Sufficient Reasons for Classifier Decisions in the Presence of Domain Constraints

Niku Gorji, Sasha Rubin

School of Computer Science, The University of Sydney, Australia
niku.gorji@sydney.edu.au, sasha.rubin@sydney.edu.au

Abstract

Recent work has unveiled a theory for reasoning about the decisions made by binary classifiers: a classifier describes a Boolean function, and the reasons behind an instance being classified as positive are the prime-implicants of the function that are satisfied by the instance. One drawback of these works is that they do not explicitly treat scenarios where the underlying data is known to be constrained, e.g., certain combinations of features may not exist, may not be observable, or may be required to be disregarded. We propose a more general theory, also based on prime-implicants, tailored to taking constraints into account. The main idea is to view classifiers as describing partial Boolean functions that are undefined on instances that do not satisfy the constraints. We prove that this simple idea results in more parsimonious reasons. That is, not taking constraints into account (e.g., ignoring, or taking them as negative instances) results in reasons that are subsumed by reasons that do take constraints into account. We illustrate this improved succinctness on synthetic classifiers and classifiers learnt from real data.

Introduction

A recent line of enquiry for providing reasons for classifier decisions is on supplying principled reasons for individual instances with formal guarantees of subset- or cardinality-minimality (Shih, Choi, and Darwiche 2018; Darwiche and Hirth 2020; Ignatiev, Narodytska, and Marques-Silva 2019a). Contrary to the more scalable heuristic approaches (Ribeiro, Singh, and Guestrin 2016; Lakkaraju et al. 2019; Iyer et al. 2018), these principled approaches guarantee the quality of produced reasons and therefore can serve to validate, benchmark, and potentially provide insights for improving the solutions of the heuristic approaches.

A noteworthy drawback of these methods however, is that they do not deal with reasoning in the presence of constraints or background knowledge.

Constraints may arise from the structure and interdependencies between features present in data (Darwiche 2020). As a simple example, consider a medical setting in which some combinations of drugs are never prescribed together and thus will not appear in any dataset: if we know that drug A and drug B are never prescribed together (i.e., the constraint), then a reason of the form "drug A was prescribed and drug B was not prescribed" is overly redundant; however, the current methods in the state of the art produce such overly complicated reasons. We argue that in this situation, it is more parsimonious to supply the reason "drug A was prescribed". Although it might be obvious how to process reasons to take such simple constraints into account, it is by no means obvious how to handle semantic constraints represented by arbitrarily complex Boolean formulas. It is the purpose of this work to provide a simple and elegant approach for doing this. More precisely, we address the problem of how to incorporate background knowledge, specifically input (domain) constraints, into supplying reasons behind individual decisions of classifiers.

Our contributions are as follows:

1. We provide a crisp formalisation of reasons for classifier decisions that takes constraints into account, resulting in reasons that are as least (and sometimes more) parsimonious, i.e., more general and more succinct, than not taking constraints into account. The central insight, both simple and powerful, is to treat a classifier as a partial function that is not defined on input instances that do not satisfy the constraint, and then to use the classic definition of prime-implicant on partial functions (Coudert 1994) as the instantiation of the word "reason". This immediately and naturally generalises the state of the art from the unconstrained setting to the constrained setting.
2. We provide a simple reduction of the computational problem of finding all reasons of a classifier's decision for a given instance in the presence of constraints to the same problem in the unconstrained setting. This allows one to reuse existing algorithms and tools from the unconstrained setting. The idea is that if the constraint is given by the Boolean formula κ, and the decision-function by φ, then reasons of φ-decisions that take κ into account are exactly the reasons of $(\kappa \rightarrow \varphi)$-decisions.[1] We prove that all other variations, including the seemingly natural variation $(\kappa \wedge \varphi)$, provide no more, and sometimes less, parsimonious reasons.
3. We show, both theoretically and empirically on syn-

[1] This simple reduction can be done in linear-time for formulas represented as strings or parse-trees, and in polynomial-time for formulas represented as OBDD$_<$ (Darwiche and Marquis 2002).

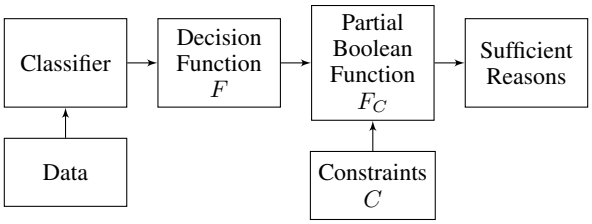

Figure 1: Workflow

thetic classifiers and classifiers learnt from data, that approaches that ignore constraints may supply reasons that are unnecessarily long since they redundantly encode knowledge already described in the constraints.

The workflow is illustrated in Figure 1. A classifier, typically learnt from data, is transformed into a decision function F. Domain constraints C are taken into account to produce a partial Boolean function F_C, which is used to compute sufficient reasons for the classifier's decisions.

Preliminaries

We begin by recalling just enough logical background to be able to explain our theory.

Boolean logic. Let $\mathbf{X} = \{X_1, X_2, \cdots, X_n\}$ be a set of n Boolean variables (aka *features*). The set of *Boolean formulas* is generated from $\mathbf{X}$, the constants $\top$ (true) and $\bot$ (false), and the logical operations $\wedge$ (conjunction),$\vee$ (disjunction),$\neg$ (negation), $\rightarrow$ (conditional) and $\leftrightarrow$ (bi-conditional). Variables X and their negations $\neg X$ are called *literals*. A *term* t is a conjunction of literals; the empty-conjunction is also denoted $\top$. The $size$ of a term t is the number of literals that occur in it. An *instance (over $\mathbf{X}$)* is an element of $\{0,1\}^n$, and is denoted $\mathbf{x}$ (intuitively, it is an instantiation of the variables $\mathbf{X}$). An instance $\mathbf{x}$ *satisfies* a formula φ if φ evaluates to true when the variables in φ are assigned truth-values according to $\mathbf{x}$. The set of instances that satisfy the formula φ is denoted $[\varphi]$, and is called the set *represented* by φ, i.e., a set C of instances is *represented* by φ if $C = [\varphi]$. If $[\varphi] = [\psi]$ then we say that φ, ψ are *logically equivalent*, i.e., they mean the same thing. For terms s, t, we say that s *subsumes* t if $[t] \subseteq [s]$, i.e., if every instance that satisfies t also satisfies s. If $[t] \subset [s]$ then we say that s *properly subsumes* t; depending on the context, we also describe this by saying that s is *more general* or *more parsimonious* than t, or s is *more succinct* than t (note that s is smaller than t).

Partial Boolean functions, and prime-implicants. A *partial Boolean function F (over $\mathbf{X}$)* is a function $\{0,1\}^n \rightarrow \{0,1,*\}$. For $i \in \{0,1,*\}$ define F^i to be the set $F^{-1}(i)$. The instances in F^1, F^0, F^* are called, respectively, the *positive, negative, undefined* instances of F. If the set F^* is empty, then F is a *total Boolean function*. If $[\varphi] = F^1$ we say that the formula φ *represents* the total Boolean function F. A term t is an *implicant of* F if $[t] \subseteq F^1 \cup F^*$; it is *prime* if no other implicant of F subsumes t. Intuitively, t is prime if removing any literal from t results in a term that is no longer an implicant. This generalises the notion of implicant and prime-implicant from total Boolean functions, cf. (Quine 1952; Shih, Choi, and Darwiche 2018; Darwiche and Hirth 2020), to partial Boolean functions, cf. (McCluskey 1956; Coudert 1994).

We state a simple but useful lemma:

Lemma 1. *If F, G are partial functions over $\mathbf{X}$ such that $F^1 \cup F^* \subseteq G^1 \cup G^*$, then every prime implicant of F is subsumed by some prime implicant of G.*

Proof. Clearly, every implicant of F is an implicant of G. Now, apply the fact that every implicant of a function is subsumed by some prime-implicant of that function. □

Decision-functions. Total Boolean functions naturally arise as the decision-functions of threshold-based binary classifiers (Choi et al. 2019; Shih, Choi, and Darwiche 2018): the *decision-function* F of a threshold-based classifier is the function that maps an instance $\mathbf{x}$ to 1 if $Pr(d = 1|\mathbf{x}) \geqslant T$, and to 0 otherwise; here d is a binary class variable, and Pr is the distribution specified by the classifier, and T is a user-defined classification threshold.

Definition 1 (Sufficient reasons for total functions)**.** *(Darwiche and Hirth 2020) Let F be a total Boolean function and let $\mathbf{x}$ be a positive instance of F. A term t is a* sufficient reason of the decision $F(\mathbf{x}) = 1$ *if (i) t is a prime-implicant of F, and (ii) t is satisfied by $\mathbf{x}$.*

Sufficient reasons are called *PI-explanation* in (Shih, Choi, and Darwiche 2018), and *abductive explanations* in (Ignatiev, Narodytska, and Marques-Silva 2019a).

Standard convention. We freely interchange between total Boolean functions and the formulas that represent them. In particular, if φ represents the total Boolean function F, we may refer to implicants, prime-implicants, and sufficient reasons of φ (instead of F).

Problem Setting

The problem we address is how to define reasons behind the decisions of a classifier in the presence of domain constraints.

Definition 2. *A* constraint *is a set C of instances over $\mathbf{X}$.*

We typically represent constraints by Boolean formulas. Here are just a few examples that show that constraints are ubiquitous. In a medical setting, constraints of the form $(X_1 \rightarrow X_2)$ may reflect that people with condition X_1 also have condition X_2, e.g., X_1 may mean "is pregnant" and X_2 may mean "is a woman". In a university degree structure: constraints of the form $X_1 \rightarrow (X_2 \wedge X_3)$ may reflect that X_2 and X_3 are prerequisites to X_1; constraints of the form $X_1 \rightarrow \neg(X_2 \vee X_3)$ may reflect prohibitions; and constraints of the form $X_1 \wedge X_2$ may reflect compulsory courses. In configuration problems, e.g., that arise when users purchase products, the user may be required to configure their product subject to certain constraints, and constraints of the form $(X_1 \vee X_2) \wedge \neg(X_1 \wedge X_2)$ may reflect that the user needs to choose between two basic models. These constraints also result from one-hot encodings of a categorical variables, e.g.,

if M is a 12-valued variable representing months, and X_i for $i = 1, \cdots, 12$ is Boolean variable, then the induced constraint is $(\bigvee_i X_i) \wedge \left(\bigwedge_{i \neq j} \neg(X_i \wedge X_j)\right)$. Combinatorial objects have natural constraints, e.g., rankings are ordered sets, trees are acyclic graphs, and games have rules, see the Case Studies and Validation section. Finally, the assumption in this paper is that constraints are *hard*, i.e., instances that are not in C will not appear in any data and can be ignored (e.g., they will not appear in training or testing data).

Recall that threshold-based classifiers produce representations of *total* Boolean functions. This suggests the following useful terminology:

Definition 3. *A* constrained decision-function *is a pair (F, C) consisting of a total Boolean function F and a constraint C.*

We thus ask:

How should one define reasons behind decisions of constrained decision-functions?

We posit that a suitable notion of "reason" that takes constraints into account:

D1. does not depend on the representations of F or C, i.e., it is a semantic notion;

D2. does not depend on the values $F(\mathbf{x})$ for $\mathbf{x} \notin C$, i.e., if F, G agree on C (and perhaps disagree on the complement of C), then reasons for (F, C) should be the same as reasons for (G, C);

D3. in case there are no constraints, i.e., $C = \{0,1\}^n$, recovers the notion of sufficient reasons from Definition 1;

D4. eliminates redundancies that are captured by the constraints.

We offer a formalisation that satisfies these desiderata.

Reasons in the Presence of Constraints

In this section we provide the main definition of reasons in the presence of constraints (Definition 5) and show that it satisfies all of the desired properties D1-D4 listed in the Problem Setting section.

D1 and D2 motivate the insight that constrained decision-functions should be treated as partial Boolean functions:

Definition 4. *For a constrained decision-function (F, C), let F_C be the partial Boolean function that maps $\boldsymbol{x}$ to $F(\boldsymbol{x})$ if $\boldsymbol{x} \in C$, and to $*$ otherwise.*

Thus, we now define sufficient reasons that take constraints into account by considering the partial function F_C.

Definition 5 (Sufficient reasons that take constraints into account)**.** *Let (F, C) be a constrained decision function, and let $\boldsymbol{x}$ be a positive instance of F such that $\boldsymbol{x} \in C$. A term t is a* sufficient reason of the decision $F(\mathbf{x}) = 1$ that takes the constraint C into account *if (i) t is a prime-implicant of the partial Boolean function F_C, and (ii) t is satisfied by $\boldsymbol{x}$.*

In this case, we will also say that t is a sufficient reason of the decision $F_C(\mathbf{x}) = 1$. *We will also call prime-implicants of F_C,* sufficient reasons using F_C.

To see that D3 holds, simply note that if $C = \{0,1\}^n$ then $F_C = F$ is a total function. Thus, a term t is a sufficient reason of the decision $F(\mathbf{x}) = 1$ that takes C into account iff it is a sufficient reason of the decision $F(\mathbf{x}) = 1$ according to Definition 1.

Remark 1. *Sufficient reasons of negative instances $\boldsymbol{x}$ can be defined and handled dually: a term t is a* sufficient reason of the decision $F_C(\mathbf{x}) = 0$ *if it is a sufficient reason of the decision $G_C(\boldsymbol{x}) = 1$ where G is the "negation of F", i.e., $G(\boldsymbol{x}) := 0$ if $F(\boldsymbol{x}) = 1$, and $G(\boldsymbol{x}) := 1$ if $F(\boldsymbol{x}) = 0$. Thus, we can reduce reasoning about negative instances of F_C to reasoning about positive instances of G_C.*

Example 1. *Consider the total Boolean function F over $\boldsymbol{X} = \{X_1, X_2\}$ represented by the formula $(X_1 \leftrightarrow X_2)$. Suppose a constraint C is represented by the formula $(X_1 \rightarrow X_2)$, thus $C = \{(0,0), (0,1), (1,1)\}$. Table 1 provides both F and the partial Boolean function F_C. The prime-implicants of F are $(X_1 \wedge X_2)$ and $(\neg X_1 \wedge \neg X_2)$. The only sufficient reason of the decision $F(0,0) = 1$ is the term $(\neg X_1 \wedge \neg X_2)$, and the only sufficient reason of the decision $F(1,1) = 1$ is the term $(X_1 \wedge X_2)$. The prime-implicants of F_C are $\neg X_2$ and X_1. The only sufficient reason of the decision $F_C(0,0) = 1$ is $\neg X_2$, and the only sufficient reason of the decision $F_C(1,1) = 1$ is X_1.*

X_1	X_2	F	F_C
0	0	1	1
0	1	0	0
1	0	0	*
1	1	1	1

Table 1: The row corresponding to the instance not in the constraint is greyed out.

Finally, we provide a simple theorem that formalises D4. We prove that every sufficient reason that does not take constraints into account is subsumed by some sufficient reason that does.

Theorem 1. *Suppose $\boldsymbol{x}$ is a positive instance of F_C. Then every sufficient reason of the decision $F(\boldsymbol{x}) = 1$ is subsumed by some sufficient reason of the decision $F_C(\boldsymbol{x}) = 1$.*

Proof. Let $\mathbf{x}$ be a positive instance of F_C. In particular, it is a positive instance of F. Let t be a prime implicant of F that is satisfied by $\mathbf{x}$. We show that there is some prime implicant t' of F_C that subsumes t (and thus is satisfied by $\mathbf{x}$). To see this, apply Lemma 1 taking $G = F_C$. The hypothesis of the Lemma holds (i.e., that $F^1 \cup F^* \subseteq G^1 \cup G^*$) since $F^1 \cup F^* = F^1$ (since F is total) and $(F_C)^1 \cup (F_C)^* = (F^1 \cap C) \cup \overline{C}$ (by definition of F_C). □

To complement this theorem, we show that simply considering reasons of the total Boolean function F (and ignoring the constraint C), may actually supply strictly less succinct reasons.

Example 2. *Continuing Example 1, note that the only sufficient reason for $F(0,0) = 1$ is subsumed by a sufficient*

reason of $F_C(0,0) = 1$, i.e., $(\neg X_1 \wedge \neg X_2)$ is subsumed by $\neg X_2$. Similarly, the only sufficient reason for $F(1,1) = 1$ is subsumed by a sufficient reason of $F_C(1,1) = 1$, i.e., $(X_1 \wedge X_2)$ is subsumed by X_1. This accords with the intuition that, in light of the constraint $(X_1 \rightarrow X_2)$, reason X_1 is preferred to reason $(X_1 \wedge X_2)$.

It is not hard to find examples where every sufficient reason of $F(\mathbf{x}) = 1$ is much larger than every sufficient reason of $F_C(\mathbf{x}) = 1$. E.g., let F be the function $X_1 \wedge X_2 \wedge \cdots \wedge X_n$, and C be the constraint $X_1 \rightarrow (X_2 \wedge X_3 \wedge \cdots \wedge X_n)$; then the only reason of the decision $F(1,1,\cdots,1) = 1$ is $X_1 \wedge X_2 \wedge \cdots \wedge X_n$, which is subsumed by the reason X_1 of the decision $F_C(1,1,\cdots,1) = 1$.

Constraint-equivalent reasons. If one is interested in the meaning of a reason, and not its syntactic structure, then one should consider sufficient reasons up to logical-equivalence modulo the constraints. That is, terms t, s are *C-equivalent* (or simply, *constraint-equivalent* when the constraint is understood), if $C \cap [s] = C \cap [t]$. For instance, if C is represented by $(X_1 \vee X_2) \wedge \neg(X_1 \wedge X_2)$ then $t = \neg X_1$ is C-equivalent to $s = X_2$, and thus s and t may be identified as the same reason in the presence of C.

Variations and Parsimony of Reasons

Subtle changes in the definition of sufficient reasons result in radically different types of reasons. First, we have seen in the Examples that ignoring the constraints does not provide the most parsimonious reasons. Second, consider the variation in which, instead of using reasons of the partial function F_C, one uses reasons of the total function that agrees with F on C and assigns 0 to instances not in C. Although seemingly natural, it is not hard to see using Lemma 1, that this results in *less parsimonious* reasons. Moreover, if F, C are represented by the Boolean formulas φ and κ respectively, then this total function is represented by the formula $\kappa \wedge \varphi$. In the next section we will see that sufficient reasons using F_C are the same as using the total function corresponding to the formula $\kappa \rightarrow \varphi$. We find it striking that this change of perspective drastically changes the parsimony of the produced reasons; we provide an example of this difference in the discussion of Case Study 1.

Computing Sufficient Reasons

In this section we discuss how to computationally find sufficient reasons in the presence of constraints. In particular, we show how to reduce this to the unconstrained case.

Definition 6 (Computational problems)**.** *Given a constrained decision-function (F, C), and a positive instance $\boldsymbol{x}$ of F_C, find all (resp. one) sufficient reasons for the decision $F_C(\boldsymbol{x}) = 1$.*

As usual (see the Preliminaries), we can think of the total function F and the set of instances C as Boolean formulas, say $F^1 = [\varphi]$ and $C = [\kappa]$ (we are agnostic about exactly how to represent these formulas until we discuss complexity and the experiments). The following proposition says that we can reduce the computational problem of the constrained case to the unconstrained case using the formula $(\kappa \rightarrow \varphi)$.

Proposition 1. *Suppose φ represents F and κ represents C. For a positive instance $\boldsymbol{x}$ of F_C, the sufficient reasons of the decision $F_C(\boldsymbol{x}) = 1$ are exactly the sufficient reasons of the decision $G(\boldsymbol{x}) = 1$ where G is the total function represented by the Boolean formula $(\kappa \rightarrow \varphi)$.*

Proof. First, note that $\mathbf{x}$ is a positive instance of G. Indeed, since $F_C(\mathbf{x}) = 1$ we know that $\mathbf{x} \in C \cap F^1$, i.e., $\mathbf{x} \models \kappa \wedge \varphi$, and thus also $\mathbf{x} \models \kappa \rightarrow \varphi$. Thus, it is sufficient to show that a term t is an implicant of F_C iff it is an implicant of G. By definition, t is an implicant of F_C iff $[t] \subseteq (F_C)^1 \cup (F_C)^*$. But $(F_C)^1 = F^1 \cap C$ and $(F_C)^* = \overline{C}$ (Definition 4). On the other hand, t is an implicant of the total function G iff $[t] \subseteq G^1$. But $G^1 = \overline{C} \cup F^1$. Thus $G^1 = (F_C)^* \cup (F_C)^1$. □

The significance of Proposition 1 is that it shows how to reuse algorithms and tools that are already developed for reasoning about total Boolean functions. Indeed, as long as the formulas κ, φ are represented in a language that allows one to form the conditional $\kappa \rightarrow \varphi$ formula in polynomial time in the sizes of κ, φ, we have a polynomial time reduction of the problem of finding reasons with constraints to those without. On the other hand, reasoning without constraints is a special case of reasoning with constraints, i.e., there is a trivial reduction in the other direction too, simply take $\kappa = \mathtt{true}$. We summarise this important computational fact as follows:

Theorem 2. *Assume that formulas are represented in a formalism that allows one to form the conditional of two formulas in polynomial time. Then, the problem of finding all (resp. one) sufficient reasons for a decision that takes constraints into account is polynomial time interreducible with the problem of finding all (resp. one) sufficient reasons for a decision (without constraints).*

Thus, if one uses representations that also allow one to compute sufficient reasons of total Boolean functions in polynomial time, then, by first applying the reduction in Theorem 2 one can find sufficient reasons for constrained decision-functions in polynomial time too.

We mention the two main approaches comprising the state of the art for computing sufficient reasons for total Boolean functions. First, (Shih, Choi, and Darwiche 2018) represent formulas using OBDDs, which support polynomial negation and conjunction (and thus implication). Their approach provides a polynomial time procedure for finding all sufficient reasons, using the fact that OBDDs support polynomial-time validity and entailment checking. To reuse their algorithm in our setting, simply run it on the OBDD representation of the formula $(\kappa \rightarrow \varphi)$. Second, (Ignatiev, Narodytska, and Marques-Silva 2019a) take an agnostic view on the representation of formulas, and only require that the chosen representation allows polynomial time entailment checking. To reuse their approach in the presence of constraints, one may use it on formulas of the form $(\kappa \rightarrow \varphi)$ instead of φ.

Note that if a representation also allows (a) polynomial time validity checking, and (b) forming the conjunction of a term and formula in polynomial time, then one can decide if two terms are constraint-equivalent in polynomial time. Thus, if one is interested in computing reasons up to

constraint-equivalence one can compute a set of representatives by, for instance, checking each pair of reasons for constraint-equivalence.

Illustration

To clarify the introduced concepts, we illustrate sufficient reasons on a complete synthetic example of a learnt classifier, inspired by an example in (Kisa et al. 2014).

Consider a tech-company that is shortlisting recent CS graduates for a job interview. The company considers candidates who took courses on Probability (P), Logic (L), Artificial Intelligence (A) or Knowledge Representation (K) during their studies. Suppose that the company uses data on candidates who were hired in the past to learn a threshold-based classifier, and let F be the associated *total* decision-function over $\mathbf{X} = \{L, K, P, A\}$ with $F^1 = \{(0011), (0110), (0111), (1100), (1101), (1110), (1111)\}$.

Consider an instance $\mathbf{x} = (0011)$ corresponding to candidates that did not take L or K, but did take P and A. Note that $F(\mathbf{x}) = 1$, i.e., the classifier decides to grant such candidates interviews. What is the reason behind this decision? Table 2 gives the reasons which were computed using (Shih, Choi, and Darwiche 2018). We see that the only reason behind the decision of F for $\mathbf{x} = (0011)$ is $(\neg L \wedge P \wedge A)$, i.e., that the candidate did not take L, but did take P and A.

L	K	P	A	Reasons using F	using F_C
0	0	1	1	$(\neg L \wedge P \wedge A)$	$(\neg L \wedge A)$
0	1	1	1	$(\neg L \wedge P \wedge A), (K \wedge P)$	$(\neg L \wedge A), K$
1	1	0	0	$(L \wedge K)$	K
1	1	1	0	$(L \wedge K), (K \wedge P)$	K
1	1	1	1	$(L \wedge K), (K \wedge P)$	K

Table 2: Rows list the positive instances that satisfy the constraints, along with their reasons using F and using F_C.

Suppose, that a student's enrolments must satisfy the following constraints C: a student must take P or L, $(P \vee L)$; the prerequisite for A is P, $(A \rightarrow P)$; the prerequisite for K is A or L, $(K \rightarrow (A \vee L))$. Reasons of the constrained decision-function F_C are given in Table 2. Note $(\neg L \wedge A)$ and K are not constraint-equivalent.

Consider the reason behind the decision $F_C(\mathbf{x}) = 1$ where $\mathbf{x} = (0011)$, i.e., $\neg L \wedge A$. This reason strictly subsumes the reason $\neg L \wedge P \wedge A$ used by the original (unconstrained) classifier F. This phenomenon, that for every positive instance $\mathbf{x}$ in C, every sufficient reason of $F(\mathbf{x}) = 1$ is subsumed by some sufficient reason of $F_C(\mathbf{x}) = 1$, can be seen in all other rows of Table 2. This illustrates that our notion of sufficient reason (Definition 5) eliminates such redundancies, a fact we formalised in Theorem 1.

Case Studies and Validation

In this section we validate our theory on constrained decision-functions learnt from binary data.[2] We provide a prototype using a type of classifier that is often considered interpretable, i.e., decision trees. The purpose of the prototype is to provide a proof of concept that shows that by using constrained decision-functions: (1) we get no less succinct, and sometimes more succinct, reasons compared with the unconstrained setting; (2) we can seamlessly integrate two major types of constraints that can arise in AI, namely constraints due to pre-processing of data (e.g. one-hot, or other categorical, encodings), and semantic constraints that are inherent to the input domain.

Representation As discussed earlier, we can compute reasons by reducing to the unconstrained case. We reuse the algorithms in (Shih, Choi, and Darwiche 2018) by simply building an OBDD representing $\kappa \rightarrow \varphi$ (using the OBDD operations for complementation and disjunction), and pass this OBDD as input to their tool that computes sufficient reasons for a given instance.

Case Study 1. We used the dataset of Corticosteroid Randomization after Significant Head Injury (CRASH) trial (Collaborators et al. 2008) to predict the condition of a patient after a traumatic head injury. There are eleven clinically relevant input variables, including demographics, injury characteristics and image findings, see (Zador, Sperrin, and King 2016) for a detailed description of the dataset. Six variables are categorical, and the rest are Boolean.[3] The outcome variable indicates moderate or full recovery at 6 months versus death or severe disability.

Categorical variables are encoded using a one-hot encoding, which induces the constraint C as follows. For a categorical variable X, let D denote a set of Boolean variables corresponding to the set of categories of X. The corresponding constraint says that exactly one of the variables in D must be true. For example, variable Eye (shortened to E) has 4 categories, which we encode by the Boolean variables in $D_E = \{E_1, E_2, E_3, E_4\}$. The corresponding constraint is $\bigvee_i E_i \wedge \bigwedge_{i \neq j} \neg(E_i \wedge E_j)$, where i, j vary over $\{1, 2, 3, 4\}$. The constraint C is the conjunction of all such constraints, one for each categorical variable.

Following (Steyerberg et al. 2008) we base our example on 6945 cases with no missing values. RPART (seed: 25, train: 0.75, cp: 0.005) correctly classifies 75.69% of instances in the test set (ROC 0.77). Figure 2 shows the model.

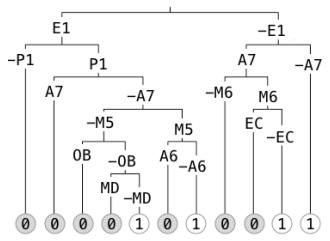

Figure 2: RPART decision tree for Case Study 1.

Consider the instance $\mathbf{x}$ that maps A_1, E_1, M_5, V_2, P_1, OB, MD to 1, and the remaining four variables to 0. The

[2] Continuous data can be discretised, and discrete/categorical data can be binarised (Breiman et al. 1984).

[3] Categorical are: **A**ge(1-7), **E**ye(1-4) **M**otor(1-6), **V**erbal(1-5), **P**upils(1-3). Boolean are: EC, PH, OB, SA, MD, HM.

decision-rule in the decision tree that explains why **x** is positive is $E_1 \wedge P_1 \wedge \neg A_7 \wedge M_5 \wedge \neg A_6$ (size: 5). There is one sufficient reason using F: $\neg A_6 \wedge \neg A_7 \wedge M_5 \wedge P_1$ (size: 4). Up to constraint-equivalence there are two sufficient reasons using F_C: (i) $A_1 \wedge M_5 \wedge P_1$ (size: 3), (ii) $\neg A_6 \wedge \neg A_7 \wedge M_5 \wedge P_1$ (size: 4).

Discussion of Case-Study 1. The explanation using the decision tree is strictly subsumed by the sufficient reason using F. This shows that decision-rules may not be the most succinct reasons. Further, incorporating constraints resulted in having a smaller reason which would be missed if one just used F. The reason using F is subsumed by some reason using F_C, in fact it appears as reason (ii); cf. Theorem 1.

Note that reasons (i) and (ii) are not constraint-equivalent (and thus should be considered different reasons). Which reason should one prefer? On the one hand, (i) is more succinct, but on the other hand (ii) strictly constraint-subsumes (i), i.e., it applies to more instances. Without additional preferences there is no basis to prefer one over the other, and thus we report both of them.

If one incorporated constraints by instead using the function represented by the formula $(\kappa \wedge \varphi)$ one would get one sufficient reason for this decision that is highly redundant in light of the constraint (as discussed in the Variations section), i.e., $(A_1 \wedge E_1 \wedge M_5 \wedge V_2 \wedge P_1 \wedge \bigwedge_X \neg X)$ where the conjunction is over all the remaining variables $A_2, A_3, \cdots, E_2, E_3, \cdots$.

Finally, the histogram in Figure 3 compares the sizes of shortest reasons using F and F_C (omitting size 2 reasons which would dominate the graph). Note that the percentage of reasons using F increases with size, while those using F_C decreases with size.

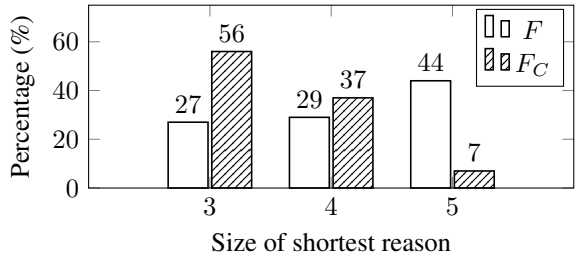

Figure 3: Distribution of shortest reasons, restricted to instances without length $\leqslant 2$ reasons (i.e., 5440 of 109120 instances). Percentages are rounded to the nearest decimal.

In summary, this case study empirically validates that reasons that take constraints into account may be more succinct.

Case Study 2. To study semantic constraints, we used the Tic-Tac-Toe (TTT) Endgame dataset from the UCI machine learning repository (Dua and Graff 2017). This dataset contains the complete set of board configurations that result from X going first, until the game ends. The target concept is "player X has three-in-a-row".

We binarise the dataset as in (Verwer and Zhang 2019). For each of the 9 board positions (labelled as in Table 3i.) introduce variables $V_{i,O}$ (resp. $V_{i,X}$) capturing whether or not O (resp. X) was placed in position i. We trained a classifier on this dataset using RPART (seed 1, train: 0.7, cp 0.01); with 93% accuracy for the test set (ROC 0.97), see Figure 4.

i.

0	1	2
3	4	5
6	7	8

ii.

X	X	X
O		O

iii.

01	01	01
00	00	00
10	00	10

Table 3: i. TTT board; ii. Positive instance; iii. Encoded instance (cell i is labelled by the values of $V_{i,O}V_{i,X}$).

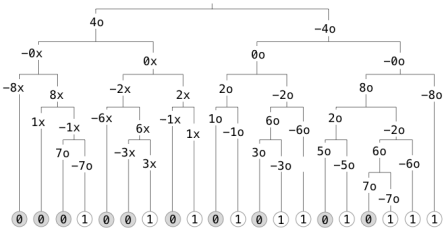

Figure 4: RPART decision tree for case study 2. We drop V and write, e.g., $4o$ instead of $V_{4,O}$ for readability.

Let F be the corresponding decision-function. In what follows we focus on sufficient reasons for the instance in Table 3iii. The sufficient reasons using F are given in Table 4.

1. 0– –– 0– –– 0– –– –– 0– ––	2. 0– –– –– –– 0– 0– –– 0– ––	3. –– –– 0– 0– 0– –– –– 0– ––	4. –– 0– –– 0– 0– 0– –– 0– ––
5. 01 –1 01 –– –– –– –– 0– ––	6. 01 –1 –1 –– –– 0– –– 0– ––	7. –1 –1 01 0– –– –– –– 0– ––	8. –1 01 –1 0– –– 0– –– 0– ––

Table 4: Reasons using F

A. –1 –1 –1 –– –– –– –– 0– ––	B. –– –– –– –– 00 00 –– 00 1–

Table 5: (A) a reason using F_C, (B) a reason using $F_{C'}$

Simple constraints for TTT The encoding induces a constraint C that expresses that no position contains both an O and an X, although, unlike the one-hot-constraints (as in Case Study 1), it may have neither, i.e., C is given by $\bigwedge_{0 \leqslant i \leqslant 8} \neg(V_{i,O} \wedge V_{i,X})$. Again, consider the positive instance in Table 3iii. The reasons for the decision using F_C include Reasons 1-4 in Table 4, as well as Reason A from Table 5 which strictly subsumes Reasons 5-8 in Table 4.

This shows that some reasons of F are redundant in light of the constraint C, e.g., as witnessed by the inclusion of the literals $\neg V_{0,O}$ and $V_{0,X}$ in reason 5.

More complex constraints: adding game rules Define the constraint C' to include C as well as saying that the board is the result of valid play, i.e., that X moves first and players alternate moves. The additional constraint is $\bigvee_{S,T}(\psi_S \wedge \varphi_T)$ where S, T vary over all subsets of $U = \{0, 1, 2, \cdots, 8\}$ such that $S \cap T = \varnothing$, and $0 \leqslant |S| - |T| \leqslant 1$, and ψ_S is $(\bigwedge_{i \in S} V_{i,X}) \wedge (\bigwedge_{i \in U \setminus S} \neg V_{i,X})$ and φ_T is $(\bigwedge_{i \in T} V_{i,O}) \wedge (\bigwedge_{i \in U \setminus T} \neg V_{i,O})$. The formula expresses that the set S of positions where X has played is disjoint from the set T where O has played, and that either there are the same number of moves, or X has played one more. Using $F_{C'}$, the sufficient reasons for the instance above include Reason B in Table 5. This reason can be interpreted as follows: *in light of the constraint C', which says that the board is the result of a valid play*, if positions 4,5,7 are blank and position 8 has an O, then player X must have won. This is indeed correct: player O could not have won since with 5 moves in the game player O can only move twice, and there could not be a draw because not all positions were filled yet.

Discussion of Case-Study 2. This case study illustrates how our framework seamlessly takes complex semantic constraints, such as combinatorial constraints, into account when producing reasons. This should be contrasted with potential ad-hoc algorithms for incorporating any fixed constraint.

Related Work

We have generalised certain aspects of (Shih, Choi, and Darwiche 2018) and (Ignatiev, Narodytska, and Marques-Silva 2019a) by incorporating domain constraints. We do not deal with constraints on the possible outputs of a classifier (Xu et al. 2018). To provide sufficient reasons in the presence of domain constraints, we show how to reduce the constrained case to the unconstrained case, thus allowing one to reuse existing symbolic algorithms and tools (Shih, Choi, and Darwiche 2018). (Marques-Silva and Ignatiev 2022) provide a thorough summary of the recent developments in formal explanations in AI.

For explanations of classifiers with multi-valued features, (Choi et al. 2020) study three different types of encoding. They settle on a special type of one-hot encoding, and take into account the induced exactly-one constraints in the same manner that we have studied in this paper. Concurrently, (Cooper and Silva 2021) define explanations in the presence of general constraints, but do not study the properties and benefits of this definition. Their focus, instead, is on properties of decision functions that ensure finding explanations is tractable.

The ML literature has techniques for producing (post-hoc local) rule-based explanations which are similar in spirit to the logic-based method of this paper. Notably, the *Anchors* of (Ribeiro, Singh, and Guestrin 2018) provide explanations that may be more succinct than sufficient reasons, but may fail to be implicants of F as shown in (Ignatiev, Narodytska, and Marques-Silva 2019b). In Case Study 1, we observe that decision-tree branches may not supply the most succinct reasons. This fact was studied by (Izza, Ignatiev, and Marques-Silva 2020) who demonstrate the significant explanation-redundancy of decision tree branches.

Another notable parallel is from social science research, namely in Configurational Comparative Methods (CCMs) (Thiem 2014; Baumgartner and Thiem 2015; Thiem and Duşa 2013) where prime-implicants of partial Boolean functions are used for causality analysis.

Discussion

The crux of this paper shows how to handle constraints in a principled manner, and establishes that ignoring constraints could result in unnecessarily long/complex reasons, as well as missing some reasons altogether. For computing reasons, our approach reduces the constrained case to the unconstrained case. Thus, any advance in the efficiency of tools for solving the latter will yield benefits for the former.

A general critique of the prime-implicant based approach is that reasons may become too large to comprehend when the number of variables is large. Notice that our method is a step towards improving this problem in the presence of constraints. If the shortest reason in the presence of constraints is still too large to comprehend, not taking constraints into account may result in reasons that are even larger and even harder to comprehend. Observe, from the case studies, that while adding constraints may decrease or increase the number of reasons, it never increases the size of the shortest reasons (a fact that is guaranteed by Theorem 1). In cases of multiple (constraint-inequivalent) reasons for a decision (even amongst the shortest ones), we do not supply a way to pick one reason over another, a challenging problem (Lakkaraju et al. 2019). Indeed, preferring one reason over another would require *additional assumptions* about preferred reasons, e.g., favouring shorter reasons (Miller 2019).

Our framework for handling constraints is model-agnostic, i.e., it supplies the underlying principle for handling domain constraints for decision-functions that correspond to binary classifiers, no matter how the classifiers were learnt or modelled. As a proof-of-concept, we illustrated this by compiling decision trees into OBDDs. The general problem of compiling ML models into compact circuits is being actively researched, e.g., (Shih, Choi, and Darwiche 2018) for Bayesian networks, (Choi et al. 2019) for Neural Networks, and (Audemard, Koriche, and Marquis 2020) for Random Forests.

Our work opens up applications that are currently only available in the unconstrained setting, including the study of classifier bias and counterfactual decisions (Darwiche and Hirth 2020).

Acknowledgements

We thank Alrik Thiem and Lusine Mkrtchyan for insightful discussions and comments on the utility of prime-implicants of partial functions in social science research.

References

Audemard, G.; Koriche, F.; and Marquis, P. 2020. On tractable XAI queries based on compiled representations. In *KR*.

Baumgartner, M.; and Thiem, A. 2015. Identifying Complex Causal Dependencies in Configurational Data with Coincidence Analysis. *R J.Jun 1;7(1):176*.

Breiman, L.; Friedman, J.; Stone, C. J.; and Olshen, R. A. 1984. *Classification and regression trees*. CRC press.

Choi, A.; Shi, W.; Shih, A.; and Darwiche, A. 2019. Compiling neural networks into tractable Boolean circuits. *AAAI VNN*.

Choi, A.; Shih, A.; Goyanka, A.; and Darwiche, A. 2020. On Symbolically Encoding the Behavior of Random Forests. *CoRR abs/2007.01493*.

Collaborators, M. C. T.; et al. 2008. Predicting outcome after traumatic brain injury: practical prognostic models based on large cohort of international patients. *BMJ*, 336(7641).

Cooper, M.; and Silva, J. M. 2021. On the tractability of explaining decisions of classifiers. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*, 21. Leibniz-Zentrum für Informatik.

Coudert, O. 1994. Two-level logic minimization: an overview. *Integration*, 17(2).

Darwiche, A. 2020. Three Modern Roles for Logic in AI. In *PODS*.

Darwiche, A.; and Hirth, A. 2020. On The Reasons Behind Decisions. In *ECAI*.

Darwiche, A.; and Marquis, P. 2002. A knowledge compilation map. *Journal of Artif. Intell. Research*, 17: 229–264.

Dua, D.; and Graff, C. 2017. UCI Machine Learning Repository.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019a. Abduction-based explanations for machine learning models. In *AAAI*.

Ignatiev, A.; Narodytska, N.; and Marques-Silva, J. 2019b. On validating, repairing and refining heuristic ML explanations. *CoRR abs/1907.02509*.

Iyer, R.; Li, Y.; Li, H.; Lewis, M.; Sundar, R.; and Sycara, K. 2018. Transparency and explanation in deep reinforcement learning neural networks. In *AAAI*.

Izza, Y.; Ignatiev, A.; and Marques-Silva, J. 2020. On explaining decision trees. *arXiv preprint arXiv:2010.11034*.

Kisa, D.; Van den Broeck, G.; Choi, A.; and Darwiche, A. 2014. Probabilistic sentential decision diagrams: Learning with massive logical constraints. In *ICML LTPM*.

Lakkaraju, H.; Kamar, E.; Caruana, R.; and Leskovec, J. 2019. Faithful and customizable explanations of black box models. In *AAAI*.

Marques-Silva, J.; and Ignatiev, A. 2022. Delivering Trustworthy AI through Formal XAI. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

McCluskey, E. J. 1956. Minimization of Boolean functions. *Bell Syst.*, 35(6): 1417–1444.

Miller, T. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artif. Intell.*, 267.

Quine, W. V. 1952. The problem of simplifying truth functions. *Am Math Mon*, 59(8).

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why should I trust you?" Explaining the predictions of any classifier. In *SIGKDD*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Anchors: High-Precision Model-Agnostic Explanations. *AAAI*.

Shih, A.; Choi, A.; and Darwiche, A. 2018. A symbolic approach to explaining bayesian network classifiers. In *IJCAI*.

Steyerberg, E. W.; Mushkudiani, N.; Perel, P.; Butcher, I.; Lu, J.; McHugh, G. S.; Murray, G. D.; Marmarou, A.; Roberts, I.; Habbema, J. D. F.; et al. 2008. Predicting outcome after traumatic brain injury: development and international validation of prognostic scores based on admission characteristics. *PLoS medicine*, 5(8).

Thiem, A. 2014. Unifying configurational comparative methods: Generalized-set qualitative comparative analysis. *Sociological Methods & Research*, 43(2): 313–337.

Thiem, A.; and Duşa, A. 2013. Boolean minimization in social science research: A review of current software for Qualitative Comparative Analysis (QCA). *Soc. Sci. Comput. Rev.*, 31(4).

Verwer, S.; and Zhang, Y. 2019. Learning optimal classification trees using a binary linear program formulation. In *AAAI*.

Xu, J.; Zhang, Z.; Friedman, T.; Liang, Y.; and Broeck, G. 2018. A semantic loss function for deep learning with symbolic knowledge. In *ICML*.

Zador, Z.; Sperrin, M.; and King, A. T. 2016. Predictors of outcome in traumatic brain injury: new insight using receiver operating curve indices and Bayesian network analysis. *PLoS one*, 11(7).