. To make it clear, we must define the hypothesis of each argument in mathematical logic rules, which will be discussed in detail in the following parts of this paper.

CHAPTER 3

MODEL THEORY BACKGROUND KNOWLEDGE

The research of model theory discussed in this paper is mainly about model theory of propositional logic. Therefore, if we want to apply it to data, the data needs to be binary to have a logical meaning. This could be the result of preprocessing, such as binarizing an image, or the data itself could have logical meaning by itself. Therefore, we take the binary dataset as an analogy of the atomic model to enable the analysis, and this will be called the analogy between atomic models and datasets.

(**Valuation**): a valuation is any function from the set of wffs of L (the language of the logical system) to the set $\{t, f\}$ of truth-values such that it assigns f to at least one wff. If a truth-value assignment only defines truth-values for atomic wffs, this truth-value assignment is called an atomic valuation.

(**Model**) A model is a set of valuations. When there are only atomic valuations inside, the model is called an atomic model.

(**The analogy between atomic models and datasets**) Assume we are given a dataset having only binary attributes. Each binary attribute is viewed as an atomic wff, and each data sample in the dataset is viewed as an atomic valuation by assigning the value true to those binary attributes which appear in the dataset and false otherwise, and then this dataset is viewed as an atomic model.

This part will show some relevant background knowledge of model theory and further propose the analogy between mathematical logic rules and knowledge. As the name suggests, this analogy finds rules as the knowledge of data. If we think that knowledge should at least be correct on the data, then the rules should also at least be correct on the model based on the analogy between atomic models and datasets. Discussing whether the knowledge is correct on data is relatively conceptual, but discussing whether rules are correct on models is much more formal, which is introduced in model of rules. Part 3.1 will introduce three formal definitions of model of rules (model of rules is different from model), namely deductive model, local model, and global model. In the end, I will show that the global model is the desired definition.

Model of rules discusses whether a set of rules will take a model (a set of valuations) as a global model, but since such a definition is a necessary and sufficient condition, it also describes to which rules a model will become a global model, which is not mentioned explicitly in [14]. To this point, I will introduce the definition of global rule and the analogy between mathematical logic rules and knowledge in Part 3.2.

3.1 – Model Of Rules

When talking about a model of a wff, we only need to check whether all valuations in a model assign the wff the truth-value of t, which is an enumerative way of checking. But when considering the model of rules, rules, as an inference system, will generate new wffs infinitely. For example, if I only have one wff A, and (& *Introduction*) as the only rule at the beginning, it will automatically take the new wff $A\&A, (A\&A)\&A, etc.$ into account.

$$(\&\, Introduction)\, \frac{\begin{matrix} H \vdash p \\ H \vdash q \end{matrix}}{H \vdash p\&q}$$

Therefore, endless new wffs are generated when considering model of rules, so we also need some methods assigning newly generated wffs truth-values.

Using truth tables is a good method. It is a necessary and sufficient condition for a wff to be assigned true or false, therefore no wffs have no or two truth-values. Such a condition determining the unique truth-value of each possible wff in a formal language is called a property. In addition to classical truth tables, there are many other properties, which will be discussed later.

> (**Property of $\boldsymbol{p}$ on $\boldsymbol{V}$**) Given p is a wff and V is a model, a property of p is a necessary and sufficient condition for p to be assigned t or f on V, so that the wff p must have one and only one truth-value.

Now we have a syntactic side, which is an inference system generating wffs, and we also have a semantic side assigning new truth-values based on truth-values (such as the truth-value of $t\&t$ is t). But when discussing merely on the syntactic side, we will not need to discuss the truth-values of wffs, and when discussing the property merely, the Boolean expressions of wffs do not play a critical role. So, these two sides are separate. To attach truth-values to wffs and thus link these two sides, the idea of a model of rules is introduced.

However, when the truth-value of a wff is determined by a model of rules, its truth-values could also be determined by the semantic side. For example, consider the property "if a certain wff can be obtained by natural deduction, then its truth-value is t." Suppose p and q are two atomic proposition symbols, since both $p \to p$ and $q \to q$ could be obtained by natural deduction (see the appendix for more details), their truth-values are t according to the property. And since $(p \to p)\&(q \to q)$ is also derivable, its truth-value is t as well. On the other hand, if the classical truth table of & is used, the truth-value of $(p \to p)\&(q \to q)$ is also determined by the truth-table according to the truth-values of $p \to p$ and $q \to q$.

Here, we are fortunate enough that the truth-values of $(p \to p)\&(q \to q)$ from the property and from the truth table are the same. But this is not guaranteed when using other properties or definitions of model of rules. Such as, following the previous property definition, the truth-value of p is f since it is not derivable, and the truth-value of $p \to p$ is t since it is derivable. As a result, the following row of a truth table of $\to$ could be concluded, which is consistent with the classical truth-table of $\to$.

Table 1: A row of the truth table defined by the definition of deductive model

$v(p)$	$v(p)$	$v(p \to p)$
f	f	t

But since q and $p \to q$ are not derivable, the truth-values of them are both f, which yields a controversial row.

Table 2: Another row of the truth table defined by the definition of deductive model

$v(p)$	$v(q)$	$v(p \to q)$
f	f	f

The above definition of the model of rules is called *deductive model*, which contains the following definitions:

(**Satisfaction of an argument**): a valuation v satisfies argument H/C iff whenever v assigns t to all members of H, v assigns t to C; if v assigns f to any members of H, H/C is satisfied vacuously.

(**$\boldsymbol{V}$-valid**): an argument is V-valid iff it is satisfied by every valuation in V.

(**Deductive model**): V is a deductive model of a rule iff the provable arguments of this rule are all V-valid.

As the example of the truth table of → shown, this definition is a bit brute force, which makes it not necessarily functional.

(**Functionality**) If a model's definition of the truth-value of wffs can make an atomic valuation uniquely correspond to an extended valuation, the model is said to be functional.

Therefore, deductive model fails to define a functional property, which means the calculation of truth-value is weaker than logical reasoning. Even if a wff can be obtained through natural deduction, we might get controversial truth-values. On the other hand, another definition which is called *local model* is defined as the following.

(**Valuation v satisfies rule $\boldsymbol{R}$**) v satisfies rule R iff if v satisfies the inputs of R, then v satisfies the output of R.

(**Local model of a rule**) V is a local model of a rule R iff every member of V satisfies R.

Consider the two connective rules of →:

$$(\rightarrow Introduction)\frac{H, p \vdash q}{H \vdash p \rightarrow q} \quad (\rightarrow Elimination)\frac{\begin{matrix} H \vdash p \\ H \vdash p \rightarrow q \end{matrix}}{H \vdash q}$$

Suppose H is satisfied and a valuation v assigns false to p and q, namely $v(p) = v(q) = f$. According to the local definition, the premise of the input of $(\rightarrow Introduction)$ is vacuously satisfied, so the rule is also satisfied, therefore $v(p \rightarrow q) = t$, which is consistent with the classical truth table of →. In fact, not only this row of the truth table, the other three rows are also consistent with the classical one, which is proved in [14, p37, Local Expression Theorem]. Hence, a local model of the two connective rules of → must fix the classical truth table of →.

However, local model also has a problem, which can be found in Peirce's Law $((A \rightarrow B) \rightarrow A) \rightarrow A$. This wff cannot be obtained by natural deduction using only *(→ Introduction)* and *(→ Elimination)* but its truth-value could be defined by local model of these two rules [14, p37, 3.2]. In another word, under the definition of local model, <u>logical reasoning is weaker than truth-value calculation</u>.

To mitigate the shortcomings of the above two definitions, while retaining their advantages, we introduce the definition of global model, which is defined as follows:

(**Preserve validity**) a rule preserves validity iff whenever the inputs of R are all V-valid, the output is also V-valid.

(**Global model of a rule**) V is a global model of a rule iff the rule preserves validity.

(**Global model of a set of rules**) V is a global model of a set of rules iff V is a global model of all rules.

Although the claim is not without its controversial side, Garson effectively argues in [14] that the notion of global model achieves the best, because most natural, matching of syntactic and semantic capacities. Since we will be focusing on global models exclusively in the following discussion, at this point I will refer simply to "models," understanding always that global models are meant.

3.2 – Global Rules

Part 3.1 introduces the reason for choosing the global model as the right definition for bridging the semantic and syntactic side. This part will explain why we need to bridge them and make sure they work "harmoniously."

An atomic model can be extended using properties, and the extended model will have the property directly. For example, "$v(p\&q) = t$ iff $v(p) = t$ and $v(q) = t$" is one property, and it is not just a property for a specific wff whose Boolean representation is $p\&q$, it is a property schema in which p, q could be substituted for any wffs. Therefore, it is a property that holds in all possible models in the langue of propositional logic without $\vee$ as a connective and where $\sim p$ is defined as $p \to \perp$ and where $p \leftrightarrow q$ is defined as $(p \to q)\&(q \to p)$. The expansion is endless and all wffs with & as the main connective are assigned a truth-value following the property. Note that here the description of the language is unusual. Further explanations will be given in the following discussion. And since we will be focusing on the mentioned language exclusively in this paper, I will call the language at issue simply "this language" or "the language."

Including "$v(p\&q) = t$ iff $v(p) = t$ and $v(q) = t$," there is a compound property $\|SI\|$ defined as follows:

$\|\bot\|$: $v(\bot) = f$

$\|\&\|$: $(A\&B) = t$ iff $v(A) = t$ and $v(B) = t$

$\|\rightarrow\|$: $v(A \rightarrow B) = t$ iff for all v' in V, if $v \leq v'$, then $v'(A) = f$ or $v'(B) = t$

$\|\leftrightarrow\|$: $v(A \leftrightarrow B) = t$ iff for all v' in V, if $v \leq v'$, then $v'(A) = v'(B)$

$\|\neg\|$: $v({\sim}A) = t$ iff for all v' in V, if $v \leq v'$, then $v'(A) = f$

$\|\leq\|$: $v \leq v'$ iff for all wffs p in A_{full}, if $v(p) = t$, then $v'(p) = t$

Correspondingly, as the semantic side, its paired syntactical side according to the global definition of model of rules is an inference system SI including the following rules: (hypothesis), (weakening) (restricted cut), (& introduction), (& elimination), (→ introduction), (→ elimination), (↔ introduction), (↔ elimination), (∼ introduction) and (∼ elimination) [14, p35]. You can also check the appendix to have a quick glance.

Note that except for $\|\bot\|$ and $\|\&\|$, other properties in $\|SI\|$ are largely different from the corresponding classical truth tables, and it does not have any property pertaining to ∨. This is because after choosing the global model as the definition of model of rules, such properties are designed specifically [14, p79, SI Theorem]. If I want to add connective rules for ∨, or to define ∼ and ↔ as independent connectives, the corresponding properties will become much more complicated. Therefore, for simplicity, I do not include these rules in the discussion of this paper. Using the property $\|SI\|$, we could extend an atomic model, namely V_{atomic}, to a model V of the full propositional algebra A_f generated from the atomics in group A (i.e., all formulas generated recursively from the atomics by using rule formation for whatever logical connectives are available in SI.) The extension described above, and its uniqueness are guaranteed by the *SI Theorem*:

(**SI Theorem**) $\|SI\|$ is a functional natural semantics for SI. [14, p79, SI Theorem]

Discussing more details here, and in particular what a functional natural semantics consists in, is not helpful in explaining the method, so I will just use a part of the conclusion of the *SI Theorem*, that this theorem assures SI globally expresses $\|SI\|$. The reader who wishes to understand the full sense of the theorem may consult the original text cited above. Since from here on I will only be concerned with global expression, as an abbreviation, I will use "express" instead of "globally express" in the following discussion.

Therefore, SI will take this extended model V as a global model. In fact, not only this model, but for any possible models in this language, if the model obeys $\|SI\|$, it is a global model of SI, and vice versa.

(**Global expression**) A system S globally expresses property P iff for all models V for a language L for S, V is a global model of the rules of S exactly when V obeys P.

This provides a way of judging whether a model obeys $\|SI\|$ without referring to $\|SI\|$ itself directly. Here, it seems it is not meaningful since $\|SI\|$ is already explicitly represented. But when the properties are difficult or impossible to represent explicitly, the importance of the above method will become apparent.

The above discussion pertains to all possible models in the language, but when a model containing only a few valuations is considered, other properties may also be expressed in addition to $\|SI\|$. For example, consider a model V_{toy} with only four valuations (since $\|SI\|$ is functional by the *SI Theorem*, V_{toy} can be represented by its corresponding atomic model).

Table 3: The atomic model of V_{toy}

#	A	B	C	D
1	t	t	t	t
2	t	t	t	t

In addition to $\|SI\|$, such a model also expresses the property $\|V_{toy}\|$ that is "$v(A) = t$ iff $v(B) = t$." It is definitely not the case on an arbitrary model in this language.

It seems that a property expressed only on a single model in a language is not meaningful. However, this is not a problem when the analogy between atomic models and the datasets is considered. First, when the models are analogies for datasets, a valuation represents a data sample. In collecting a dataset, some data samples will simply never appear due to the source of data, and so neither will their corresponding valuations. For example, consider a dataset about apples, and suppose one attribute "made of iron" is used. Such an attribute will clearly be false in all data samples, which means the valuations with the atomic wff corresponding to this attribute being true will never appear. We now introduce some definitions to aid in discussion.

(**Concept of people**) A concept cp is a way of thinking agreed upon by a few people. For this group of people, if they encounter something, they will draw the same conclusion of t or f that whether this thing follows cp or not.

For example, red apples follow the concept of "apples" and oranges do not follow this concept in human knowledge. Note that the "people" in the definition is very important since different people might have completely different concepts. But in this paper, there is a hypothesis that the group of people who hold the concept is fixed, or we can also say the group is not fixed, but the concept is shared in all possible groups of people, just like the so-called the commonsense. Therefore, as an abbreviation, I will simply call it "concept".

Concepts could be used as attributes to collect data.

(**Attributes used to collect data samples**) A set $Attr$ is a set of concepts (also called attributes when they are used to collect data samples) used to turn things encountered into data samples.

For example, a set $Attr_{toy}$ has in it three concepts, namely "is apple," "shape is round" and "color is red." Use $Attr_{toy}$ to turn a red apple instance encountered into a data sample, it will be represented as an atomic valuation as the following.

Table 4: A recorded atomic valuation

#	$is\ apple$	$shape\ is\ round$	$color\ is\ red$
1	t	t	t

Correspondingly, if we use $Attr_{toy}$ to turn a few apple instances into atomic valuations, we will get an atomic model.

In collecting data samples, we usually have a purpose, such as $Attr_{toy}$, which is used to collect apples. Such a concept is called the main concept, namely cp. Including cp, according to the attributes used, as said, some valuations will never appear. For those valuations that will appear, I will call them possibly-collected valuations of cp, and a corresponding set containing all such valuations is called W_{cp}. Similarly, we could have possibly-collected models of cp, and each of them is called W_i. Note that all possibly-collected valuations are also extended by $\|SI\|$, but since $\|SI\|$ is functional, they are only represented in an atomic format. In another word, we take all valuations in W_{cp} to be extended in the unique way from the corresponding atomic valuations. Therefore, even though we cannot really collect the truth-value assignments for all compound wffs, they are still called possibly-collected valuations.

Specifically, in this paper, we are going to collect data for machines to solve classification problems, and the attribute to be classified should always be for the purpose of collecting data. So, I will just call them possibly-collected valuations instead of possibly-collected valuations of cp for simplicity, and analogously for the possibly-collected models of cp.

*(***Possibly-collected model of a concept***)* Take any subset of the valuations in W_{cp} to form a model W_i, which is called a possibly-collected model. A set which contains all possibly-collected models is called W.

Note that W is a set of models, and it is NOT a set of valuations. However, W_{cp} is a set of valuations. Since there might be some atomic valuations that will never appear and W_i contains extended valuations, we might be able to get some other properties, just like finding $\|V_{toy}\|$ in V_{toy}.

($\|\boldsymbol{P_i}\|$ **of** $\boldsymbol{cp}$ **on** $\boldsymbol{W_i}$) Consider an arbitrary model W_i, it is possible to express properties of the main concept other than $\|SI\|$, namely expressing $\|P_i\|$. Specially, since W_{cp} contains all possibly-collected valuations, it expresses the most general property $\|P\|$ which holds for all possibly-collected models.

We collect data samples for machines to find such properties. Whenever possible, we do not want to challenge machines. As a result, we prefer easier ways for machines to get these properties so that we are willing to take whatever measures are necessary to get appropriate data samples. Though it is still not guaranteed that desired properties could be found even in such appropriate datasets, we will make efforts to approach this purpose. Since we are willing to let the machine know how to act in an easy way, I will assume that such a property $\|P_i\|$ always exists.

But this does not mean these properties necessarily follow our thinking. Suppose we use the attribute "made of iron" to collect some apple instances, since this attribute is always assigned f, we can get a property "$v(apple) = t$ iff $v(made\ of\ iron) = f$." This is correct on this dataset, but clearly, it is not consistent with our cognition. We know there are many other things made of iron that are not apples, but since these things are not reflected on this dataset, I think the property "$v(apple) = t$ iff $v(made\ of\ iron) = f$" found is also okay. This is inevitable since we cannot typically collect an extremely comprehensive dataset like an encyclopedia for machines to learn.

Now properties $\|P_i\|$ are expected to exist, but it is not usually easy to get them. We have no idea of their formal representations, and we even do not know how many properties a dataset can express. Although we can give some examples to show their existence, we cannot find a systematic way to get all of them. However, since we are going to solve classification problems, it is not necessary to know what properties the dataset expresses, we only need to judge whether the testing data has the same property of the main concept as the training data. Naturally, we are going to ask whether we can imitate the *SI Theorem* to judge its existence indirectly? The answer is yes.

In the most general case when we get the model W_{cp} which contains all possibly-collected valuations, the most common property $\|P\|$ is assumed to exist. What's next is to find some rules that take W_{cp} as a global model, so we can use them to judge whether a model in W express the property. To create such rules, the following definitions are built.

(**Constant true wff**) Let V be a set of full valuations, and consider a wff p, if $v(p) = t$ for all $v \in V$, we say p is a constant-valid wff on V.

(**Constant false wff**) Let V be a set of full valuations, and consider a wff p, if $v(p) = f$ for all $v \in V$, we say p is a constant-invalid wff on V.

(**Variable wff**) Let V be a set of full valuations, and consider a wff p, if $v(p) = t$ for some $v \in V$, and if $v'(p) = f$ for some other $v' \in V$, we say p is a variable wff on V.

(**Additional rule**) Let $R = I/O$ be a rule with at least one input argument and only one output argument. If this rule is not a rule in SI, and if whenever its input argument is valid on a model V, then so is its output argument, we say this rule is an additional rule on V.

For example, in V_{toy}, there are three atomic constant true wffs: A, B, D. Therefore, the following arguments are V_{toy}-valid: $D \vdash A, D \vdash B$. Take $D \vdash A$ as the premise of the rule and $D \vdash B$ as the result, an additional rule R_{toy} is found:

$$R_{toy} = \frac{D \vdash A}{D \vdash B}$$

We can create some additional rules on V based on the following Algorithm I.

(**Algorithm I**) This algorithm is used to generate additional rules. We follow the steps:

1. Choose one or more constant true wffs on V to form the hypothesis H_i of an argument, and then choose a constant true wff on V as the conclusion I_i of this argument, then an argument $H_i \vdash I_i$ if is found. For an arbitrary valuation $v \in V$, all the wffs in H_i are assigned $true$, and the wff in I_i is also assigned $true$, therefore, the argument $H_i \vdash I_i$ is satisfied by all valuations $v \in V$, which means that the argument $H_i \vdash I_i$ is V-valid.
2. Find such V-valid arguments, use any of them as the premise I of a rule, and use an argument that is different from what is used to constitute the input as the conclusion O of a rule, and then one rule I/O is found. If this rule is not a rule in SI, since I is valid and so is O, this rule is an additional rule on V.

Including such additional rules found and rules in SI, a set of global rules is created. All global rules form an inference system S_g.

(**Global rules**) Take a rule $R = I/O$, if this rule is from SI or this rule is an additional rule on V, we say R is a global rule on V. All global rules form an inference system S_g.

In the analogy between atomic models and datasets, the following proposition holds: For any model in W, a model is a global model of S_g iff it obeys $||P||$ (as defined with respect to cp above). The description is straightforward, since the additional rules are designed to make every model in W become its global model, and because of the $||SI||$ *Theorem*, the model in W is also the global model of SI. Therefore, all models in W must be global models of S_g. On the other hand, $\|P\|$ is defined as the most general property, so every model in W should express $\|P\|$. This shows one direction of the description.

Since all models encountered must be in W as the analogy defines and $\|P\|$ is the most general property, every model in W must express $\|P\|$, and since the additional rules are designed to make every model in W become its global model, these models in W must be global models of S_g. This shows the other direction of the be-implication.

Simply speaking, one direction of the proof is by definition, and the other direction is by deliberate design.

Strictly, we are talking about rules on an arbitrary W_i in W, so we do not need to specify what $\|P\|$ is explicitly. We only need to judge whether a model is S_g's global model, and then I can judge whether this model obeys $\|P\|$ or not. Although this proposition looks very similar to the $\|SI\|$ *Theorem*, there is the main difference that the $\|SI\|$ *Theorem* applies to all models in the language, but this proposition only applies to models in W.

> (**The analogy between mathematical logic rules and knowledge**) Given a dataset and the analogy between the atomic model and the dataset, the knowledge on the dataset is viewed as global rules of the model corresponding to the dataset.

Though W can be much smaller than a set including all possible models in the language, considering collecting a dataset like W is still a bit too ambitious. Even if W only contains those models that will appear, it is typically still too rich compared to the data that can be really recorded by us. For example, under the main concept cp, suppose a set $\{A\}$ with one hundred concepts is used, then there are 1.26e30 possible valuations. Say 1.26e10 valuations are recorded, there are still too many valuations. Perhaps we can only really collect 1.26e5 of them. What's more, once we know what W is, there is no need for us to judge whether a model is a global model of S_g. We can simply check whether this model is a member of W. That is to say, it is hard and impossible for us to get W, and therefore, I hope that when only a part of W is obtained, namely W_s, some rules in S_g can still be discovered.

It is clear that the property on the model that contains the greatest number of valuations in W_s (namely $||P_s||$) is only a special case of $||P||$, so the global rule which takes all members in W_s as the global model (namely S_{gs}) does not necessarily take all members in W as global models as well. However, in contrast, S_g, which takes all members in W as global models, must also take all members in W_s as a global model, so the set consists of all global rules found in W_s, namely S_{gs}, may have a chance to be in S_g, at least "approximately."

Anyway, this is only a possible situation after all, but with the increase of the size of W_s, this situation will become more and more significant. When W_s is W itself, S_{gs} is S_g. Therefore, with the increase of W_s, we only need to keep the rules that are always global rules while adding new data items to W_s, and so S_{gs} can get closer and closer to S_g.

CHAPTER 4

FINDING CONSTANT WFFS

Following Algorithm I in Part 3, we need to look for constant true wffs on a model to create mathematical logic rules that take the model as a global model. But by $\|\neg\|$, constant false wffs could be transferred into constant true wffs simply by adding ~ at the front of them. So, actually, we need constant wffs, no matter whether true or false.

To do so, one cannot just take atomic wffs into consideration. In the worst case, when a model contains merely two different valuations, all atomic wffs may become variable wffs (the truth-value assignment of the second valuation for each atomic wff is just the opposite of the first one). Therefore, it is necessary to consider finding compound constant wffs. However, there are infinite compound wffs, and the exhaustive method clearly cannot work.

Compound wffs contain connectives and wffs (atomic or compound wffs). For convenience, the wffs used to create a compound wff are called material wffs, and the compound wff is also called the result wff.

(**Material/result wff**) A compound wff must have a main connective, the two (or one if the main connective is ~) wffs connected by the main connective are(/is) called the material wffs, and the original compound wff itself is also called the result wff.

Although atomic valuations can be expanded through $\|SI\|$, it is endless. So, we cannot keep expanding blindly. After converting the dataset to an atomic model, only atomic wffs are at hand. Although the truth-values of other compound wffs have already been determined by $\|SI\|$, the compound constant wffs are not at hand. Our purpose is to make some of the compound constant wffs at hand but not all of them.

(**At hand wff set**) Given an atomic model, by $\|SI\|$, there are infinite constant wffs, but some of them have not been found yet. By using different methods, a finite number of constant wffs could be found. For those constant wffs found, we say that they consist of an at hand wff set.

If one wants to find some constant compound wffs directly on a model containing multiple valuations, the only possible way I can think is filtering all compound wffs through enumeration, but it is impossible in an acceptable time. However, it is observed that when a model contains only one valuation, all atomic wffs are directly at hand constant wffs as desired. Therefore, if we only have one valuation, we already get some constant wffs. If we do not want to continue, it is done. But usually, a model contains more valuations, so we need to consider adding (updating) other valuations, which also makes finding at hand compound constant wffs an update-by-update process. With the addition of valuations, original constant wffs may remain constant or become variable. I will propose a method of finding not just atomic but also compound constant wffs based on the update of valuations in this part. Finally, a certain number of constant wffs are found within an acceptable time.

To make it clear, I will introduce some examples on the Lenses dataset [18] to help explain. For details of the dataset, please see the appendix.

4.1.1 – Source-0 Wffs

There are some compound wffs, even if they are indeed constant, they cannot always be found out as at hand wffs. For example, suppose A is a constant wff, according to $\|\neg\|$, $\sim A$ must also be a constant wff.

$$\|\neg\| \; v(\sim A) = t \text{ iff for all } v' \in V, \text{if } v \leq v', v'(A) = f$$

After one update, if A becomes variable, then $\sim A$ must also become variable. Therefore, bringing $\sim A$ to our at hand wff set is clearly unnecessary. Therefore, an independent relationship between constant wffs is desirable, so that being variable of one of them will not affect other wffs.

In $\|\neg\|$, the truth-value of compound wffs using $\sim$ as the main connective is completely dependent on their material wffs, so any such compound wffs and their corresponding material wffs should not be in a wff set at the same time. And by Occam's razor, suppose that there is a constant wff A, since A is shorter than $\sim A$, we will only keep A in the at hand wff set whenever possible.

Compound wffs using binary connectives still have the same problem. Suppose A is a constant true wff and B is a constant false wff, then $A\&B$ is a constant false wff. When B becomes variable, $A\&B$ will also become variable. But unlike the case of $\sim A$, $A\&B$ also shows some independence. When only A becomes variable, $A\&B$ is still constant. Therefore, though there are still some dependences, they are weaker. In addition, other binary connectives have similar situations which will be analyzed in detail in Part 4.2. In each case, I will analyze how the truth-value of the compound wff is related to which material wff.

If a material wff changes from constant to variable, and this will also cause the result wff to change from constant to variable (or flips its truth-value), we call this material wff a source of the result wff. Since all compound wffs discussed below are using binary connectives, based on the number of its source, there are four different cases. The first case is that both material wffs are source wffs, namely source-2. The second case is that only one of the material wffs is the source, namely source-1. Naturally, we may think the third case is that there are no source wffs at all, namely source-0. But this will be shown to be problematic. Therefore, we will just call it source-0* as an alternative. In addition, there is a special case in which we do not know whether a material wff is a source wff or not, namely source-x (note that source-x is not an abbreviation of source-0,1,2).

(**Source wff**) The truth-value of a compound wff might be subject to some of its material wffs considering $||SI||$. This means if one material wff changes from constant to variable (and the other remains the same), the result wff will also change from constant to variable, i.e. its truth-value will be flipped. If so, these material wffs are called the source wffs of the result wff. Since we are discussing the case that those wffs are in the at hand wff set, it is required that all source wffs need to be constant.

(**Source-1 wff**) If a compound wff only has one source wff, it is called a source-1 wff. All source-1 wffs are required to be constant.

(**Source-2 wff**) If a compound wff has two source wffs, it is called a source-2 wff. All source-2 wffs are required to be constant.

(**Source-0* wff**) If a compound wff does not have source wffs, the original compound wff is called a source-0 wff. If a source-0* wff is in the at hand wff set, it must be constant.

4.1.2 – The Case Of Source-x Wffs

The case of source-x wffs might be a little confusing. To make it clear, we need to talk about $\|\leq\|$ at first, which represents a precedence relationship on valuations (not only atomic valuations).

$$\|\leq\| \; v \leq v' \text{ iff for all wffs } A, \text{if } v(A) = t \text{ then } v'(A) = t$$

Since any valuation is endless, it is impossible to check all truth-value assignments inside them. However, Garson has proved that for the ordering of valuations we only need to check atomic valuations [14, p.63] whenever $\|SI\|$ is expressed by a model of those valuations.
Therefore, according to $\|\leq\|$, valuations in a model may make the model a graph of valuations, namely a valuation graph.

(**Valuation graph**) According to the ordering of valuations, a valuation v and a valuation v' having the smallest Hamming distance in the natural way are connected by a directed arrow from v to v' if $v \leq v'$. If all valuations are connected in this way, the resulting directed acyclic graph (DAG) is called a valuation graph.

For example, in a world with three atomic propositions (A, B, C), a model with all eight possible valuations is presented in the following valuation graph.

$$\begin{tikzcd}
 & (t, f, f) \arrow[r] \arrow[dr] & (t, t, f) \arrow[dr] & \\
(f, f, f) \arrow[ur] \arrow[r] \arrow[dr] & (f, t, f) \arrow[ur] \arrow[r] \arrow[dr] & (t, f, t) \arrow[r] & (t, t, t) \\
 & (f, f, t) \arrow[ur] \arrow[r] & (f, t, t) \arrow[ur] &
\end{tikzcd}$$

Figure 1: A valuation graph containing eight valuations

Note that although ||≤|| can be used to sort the valuation according to the truth-values of atomic wffs, it cannot be used to determine the truth-value of compound wffs when the valuation graph changes. When the graph is static, of course, the truth-values of wffs can be determined according to the graph.

For example, suppose that in a model with three atomic propositions, namely (A, B, C), there are two valuations, namely (f, f, f) and $(t, f, f)|$. Recall the specification of the property ||→||: $v(A \to B) = t$ iff for all v' in V, if $v \leq v'$, then $v'(A) = f$ or $v'(B) = t$. If (f, f, f) is the first valuation introduced, then in the model consisting of this valuation alone $A \to B$ is a constant true wff. When a new valuation (t, f, f) is added and we consider the model including both valuations, according to ||≤||, since $(f, f, f) \leq (t, f, f)$, and because of the truth-value of $A \to B$ is t on (f, f, f), then it might seem that the truth-value of $A \to B$ should also be t on (t, f, f). However, according to ||→||, the truth-value of $A \to B$ should be f on (t, f, f), and so, conversely, the truth-value of $A \to B$ now becomes f on (f, f, f) in this new model.

The above example shows when considering $||\leq||$, even if only one valuation is added, it may change the structure of the valuation graph, and then the truth-values of wffs may also change on all the other valuations. So, after adding a new valuation, the truth-value of each constant wff of the unadded valuation graph must be recalculated. As an abbreviation, after adding a new valuation, the valuation graph consisted of all previous valuations is called the old valuation graph. Correspondingly, constant wffs on the old valuation graph are called old constant wffs. In contrast, there is a new valuation graph and there are new constant wffs.

As mentioned in Part 3, some atomic valuations will never appear, so the valuation graph of a dataset will not in general be complete. Even so, the number of valuations may be very large. If one wants to calculate the truth-value of a wff that needs to consider $||\leq||$, it needs to consider all precedence relationships, which is time-consuming. And if the timeliness of the dataset plays a role, some of the valuations in the valuation graph may expire after running for a while, so except for adding new valuations, we also need to consider removing some valuations from the valuation graph.

To increase the computational efficiency and make the valuations in the graph more time-sensitive, I have set a maximum number of valuations for a valuation graph, which is called the memory of the valuation graph. When the memory of the graph is exhausted (we will also say the graph is full) and a new valuation is added, the valuation that was first added will be deleted. Specifically, if the valuation graph will be full after adding a valuation, we need to delete the first added valuation in the old graph and re-evaluate whether all constant wffs are still constant. Those who are no longer constant will also be deleted. Since in the previous and following analysis, I always assume that after adding a valuation, no valuations will be removed from the graph. Due to limiting of memory of the valuation graph, this is necessary.

This may also weaken constant wffs. If the memory of a valuation graph is smaller than the number of data items, there will be multiple full valuation graphs. The constant wffs found in one valuation graph may not be constant in another graph. However, the method proposed is data-driven. Therefore, this could also be used to mitigate overfitting. Suppose that in 999 updates, a wff A keeps constant, but at the 1,000th update, the new data item makes A variable. Though following the definition, A is indeed not the constant wff, but I also agree that it is unfair that just because of one data item, a constant wff on 999 data items is eliminated.

Although the valuation added to the graph is completely unpredictable, nonetheless the truth-values of old constant wffs do not change completely irregularly. Two valuations in a valuation graph either have a precedence relationship or not, and there are no exceptions. For example, (f, f, f) and (t, f, f) have a precedence relationship, while (t, f, f) and (f, t, f) do not. When there are two valuations in a model having a smallest Hamming distance, and if they therefore have a precedence relationship, we say they are partial valuation neighbors. On the other hand, if they do not have a precedence relationship, we will just say they are two separated valuations.

(**Partial valuation neighbors**) If two valuations (v, v') in a model having a smallest Hamming distance, and if they therefore have a precedence relationship, namely $v \leq v'$ or $v' \leq v$, then we say the two valuations constitute partial valuation neighbors. Assume that $v \leq v'$ and they are neighbors, then v is called the front valuation, and v' is called the back valuation.

Suppose that there is a constant wff $A \rightarrow B$, in which both of them are constant false wffs, we are going to discuss the truth-value dependence between the result wff and the two material wffs.

1. When A becomes variable after an update and B remains the same.

   Since A is variable in the new valuation graph, at least one of the following cases must be satisfied.

   a. There is a partial valuation neighbors, in which two inside valuations assign different truth-values to A.

   b. There are two separated valuations assign A different truth-values.

   In the first case, by $\|\leq\|$, the valuation assigns A false must be the front valuation (namely v), and the back valuation (namely v') must assigns A true. Therefore, $v'(A \rightarrow B) = f$, and so does $v(A \rightarrow B) = f$. Though $A \rightarrow B$ is still constant, its truth-value flips, so we say A is a source of $A \rightarrow B$ in this case.

   In the second case, clearly, we have both $t \rightarrow f$ and $f \rightarrow f$ and $\|\leq\|$ does not apply to them. Note it is possible that there are two valuations that $\|\leq\|$ applies to them, but they are not partial valuations neighbors. For example, if we have three atomic valuations $(f,f,f), (t,f,f), (t,t,f)$, it is clear that $(f,f,f) \leq (t,t,f)$ but they are not partial valuation neighbors. However, this does not matter. Suppose there are two valuations v_1, v_2 $(v_1 \leq v_2)$ that assign different truth-values to a wff A, such that $v_1(A) = f$ and $v_2(A) = t$ by $\|\leq\|$. If they are not partial valuation neighbors, which means that there must be other valuations between them, such as $v_1 \leq \cdots \leq v_2$. Still, by $\|\leq\|$, since $v_1(A) = f$ and $v_2(A) = t$, among those valuations between v_1 and v_2, there is a threshold valuation v_t, that $v(A) = t$ for all valuations $v_t \leq v, v \neq v_t$, and $v(A) = f$ for all valuations $v \leq v_t, v \neq v_t$. In another word, although v_1 and v_2 are not neighbors, there must be valuations between them that constitute partial valuation neighbors in which the two inside valuations assign A different truth-values. Therefore, when I say "$\|\leq\|$ does not apply to them," I mean they do not have a precedency relationship at all, such as (t, f, f) and (f, t, f). Therefore, $A \to B$ is variable in both cases, so we say A is a source of $A \to B$ in this case.

In summary, since at least one of the above cases must be satisfied. No matter what A is, it is a source of $A \to B$.

2. When B becomes variable and A remains the same.

   Similarly, there are two cases.

   - There is a partial valuation neighbors, in which two inside valuations assign different truth-values to B.
   - There are two separated valuations which assign B different truth-values.

   In the first case, by $\|\leq\|$, the valuation assigns B false must be the front valuation (namely v), and the back valuation (namely v') must assigns A true. Therefore, $v'(A \to B) = t$, so does $v(A \to B) = t$. $A \to B$ is still constant, and its truth-value remains.

   In the second case, clearly, we have both $f \to t$ and $f \to f$ and $\|\leq\|$ does not apply to them. Therefore, $A \to B$ is variable, so we say B is a source of $A \to B$ in this case.

   In summary, B becoming variable may or may not make $A \to B$ become variable or flip its truth-value, which totally depends on the structure of the new valuation graph. But the structure is unpredictable, as a result, we do not know whether B is a source of $A \to B$. Although in one case B is indeed a source of $A \to B$, so that we will just call B a para-source.

In conclusion, A is a source of $A \to B$ and B is just a para-source, and we will call $A \to B$ a source-x wff.

(**Para-source**) If one material wff changes from constant to variable (and the other remains the same), the result wff will or will not change from constant to variable, or its truth-value will or will not be flipped. If so, these material wffs are called the para-source wffs of the result wff. All para-source wffs need to be constant.

(**Source-x wff**) If a compound wff has at least one para-source, we say the compound wff is a source-x wff. All source-x wffs need to be constant.

4.2 – Analyzing The Combination Of Connectives And Constant Wffs

This part intends to analyze all twelve combinations, but I will only show the detailed analysis of some of them, and the other is left in the appendix. After that I will give the conclusion directly.

When using & as the main connective.

$$\|\&\| \; v(A\&B) = t \text{ iff } v(A) = v(B) = t$$

According to the truth-value of the material wffs, there are three sub situations.

a. Constant true & constant true

   Any material wff becoming variable will make the result wff be variable as well, so the result wff is a source-2 wff.

b. Constant true & constant false or constant false & constant true

   When only the material wff whose truth-value is t becomes variable, the result wff is still a constant false wff. When only the material wff whose truth-value is f becomes variable, the result wff will also become variable, so the result wff is a source-1 wff and the constant false wff is its source.

c. Constant false & constant false

Becoming variable of any of its material wffs will not make the result wff become variable, so the result wff here is a source-0* wff.

So far, we have shown cases of source-0*,1,2,x wffs, and I think it is enough to give the conclusion of the analysis. As the result, we have the following table of compound wffs.

- Source-0* wff: constant false & constant false.
- Source-1 wffs: constant true & constant false, constant false & constant true. (In which the source is marked in red.)
- Source-2 wffs: constant true & constant true, constant true ↔ constant false, constant false ↔ constant false.
- Source-x wffs: constant true → constant true, constant true → constant false, constant false → constant true, constant false → constant false, constant true ↔ constant false, constant false ↔ constant true. (In which the source is marked in red, and the para-source is marked in orange.)

It is the case that all compound wffs are dependent (or possibly dependent) on at least one of its material wffs. So that we cannot have any compound wffs and its source wffs (or para-source wffs) in an at hand wff set at the same time.

Here is a description.

1. All source-2 and source-x wffs are dependent (or possibly dependent) on both of their material wffs, so we cannot put any source-2,x wffs and any of their material wffs in the same at hand wff set.
2. Source-0* wffs (the case of false & false) look good, but after an update, it is possible that two material wffs both become variable, and this will possibly make the result wff become variable,

so we still cannot put any source-0* wffs and any of their material wffs in the same at hand wff set.

3. It also seems that it does not prohibit containing a source-1 wff and its material wff that is not its source in an at hand wff set. For example, suppose that A is a constant false wff and B is a constant true wff. The truth-values of B and $A\&B$ are not dependent, but according to Occam's razor, since A is shorter than $A\&B$, we do not need to contain $A\&B$ in the wff set whenever A is constant.

In summary, if a compound constant wff is in the wff set, its material wffs cannot be in the same wff set. And this yields the definition of source-0 wff.

(**Source-0 wff**) Constant atomic wffs are source-0 wffs. If a constant compound wff's material wffs are variable, we say the compound wff is a source-0 wff.

4.3 – Finding At Hand Source-0 Wffs

According to the analysis in Part 4.2, we cannot use two constant wffs as materials to create a constant compound wff, which means we can only use variable wffs as materials. But if we enumerate all possible combinations of variable wffs, suppose there are n variable wffs after one update, the time complexity is $O(n^2)$, which is not desirable. I will propose my method and then analyze its time complexity.

Let us first discuss a relatively simple situation, that is, the result wff contains one constant material wff and a variable material wff. The method is that, if the result wff is a constant wff, we are going to make the constant material wff as its source, such that the truth-value of the compound wff is dependent on it. Or put it another way, we are going to find source-1 wffs.

Since the truth-value of these wffs are dependent on some wffs in the wff set, we cannot put them in the at hand wff set directly, instead, we just put them in a waitlist. When the only source of this compound wff becomes variable, the two material wffs are both variable, which makes the truth-value of the result wff independent. Except for this, to be a constant wff, its truth-value must be further calculated on the valuation graph. If the result wff is indeed constant, then it is a source-0 wff.

By using this method, we also guarantee that if a wff is constant, no constant wffs take it as a material will be added to the wff set, which avoids the previously mentioned problem where Occam's razor is applied.

In the previous analysis, both material wffs of a source-1 wff are constant wffs, but since a source-1 wff's truth-value is only dependent on one material wff, source-1 wffs could be generated by combining variable and constant wffs. It has another advantage that the atomic symbols in the variable wff could be kept.

What's more, atomic symbols are not only atomic wffs. For example, suppose that A and B are two atomic symbols, and $A\&B$ is a source-0 wff. Therefore, A and B must both be variable. Although A and B are not atomic wffs in the constant wff set, they show up as atomic symbols in $A\&B$. Since we are going to find sub-knowledge in processing different data, if a certain attribute (concept) never appears, the knowledge related to it will completely disappear. This is very likely to happen when updating the valuation graph. Following the above example, suppose $A\&B$ is the only at hand wff containing the atomic symbol A, if $A\&B$ becomes variable and we do not use the method proposed to keep $A\&B$ as a part of some source-1 wffs, the atomic symbol A will disappear forever. Therefore, if we are going to use the at hand source-0 wffs to create rules, A as an atomic symbol will never be used, and all sub-knowledge related to A will disappear.

Another comment, note that two constant wffs are not used here to generate source-2 wffs, since it is possible to wait until one of the constant wffs becomes variable and then generate source-1 wffs or wait until both of them become variable wffs.

According to the conclusion in Part 4.2, there are five ways to create source-1 wff, namely S1-1,2,3,4,5:

(**S1-1**) Variable & constant false (The result is a constant false wff.)

(**S1-2**) Constant false → variable (The result is a constant true wff.)

(**S1-3**) Variable → constant true (The result is a constant true wff.)

(**S1-4**) Variable → constant false (The result is a constant false wff.)

(**S1-5**) Variable ↔ constant false (The result is a constant false wff.)

For simplicity, the experiments of this paper only use the first three ways. If so, at most two combinations will be attached to a variable wff. Therefore, after an update, suppose there are n variable wffs, we only need to check their attached source-1 wffs, which has the time complexity of $O(n)$.

Except for creating source-0 wffs using the above methods, it is possible to use two variable wffs to get source-0 wffs directly. But since there might be many variable wffs, enumerating all combinations is impossible. Therefore, I only propose a relatively straightforward approach. The point is neither to collect all constant wffs nor to use an approach that is quick but rarely collects constant wffs. Rather, the goal is to collect a robust set of constant formulas in a relatively efficient way.

As introduced in Part 4.2, to determine which constant wffs remain constant after a new valuation is added, it is inevitable to recalculate the truth-values of the previous constant wffs. Sometimes, adding a new valuation will not change the truth-value of a variable wff on the old valuations. In this case, we say the truth-value of the wff has a truth-value conversion pattern (TCP), such as $t \Rightarrow f$ or $f \Rightarrow t$. According to the similarities and differences of the TCPs, there are three following combinations can produce source-0 wffs, which are called S0-1, 2, 3.

(**Truth-value conversion pattern**) Let the truth-value of a constant wff be tv_0, after adding a new valuation to a model, if the truth-value of the wff remains tv_0 on the old valuations, and only on this newly added valuation the truth-value is the opposite of tv_0 (namely $\sim tv_0$), it is said that this variable wff has a truth-value conversion pattern, which is $(tv_0 \Rightarrow \sim tv_0)$.

(**S0-1**) According to ‖&‖, two variable wffs with the opposite TCPs can be connected by & to get a constant false wff. Taking & as the main connective only needs to consider each valuation separately. Since the TCPs are different, the variable wff has the opposite truth-values on the old valuations, and the opposite truth-values on the newly added valuation as well.

(**S0-2**) According to ‖↔‖, two variable wffs with the same TCPs can be connected by ↔ to get a constant true wff, since the truth-value of these two material wffs are the same on all valuations.

(**S0-3**) Correspondingly, according to $\|\leftrightarrow\|$, two variable wffs with the opposite TCPs can be connected by $\leftrightarrow$ to get a constant false wff. Since in all valuations, the truth-values of these two material wffs are the opposite.

4.4 – Considering Computing Resources

Suppose the number of constant wffs before adding a new valuation is n, in the most extreme case, half of them will become variable after an update. Therefore, there will be $O(n^2)$ source-1 wffs attached to the constant wffs. If all of them are justified as source-0 wffs, the number of source-0 wffs may grow exponentially. So, the method proposed also limits the number of source-0 wffs retained after each update.

Since now we must select among source-0 wffs, there needs to be a way to judge how satisfying a source-0 wff is. Concerning the purpose of finding source-0 wffs, it is only about using them to create mathematical logic rules that meet the definition. So, the only requirement is that source-0 wffs should be constant. Of course, they should also be truth-value independent according to Part 4.3, i.e. not belong to the same pair of partial valuation neighbors. Once these requirements are met, Occam's razor is applied to give priority to the shortest source-0 wffs found.

What's more, some source-0 wffs were discovered at the beginning of an update, while some source-0 wffs were discovered at a later stage. Compared with the source-0 wffs discovered at the end of the update, the wffs discovered earlier has obviously undergone more verification and are more credible, which gives different source-0 wffs a measurement of credibility. The default value is 1, and each time when a new valuation is added, if a source-0 wff remains constant, its credibility will increase by 1. And after adding a valuation, the retained wffs will be selected according to these two indicators, in which the length is the first to be considered. If the length is the same, the wff with higher credibility will be selected.

Generally, source-0 wffs found in the later stage are all generated by the combination of the source-0 wff discovered at an earlier stage, so it must be longer. But since we have limited the size of the valuation graph, a variable atomic wff may be changed back to a constant wff due to the deletion of old valuations. For example, in a world where 3 atomic propositions, namely (A, B, C), there is a model containing the following two valuations, and this model is limited to a maximum of two valuations.

Table 5: Two valuations in a valuation graph (before updating)

#	A	B	C
1	t	f	f
2	f	f	t

Obviously, A is not a source-0 wff, but when the new valuation is added, A may become a source-0 wff.

Table 6: Two valuations in a valuation graph (after updating)

#	A	B	C
2	f	f	t
3	f	t	t

But sometimes, we might want more than the limited number of source-0 wffs. I observe that the input order of valuations will also affect the source-0 wffs finally found. When the timeliness of the input is not so important, changing the input order and run the method proposed several times can produce more constant wffs in a linear time.

For example, suppose there are three valuations, $(f,f,f),(t,f,f),(t,t,f)$ in a world with three atomic propositions, namely (A,B,C). If (f,f,f) is used as the initial valuation in the model, there are three constant wffs at this time, which are A,B and C, all of which are constant false wffs. Note that at here there are no "after adding" formulas in the following table, since the valuation is used for initialization.

Table 7: initialization, valuations in a valuation graph and corresponding source-0 wffs

$valuation\ added$ $\big(v(A),v(B),v(C)\big)$	$at\ hand,$ $constant\ wff$	$after\ adding,$ $constant\ wff$	$after\ adding,$ $variable\ wff$
(f,f,f)	A,B,C	/	/

Table 8: initialization, source-1 wffs attached

$at\ hand, constant\ wff$	$corresponding\ source-1\ wff$
A	/
B	/
C	/

If (t,f,f) is the first input valuation, A becomes a variable wff, then according to S1-1 and S1-2, $B\&A, B\rightarrow A$ is created as B's source-1 wffs, and similarly, $C\&A, C\rightarrow A$ is associated with C.

Table 9: update once, valuations in a valuation graph and corresponding source-0 wffs

$valuation\ added$ $\big(v(A),v(B),v(C)\big)$	$at\ hand,$ $constant\ wff$	$after\ adding,$ $constant\ wff$	$after\ adding,$ $variable\ wff$
(f,f,f)	A,B,C	/	/
(t,f,f)	B,C	B	A

Table 10: update once, source-1 wffs attached

$at\ hand, constant\ wff$	$corresponding\ source - 1\ wff$
B	$B\&A, B \to A$
C	$C\&A, C \to A$

When the valuation (t, t, f) is then added, B also becomes a variable wff, whose inside source-1 wffs are therefore released. But only $B \to A$ is indeed constant, so after entering three valuations, there are only two source-0 wffs at hand, one is $B \to A$, and the other is C.

Table 11: update twice, valuations in a valuation graph and corresponding source-0 wffs

$valuation\ added$	$at\ hand,$	$after\ adding,$	$after\ adding,$
$(v(A), v(B), v(C))$	$unchanging\ wff$	$unchanging\ wff$	$changing\ wff$
(f, f, f)	A, B, C	/	/
(t, f, f)	B, C	B, C	A
(t, t, f)	$C, B \to A$	C	B

Table 12: update twice, source-1 wffs attached

$at\ hand, unchanging\ wff$	$corresponding\ source - 1\ wff$
C	$C\&A, C \to A, C\&B, C \to B$
$B \to A$	/

If the input order of valuation is changed, input (t, t, f) first (the initialization remains the same), A and B will become variable wffs at the same time. As a result, $C\&B, C \to B, C\&A, C \to A$ will be added to C as its inside source-1 wffs. Since A and B are variable wffs with the same TCP, then $A \leftrightarrow B$ is directly added as a source-0 wff.

Table 13: update once, change the order, valuations in a valuation graph and corresponding source-0 wffs

$valuation\ added$ $(v(A), v(B), v(C))$	$at\ hand,$ $constant\ wff$	$after\ adding,$ $constant\ wff$	$after\ adding,$ $variable\ wff$
(f, f, f)	A, B, C	/	/
(t, t, f)	$C, A \leftrightarrow B$	C	A, B

Table 14: update once, change the order, source-1 wffs attached

$at\ hand, constant\ wff$	$corresponding\ source - 1\ wff$
C	$C\&A, C \rightarrow A, C\&B, C \rightarrow B$
$A \leftrightarrow B$	/

After adding the valuation (t, f, f), $A \leftrightarrow B$ becomes a variable wff, so only $C \rightarrow (A \leftrightarrow B)$ and $C\&(A \leftrightarrow B)$ will be added to C as source-1 wffs, and only C is the constant wff at hand.

Table 15: update twice, change the order, valuations in a valuation graph and corresponding source-0 wffs

$valuation\ added$ $(v(A), v(B), v(C))$	$at\ hand,$ $constant\ wff$	$after\ adding,$ $constant\ wff$	$after\ adding,$ $variable\ wff$
(f, f, f)	A, B, C	/	/
(t, t, f)	$C, A \leftrightarrow B$	C	A, B
(t, f, f)	C	C	$A \leftrightarrow B$

Table 16: update twice, change the order, source-1 wffs attached

$at\ hand, constant\ wff$	$corresponding\ source - 1\ wff$
C	$C\&A, C \to A, C\&B, C \to B,$ $C\&(A \leftrightarrow B), C \to (A \leftrightarrow B)$

This can also make up for the weakening of constant wffs caused by the limitation on the valuation graph. Although the constant wffs found are not necessarily constant on all possible valuation graphs, the more frequently the constant wff is found, the more likely it is to remain constant on larger valuation graphs. Therefore, while changing the input order and running several times, we can find a set S_0 that keeps these wffs and their corresponding times found (namely its support). For each time of inputting, if the valuation graph is full after an update, all at hand constant wffs will be forwarded to S_0 and their support will increase by 1. If the wff is the first time introduced to S_0, its support will be set to 1.

4.5 – An Example On The Lenses Dataset

In the previous explanation, I have been using relatively abstract logical symbols such as A, B, and C. In this part, I will conduct small-scale experiments on a dataset with concrete logical meaning, and I will give some explanations about the results. Data items whose lenses type are "No Lenses" are used as the training data (a small part of them will be used for testing), together with the data of the other lenses types as the testing data.
Data items used for training are as follows.

Table 17: training data items from the Lenses dataset

#	Age	Prescript	Astigmatic	Tear Rate	Lenses Type
5	young	hyper	no	reduced	no
7	young	hyper	yes	reduced	no
9	pre	myope	no	reduced	no
11	pre	myope	yes	reduced	no
13	pre	hyper	no	reduced	no
15	pre	hyper	yes	reduced	no
16	pre	hyper	yes	normal	no

Its corresponding valuation graph is the following.

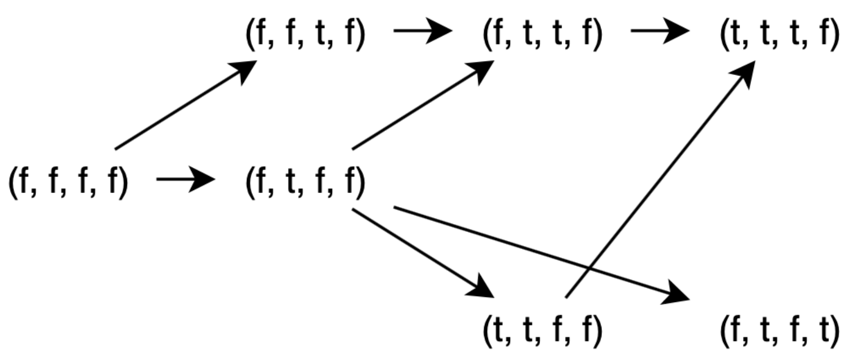
$$\begin{tikzcd}
 & (f,f,t,f) \arrow[r] & (f,t,t,f) \arrow[r] & (t,t,t,f) \\
(f,f,f,f) \arrow[r] \arrow[ur] & (f,t,f,f) \arrow[ur] \arrow[dr] \arrow[drr] & & \\
 & & (t,t,f,f) \arrow[uur] & (f,t,f,t)
\end{tikzcd}$$

Figure 2: a valuation graph corresponding to the training data

Data items used for testing. (Lenses type is “No Lenses.”)

Table 18: testing data items from the Lenses dataset

#	Age	Prescript	Astigmatic	Tear Rate	Lenses Type
1	young	myope	no	reduced	no
3	young	myope	yes	reduced	no

In each run of input, the number of wffs retained after one update is set to 5, and the memory of valuation graphs is set to 7. The memory of the valuation graph is the same as the size of the training data, which means no valuations will be removed from the graph in this example. I will change the order of the training data, and then run 5,10,15,20,25,30,35,40,45,50 times to show the impact of the number of runs on the constant wffs found. For the convenience of description, I use propositional symbol A instead of Age=young, propositional symbol B instead of Prescript=hyper, propositional symbol C instead of Astigmatic=no, and propositional symbol D instead of Tear rate=normal. The result of a random initialization is as follows.

Table 19: initialization, valuations in a valuation graph and corresponding source-0 wffs

$valuation\ added$ $(v(A), v(B), v(C), v(D))$	$at\ hand,$ $constant\ wff$	$after\ adding,$ $constant\ wff$	$after\ adding,$ $variable\ wff$
(t, t, t, f)	A, B, C, D	/	/

Table 20: initialization, source-1 wffs attached

$at\ hand, constant\ wff$	$corresponding\ source-1\ wff$
A	/
B	/
C	/
D	/

When a new valuation is added, A, B, C become variable but have the same TCPs. Therefore, we can use S0-2 to create new source-0 wffs. And since D is still a constant false wff, we can use S1-1 and S1-2 to create corresponding source-1 wffs in D.

Table 21: update once, valuations in a valuation graph and corresponding source-0 wffs

$valuation\ added$ $(v(A), v(B), v(C), v(D))$	$at\ hand,$ $constant\ wff$	$after\ adding,$ $constant\ wff$	$after\ adding,$ $variable\ wff$
(t, t, t, f)	A, B, C, D	/	/
(f, f, f, f)	$A \leftrightarrow B, B \leftrightarrow C, A \leftrightarrow C, D$	/	/

Table 22: update once, source-1 wffs attached

$at\ hand, constant\ wff$	$corresponding\ source - 1\ wff$
$A \leftrightarrow B$	/
$B \leftrightarrow C$	/
$A \leftrightarrow C$	/
D	$D\&A, D \to A, D\&B, D \to B, D\&C, D \to C$

After input the remaining 5 valuations, there are 5 source-0 wffs at hand.

- $C\&D$ (Constant false): To verify this is a source-0 wff, we do not need to consider the valuation graph. So, we just need to see whether C and D are not both assigned t in all valuations.
- $A\&D$ (Constant false): Just like $C\&D$.
- $D \to B$ (Constant true): Now we need to consider the structure of the valuation graph. For each inside valuation, we need to verify whether all its following valuations assign D the truth-value f or B the truth-value t. This could be finished by checking the valuation backwardly.
- $D \to (A \leftrightarrow C)$ (Constant true): just like the previous case, but we need to find the truth-value of $A \leftrightarrow C$ at first. In the given valuation graph, the truth-values of $A \leftrightarrow C$ on different valuations are as follows.

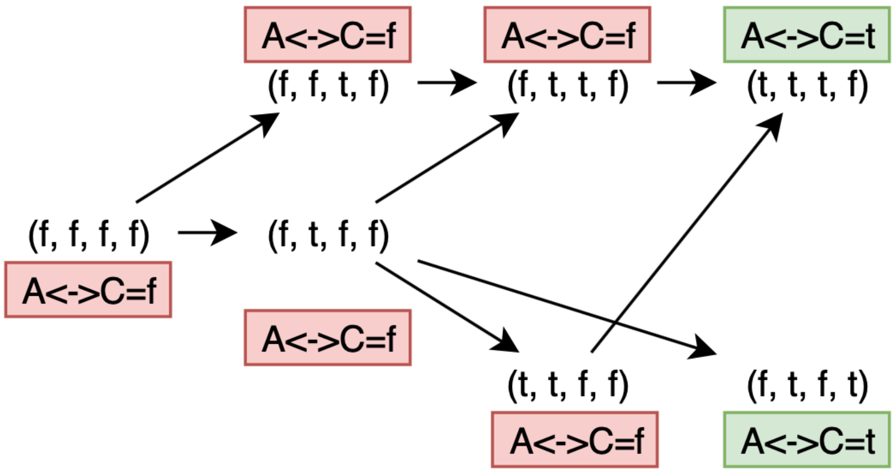

Figure 3: a valuation graph and the truth-value of $A \leftrightarrow C$ on different valuations

As an abbreviation, we can take $A \leftrightarrow C$ as a new proposition symbol E. If we want to verify $D \rightarrow E$ as constant true wff, we need to check whether for each valuation, all its following valuations assign D the truth-value f or E the truth-value t.

- $D\&(A \leftrightarrow B)$ (Constant false): like the previous case, we need to check the truth-value of $(A \leftrightarrow B)$ at first.

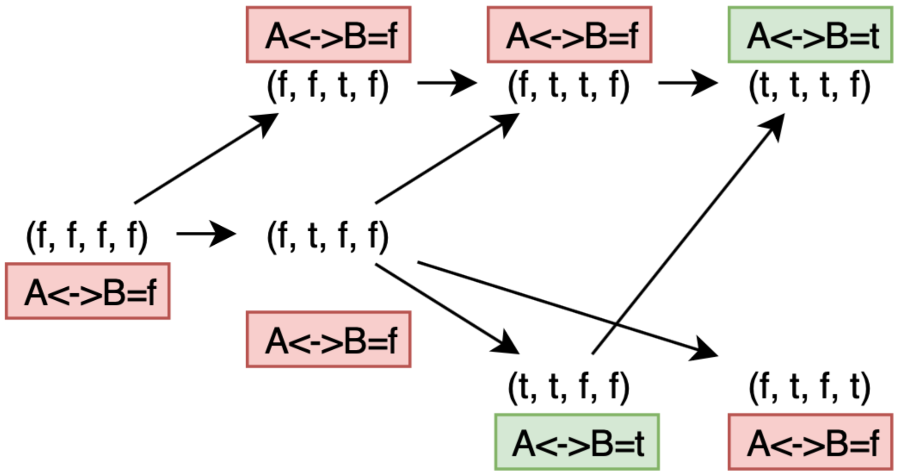

Figure 4: a valuation graph and the truth-value of $A \leftrightarrow C$ on different valuations

$(A \leftrightarrow B)$ could also be viewed as a new propositional symbol F as an abbreviation, and the following verification is relatively straightforward.

These wffs have relatively stable logical meanings, and therefore have interpretability far beyond the deep features found by traditional machine learning algorithms such as NNs. For example, in the constant wffs found above, $A\&D$ is a constant false wff, which means that for each data item whose lenses type is “No lenses,” A and D cannot be true at the same time, that is, age cannot be young in the same case that tear rate is not normal. Explaining wffs using $\rightarrow$ as the main connective is relatively complicated. For example, $D \rightarrow B$ is a constant true wff, which means that there will be no case where D is true and B is false on every valuation. That is, for each data item whose lenses type is “No lenses,” prescript cannot be myope in the same case that tear rate is not normal. Longer wff can also be analyzed in this way, but as the length of wff increases, the meaning will become more and more complicated.

After five runs, there are 11 source-0 wffs found, and after fifty runs, there are 75 source-0 wffs found. In machine learning tasks, different test samples are usually considered independent. Therefore, while discussing the testing model in the following discussion, the testing model contains only one valuation, and I will call the testing model the testing valuation. Using the wff found on different testing sets, regardless of the valuation graph of testing valuations, if the truth-value of the wff found is the same as the truth-value calculated on the testing valuation, this testing valuation has the supports of this source-0 wff. By focusing on how much support a testing valuation gets, we can know to what extent the testing valuation has the property of the training set.

For example, in the 75 source-0 wffs found, $C\&D$ is a constant false wff with 55 supports. For this testing valuation, the truth-value of $C\&D$ calculated is the same as the truth-value of $C\&D$ as a source-0 wff. Therefore, this testing valuation will have the 55 supports.

Table 23: a testing data item

#	*Age*	*Prescript*	*Astigmatic*	*Tear Rate*	*Lenses Type*
3	*young*	*myope*	*yes*	*reduced*	*no*

According to the 75 source-0 wffs found, I calculate the support of each testing valuation. The average supports of different test sets on different number of runs are as follows.

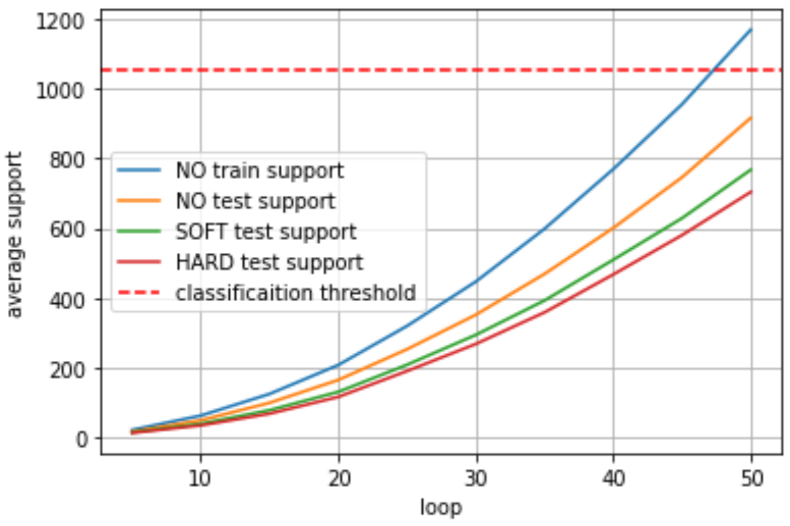

Figure 5: average support obtained by different datasets under different loops

The uppermost blue and orange curves show the average support of the testing/training set whose lenses type is classified as "No lenses" on different numbers of runs. Clearly, with more runs, the difference in supports among different lenses types is more and more apparent. The red straight-line could be given simply as a threshold to make a binary justification of whether a data item has the property of the training set.

CHAPTER 5

CREATING RULES FOR CLASSIFICATION

A mathematical logic rule can be obtained following the Algorithm I given the wffs found in the previous part.

$$\frac{\begin{array}{c} H_1 \vdash I_1 \\ \vdots \\ H_i \vdash I_i \end{array}}{H_{i+1} \vdash O}$$

Recall that the model corresponding to the training data according to the analogy between atomic models and datasets is V. In the above rule form, O is the atomic wff corresponding to the main concept. Since the truth-values of wffs used are all constant, each argument must be satisfied by each valuation in V. Therefore, the arguments in the premise must be V-valid, as is the argument in the conclusion, which means this rule takes V as a global model. According to the assumptions in Part 3, V consists of one model in all possibly-collected models W. The rules that take V as the global model are expected to take the largest model W_{cp} in W as the global model. If we want to judge whether a testing valuation/model from W has the same property as the training model V, we only need to judge whether the rules obtained also take the testing model as a global model.

Take the previous example on the Lenses dataset, the following table shows the training model V. Recall that the propositional symbol A stands for $Age = young$, propositional symbol B stands for $Prescript = hyper$, propositional symbol C stands for $Astigmatic = no$, and propositional symbol D stands for $Tear\ rate = normal$. Here in addition I will use a new propositional symbol E instead of $Lenses\ Type = no$.

Table 24: modified training data items from the Lenses dataset

#	A	B	C	D	E
5	t	t	t	f	t
7	t	t	f	f	t
9	f	f	t	f	t
11	f	f	f	f	t
13	f	t	t	f	t
15	f	t	f	f	t
16	f	t	f	t	t

As said, there are 5 constant wffs found, which are $C\&D$ (constant false), $A\&D$ (constant false), $D \rightarrow B$ (constant true), $D \rightarrow (A \leftrightarrow C)$ (constant true), $D\&(A \leftrightarrow B)$ (Constant false), of course, there is a constant atomic wff E. As a result, such a rule R_0 that takes V as the global model could be created following Algorithm I.

$$R_0: \frac{\sim(A\&D), \sim(C\&D) \vdash \sim(A\&D)}{\sim(A\&D), \sim(C\&D) \vdash E}$$

For a testing valuation (namely #3), if this valuation has the same property as the property of V, it should satisfy R_0.

Table 25: a modified testing data item

#	A	B	C	D	E
3	t	f	f	f	t

In this testing valuation, $\sim(A\&D)$ and $\sim(C\&D)$ are assigned t, so this valuation satisfies the argument in the premise of R_0. Since R_0 takes V as a global model, we expect the testing model (containing only #3) also takes R_0 as a global model. Since the premise of R_0 is also valid on the testing model, the argument in the result of R_0 is also satisfied by #3. And since $\sim(A\&D)$ and $\sim(C\&D)$ are both satisfied, E should also be satisfied, the lenses type of the testing valuation is "No lenses," which is consistent with this data item.

If a valuation (namely #4, in which F is a new propositional symbol, which means the lenses type is "Hard lenses.") whose lenses type is not "No lenses" is tested, we need to show whether this valuation does not take R_0 as a global rule.

Table 26: another modified testing data item

#	A	B	C	D	F
4	t	f	f	t	t

Now $\sim(A\&D)$ is not satisfied by this valuation, so the argument in the premise of R_0 is satisfied vacuously, and then the testing model containing #4 also satisfies R_0 vacuously. That is to say, #4 does take R_0 as a global model, but only vacuously. Therefore, it seems that we should consider the lenses type of #4 to be "No Lenses." However, such a judgment is made when the hypothesis of the argument is not satisfied, so unlike the case in #3, the truth-value of E is not given, so we can think that the rule does not make a judgment here, and this data was rejected. From this, we can see the status of the hypothesis of the rule in the method proposed. Even if R_0 is not suitable for judging #4 here, R_0 itself will not incorrectly give the truth-value of E.

Next, we introduce an example to show what problems arise when we mix the rules found in different dataset sharing the same attributes. Suppose there are two datasets sharing the same attributes, one dataset contains data items about red apples, and another contains data items about red bricks. If there are two shared attributes X and Y, in which X means $color = red$ and Y means $material = nonbiological$. Then in the dataset containing red apples, I may create the following rule R_1 (H is empty).

$$R_1: \frac{\vdash X}{\vdash brick}$$

And in the dataset containing red bricks, I may create the following rule R_2.

$$R_2: \frac{\vdash X}{\vdash apple}$$

When there is a testing valuation that needs to be classified, if X is satisfied when the hypothesis is empty, then according to R_1 and R_2, the two rules will both be valid on the testing model and give truth-values of $brick$ and $apple$. But we know the testing valuation could not be both a piece of brick and an apple, so the judgments made by R_1 and R_2 are conflicting. Still, we can use Y as the hypothesis to mitigate this problem, which yields R_1' and R_2'.

$$R_1': \frac{Y \vdash X}{Y \vdash brick} \quad R_2': \frac{\sim Y \vdash X}{\sim Y \vdash apple}$$

Come back to the example about the Lenses dataset, if we create a rule R_3 finding an appropriate hypothesis, the classification can be carried out successfully.

$$R_3: \frac{D \rightarrow (A \leftrightarrow C) \vdash \sim(A\&D)}{D \rightarrow (A \leftrightarrow C) \vdash E}$$

Currently, no matter the testing valuation is #3 or #4, $D \rightarrow (A \leftrightarrow C)$ is always satisfied. But on #3, $\sim(A\&D)$ is satisfied, which is not satisfied on #4, therefore, E is satisfied only on #3. That is to say, the lenses type of the data sample corresponding to #3 is "No Lenses" and the lenses type of the data sample corresponding to #3 is not "No Lenses."

In summary, to mix the rules found in different dataset sharing the same attributes, we cannot arbitrarily pick wffs found in part 4 to create rules following Algorithm I. This part will show a step-by-step method of using constant wffs to construct rules for classification in three situations.

5.1 – One-class Classification Problem In A Single Dataset

The one-class classification problem is the most basic situation. In this kind of problems, the training set only contains data items sharing the same main concept, and the purpose of the classification is to find a way to recognize whether the other data items are under this concept.

If we want to deal with this kind of problem using mathematical logic rules, we can consider all possibly-collected data items have the same class label, which means all the rules used for classification have the same result O in the rule form. Therefore, it is not necessary to consider O. When all valuations come from the same source, a valuation should not be rejected. Therefore, it is also unnecessary to consider the hypothesis.

Not considering the hypothesis and the result does not mean they could be assigned arbitrarily. This just says they are indifferent in making judgments. So, we only need to pick any number of constant wffs as conclusions to represent the rules.

But when there are n constant wffs, in such a way of finding rules, there are 2^n different rules that could be found. The number of the rules grows exponentially with the growth of n, so clearly, we cannot enumerate the rules and judge whether they are global rules one by one. And since there are multiple rules found, to test whether a testing valuation (namely v, when it is more appropriate to talk about its corresponding model, I will call it V) to what extent satisfies the property of the main concept of the training set, we need to find how many rules preserve validity on V.

However, if a rule preserves validity on V, regardless of the hypothesis and the result, v needs to satisfy all $I_i, i \in [1,2, \ldots, n]$ in the rule form. So, we can check which constant wff is satisfied at first. Only the wffs satisfied by all $I_i, i \in [1,2, \ldots, n]$ could be used to create rules preserve validity. Suppose the number of wffs satisfied is n_s, then the number of rules that preserve validity is 2^{n_s}, and n_s is positively correlated to 2^{n_s}. Therefore, to judge the number of V-valid rules, we can only consider the number of constant wffs satisfied instead. According to the average support on the training set (or the validation set), we can set a threshold that when the support got by a valuation does not meet the threshold, it is not classified as in this class label.

At the end of Part 4, I have mentioned the constant wffs found might not completely follow the definition, and the extent to which it meets the definition depends on its support. As a result, to count the number of V-valid rules, we consider the summation of the support of the constant wffs satisfied as an alternative.

5.2.1 – Multiple-class Classification Problem In A Single Dataset

When there are multiple class labels, we have to consider the result of the rule, but since there is only one dataset here, we still do not need to consider the hypothesis.
According to different class labels, data items in a dataset may be separated into different sub-datasets having different main concepts. Though different sub-datasets can find their constant wffs independently, we cannot guarantee that the constant wffs found in one sub-dataset will not be a constant wff with the same truth-value in another sub-dataset. For example, in the case used in Part 4 about the Lenses dataset, $C\&D$ is a constant false wff on the training model. After adding two more valuations (whose lenses type is "Hard Lenses"), $C\&D$ is still a constant false wff. No matter in a sub-dataset whose lenses type is "No Lenses" (namely V_{NO}) or a sub-dataset whose lenses type is "Hard Lenses" (namely V_{HARD}), $C\&D$ is always a constant false wff.

Table 27: adding more valuations to the training set

#	A	B	C	D	$Lenses\ Type$
5	t	t	t	f	no
7	t	t	f	f	no
9	f	f	t	f	no
11	f	f	f	f	no
13	f	t	t	f	no
15	f	t	f	f	no
16	f	t	f	t	no
4	t	f	f	t	$hard$
8	t	t	f	t	$hard$

If there are two rules R_3 and R_4, no matter a testing valuation's lenses type is "No Lenses" or "Hard Lenses," it satisfies the arguments in the premise of these two rules, and then conflicting results will be achieved. In this case, the hypothesis (namely H in the following discussion) of each argument is omitted, E represents that the lenses type is "No Lenses," and F represents that the lenses type is "Hard Lenses."

$$R_3 : \frac{H \vdash \sim(C\&D)}{H \vdash E} \quad R_4 : \frac{H \vdash \sim(C\&D)}{H \vdash F}$$

In contrast, $A\&D$ is a constant false wff on V_{NO}, but it is a constant true wff on V_{HARD}. If we replace $C\&D$ for $A\&D$, there are two new rules R_3' and R_4'.

$$R_3': \frac{H \vdash \sim(A\&D)}{H \vdash E} \quad R_4': \frac{H \vdash A\&D}{H \vdash F}$$

Now, when the lenses type of the testing valuation is “No Lenses,” it will satisfy R_3'. Or conversely, it will satisfy R_4'. A wff like $A\&D$ is constant in these two sub-datasets, but its truth-value on one dataset is the opposite of the truth-value on another sub-dataset. Therefore, when it is satisfied on one dataset, it is not able to be satisfied on another dataset, and we will call such a wff a decisive difference of a dataset.

(**Decisive difference of a dataset**) In different sub-datasets, if there is a wff that is constant in all sub-datasets, but its truth-value on one dataset V is the opposite with the truth-values on the other sub-datasets, we call this wff a decisive difference of V.

Therefore, to avoid the conflicts of rules found in different sub-datasets, constructing the rules using merely constant wffs in each sub-dataset respectively is not enough, we must use decisive differences of each sub-datasets instead. Still, though decisive differences are indeed constant wffs, it is much rarer, which makes it difficult to obtain through the screening of constant wffs found, especially when the number of constant wffs retained after each update is limited. So, in the following chapter, I will introduce a method of finding decisive differences.

5.2.2 – Finding Decisive Differences

All subsequent discussions will be based on finding decisive differences in two sub-datasets. It is not to say there are only two class labels, but for each class label c_i, all training data can be divided into data whose class label is c_i (namely the self-class) and data whose class label is not c_i (namely the other-class). Based on the truth-values of the constant wffs in the other-class, constant wffs could be further divided into three types, namely type-A constant wffs, type-B constant wffs, and type-C constant wffs.

Type-A wffs are constant wffs in both the self-class and the other-class, but their truth-values in the self-class and the other-class are the same. Or we can say type-A wffs are the common features of them. Type-B wffs are still constant wffs in both datasets, but their truth-values on the self-class are different from their truth-values in the other-class. So, we can say type-B wffs are the decisive differences of the self-class. For example, in the case of the Lenses dataset, $\sim(C\&D)$ is a type-A wff and $\sim(A\&D)$ is a type-B wff of the sub-dataset whose lenses type is "No Lenses." Type-C wffs are different, they are only constant wffs on the self-class, but they are variable wffs on the other-class.

What we want are just type-B wffs now. But as said, decisive differences are relatively rare. The first problem occurs when there are no type-B wffs in the constant wffs found at all. However, we can still use type-C wffs to generate some type-B wffs. One thing worthy of addressing is that while using type-C wffs to create type-B wffs, the created type-B wff consists of constant wffs, which is not allowed since that will bring dependent truth-value among constant wffs. But here we do not need to worry about it since there are only type-B wffs will be retained, all material type-C wffs will be removed from the set of at hand wffs.

To discuss the truth-value of the compound wff created by connecting two type-C wffs, there are the following abbreviations. While talking about the truth-values of type-C wffs, when the truth-value of the type-C wff is t, we call the wff C_t. In contrast, we call the wff C_f. Since it is necessary to check the truth-values of type-C wffs on other-classes, in the other-class, we will say the valuations that assign $true$ to the type-C wff consists of a set called C_{Ot}. Similarly, there is a set called C_{Of}. And we call the compound wff CC.

1. $C_{t1}\&C_{t2}$

   CC is a constant true wff in the self-class, if we want CC to be a type-B wff, then CC must be a constant false wff in the other-class. Since the two material wffs are connected by &, then the valuations assign CC false must be the union of ${C_{t1}}_{Of}$ and ${C_{t2}}_{Of}$. When this union contains all valuations in other-classes, we can say $C_{t1}\&C_{t2}$ is a type-B wff.

2. $C_t\&C_f$ or $C_f\&C_t$

   CC is a constant false wff in the self-class, if we want CC to be a type-B wff, then CC must be a constant true wff in the other-class. That means in all valuations in the other-class, the truth-values of C_t and C_f must both be t, which means the two material wffs are not type-C wffs at all. Therefore, in this case, CC could not be a type-B wff.

3. $C_{f1}\&C_{f2}$

   Same as the second case.

4. $C_{t1} \to C_{t2}$

   CC is a constant true wff in the self-class. According to $\|\to\|$, $v(A \to B) = t$ iff for all $v', v \leq v'$, $v'(A) = f$ or $v'(B) = t$. The contrapositive is $v(A \to B) = f$ iff for some $v', v \leq v'$, $v'(A) = t$ and $v'(B) = f$. If we want CC to be a type-B wff, then CC must be a constant false wff in the other-class, which means ${C_{t1}}_{Ot}$ and ${C_{t2}}_{Of}$ must have an intersection, and this intersection must have all valuations at the end of the valuation graph of the other-class, in which the end (namely v_e) means there are no other valuations v' such that $v_e \leq v'$.

5. $C_t \to C_f$

CC is a constant false wff in the self-class, if we want CC to be a type-B wff, then CC must be a constant true wff in other-classes. That is to say ${C_{t1}}_{Ot}$ and ${C_{t2}}_{Of}$ cannot have an intersection.

6. $C_f \rightarrow C_t$

   Same as the fourth case.

7. $C_{f1} \rightarrow C_{f2}$

   Same as the fourth case.

8. $C_{t1} \leftrightarrow C_{t2}$

   CC is a constant true wff in the self-class, if we want CC to be a type-B wff, then CC must be a constant false wff in the other-class. To put it another way, for all valuations at the end of the valuation graph of other-classes, they must assign different truth-values for C_{t1} and C_{t2}.

9. $C_t \leftrightarrow C_f$ or $C_f \leftrightarrow C_t$

   CC is a constant false wff in the self-class, if we want CC to be a type-B wff, then CC must be a constant true wff in other-classes. So, ${C_t}_{Ot}$ and ${C_f}_{Of}$ cannot have an intersection, and ${C_t}_{Of}$ and ${C_f}_{Ot}$ cannot have an intersection as well.

10. $C_{f1} \leftrightarrow C_{f2}$

    Same as the eighth case.

So far, we have found eight ways to use two type-C wffs to generate a type-B wffs.

(**B-1**) Suppose there are two type-C wffs in the self-class, both are constant true wffs, namely C_{t1} and C_{t2}. If the union of ${C_{t1}}_{Of}$ and ${C_{t2}}_{Of}$ contains all valuations in other-classes, $C_{t1} \& C_{t2}$ is a type-B wff.

(**B-2**) Suppose there are two type-C wffs in the self-class, both are constant true wffs, namely C_{t1} and C_{t2}. If the intersection of $C_{t1_{Ot}}$ and $C_{t2_{Of}}$ contains all valuations at the end in the valuation graph of other-classes, $C_{t1} \rightarrow C_{t2}$ is a type-B wff.

(**B-3**) Suppose there are two type-C wffs in the self-class, C_t is a constant true wff, and C_f is a constant false wff. If there is no intersection of $C_{t_{Ot}}$ and $C_{f_{Of}}$, $C_t \rightarrow C_f$ is a type-B wff.

(**B-4**) Suppose there are two type-C wffs in the self-class, C_t is a constant true wff, and C_f is a constant false wff. If there is no intersection of $C_{f_{Ot}}$ and $C_{t_{Of}}$, $C_f \rightarrow C_t$ is a type-B wff.

(**B-5**) Suppose there are two type-C wffs in the self-class, both are constant false wffs, namely C_{f1} and C_{f2}. If the union of $C_{f1_{Ot}}$ and $C_{f2_{Of}}$ contains all valuations in other-classes, $C_{f1} \rightarrow C_{f2}$ is a type-B wff.

(**B-6**) Suppose there are two type-C wffs in the self-class, both are constant true wffs, namely C_{t1} and C_{t2}. If the union of the intersection of $C_{t1_{Ot}}$ and $C_{t2_{Of}}$ and the intersection of $C_{t1_{Of}}$ and $C_{t2_{Ot}}$ contains all valuations at the end in the valuation graph of other-classes, $C_{t1} \leftrightarrow C_{t2}$ is a type-B wff.

(**B-7**) Suppose there are two type-C wffs in the self-class, C_t is a constant true wff, and C_f is a constant false wff. If there is no intersection of $C_{f_{Ot}}$ and $C_{t_{Of}}$, and there is no intersection of $C_{f_{Of}}$ and $C_{t_{Ot}}$, $C_f \leftrightarrow C_t$ and $C_t \leftrightarrow C_f$ are two type-B wffs.

(**B-8**) Suppose there are two type-C wffs in the self-class, both are constant false wffs, namely C_{f1} and C_{f2}. If the union of the intersection of $C_{f1_{Ot}}$ and $C_{f2_{Of}}$ and the intersection of $C_{f1_{Of}}$ and $C_{f2_{Ot}}$ contains all valuations at the end in the valuation graph of other-classes, $C_{f1} \leftrightarrow C_{f2}$ is a type-B wff.

For example, in the case of the Lenses dataset mentioned in Part 5.2.1, there are two class labels, and there are two corresponding models.

Table 28: a set of valuations whose lenses type is "NO"

#	A	B	C	D	$Lenses\ Type$
5	t	t	t	f	no
7	t	t	f	f	no
9	f	f	t	f	no
11	f	f	f	f	no
13	f	t	t	f	no
15	f	t	f	f	no
16	f	t	f	t	no

Table 29: a set of valuations whose lenses type is "HARD"

#	A	B	C	D	$Lenses\ Type$
4	t	f	f	t	$hard$
8	t	t	f	t	$hard$

In the first model, there are 5 constant wffs found, which are $C\&D$ (constant false), $A\&D$ (constant false), $D \rightarrow B$ (constant true), $D \rightarrow (A \leftrightarrow C)$ (constant true), and $D\&(A \leftrightarrow B)$ (constant false). If we take the first model whose lenses type is "No Lenses" as the self-class, the second model could be viewed as the other-class. Based on the valuation graph of the other class:

- $C\&D$ is a constant false wff in the other-class, so it is a type-A wff.

- $A\&D$ is a constant true wff in the other-class, so it is a type-B wff.
- $D \rightarrow B$ is a variable wff in the other-class, its truth-value on (t, f, f, t) is f and its truth-value on (t, t, f, t) is t, so it is a type-C wff.
- Discussing the truth-value of $D \rightarrow (A \leftrightarrow C)$ requires the truth-value of $A \leftrightarrow C$. In the other-class, $A \leftrightarrow C$ is a constant false wff, therefore, $D \rightarrow (A \leftrightarrow C)$ is a constant true wff, and so it is a type-B wff.
- Discussing the truth-value of $D\&(A \leftrightarrow B)$ requires the truth-value of $A \leftrightarrow B$. In the other-class, $A \leftrightarrow B$ is a variable wff, its truth-value on (t, f, f, t) is f and its truth-value on (t, t, f, t) is t, so it is a type-C wff, which also makes $D\&(A \leftrightarrow B)$ a type-C wff.

$$(t, f, f, t) \rightarrow (t, t, f, t)$$

According to B-3, using $D \rightarrow B$ and $D\&(A \leftrightarrow B)$ could create a new type-B wff, $(D \rightarrow B) \rightarrow \big(D\&(A \leftrightarrow B)\big)$. Since in the second model, the truth-values of $D \rightarrow B$ and $D\&(A \leftrightarrow B)$ are both t on (t, t, f, t) and f on (t, f, f, t), so the truth-value of $(D \rightarrow B) \rightarrow \big(D\&(A \leftrightarrow B)\big)$ is t on both valuations, and then it is a constant true wff in the other-class. But on the first model, it is a constant false wff, so $(D \rightarrow B) \rightarrow \big(D\&(A \leftrightarrow B)\big)$ is a type-B wff.

Though the above methods could be used to generate some type-B wffs, after each update, the number of type-B wffs may still be relatively small sometimes. It seems that we could use a type-A (or type-C) wff and a type-B wff to generate a new type-B wff. But after creating this wff, only the type-A (or type-C) wff is removed, if the remaining type-B wff is the source of the compound wff, the truth-values of these two wffs will be dependent. After analyzing all possible cases, the remaining type-B wff is always the source of the compound wff, so using type-A (or type-C) wffs and type-B wffs to generate a new wff are forbidden. For detailed analysis, please check the appendix.

What's more, the above methods only consider generating type-B wffs using two material wffs, therefore, after one update, it is still possible that there are no type-B wffs that could be generated at all. In this case, we need to take the input as an exception, and withdraw the input of the updating valuation, return to the state before the input, and then make another updating valuation.

5.3 – Construct Rules For Multiple-class Classification Under A Single Dataset

Suppose in a multiple-class classification problem, there are n class labels, which are $X_1, X_2, \ldots, X_n$. Their corresponding sets of decisive differences are $B_1, B_2, \ldots, B_n$. For each class label, pick any number of decisive differences as the wff in the result to create corresponding arguments, a piece of the rule used for classification for one class label could be represented. To determine which property the testing model has, we need to compare how many rules there are that preserve validity on the testing model. As when dealing with one-class classification problems, the wffs that can be satisfied by the testing valuation should be found first. Only the rules created by these wffs can preserve validity on the testing model. The more wffs the testing valuation satisfies, the more the rules preserve validity. According to the support of decisive differences, the more support the testing valuation gest from a set of decisive differences of a class label, the more the testing valuation has the property of the corresponding main concept.

Note that in the one-class classification problems, I will set a threshold for each classifier. If the support obtained by a testing valuation is below the threshold, the valuation will not be classified in this class. But in the multiple-class classification problems, the decisive differences of each class label not only describe their own features but also the difference among class labels. There is no need to set a minimum threshold to reject the testing valuation.

5.4 – Different Classification Problems Using The Same Attributes

When there are multiple classification problems, similar decisive differences might also be found, which will typically cause the rules that make correct judgments in each classification problem may conflict or produce errors when put together. This is just like the example of ResNet-18 mentioned in the introduction. To mitigate this problem, we need to consider the hypotheses of arguments in the rule form.

The purpose of defining the hypothesis is to make sure all testing valuations are distributed to the correct classifier, which seems a bit like a classification problem of classifiers. But if we think about it this way, this is equivalent to another multi-classification problem similar to Part 5.2. However, we must consider that multiple classification problems are not trained at the same time, and if according to the method in Part 5.2, any testing valuation must belong to one and only one classifier, so that no testing valuations will be rejected, this is also contrary to the original intention of this method proposed. Therefore, compared with the method proposed in Part 5.2, this is closer to the problem raised in Part 5.1. Each classifier is trained separately. In training, in addition to distinguishing different inside-class labels, it is also necessary to look for constant wffs on the entire training data. In this way, multiple classifiers can be accumulated. When there is a testing valuation, according to the constant wff and threshold of each classifier, it is judged which classifier should be used for this valuation or whether the existing classifier could be used to make judgments.

Of course, although in Part 4.5, using constant wffs has certain judgment ability, with the increase of the number of classifiers, only using constant wffs may not be sufficient to correctly judge whether a testing valuation should be accepted, or how to forward a testing valuation to an inside classifier. Therefore, when there is an error in the judgment between two classifiers, it means that their constant wffs can no longer distinguish between them. At this time, we need to use the method proposed in Part 5.2 to use decisive differences of the two classifiers to replace the constant wffs. In this way, the two classifiers are completely separated. Using the above methods, the knowledge in each data processed can be used together without conflict when trained separately.

CHAPTER 6

EXPERIMENTS AND RESULTS

In previous parts, I have discussed the basic methods and use some toy examples to explain them. This part will use the methods proposed to solve some real classification problems.

6.1.1 – One-class Classification In A Single Dataset, Small-scale Experiment

The experiment follows the experiments in Part 4.5. As a reminder, this is carried out on the Lenses dataset, 7 data samples whose lenses types are "No Lenses" are used for training. 2 data samples whose lenses types are "No Lenses" and 2 data samples whose lenses types are "Soft Lenses" are used for testing. The size of the valuation graph is 7, and after each update, only 5 constant wffs are retained. After inputting all training data samples, the constant wffs retained will be forwarded to a final constant wff set S_0 which is used for classification. The input data samples will be shuffled and input again 1 to 50 times. The curve of the average support obtained from S_0 and the number of runs is as follows.

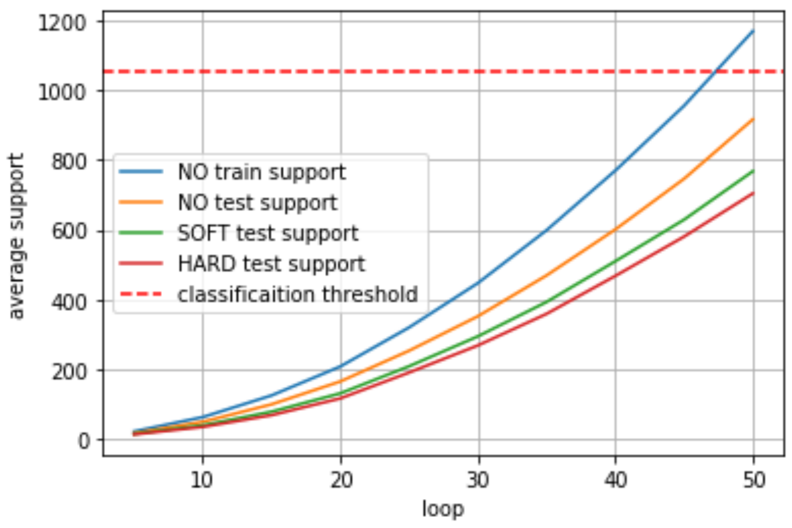

Figure 6: average support obtained by different datasets under different loops, without threshold

We can see, along with the increase of repeated input, the size of S_0 also increases, and the support obtained by testing data samples having different lenses types also increases. But since S_0 is found in the dataset whose lenses type is "No Lenses," it always gives more support to the testing data samples with the same lenses type.

According to the average supports $\bar{s}$ obtained by the testing data sample whose lenses type is "No Lenses," we can give a hyperparameter $\lambda, \lambda \in [0,1]$ and let $\lambda \cdot \bar{s}$ be the threshold. For each testing valuation, if the support obtained by the valuation is below the threshold, we will say this valuation is classified as not having the main concept as the corresponding training data, namely accepted. Otherwise, it has the main concept, namely rejected. When the testing valuation's lenses type is "No Lenses," if it is rejected, we also say it is misclassified. After repeated inputting 50 times, according to different values of λ, the curves of classification accuracy for testing data having different lenses types are as follows.

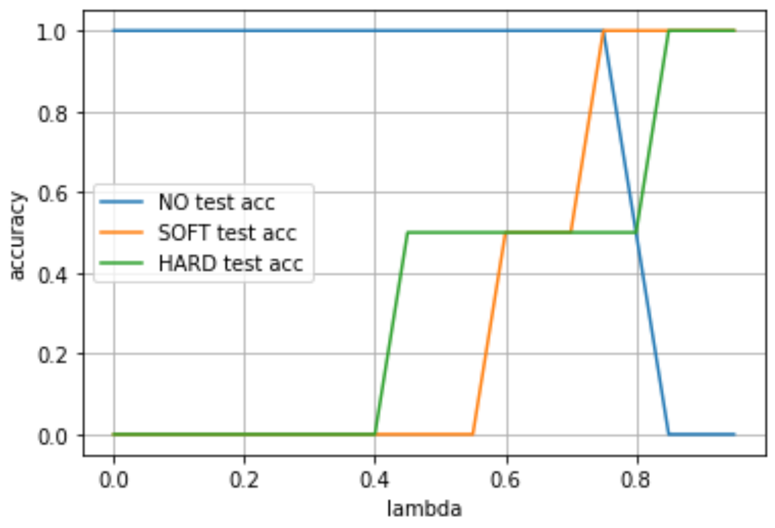

Figure 7: one-class classification accuracy under different values of λ

As the increasing of λ, the classification accuracy for the testing valuations whose lenses types are not "No Lenses" increases, and the classification accuracy for the testing valuations whose lenses types are not "No Lenses" decreases. When $\lambda \in [0.6,1]$, we can see all three testing datasets could get a great accuracy.

6.1.2 – One-class Classification In A Single Dataset, Large-scale Experiment

In order to verify that the method proposed can still have a good performance when the amount of data or the length of the data is large, I choose to conduct experiments on the MNIST dataset [13]. I reshape each MNIST data sample from a 28×28 tensor to a 14×14 tensor and reshape it into a 196-bit vector, then use 128 as the threshold of the gray degree value to convert this continuous vector into a binary vector. After that, although it can be said that each bit represents a binary attribute, obviously this does not have a concrete logical meaning, but due to considerations of size this is the only method that could be used to process MNIST data samples. Since this experiment is about one-class classification problems, only the MNIST data samples with the number 1 participate in the training, and the data samples with the numbers 1, 2, and 3 participate in the testing. If after inputting a certain data sample, the valuation graph reaches the maximum size or is still the maximum size, 5 decisive differences retained will be forward to the final set S_0 of decisive differences.

Different from the small-scale experiment, there are many MNIST data samples, so that we can compare the method proposed with the classical machine learning approaches solving one-class classification problems, which is the one-class support vector machine (OCSVM [16]). Using different amounts of training samples, OCSVM gets the following results with 600 testing samples (600 samples for data samples with the numbers 1, 2, and 3 each).

Table 30: experiment results of OCSVM on the MNIST dataset

	Training = 50	Training = 100	Training = 500	Training = 2,000
MNIST #1	0/600	7/539	203/397	361/239
MNIST #2	0/600	0/600	0/600	0/600
MNIST #3	0/600	0/600	0/600	0/600

In the above table, take $361/239$ as an example, the number at the left of $/$ means the number of data samples classified as the data with the number 1, and the number at the right of $/$ means the number of data samples classified as the data with the other numbers. Here among 600 testing data items with the number 1, 361 of them are classified as data with number 1. Or put it another way, they are classified correctly. And 239 of them are classified as data with other numbers, which means they are classified incorrectly.

When there are 2,000 training samples, the accuracy of OCSVM just exceeds 50%. It says when there are not enough training samples, OCSVM is not a good choice. But we can also say that the accuracy of OCSVM is always 100% on the data with a number other than 1.

When the method proposed is used, the size of the valuation graph is set to 10, and 200 constant wffs are retained after each update. Using 50 data samples for training, and repeated inputting 50 times, the result is as follows.

Table 31: average support ratio obtained by different testing MNIST datasets

	MNIST #1	MNIST #2	MNIST #3
NW200 – NVG10 – T50/5	0.994050	0.921567	0.935867

Since the number of constant wffs retrained is relatively large, therefore, after multiple repeated inputting, the size of S_0 will be very large, so I use the length of wff as the first evaluation criterion (the shorter the better) and the support of wff as the second criterion (the larger the better) to find 200 constant wffs from S_0 for classification, which is called S_1.

The number in the table does not mean the classification accuracy, but for a testing valuation, the average support obtained from S_1 for the total support in S_1. It is clear that the testing data with the number 1 have larger support than the data with the other numbers.

When the threshold is set to 0.97, that is, when the proportion of the support obtained by a testing valuation exceeds 0.97, it is classified as the data with the number 1, otherwise it is classified as the data with the other numbers. The method proposed gets an accuracy of 98.5% on the testing data with the number 1, 85.5% on the data with the number 2, and 94.2% on the data with the number 3.

Compared with OCSVM, the performance of the method proposed is not as good on data samples with numbers 2 and 3, but when there are only 50 training samples, OCSVM does not work at all. So, the method proposed has a good performance when the training set is not small.

For a detailed discussion, I randomly pick 200 data samples with number 1, 2 and 3 respectively. The analysis of the 200 constant wffs used for classification is as follows.

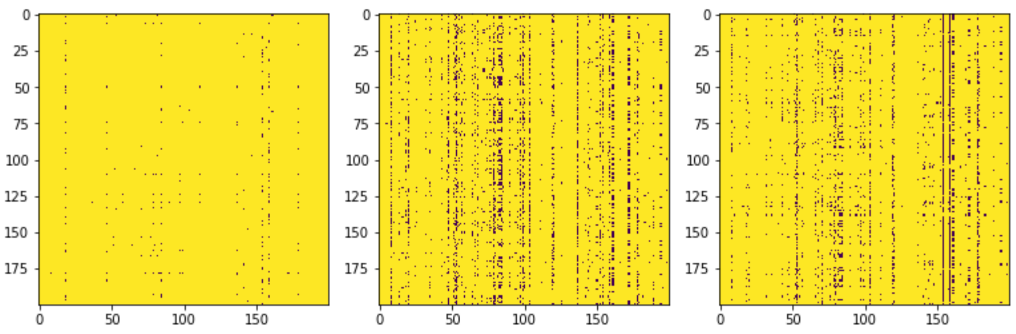

Figure 8: decisive differences are satisfied by different testing data items

The horizontal axis of each image represents different constant wffs, and the vertical axis represents different test data samples. In which the yellow pixels represent that a constant wff is satisfied by a test data sample, and purple pixels mean the opposite. There are few purple pixels in the testing data with the number 1, which also reflects that the constant wffs found fit the class label of the training set. But when the numbers are 2 and 3, in their testing sets, we can see some purple vertical lines, which means that there is a constant wff found that is almost not satisfied by all testing samples, which shows that such wff is the key feature used to distinguish different numbers, which are decisive differences as discussed.

6.2.1 – Multiple-class Classification In A Single Dataset, Small-scale Experiment

The experiment is carried out on the Lenses dataset, 7 data samples whose lenses types are "No Lenses" and 3 data samples whose lenses types are "Soft Lenses" are used for training. 2 data samples whose lenses types are “No Lenses” and 2 data samples whose lenses types are “Soft Lenses” are used for testing. Since there are two class labels, there are two classifiers. For the classifier corresponding to the data samples whose lenses type is “No lenses,” the size of the valuation graph is 7, and 5 constant wffs are retained after each update. For the classifier corresponding to the data samples whose lenses type us “Soft Lenses,” the size of the valuation graph is 2, and 5 constant wffs are retained after each update. After inputting all training data samples, the constant wffs retained will be forward to a final constant wff set which is used for testing, and the input data samples will be shuffled and input again 1 to 50 times. The decisive differences found will also be restored in a set for testing, for the first classifier for data whose lenses type is “No Lenses,” the set is called S_{NO}, and S_{SOFT} for data whose lenses type is “Soft Lenses.” The figure below shows the average support difference obtained from the two decisive difference sets for the testing data samples of different lenses types.

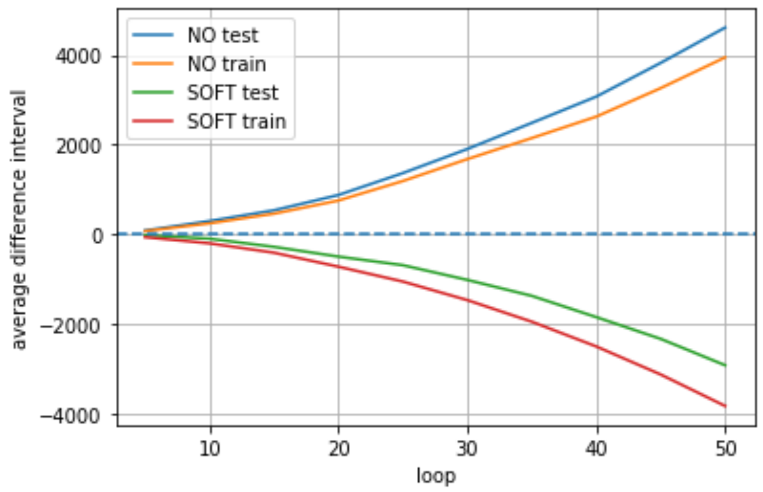

Figure 9: average support difference obtained from two sets of decisive differences by different testing datasets

The average support difference larger than 0 means the support obtained from S_{NO} is larger than the support obtained from S_{SOFT}, and vice versa. We can see the support difference obtained by the data whose lenses type is “No Lenses” is always greater than 0, and this situation becomes more apparent with the growth of the number of runs.

This shows that the decisive differences found are indeed key features in classification. If we use the amount of support obtained for classification, we can see that no matter how many times the loop is repeated, the accuracy of the two types of testing data is always 100%.

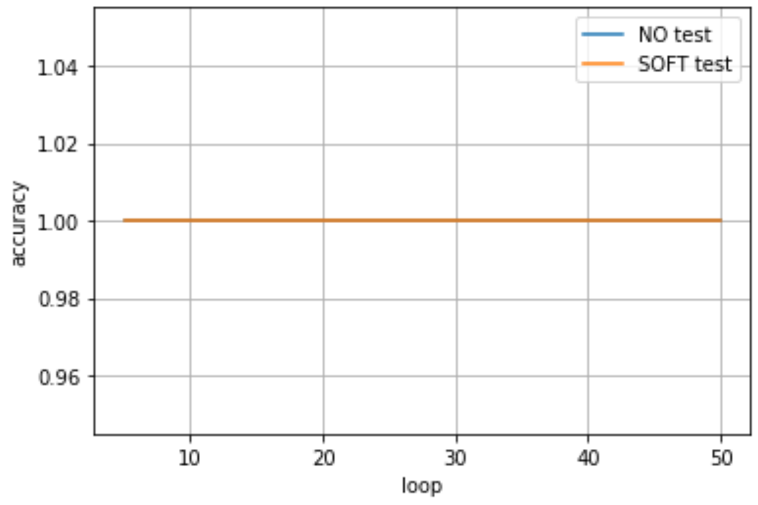

Figure 10: the accuracy of two testing datasets

6.2.2 – Multiple-class Classification In A Single Dataset, Large-scale Experiment

To verify that the decisive differences found can still have a good performance when the amount of data or the length of the data is large, I am going to carry the experiments on the MNIST dataset with the same data preprocessing methods as in Part 6.1.2. I will try to use the decisive differences found to classify data samples with numbers 1 and 2.

I use 40 number 2 data samples and 40 number 3 data samples for training, and test on 3,000 samples (3,000 samples of number 2 and 3,000 samples of number). While training, the size of the value graph is 10, and the number of wffs retained after each update is 5. If after inputting a certain data sample, the valuation graph reaches the maximum size or is still the maximum size, the 5 decisive differences retained after the input will be forward to the final set S_D of decisive differences. After inputting all data samples, change the input order and re-input 1 to 50 times. The following curves show the performance of S_D on different testing data under different repeated times. Among them, s_1 represents the number 2 training data, and s_2 represents the number 3 training data.

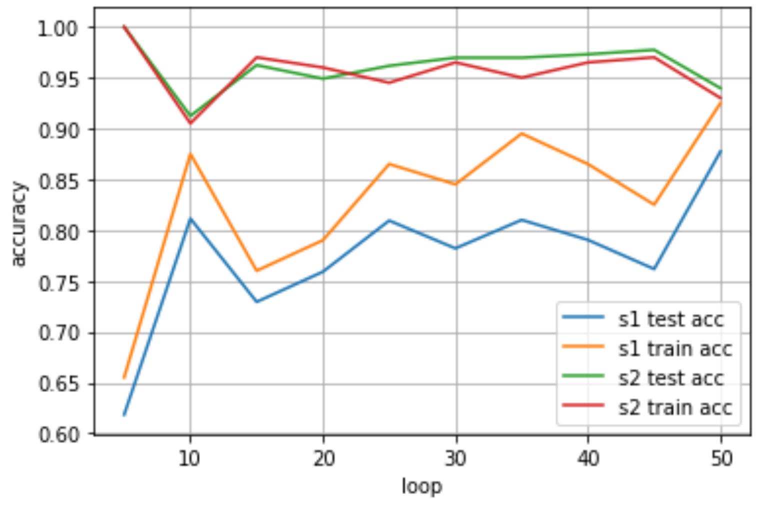

Figure 11: by different times of loops, the accuracy of the method proposed on different testing datasets

As the number of runs increases, no matter the testing data is with number 2 or with number 3, their accuracy on the testing set and training set will both have a tendency of increasing. To confirm the necessity of using S_D of decisive differences, I will show the following figure, comparing the accuracy of classification without using S_D but only the 5 decisive differences retained.

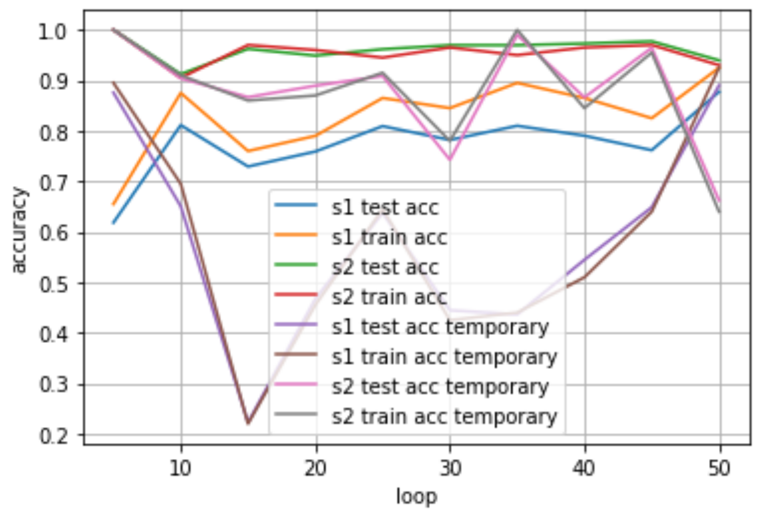

Figure 12: compared with not using the accumulated wff set, by different times of loops, the accuracy of the method proposed on different testing datasets

It can be seen that such curves do not have a tendency of increasing accuracy as the number of runs increases.

6.3 – Different Classification Problems Using Same Attributes

This experiment is still carried out on the Lenses dataset. Performing the experiment requires at least two different classification problems using the same attributes. The first classification problem is classifying data whose lenses type is “No Lenses” from data whose lenses type is “Soft Lenses.” And the second classification problem is classifying data whose lenses type is “No Lenses” from data whose lenses type is “Hard Lenses.”

The first classifier is trained with 7 samples whose lenses type is “No Lenses” and 2 samples whose lenses type is “Soft Lenses.” The second classifier is trained with 7 samples whose lenses type is “No Lenses” and 1 sample whose lenses type is “Hard Lenses.” Except for finding decisive differences in each classifier, constant wffs are found for each classification problem as well.

Since these two classifiers are trained independently, it is possible that the constant wffs found in each classifier are not enough to tell where a testing valuation should be moved to, or whether a testing valuation should be accepted. I use the data whose lenses type is not “No Lenses” in the two classifiers for testing. Ideally, the sample whose lenses type is “Hard Lenses” should be rejected by the first classifier, and the sample whose lenses type is “Soft Lenses” should be rejected by the second classifier. However, the experimental results show that only one of the two samples whose lenses type is “Hard Lenses” is correctly rejected, while the two samples whose lenses type is “Soft Lenses” are all incorrectly accepted.

Therefore, I use the method proposed in Part 5.2 to find decisive differences between samples whose lenses type is “Soft Lenses” and the samples whose lenses type is “Hard Lenses,” instead of using constant wffs. After that, the experiments showed that all samples are classified correctly as desired.

CHAPTER 7

CONCLUSION

Through the analysis of NN and ILP, this paper suggests that if an intelligent agent is to accumulate knowledge in the process of solving multiple problems, it must have two characteristics. First, since the problems to be solved are usually specific, the knowledge obtained by the agent may contain sub-knowledge. The sub-knowledge needs to be separated and understood by people, so they can be mixed in a meaningful way. Secondly, the sub-knowledge should also be restricted to where they were effective to avoid conflicts in combining with another sub-knowledge.

Based on the study of model theory, this paper puts forward the concept of global rule based on the concept of global model, finds rules in models that do not contain all valuations in a language, and then uses such rules to solve classification problems based on the analogy between atomic models & datasets and the analogy between rules & knowledge. Through small-scale experiments on the Lenses dataset and large-scale experiments on the MNIST dataset, the method proposed performs well even using only a small amount of data.

REFERENCES

[1] Muggleton, Stephen, and Luc De Raedt. "Inductive logic programming: Theory and methods." The Journal of Logic Programming 19 (1994): 629-679.

[2] Zeiler, Matthew D., and Rob Fergus. "Visualizing and understanding convolutional networks." European conference on computer vision. Springer, Cham, 2014.

[3] Bau, David, et al. "Network dissection: Quantifying interpretability of deep visual representations." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

[4] Garson, David G. "Interpreting neural network connection weights." (1991): 47-51.

[5] Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. "" Why should i trust you?" Explaining the predictions of any classifier." Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. 2016.

[6] Koh, Pang Wei, and Percy Liang. "Understanding black-box predictions via influence functions." International Conference on Machine Learning. PMLR, 2017.

[7] Advances in Neural Information Processing Systems 27, pages 3320-3328. Dec. 2014

[8] Hinton, Geoffrey, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network." arXiv preprint arXiv:1503.02531 (2015).

[9] Gou, Jianping, et al. "Knowledge distillation: A survey." International Journal of Computer Vision 129.6 (2021): 1789-1819.

[10] Cropper, Andrew, Alireza Tamaddoni-Nezhad, and Stephen H. Muggleton. "Meta-interpretive learning of data transformation programs." International Conference on Inductive Logic Programming. Springer, Cham, 2015.

[11] Dai, Wang-Zhou, Stephen H. Muggleton, and Zhi-Hua Zhou. "Logical vision: Meta-interpretive learning for simple geometrical concepts." ILP (Late Breaking Papers). 2015.

[12] Muggleton, Stephen, and Wray Buntine. "Machine invention of first-order predicates by inverting resolution." Machine Learning Proceedings 1988. Morgan Kaufmann, 1988. 339-352.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition." Proceedings of the IEEE, 86(11):2278-2324, November 1998.

[14] Garson, James W. What logics mean: from proof theory to model-theoretic semantics. Cambridge University Press, 2013

[15] Zhou, Zhi-Hua. "Rule extraction: Using neural networks or for neural networks?." Journal of Computer Science and Technology 19.2 (2004): 249-253.

[16] Heller, Katherine, et al. "One class support vector machines for detecting anomalous windows registry accesses." (2003).

[17] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[18] C. J. Merz and P. Murphy. Uci repos- itory of machine learning databases. In http://www.cs.uci.edu/m ˜learn/MLRepository.html, 1996.

APPENDICES

A. The Truth-value Dependence Analysis Of The Other Combinations

We have analyzed all four combinations with & as the main connective, and the case of $f \rightarrow f$. Here I am going to introduce the analysis of the other cases.

1. Constant true → constant true

   There are two different cases.

   a. When A becomes variable after an update and B remains the same.

      Since A is variable in the new valuation graph, at least one of the following cases must be satisfied.

      - There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to A.
      - There are two separated valuations assign A different truth-values.

      In the first case, by $\|\leq\|$, the valuation assigns A false must be the front valuation (namely v), and the back valuation (namely v') must assigns A true. Therefore, $v'(A \rightarrow B) = t$, so does $v(A \rightarrow B) = t$. $A \rightarrow B$ is still constant, and its truth-value does not flip, so A is not a source of $A \rightarrow B$ in this case.

      In the second case, clearly, we have both $t \rightarrow f$ and $f \rightarrow f$ and $\|\leq\|$ does not apply to them. Therefore, $A \rightarrow B$ is variable, so we say A is a source of $A \rightarrow B$ in this case.

To know whether A becoming variable will or will not make $A \rightarrow B$ become variable as well, we need to know the structure of the valuation graph, which is completely unpredictable. So, we can only say A is a para-source.

b. When B becomes variable and A remains the same.

   Similarly, there are two cases.

   - There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to B.
   - There are two separated valuations assign B different truth-values.

   In the first case, by $\|\leq\|$, the valuation assigns A false must be the front valuation (namely v), and the back valuation (namely v') must assigns A true. Therefore, $v'(A \rightarrow B) = t$, and $v(A \rightarrow B) = f$. $A \rightarrow B$ is variable, so we say B is a source of $A \rightarrow B$ in this case.

   In the second case, clearly, we have both $f \rightarrow t$ and $f \rightarrow f$ and $\|\leq\|$ does not apply to them. Therefore, $A \rightarrow B$ is variable, so we say B is a source of $A \rightarrow B$ in this case.

In summary, $A \rightarrow B$ is a source-x wff, in which A is its para-source and B is its source.

2. Constant true → constant false

   There are two different cases.

   a. When A becomes variable after an update and B remains the same.

      Since A is variable in the new valuation graph, at least one of the following cases must be satisfied.

      - There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to A.
      - There are two separated valuations assign A different truth-values.

In the first case, by $\|\leq\|$, the valuation assigns A false must be the front valuation (namely v), and the back valuation (namely v') must assigns A true. Therefore, $v'(A \to B) = f$, so does $v(A \to B) = f$. $A \to B$ is still constant, and its truth-value does not flip, so A is not a source of $A \to B$ in this case.

In the second case, clearly, we have both $t \to f$ and $f \to f$ and $\|\leq\|$ does not apply to them. Therefore, $A \to B$ is variable, so we say A is a source of $A \to B$ in this case.

In summary, A is just a para-source.

b. When B becomes variable and A remains the same.

Similarly, there are two cases.

- There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to B.
- There are two separated valuations assign B different truth-values.

In the first case, by $\|\leq\|$, the valuation assigns B false must be the front valuation (namely v), and the back valuation (namely v') must assigns A true. Therefore, $v'(A \to B) = t$, so does $v(A \to B) = f$. $A \to B$ is variable, so we say B is a source of $A \to B$ in this case.

In the second case, clearly, we have both $f \to t$ and $f \to f$ and $\|\leq\|$ does not apply to them. Therefore, $A \to B$ is variable, so we say B is a source of $A \to B$ in this case.

In summary, $A \to B$ is a source-x wff, in which A is its para-source and B is its source.

3. Constant false → constant true

There are two different cases.

a. When A becomes variable after an update and B remains the same.

Since A is variable in the new valuation graph, at least one of the following cases must be satisfied.

- There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to A.
- There are two separated valuations assign A different truth-values.

In the first case, by $\|\leq\|$, the valuation assigns A false must be the front valuation (namely v), and the back valuation (namely v') must assigns A true. Therefore, $v'(A \to B) = t$, so does $v(A \to B) = t$. $A \to B$ is still constant, and its truth-value does not flip, so A is not a source of $A \to B$ in this case.

In the second case, clearly, we have both $t \to f$ and $f \to f$ and $\|\leq\|$ does not apply to them. Therefore, $A \to B$ is variable, so we say A is a source of $A \to B$ in this case.

In summary, A is just a para-source of $A \leftrightarrow B$.

b. When B becomes variable and A remains the same.

Similarly, there are two cases.

- There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to B.
- There are two separated valuations assign B different truth-values.

In the first case, by $\|\leq\|$, the valuation which assigns B false must be the front valuation (namely v), and the back valuation (namely v') must assign A true. Therefore, $v'(A \to B) = t$, so $v(A \to B) = t$. $A \to B$ is still constant, and its truth-value does not flip, so B becoming variable will not make $A \to B$ become variable or flip its truth-value in this case.

In the second case, clearly, we have both $f \to t$ and $f \to f$ and $\|\leq\|$ does not apply to them. Therefore, $A \to B$ is variable, so we say B is a source of $A \to B$ in this case.

In summary, $A \to B$ is a source-x wff, in which A and B are two of its para-sources.

4. Constant true $\leftrightarrow$ constant true

   There are two different cases.

   a. When A becomes variable after an update and B remains the same.

      Since A is variable in the new valuation graph, at least one of the following cases must be satisfied.

      - There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to A.
      - There are two separated valuations which assign A different truth-values.

      In the first case, by $\|\leq\|$, the valuation assigns A false must be the front valuation (namely v), and the back valuation (namely v') must assign A true. Therefore, $v'(A \leftrightarrow B) = t$, so $v(A \rightarrow B) = f$. $A \rightarrow B$ is variable, so A is a source of $A \rightarrow B$ in this case.

      In the second case, clearly, we have both $t \rightarrow f$ and $f \rightarrow f$ and $\|\leq\|$ does not apply to them. Therefore, $A \rightarrow B$ is variable, so we say A is a source of $A \rightarrow B$ in this case.

      In summary, A is a source of $A \leftrightarrow B$.

   b. When B becomes variable and A remains the same.

      Similarly, there are two cases.

      - There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to B.
      - There are two separated valuations assign B different truth-values.

      In the first case, by $\|\leq\|$, the valuation assigns B false must be the front valuation (namely v), and the back valuation (namely v') must assign A true. Therefore, $v'(A \rightarrow B) = t$, so $v(A \rightarrow B) = f$. $A \rightarrow B$ is variable, so B is also a source of $A \rightarrow B$ in this case.

In the second case, clearly, we have both $f \to t$ and $f \to f$ and $\|\leq\|$ does not apply to them. Therefore, $A \to B$ is variable, so we say B is a source of $A \to B$ in this case.

In summary, $A \leftrightarrow B$ is a source-2 wff. The case of constant false $\leftrightarrow$ constant false is similar, so it is omitted.

5. Constant true $\leftrightarrow$ constant false

   There are two different cases.

   a. When A becomes variable after an update and B remains the same.

      Since A is variable in the new valuation graph, at least one of the following cases must be satisfied.

      - There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to A.
      - There are two separated valuations assign A different truth-values.

      In the first case, by $\|\leq\|$, the valuation assigns A false must be the front valuation (namely v), and the back valuation (namely v') must assign A true. Therefore, $v'(A \leftrightarrow B) = f$, so $v(A \to B) = f$. $A \to B$ is constant and its truth-value remains, so A is not a source of $A \to B$ in this case.

      In the second case, clearly, we have both $t \to f$ and $f \to f$ and $\|\leq\|$ does not apply to them. Therefore, $A \to B$ is variable, so we say A is a source of $A \to B$ in this case.

      In summary, we do not know whether A is a source of $A \leftrightarrow B$.

   b. When B becomes variable and A remains the same.

      Similarly, there are two cases.

      - There is a pair of partial valuation neighbors, in which two inside valuations assign different truth-values to B.

- There are two separated valuations assign B different truth-values.

  In the first case, by $\|\leq\|$, the valuation assigns B false must be the front valuation (namely v), and the back valuation (namely v') must assign A true. Therefore, $v'(A \to B) = t$, so $v(A \to B) = f$. $A \to B$ is variable, so B is also a source of $A \to B$ in this case.

  In the second case, clearly, we have both $f \to t$ and $f \to f$ and $\|\leq\|$ does not apply to them. Therefore, $A \to B$ is variable, so we say B is a source of $A \to B$ in this case.

In summary, $A \leftrightarrow B$ is a source-x wff, in which A is its para-source and B is its source. The case of constant false ↔ constant true is similar, so it is omitted.

As the result, we have the following table of compound wffs.

- Source-0* wff: constant false & constant false.
- Source-1 wffs: constant true & constant false, constant false & constant true. (In which the source is marked in red.)
- Source-2 wffs: constant true & constant true, constant true ↔ constant false, constant false ↔ constant false.
- Source-x wffs: constant true → constant true, constant true → constant false, constant false → constant true, constant false → constant false, constant true ↔ constant false, constant false ↔ constant true. (In which the source is marked in red, and the para-source is marked in orange.)

B. The Lenses Dataset

There are only 16 data items in this dataset, but each attribute has a concrete logical meaning.

Table 32: the whole Lenses dataset

#	*Age*	*Prescript*	*Astigmatic*	*Tear Rate*	*Lenses Type*
1	*young*	*myope*	*no*	*reduced*	*no*
2	*young*	*myope*	*no*	*normal*	*soft*
3	*young*	*myope*	*yes*	*reduced*	*no*
4	*young*	*myope*	*yes*	*normal*	*hard*
5	*young*	*hyper*	*no*	*reduced*	*no*
6	*young*	*hyper*	*no*	*normal*	*soft*
7	*young*	*hyper*	*yes*	*reduced*	*no*
8	*young*	*hyper*	*yes*	*normal*	*hard*
9	*pre*	*myope*	*no*	*reduced*	*no*
10	*pre*	*myope*	*no*	*normal*	*soft*
11	*pre*	*myope*	*yes*	*reduced*	*no*
12	*pre*	*myope*	*yes*	*normal*	*hard*
13	*pre*	*hyper*	*no*	*reduced*	*no*
14	*pre*	*hyper*	*no*	*normal*	*soft*
15	*pre*	*hyper*	*yes*	*reduced*	*no*
16	*pre*	*hyper*	*yes*	*normal*	*no*

C. The Lenses Dataset (Atomic Model)

As an abbreviation, I have changed the name of each attribute to atomic wffs. In which, A means age=young, B means prescript=hyper, C means astigmatic=no, D means tear rate=normal。

Table 33: the whole Lenses dataset (atomic model)

#	A	B	C	D	$Lenses\ Type$
1	t	f	t	f	no
2	t	f	t	t	$soft$
3	t	f	f	f	no
4	t	f	f	t	$hard$
5	t	t	t	f	no
6	t	t	t	t	$soft$
7	t	t	f	f	no
8	t	t	f	t	$hard$
9	f	f	t	f	no
10	f	f	t	t	$soft$
11	f	f	f	f	no
12	f	f	f	t	$hard$
13	f	t	t	f	no
14	f	t	t	t	$soft$
15	f	t	f	f	no
16	f	t	f	t	no

D. Creating Type-B Wffs Using Type-A Wffs And Type-B Wffs

As an abbreviation, each type-A wff is called A, when it is a constant true wff in the self-class, we call it A_t. In contrast, we call it A_f. Similarly, we can have B_t and B_f. The compound wff is called AB.

1. $A_t \& B_t$

   AB is a source-2 wff, so B_t as a source of AB will be exist in the constant wff set at the same time as AB, which is forbidden.

2. $A_t \& B_f$

   AB is a source-1 wff, so B_f as the source of AB will be exist in the constant wff set at the same time as AB, which is forbidden.

3. $A_f \& B_t$

   AB is a source-1 wff, and A_f as the source of AB will be removed from the constant wff set. But here, AB is a constant false wff in the self-class, and it is also a constant false wff in other-classes. Therefore, AB is not a type-B wff.

4. $A_f \& B_f$

   AB is a special source-1 wff, so B_f as a source of AB will be exist in the constant wff set at the same time as AB, which is forbidden.

5. $A_t \rightarrow B_t$

   AB is a source-1 wff, so B_f as the source of AB will be exist in the constant wff set at the same time as AB, which is forbidden.

6. $A_t \rightarrow B_f$

   According to the analyze in Part 3, AB is a source-2 wff under pessimistic estimation. Therefore, B_f as a source of AB will be exist in the constant wff set at the same time as AB, which is forbidden.

7. $A_f \rightarrow B_t$

   AB is a constant true wff in the self-class, and it is also a constant true wff in other-classes. So this case is also forbidden.

8. $A_f \rightarrow B_f$

   AB is a constant true wff in the self-class, and it is also a constant true wff in other-classes. So this case is also forbidden.

9. $A_t \leftrightarrow B_t$

   According to the analyze in Part 3, AB is a source-2 wff under pessimistic estimation. Therefore, B_t as a source of AB will be exist in the constant wff set at the same time as AB, which is forbidden.

10. $A_t \leftrightarrow B_f$

    According to the analyze in Part 3, AB is a source-2 wff under pessimistic estimation. Therefore, B_f as a source of AB will be exist in the constant wff set at the same time as AB, which is forbidden.

11. $A_f \leftrightarrow B_t$

    According to the analyze in Part 3, AB is a source-2 wff under pessimistic estimation. Therefore, B_t as a source of AB will be exist in the constant wff set at the same time as AB, which is forbidden.

12. $A_f \leftrightarrow B_f$

    According to the analyze in Part 3, AB is a source-2 wff under pessimistic estimation. Therefore, B_f as a source of AB will be exist in the constant wff set at the same time as AB, which is forbidden.

In summary, all cases using type-A wffs and type-B wffs to generate type-B wffs are forbidden.

E. Creating Type-B Wffs Using Type-C Wffs And Type-B Wffs

As an abbreviation, each type-C wff is called C, when it is a constant true wff in the self-class, we call it C_t. In contrast, we call it C_f. Similarly, we can have B_t and B_f. The compound wff is called CB.

1. $C_t\&B_t$

   CB is a constant true wff in the self-class, but in other-classes, B_t becomes a constant false wff, though C_t is a variable wff in other-classes, CB is a constant false wff in other-classes, so it is also a type-B wff. But CB is a source-2 wff, B_t as a source of CB will be exist in the constant wff set at the same time as CB, which is forbidden

2. $C_t\&B_f$

   CB is a source-1 wff, and B_t as the source of CB will be exist in the constant wff set at the same time as CB, which is forbidden.

3. $C_f\&B_t$

   CB is a constant false wff in the self-class, but it is also a constant false wff in other-classes, so CB is not a type-B wff.

4. $C_f\&B_f$

   CB is a special source-1 wff, and B_f as a source of CB will be exist in the constant wff set at the same time as CB, which is forbidden.

5. $C_t \rightarrow B_t$

   CB is a source-1 wff, and B_t as the source of CB will be exist in the constant wff set at the same time as CB, which is forbidden.

6. $C_t \to B_f$

According to the analyze in Part 3, CB is a source-2 wff under pessimistic estimation. B_f as the source of CB will be exist in the constant wff set at the same time as CB, which is forbidden

7. $C_f \to B_t$

CB is a constant true wff in the self-class, but it is also a constant true wff in other-classes, so CB is not a type-B wff.

8. $C_f \to B_f$

CB is a constant true wff in the self-class, but it is also a constant true wff in other-classes, so CB is not a type-B wff.

9. $C_t \leftrightarrow B_t$

According to the analyze in Part 3, CB is a source-2 wff under pessimistic estimation. Therefore, B_t as a source of CB will be exist in the constant wff set at the same time as CB, which is forbidden.

10. $C_t \leftrightarrow B_f$

According to the analyze in Part 3, CB is a source-2 wff under pessimistic estimation. Therefore, B_f as a source of CB will be exist in the constant wff set at the same time as CB, which is forbidden.

11. $C_f \leftrightarrow B_t$

According to the analyze in Part 3, CB is a source-2 wff under pessimistic estimation. Therefore, B_t as a source of CB will be exist in the constant wff set at the same time as CB, which is forbidden.

12. $C_f \leftrightarrow B_f$

According to the analyze in Part 3, CB is a source-2 wff under pessimistic estimation. Therefore, B_f as a source of CB will be exist in the constant wff set at the same time as CB, which is forbidden.

In summary, all cases using type-C wffs and type-B wffs to generate type-B wffs are forbidden.

F. Derivation In Natural Deduction

$$\dfrac{\dfrac{\dfrac{[p]^2 \quad [\neg p]^1}{\bot}\to E}{p}\bot_1}{p \to p}\to I_2$$

$$\dfrac{\dfrac{\dfrac{[q]^2 \quad [\neg q]^1}{\bot}\to E}{q}\bot_1}{q \to q}\to I_2$$

$$\dfrac{\dfrac{\dfrac{\dfrac{[p]^2 \quad [\neg p]^1}{\bot}\to E}{p}\bot_1}{p \to p}\to I_2 \qquad \dfrac{\dfrac{\dfrac{[q]^2 \quad [\neg q]^1}{\bot}\to E}{q}\bot_1}{q \to q}\to I_2}{(p \to p)\&(q\&q)}\&I$$