Short Boolean Formulas as Explanations in Practice

Reijo Jaakkola[1], Tomi Janhunen[1], Antti Kuusisto[1,2](✉), Masood Feyzbakhsh Rankooh[1], and Miikka Vilander[1]

[1] Tampere University, Tampere, Finland
antti.kuusisto@tuni.fi
[2] University of Helsinki, Helsinki, Finland

Abstract. We investigate explainability via short Boolean formulas in the data model based on unary relations. As an explanation of length k, we take a Boolean formula of length k that minimizes the error with respect to the target attribute to be explained. We first provide novel quantitative bounds for the expected error in this scenario. We then also demonstrate how the setting works in practice by studying three concrete data sets. In each case, we calculate explanation formulas of different lengths using an encoding in Answer Set Programming. The most accurate formulas we obtain achieve errors similar to other methods on the same data sets. However, due to overfitting, these formulas are not necessarily ideal explanations, so we use cross validation to identify a suitable length for explanations. By limiting to shorter formulas, we obtain explanations that avoid overfitting but are still reasonably accurate and also, importantly, human interpretable.

Keywords: Boolean formula size · Explainability · Interpretable AI · Overfitting error · Answer Set Programming · Boolean optimization

1 Introduction

In this article we investigate explainability and classification via short Boolean formulas. As the data model, we use multisets of propositional assignments. This is one of the simplest data representations available—consisting simply of data points and properties—and corresponds precisely to relational models with unary relations. The data is given as a model M with unary relations $p_1, \ldots, p_k$ over its domain W, and furthermore, there is an additional *target predicate* $q \subseteq W$. As classifiers for recognizing q, we produce Boolean formulas φ over $p_1, \ldots, p_k$, and the corresponding error is then the percentage of points in W that disagree on φ and q over W. For each formula length ℓ, a formula producing the minimum error is chosen as a candidate classifier. Longer formulas produce smaller errors, and ultimately the process is halted based on cross validation which shows that the classifier formulas φ begin performing significantly better on training data in comparison to test data, suggesting overfitting.

Importantly, the final classifier formulas φ tend to be short and therefore *explicate* the global behavior of the classifier φ itself in a transparent way. This leads to *inherent interpretability* of our approach. Furthermore, the formulas φ can *also* be viewed as *explanations* of the target predicate q. By limiting to short formulas, we obtain explanations (or classifiers) that avoid overfitting but are still reasonably accurate and also—importantly—human interpretable.

Our contributions include theory, implementation and empirical results. We begin with some theory on the errors of Boolean formulas as explanations. We first investigate general reasons behind overfitting when using Boolean formulas. We also observe, for example, that if all distributions are equally likely, the expected *ideal theoretical error* of a distribution is 25%. The ideal theoretical error is the error of an ideal Boolean classifier for the entire distribution. We proceed by proving novel, quantitative upper and lower bounds on the expected *ideal empirical error* on a data set sampled from a distribution. The ideal empirical error is the smallest error achievable on the data set. Our bounds give concrete information on sample sizes required to avoid overfitting.

We also compute explanation formulas in practice. We use three data sets from the UCI machine learning repository: Statlog (German Credit Data), Breast Cancer Wisconsin (Original) and Ionosphere. We obtain results comparable to other experiments in the literature. In one set of our experiments, the empirical errors for the obtained classifiers for the credit, breast cancer and ionosphere data are 0.27, 0.047 and 0.14. The corresponding formulas are surprisingly short, with lengths 6, 8 and 7, respectively. This makes them highly interpretable. The length 6 formula for the credit data (predicting if a loan will be granted) is

$$\neg(a[1,1] \wedge a[2]) \vee a[17,4],$$

where $a[1,1]$ means negative account balance; $a[2]$ means above median loan duration; and $a[17,4]$ means employment on managerial level. Our errors are comparable to those obtained for the same data sets in the literature. For example, [25] obtains an error 0.25 for the credit data where our error is 0.27. Also, all our formulas are immediately interpretable. See Sect. 5 for further discussion.

On the computational side, we deploy answer set programming (ASP; see, e.g., [6,14]) where the solutions of a search problem are described declaratively in terms of rules such that the *answer sets* of the resulting logic program correspond to the solutions of the problem. Consequently, dedicated search engines, known as *answer-set solvers*, provide means to solve the problem via the computation of answer sets. The Clasp [8] and Wasp [1] solvers represent the state-of-the art of native answer set solvers, providing a comparable performance in practice. These solvers offer various reasoning modes—including prioritized optimization—which are deployed in the sequel, e.g., for the minimization of error and formula length. Besides these features, we count on the flexibility of rules offered by ASP when describing explanation tasks. More information on the technical side of ASP can be found from the de-facto reference manual [9] of the Clingo system.

The efficiency of explanation is governed by the number of hypotheses considered basically in two ways. Firstly, the search for a plausible explanation requires the exploration of the hypothesis space and, secondly, the exclusion of better explanations becomes a further computational burden, e.g., when the error with respect to data is being minimized. In computational learning approaches (cf. [17]), such as *current-best-hypothesis search* and *version space learning*, a hypothesis in a normal form is maintained while minimizing the numbers of false positive/negative examples. However, in this work, we tackle the hypothesis space somewhat differently: we rather specify the form of hypotheses and delegate their exploration to an (optimizing) logic solver. In favor of interpretability, we consider formulas based on negations, conjunctions, and disjunctions, not necessarily in a particular normal form. By changing the form of hypotheses, also other kinds of explanations such as decision trees [19] or lists could alternatively be sought.

Concerning further related work, our bounds on the difference between theoretical and expected empirical error are technically related to results in statistical learning theory [24] and PAC learning [15,23]. In PAC learning, the goal is to use a sample of labeled points drawn from an unknown distribution to find a hypothesis that gives a small true error with high probability. The use of hypotheses with small descriptions has also been considered in the PAC learning in relation to the Occam's razor principle [3–5]. One major difference between our setting and PAC learning is that in the latter, the target concept is a (usually Boolean) function of the attribute values, while in our setting we only assume that there is a probability distribution on the propositional types over the attributes.

Explanations relating to minimality notions in relation to different Boolean classifiers have been studied widely, see for example [20] for *minimum-cardinality* and *prime implicant* explanations, also in line with Occam's razor [3]. Our study relates especially to global (or general [11]) explainability, where the full behavior of a classifier is explained instead of a decision concerning a particular input instance. Boolean complexity—the length of the shortest equivalent formula—is promoted in the prominent article [7] as an empirically tested measure of the subjective difficulty of a concept. On a conceptually related note, intelligibility of various Boolean classifiers are studied in [2]. While that study places, e.g., DNF-formulas to the less intelligible category based on the complexity of explainability queries performed on classifiers, we note that with genuinely small bounds for classifier length, asymptotic complexity can sometimes be a somewhat problematic measure for intelligibility. In our study, the bounds arise already from the overfitting thresholds in real-life data. In the scenarios we studied, overfitting indeed sets natural, small bounds for classifier length. In inherently Boolean data, such bounds can be fundamental and cannot be always ignored via using different classes of classifiers. The good news is that while a length bound may be *necessary* to avoid overfitting, *shorter length increases interpretability*. This is important from the point of theory as well as applications.

We proceed as follows. After the preliminaries in Sect. 2, we present theoretical results on errors in Sect. 3. Then, Sect. 4 explains our ASP implementation. Next, we present and interpret empirical results in Sect. 5 and conclude in Sect. 6.

2 Preliminaries

The syntax of propositional logic PL[σ] over the vocabulary $\sigma = \{p_1, \ldots, p_m\}$ is given by $\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi$ where $p \in \sigma$. We also define the exclusive or $\varphi \oplus \psi := (\varphi \vee \psi) \wedge \neg(\varphi \wedge \psi)$ as an abbreviation. A **σ-model** is a structure $M = (W, V)$ where W is a finite, non-empty set referred to as the **domain** of M and $V : \sigma \to \mathcal{P}(W)$ is a **valuation** function that assigns each $p \in \sigma$ the set $V(p)$ (also denoted by p^M) of points $w \in W$ where p is considered to be true.

A σ-valuation V can be extended in the standard way to a valuation $V : \mathrm{PL}[\sigma] \to \mathcal{P}(W)$ for all $\mathrm{PL}[\sigma]$-formulas. We write $w \models \varphi$ if $w \in V(\varphi)$ and say that w **satisfies** φ. We denote by $|\varphi|_M$ the *number* of points $w \in W$ where $\varphi \in \mathrm{PL}[\sigma]$ is true. For σ-formulas φ and ψ, we write $\varphi \models \psi$ iff for all σ-models $M = (W, V)$ we have $V(\psi) \subseteq V(\varphi)$. Let $lit(\sigma)$ denote the set of **σ-literals**, i.e., formulas p and $\neg p$ for $p \in \sigma$. A **σ-type base** is a set $S \subseteq lit(\sigma)$ such that for each $p \in \sigma$, precisely one of the literals p and $\neg p$ is in S. A **σ-type** is a conjunction $\bigwedge S$. We assume some fixed bracketing and ordering of literals in $\bigwedge S$ so there is a one-to-one correspondence between type bases and types. The set of σ-types is denoted by T_σ. Note that in a σ-model $M = (W, V)$, each element w satisfies precisely one σ-type, so the domain W is partitioned by some subset of T_σ. The **size** $size(\varphi)$ of a formula $\varphi \in \mathrm{PL}[\sigma]$ is defined such that $size(p) = 1$, $size(\neg\psi) = size(\psi) + 1$, and $size(\psi \wedge \vartheta) = size(\psi \vee \vartheta) = size(\psi) + size(\vartheta) + 1$.

We will use short propositional formulas as *explanations* of target attributes in data. Throughout the paper, we shall use the vocabulary $\tau = \{p_1, \ldots, p_k\}$ for the language of explanations, while $q \notin \tau$ will be the target attribute (or target proposition) to be explained. While the set of τ-types will be denoted by T_τ, we let $T_{\tau,q}$ denote the set of $(\tau \cup \{q\})$-types in the extended language $\mathrm{PL}[\tau \cup \{q\}]$.

By a probability distribution over a vocabulary σ, or simply a σ-distribution, we mean a function $\mu_\sigma : T_\sigma \to [0, 1]$ that gives a probability to each type in T_σ. We are mainly interested in such distributions over τ and $\tau \cup \{q\}$. For notational convenience, we may write $\mu_{\tau,q}$ or simply μ instead of $\mu_{\tau \cup \{q\}}$. In the theoretical part of the paper, we assume that the studied data (i.e., $(\tau \cup \{q\})$-models) are sampled using such a distribution μ.

We then define some notions of error for explanations. Let $\tau = \{p_1, \ldots, p_k\}$. Fix a probability distribution $\mu : T_{\tau,q} \to [0, 1]$. Let φ and ψ be $(\tau \cup \{q\})$-formulas. The **probability of φ over μ** is defined as

$$\Pr_\mu(\varphi) := \sum_{t \in T_{\tau,q},\ t \models \varphi} \mu(t).$$

The **probability of ψ given φ over μ** is defined as $\Pr_\mu(\psi \,|\, \varphi) := \frac{\Pr_\mu(\psi \wedge \varphi)}{\Pr_\mu(\varphi)}$ (and 0 if $\Pr_\mu(\varphi) = 0$). For simplicity, we may write $\mu(\varphi)$ for $\Pr_\mu(\varphi)$ and $\mu(\psi \,|\, \varphi)$ for $\Pr_\mu(\psi \,|\, \varphi)$. Let $M = (W, V)$ be a $(\tau \cup \{q\})$-model. The **probability of φ over M** is $\Pr_M(\varphi) := \frac{1}{|W|}|\varphi|_M$, and the **probability of ψ given φ over M** is defined as $\Pr_M(\psi \,|\, \varphi) := \frac{|\psi \wedge \varphi|_M}{|\varphi|_M}$ (and 0 if $\Pr_M(\varphi) = 0$). The disjunction $\varphi_{id}^M := \bigvee\{\, t \in T_\tau \mid \Pr_M(q \,|\, t) \leq \frac{1}{2}\}$ is the **ideal classifier** w.r.t. M, and the disjunction $\varphi_{id}^\mu := \bigvee\{\, t \in T_\tau \mid \mu(q \,|\, t) \leq \frac{1}{2}\}$ is the **ideal classifier** w.r.t. μ.

Now, let $\psi \in \mathrm{PL}[\tau]$. The **theoretical error** (or **true error**) of ψ with respect to μ is $\mathrm{err}_\mu(\psi) := \mathrm{Pr}_\mu(\psi \oplus q)$. The **ideal theoretical error of** μ is

$$\mathrm{err}(\mu) := \min_{\psi \in \mathrm{PL}[\tau]} \mathrm{err}_\mu(\psi) = \mathrm{Pr}_\mu(\varphi^\mu_{id}) = \sum_{t \in T_\tau} \min\{\mu(t \wedge q), \mu(t \wedge \neg q)\}.$$

Let M be a $(\tau \cup \{q\})$-model. The **empirical error** of ψ with respect to M is $\mathrm{err}_M(\psi) := \mathrm{Pr}_M(\psi \oplus q)$. The **ideal empirical error** of M is

$$\mathrm{err}(M) := \min_{\psi \in \mathrm{PL}[\tau]} \mathrm{err}_M(\psi) = \mathrm{Pr}_M(\varphi^M_{id}) = \frac{1}{|W|} \sum_{t \,\in\, T_\tau} \min\{|t \wedge q|_M, |t \wedge \neg q|_M\}.$$

For a τ-type t, the **ideal error over** t **w.r.t.** μ is $\min\{\mu(q \,|\, t), \mu(\neg q \,|\, t)\}$. The **ideal error over** t **w.r.t.** M is $\min\{\mathrm{Pr}_M(q \,|\, t), \mathrm{Pr}_M(\neg q \,|\, t)\}$.

The main problem studied in this paper is the following: over a $(\tau \cup \{q\})$-model M, given a bound ℓ on formula length, find ψ with $size(\psi) \leq \ell$ and with minimal empirical error w.r.t. M. This can be formulated as a *general explanation problem* (GEP) in the sense of [11]; see in particular the extended problems in [12]. The goal in GEP is to explain the *global* behavior of a classifier rather than a reason why a particular instance was accepted or rejected.

Finally, we define $cut : [0,1] \to [0, \frac{1}{2}]$ to be the function such that $cut(x) = x$ if $x \leq \frac{1}{2}$ and otherwise $cut(x) = 1 - x$.

3 Expected Errors

In this section we consider the errors given by Boolean classifiers, including the phenomena that give rise to the errors. With no information on the distribution $\mu : T_{\tau,q} \to [0,1]$, it is difficult to predict the error of a classifier φ in $PL[\tau]$. However, some observations can be made. Consider the scenario where all distributions μ are equally likely, meaning that we consider the flat Dirichlet distribution $Dir(\alpha_1, \ldots, \alpha_{|T_{\tau,q}|})$ with each α_i equal to 1, i.e., the distribution that is uniform over its support which, in turn, is the $(|T_{\tau,q}| - 1)$-simplex. For more on Dirichlet distributions, see [16]. We begin with the following observation.

Proposition 1. *Assuming all distributions over $\tau \cup \{q\}$ are equally likely, the expected value of the ideal theoretical error is* 0.25. *Also, for any type $t \in T_\tau$ and any μ_τ with $\mu_\tau(t) > 0$, if all extensions μ of μ_τ to a $(\tau \cup \{q\})$-distribution are equally likely, the expectation of the ideal error over t w.r.t. μ is likewise* 0.25.

Proof. We prove the second claim first. Fix a μ and t. If $x = \mu(q \,|\, t)$, then the ideal error over t w.r.t. μ is given by $cut(x)$. Therefore the corresponding expected value is given by $\frac{1}{1-0} \int_0^1 cut(x)\, dx = \int_0^{\frac{1}{2}} x\, dx + \int_{\frac{1}{2}}^1 (1-x)\, dx = \frac{1}{4}$. This proves the second claim. Based on this, it is not difficult to show that the also the first claim holds; the full details are given in the full version [13]. $\square$

One of the main problems with Boolean classifiers is that the number of types is exponential in the vocabulary size, i.e., the curse of dimensionality. This leads to overfitting via overparameterization; even if the model M is faithful to an underlying distribution μ, classifiers φ_{id}^{M} tend to give empirical errors that are significantly smaller than the theoretical ones for μ. To see why, notice that in the extreme case where $|t|_M = 1$ for each $t \in T_{\tau,q}$, the ideal empirical error of M is zero. In general, when the sets $|t|_M$ are small, ideal classifiers φ_{id}^{M} benefit from that. Let us consider this issue quantitatively. Fix μ and $t \in T_\tau$. For a model M, let $\mathrm{err}(M,t)$ refer to the ideal error over t w.r.t. M. Consider models M sampled according to μ, and let $m \in \mathbb{N}$ and $\mu(q \,|\, t) = p$. Now, the expected value $E(m,p)$ of $\mathrm{err}(M,t)$ over those models M where $|t|_M = m$ is given by

$$\Big(\sum_{0\,<\,k\,\leq\,m/2}\binom{m}{k}p^k(1-p)^{m-k}\cdot\frac{k}{m}\Big)+\Big(\sum_{m/2\,<\,k\,<\,m}\binom{m}{k}p^k(1-p)^{m-k}\cdot\frac{(m-k)}{m}\Big).$$

Now for example $E(4, 0.7) = 0.2541$ and $E(2, 0.7) = 0.21$, both significantly lower than $cut(p) = cut(0.7) = 0.3$ which is the expected value of $\mathrm{err}(M,t)$ when the size restriction $|t|_M = m$ is lifted and models of increasing size are sampled according to μ. Similarly, we have $E(4, 0.5) = 0.3125$ and $E(2, 0.5) = 0.25$, significantly lower than $cut(p) = cut(0.5) = 0.5$. A natural way to avoid this phenomenon is to limit formula size, the strategy adopted in this paper. This also naturally leads to shorter and thus more interpretable formulas.

We next estimate empirical errors for general Boolean classifiers (as opposed to single types). The **expected ideal empirical error of** μ is simply the expectation $\mathbb{E}(\mathrm{err}(M))$ of $\mathrm{err}(M)$, where M is a model of size n sampled according to μ. One can show that $\mathbb{E}(\mathrm{err}(M)) \leq \mathrm{err}(\mu)$ and that $\mathbb{E}(\mathrm{err}(M)) \to \mathrm{err}(\mu)$ as $n \to \infty$. Thus it is natural to ask how the size of the difference $\mathrm{err}(\mu) - \mathbb{E}(\mathrm{err}(M))$, which we call the **bias** of empirical error, depends on n.

In the remaining part of this section we establish bounds on the expected ideal empirical error, which in turn can be used to give bounds on the bias of empirical error. Since expectation is linear, it suffices to give bounds on

$$\frac{1}{n}\sum_{t\in T_\tau}\mathbb{E}\min\{|t\wedge q|_M, |t\wedge\neg q|_M\}, \tag{1}$$

where M is a model of size n which is sampled according to μ. Here, for each type $t \in T_\tau$, $|t \wedge q|_M$ and $|t \wedge \neg q|_M$ are random variables that are distributed according to $\mathrm{Binom}(n, \mu(t \wedge q))$ and $\mathrm{Binom}(n, \mu(t \wedge \neg q))$ respectively. Since $|t \wedge q|_M + |t \wedge \neg q|_M = |t|_M$, we can replace $|t \wedge \neg q|_M$ with $|t|_M - |t \wedge q|_M$.

To simplify (1), we will first use the law of total expectation to write it as

$$\frac{1}{n}\sum_{t\in T_\tau}\sum_{m=0}^{n}\mathbb{E}(\min\{|t\wedge q|_M, m-|t\wedge q|_M\} \mid |t|_M = m)\cdot \Pr(|t|_M = m). \tag{2}$$

For each $0 \leq m \leq n$ and $t \in T_\tau$ we fix a random variable $X_{m,t,q}$ distributed according to $\mathrm{Binom}(m, \mu(q|t))$, where $\mu(q|t) := \mu(t \wedge q)/\mu(t)$. In the full version

[13] we show that (2) equals

$$\frac{1}{n}\sum_{t\in T_\tau}\sum_{m=0}^{n}\mathbb{E}\min\{X_{m,t,q},\ m-X_{m,t,q}\}\cdot\Pr(|t|_M=m). \tag{3}$$

To avoid dealing directly with the expectation of a minimum of two Binomial random variables, we simplify it via the identity $\min\{a,b\}=\frac{1}{2}(a+b-|a-b|)$. In the full version [13] we show that using this identity on (3) gives the form

$$\frac{1}{2}-\frac{1}{n}\sum_{t\in T_\tau}\sum_{m=0}^{n}\mathbb{E}\left|X_{m,t,q}-\frac{m}{2}\right|\cdot\Pr(|t|_M=m). \tag{4}$$

In the above formula the quantity $\mathbb{E}|X_{m,t,q}-\frac{m}{2}|$ is convenient since we can bound it from above using the standard deviation of $X_{m,t,q}$. Some further estimates and algebraic manipulations suffice to prove the following result.

Theorem 1. *Expected ideal empirical error is bounded from below by*

$$\mathrm{err}(\mu)-\frac{1}{\sqrt{n}}\sum_{t\in T_\tau}\sqrt{\mu(q|t)(1-\mu(q|t))\mu(t)}.$$

We note that Theorem 1 implies immediately that the bias of the empirical error is bounded from above by $\frac{1}{\sqrt{n}}\sum_{t\in T_\tau}\sqrt{\mu(q|t)(1-\mu(q|t))\mu(t)}\leq\frac{1}{2}\sqrt{\frac{|T_\tau|}{n}}$. This estimate yields quite concrete sample bounds. For instance, if we are using three attributes to explain the target attribute (so $|T_\tau|=8$) and we want the bias of the empirical error to be at most 0.045, then a sample of size at least 1000 suffices. For the credit data set with 1000 data points, this means that if three attributes are selected, then the (easily computable) ideal empirical error gives a good idea of the ideal theoretical error for those three attributes.

Obtaining an upper bound on the expected ideal empirical error is much more challenging, since in general it is not easy to give good lower bounds on $\mathbb{E}|X-\lambda|$, where X is a binomial random variable and $\lambda>0$ is a real number. Nevertheless we were able to obtain the following result.

Theorem 2. *Expected ideal empirical error is bounded from above by*

$$\frac{1}{2}-\frac{1}{\sqrt{8n}}\sum_{n\mu(t)\geq 1}\sqrt{\mu(t)}+\frac{1}{2\sqrt{8n}}\sum_{n\mu(t)\geq 1}\frac{1-\mu(t)}{\sqrt{n\mu(t)}}-\frac{1}{\sqrt{8}}\sum_{n\mu(t)<1}\mu(t)(1-\mu(t))^n.$$

The proof of Theorem 2 — which can be found in the full version [13] — can be divided into three main steps. First, we observe that the expected ideal empirical error is maximized when $\mu(q|t)=1/2$, for every $t\in T_\tau$, in which case $\mathbb{E}(X_{m,t,q})=\frac{m}{2}$. Then, we use a very recent result of [18] to obtain a good lower bound on the value $\mathbb{E}|X_{m,t,q}-\mathbb{E}(X_{m,t,q})|$. Finally, after some algebraic manipulations, we are left with the task of bounding $\mathbb{E}(\sqrt{|t|_M})$ from below, which we achieve by using an estimate that can be obtained from the Taylor expansion of $\sqrt{x}$ around 1.

To get a concrete feel for the lower bound of Theorem 2, consider the case where $\mu(q|t) = 1/2$, for every $t \in T_\tau$. In this case a *rough* use of Theorem 2 implies that the bias of the empirical error is bounded from below by

$$\frac{1}{\sqrt{8}} \sum_{n\mu(t)<1} \mu(t)(1-\mu(t))^n \geq \frac{1}{\sqrt{8}e} \cdot \frac{(n-1)}{n} \cdot \sum_{n\mu(t)<1} \mu(t),$$

where we used the fact that $(1-1/n)^{n-1} \geq 1/e$, which holds provided that $n > 1$. This lower bound very much depends on the properties of the distribution μ, but one can nevertheless make general remarks about it. For instance, if $|T_\tau|$ is much larger than n and μ is not concentrated on a small number of types (i.e., its Shannon entropy is not small), then we except $\sum_{n\mu(t)<1} \mu(t)$ to be close to one. Thus the above bound would imply that in this scenario the generalization gap is roughly $1/(\sqrt{8} \cdot e) \approx 0.13$, which is a significant deviation from zero.

4 An Overview of the Implementation in ASP

In this section, we describe our proof-of-concept implementation of the search for short formulas explaining data sets. Our implementation presumes Boolean attributes only and *complete* data sets having no missing values. In the following, we highlight the main ideas behind our ASP encoding in terms of code snippets in the Gringo syntax [9]. The complete encoding will be published under the ASPTOOLS collection[1] along with some preformatted data sets for testing purposes. Each data set is represented in terms of a predicate `val(D,A,V)` with three arguments: `D` for a data point identifier, `A` for the name of an attribute, and `V` for the value of the attribute `A` at `D`, i.e., either `0` or `1` for Boolean data.

Given a data set based on attributes $a_0, \ldots, a_n$ where a_n is the target of explanation, the hypothesis space is essentially the propositional language $\mathrm{PL}[\tau]$ with the vocabulary $\tau = \{a_0, \ldots, a_{n-1}\}$. Thus, the goal is to find a definition $a_n \leftrightarrow \varphi$ where $\varphi \in \mathrm{PL}[\tau]$ with the least error. To avoid obviously redundant hypotheses, we use only propositional connectives from the set $C = \{\neg, \wedge, \vee\}$ and represent formulas in the so-called *reversed Polish notation*. This notation omits parentheses altogether and each formula φ is encoded as a sequence $s_1, \ldots, s_k$ of symbols where $s_i \in \tau \cup C$ for each s_i. Such a sequence can be transformed into a formula by processing the symbols in the given order and by pushing formulas on a *stack* that is empty initially. If $s_i \in \tau$, it is pushed on the stack, and if $s_i \in C$, the arguments of s_i are popped from the stack and the resulting formula is pushed on the stack using s_i as the connective. Eventually, the result appears as the only formula on top of stack. For illustration, consider the sequence $a_2, a_1, \wedge, \neg, a_0, \vee$ referring to attributes a_0, a_1, and a_2. The stack evolves as follows: $a_2 \mapsto a_2, a_1 \mapsto (a_1 \wedge a_2) \mapsto \neg(a_1 \wedge a_2) \mapsto \neg(a_1 \wedge a_2), a_0 \mapsto a_0 \vee \neg(a_1 \wedge a_2)$. Thus, the formula is

[1] https://github.com/asptools/benchmarks.

Listing 1. Encoding the syntactic structure of hypotheses

```
 1  % Domains
 2  #const l=10.
 3  node(1..l).  root(l).  op(neg;and;or).
 4  data(D) :- val(D,A,B).
 5  attr(A) :- val(D,A,B).
 6
 7  % Choose the actual length
 8  {used(N)} :- node(N).
 9  used(N+1) :- used(N), node(N+1).
10  used(N) :- root(N).
11
12  %  Choose leaf nodes and inner nodes, and label them
13  {leaf(N)} :- used(N).
14  inner(N) :- used(N), not leaf(N).
15  { op(N,O): op(O) } = 1 :- inner(N).
16  { lat(N,A): attr(A) } = 1 :- leaf(N).
```

Listing 2. Checking the syntax using a stack

```
 1  % Check the size of the stack
 2  count(N,0) :- used(N), not used(N-1).
 3  count(N+1,K+1) :- leaf(N), count(N,K), node(N), K>=0, K<=2.
 4  count(N+1,K) :- count(N,K), node(N), op(N,neg).
 5  count(N+1,K-1) :- count(N,K), node(N), op(N,O), O!=neg.
 6  :- not count(l+1,1).
 7
 8  %  The step-by-step evolution of the stack
 9  stack(N+1,K+1,N) :- leaf(N), count(N,K), K>=0, K<=2.
10  stack(N+1,K,  N) :- op(N,neg), count(N,K), K>0, K<=3.
11  stack(N+1,K-1,N) :- op(N,O), O!=neg, count(N,K), K>=2, K<=3.
12
13  stack(N+1,I,  M) :- leaf(N), count(N,K), I>=0, I<=K, stack(N,I,M).
14  stack(N+1,I,  M) :- op(N,neg), count(N,K), I>0, I<K, stack(N,I,M).
15  stack(N+1,1,  M) :- op(N,O), O!=neg, count(N,3), stack(N,1,M).
```

$a_0 \vee \neg(a_1 \wedge a_2)$. For a formula φ, the respective sequence of symbols can be found by traversing the syntax tree of φ in the *post order*. There are also malformed sequences not corresponding to any formula.

Based on the reverse Polish representation, the first part of our encoding concentrates on the generation of hypotheses whose syntactic elements are defined in Listing 1. In Line 2, the maximum length of the formula is set, as a global parameter `l` of the encoding, to a default value `10`. Other values can be issued by the command-line option `-cl=<number>`. Based on the value chosen, the respective number of *nodes* for a syntax tree is defined in Line 3, out of which the last one is dedicated for the *root*. The three Boolean *operators* are introduced using the predicate `op/1`. The data points and attributes are extracted from data in Lines 4 and 5, respectively. To allow explanations shorter than `l`, the choice rule in Line 8 may take any node into use (or not). The rule in Line 9 ensures that all nodes with higher index values up to `l` are in use. The root node is always in

Listing 3. Evaluating the hypothesis at data points

```
1 true(D,N) :- data(D), leaf(N), lat(N,A), val(D,A,1).
2 {true(D,N)} :- data(D), used(N), inner(N).
3
4 % Constraints for disjunctions
5 :- data(D), op(N,or), count(N,I), stack(N,I-1,N3),
6    true(D,N), not true(D,N-1), not true(D,N3).
7 :- data(D), op(N,or), not true(D,N), true(D,N-1).
8 :- data(D), op(N,or), count(N,I), stack(N,I-1,N2),
9    not true(D,N), true(D,N2).
```

Listing 4. Encoding the objective function

```
1 % Compute error
2 error(D) :- data(D), val(D,A,0), expl(A), true(D,N), root(N).
3 error(D) :- data(D), val(D,A,1), expl(A), not true(D,N), root(N).
4
5 #minimize { 1@1,D: error(D);  1@0,N: used(N), node(N) }.
```

use by Line 10. The net effect is that the nodes `i..l` taken into use determine the actual length of the hypothesis. Thus the length may vary between `1` and `l`. In a valid syntax tree, the nodes are either *leaf* or *inner* nodes, see Lines 13 and 14, respectively. Each inner node is assigned an operator in Line 15 whereas each leaf node is assigned an attribute in Line 16, to be justified later on.

The second part of our encoding checks the syntax of the hypothesis using a stack, see Listing 2. Line 2 resets the size of the stack in the first used node. The following rules in Lines 3–5 take the respective effects of attributes, unary operators, and binary operators into account. The constraint in Line 6 ensures that the count reaches `1` after the root node. Similar constraints limit the size of the stack: at most two for leaf nodes and at least one/two for inner nodes with a unary/binary connective. The predicate `stack/3` propagates information about arguments to operators, i.e., the locations `N` where they can be found. Depending on node type, the rules in Lines 9–11 create a new reference that appears on top of the stack at the next step `N+1` (cf. the second argument `K+1`, `K`, or `K-1`). The rules in Lines 13–15 copy the items under the top item to the next step `N+1`.

The third part of our encoding evaluates the chosen hypothesis at data points `D` present in the data set given as input. For a leaf node `N`, the value is simply set based on the value of the chosen attribute `A` at `D`, see Line 1. For inner nodes `N`, we indicate a choice of the truth value in Line 2, but the choice is made deterministic in practice by the constraints in Lines 5–9, illustrating the case of the `or` operator. The constraints for the operators `neg` and `and` are analogous.

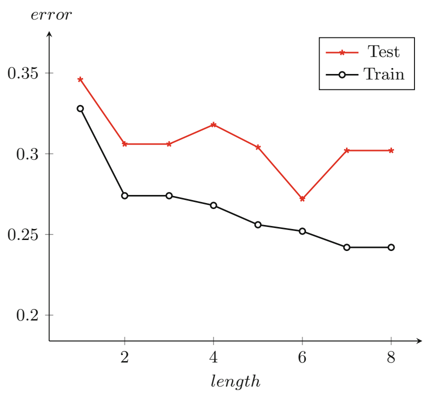
Finally, Listing 4 encodes the objective function. Lines 2 and 3 spot data points `D` that are incorrect with respect to the attribute `A` being explained and the selected hypothesis rooted at `N`. For a false positive `D`, the hypothesis is true at `D` while the value of `A` is `0`. In the opposite case, the hypothesis is false while the value of `A` at `D` is `1`. The criteria for minimization are given in Line 5. The number of errors is the first priority (at level `1`) whereas the length of the hypothesis is the secondary objective (at level `0`). Also, recall that the maximum length has been set as a parameter earlier. The optimization proceeds *lexicographically* as follows: a formula that minimizes the number of errors is sought first and, once such an explanation has been found, the length of the formula is minimized additionally. So, it is not that crucial to set the (maximum) length parameter `l` to a particular value: the smaller values are feasible, too, based on the nodes in use. The performance of our basic encoding can be improved by adding constraints to prune redundant hypotheses, sub-optimal answer sets, and candidates.

5 Results from Data and Interpretation

To empirically analyze short Boolean formulas as explanations and classifiers, we utilize three data sets from the UCI machine learning repository: Statlog (German Credit Data), Breast Cancer Wisconsin (Original) and Ionosphere. The target attributes are given as acceptance of a loan application, benignity of a tumor and "good" radar readings, respectively. The breast cancer data contains a small number of instances with missing attribute values (16 out of 699), which are excluded from the analysis. The original data sets contain categorical and numerical attributes, as well as Boolean ones. To convert a categorical attribute into Boolean format, we treat the inclusion of instances in each corresponding category as a separate Boolean attribute. For numerical attributes, we use the median across all instances as the threshold. Thus the Booleanized credit, breast cancer and ionosphere data sets consist of 1000, 683 and 351 instances, respectively, with 68, 9 and 34 Boolean attributes each, plus the target attribute. To evaluate the obtained formulas as classifiers, we randomly divide each data set into two equal parts: one serving as the training data and the other as the test data. For the training data M, target predicate q and increasing formula length bounds ℓ, we produce formulas ψ not involving q with $size(\psi) \leq \ell$ that minimize the empirical error $\mathrm{err}_M(\psi)$. We also record the error on the test data (i.e., the complement of the training data). We repeated this process 10 times. For each data set, Figs. 1, 2 and 3 record both the first experiment as an example and the average over 10 experiments on separately randomized training and test data sets. We employed Clingo (v. 5.4.0) as the answer-set solver in all experiments.

For the ionosphere data, the Booleanization via median is rough for the real-valued radar readings. Thus we expect larger errors compared to methods using real numbers. This indeed happens, but the errors are still surprisingly low.

Overfitting and Choice of Explanations. The six plots show how the error rates develop with formula length. In all plots, the error of the test data eventually stays roughly the same while the error of the training data keeps decreasing. This illustrates how the overfitting phenomenon ultimately arises. We can use these results to find a *cut-off point* for the length of the formulas to be used as explanations. Note that this should be done on a case-by-case basis and we show the average plots only to demonstrate trends. For the single tests given on the left in Figs. 1, 2 and 3, we might choose the lengths 6, 8 and 7 for the credit, breast cancer and ionosphere data sets, respectively. The errors of the chosen



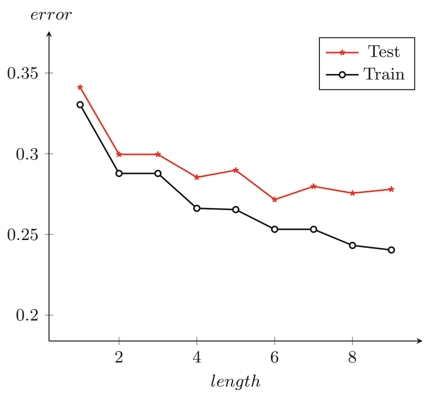
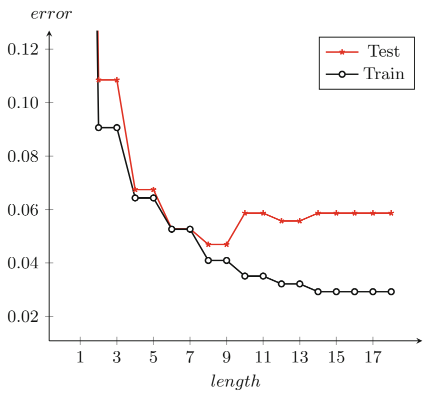


Fig. 1. Credit data set – first test (left) and average (right)



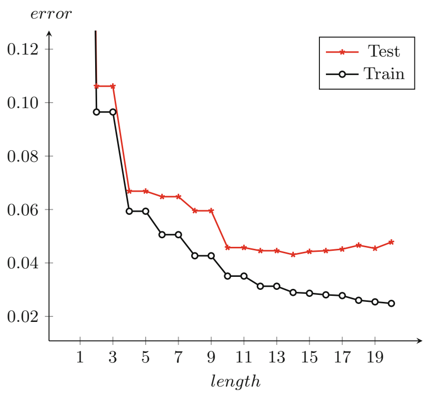
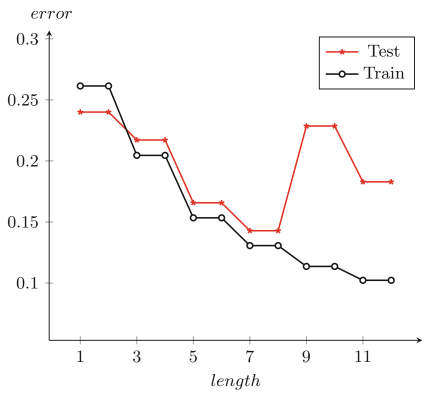


Fig. 2. Breast cancer data set – first test (left) and average (right)



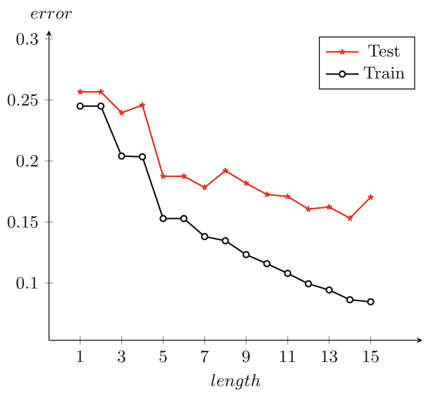


Fig. 3. Ionosphere data set – first test (left) and average (right)

formulas are 0.27, 0.047 and 0.14, respectively. We conclude that by sticking to short Boolean formulas, we can avoid overfitting in a simple way.

Interpretability. A nice feature of short Boolean formulas is their interpretability. Suppose we stop at the formula lengths 6, 8 and 7 suggested above. The related formulas are simple and indeed readable. Consider the formula

$$\neg(a[1,1] \wedge a[2]) \vee a[17,4]$$

of length 6 and a test error of 0.27 obtained from the credit data. The meanings of the attributes are as follows: $a[1,1]$ means the applicant has a checking account with negative balance, $a[2]$ means that the duration of the applied loan is above median, and $a[17,4]$ means the applicant is employed at a management or similar level. (The second number in some attributes refers to original categories in the data.) Therefore the formula states that if an applicant is looking for a short term loan, has money on their account or has a management level job, then they should get the loan. For the breast cancer data set, we choose the formula

$$\neg(((a[1] \wedge a[6]) \vee a[5]) \wedge a[3])$$

of length 8 with test error 0.047. The meanings of the attributes in the order of appearance in the formula are given as clump thickness, bare nuclei, single epithelial cell size and uniformity of cell shape. The full power of Boolean logic is utilized here, in the form of both negation and alternation between conjunction and disjunction. Finally, for the ionosphere data set, the formula

$$((a[8] \wedge a[12]) \vee a[15]) \wedge a[1]$$

of length 7 and test error 0.14 is likewise human readable as a formula. However, it must be mentioned again that the data was used here for technical reasons, and the Booleanized attributes related to radar readings are difficult to explicate.

Using the power of Boolean logic (i.e., including all the connectives $\neg$, $\wedge$, $\vee$) tends to compress the explanations suitably in addition to giving flexibility in explanations. We observe that our experiments gave short, readable formulas.

Comparing Error Rates on Test Data. In [25], all three data sets we consider are treated with *naive Bayesian classifiers* and error rates 0.25, 0.026 and 0.10 are achieved on the test data for the credit, breast cancer and ionosphere data sets, respectively. In [10], the credit data is investigated using neural networks and even there, the best reported error rate is 0.24. In [22], many different methods are compared on the breast cancer data, and the best error achieved is 0.032. For the ionosphere data, the original paper [21] uses neural networks to obtain an error of 0.04. We can see from the plots that very short Boolean formulas can achieve error rates of a similar magnitude on the credit and breast cancer data sets. For the ionosphere data, neural networks achieve a better error rate, but as explained earlier, this is unsurprising as we use a roughly Booleanized version of the underlying data. We conclude that very short Boolean formulas give surprisingly good error rates compared to other methods. Furthermore, this approach seems inherently interpretable for many different purposes.

Runtime Behavior. When computing explanations, no strict timeout was set and the runs were finished only when the optimum was found. Figure 4 depicts the average runtime (over the 10 runs) as a function of formula length. The number of attributes (i.e., 68, 34 and 9 in the order of the curves) appears to be a key factor affecting the performance. Maximum runtimes (approx. 10 hours) indicate the feasibility of our approach, as demonstrated here for realistic data sets previously explored in the literature. Besides minimal explanations, intermediate ones may also be useful.

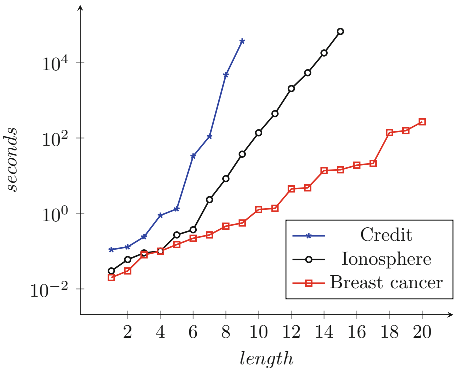


Fig. 4. Average runtimes for data sets

6 Conclusion

We have studied short Boolean formulas as a platform for producing explanations and interpretable classifiers. We have investigated the theoretical reasons behind overfitting and provided related quantitative bounds. Also, we have tested the approach with three different data sets, where the resulting formulas are indeed interpretable—all being genuinely short—and relatively accurate. In general, short formulas may sometimes be necessary to avoid overfitting, and moreover, shorter length leads to increased interpretability.

Our approach need not limit to Boolean formulas only, as we can naturally extend our work to general relational data. We can use, e.g., description logics and compute concepts $C_1, \ldots, C_k$ and then perform our procedure using $C_1, \ldots, C_k$, finding short Boolean combinations of concepts. This of course differs from the approach of computing minimal length formulas in the original description logic, but can nevertheless be fruitful and interesting. We leave this for future work. Further future directions include, e.g., knowledge discovery via computing all formulas up to some short length ℓ with errors smaller than a given threshold.

Acknowledgments. T. Janhunen, A. Kuusisto, M. F. Rankooh and M. Vilander were supported by the Academy of Finland consortium project *Explaining AI via Logic* (XAILOG), grant numbers 345633 (Janhunen) and 345612 (Kuusisto). A. Kuusisto was also supported by the Academy of Finland project *Theory of computational logics*, grant numbers 352419, 352420, 353027, 324435, 328987. The author names have been ordered on the basis of alphabetical order.

References

1. Alviano, M., Dodaro, C., Leone, N., Ricca, F.: Advances in WASP. In: LPNMR 2015, pp. 40–54 (2015)
2. Audemard, G., Bellart, S., Bounia, L., Koriche, F., Lagniez, J., Marquis, P.: On the computational intelligibility of Boolean classifiers. In: Bienvenu, M., Lakemeyer, G., Erdem, E. (eds.) Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning, KR 2021, Online event, 3–12 November 2021, pp. 74–86 (2021)
3. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Occam's razor. Inf. Process. Lett. **24**(6), 377–380 (1987)
4. Blumer, A., Ehrenfeucht, A., Haussler, D., Warmuth, M.K.: Learnability and the Vapnik-Chervonenkis dimension. J. ACM **36**(4), 929–965 (1989)
5. Board, R.A., Pitt, L.: On the necessity of Occam algorithms. Theor. Comput. Sci. **100**(1), 157–184 (1992)
6. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011)
7. Feldman, J.: Minimization of Boolean complexity in human learning. Nature **407**(6804), 630–633 (2022)
8. Gebser, M., Kaminski, R., Kaufmann, B., Romero, J., Schaub, T.: Progress in clasp series 3. In: LPNMR 2015, pp. 368–383 (2015)
9. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, Williston (2012)
10. Griffith, J., O'Dea, P., O'Riordan, C.: A neural net approach to data mining: classification of users to aid information management. In: Szczepaniak, P.S., Segovia, J., Kacprzyk, J., Zadeh, L.A. (eds.) Intelligent Exploration of the Web. Studies in Fuzziness and Soft Computing, vol. 111, pp. 389–401. Physica, Heidelberg (2003). https://doi.org/10.1007/978-3-7908-1772-0_23
11. Jaakkola, R., Janhunen, T., Kuusisto, A., Rankooh, M.F., Vilander, M.: Explainability via short formulas: the case of propositional logic with implementation. In: RCRA 2022. CEUR Workshop Proceedings, vol. 3281, pp. 64–77. CEUR-WS.org (2022)
12. Jaakkola, R., Janhunen, T., Kuusisto, A., Rankooh, M.F., Vilander, M.: Explainability via short formulas: the case of propositional logic with implementation. CoRR abs/2209.01403 (2022)
13. Jaakkola, R., Janhunen, T., Kuusisto, A., Rankooh, M.F., Vilander, M.: Short boolean formulas as explanations in practice. CoRR abs/2307.06971 (2023)
14. Janhunen, T., Niemelä, I.: The answer set programming paradigm. AI Mag. **37**(3), 13–24 (2016)
15. Kearns, M.J., Vazirani, U.: An Introduction to Computational Learning Theory. The MIT Press, Cambridge (1994)
16. Kotz, S., Balakrishnan, N., Johnson, N.: Continuous Multivariate Distributions, Volume 1: Models and Applications. Continuous Multivariate Distributions, Wiley, Hoboken (2004)
17. Mitchell, T.M.: Generalization as search. Artif. Intell. **18**(2), 203–226 (1982)
18. Pelekis, C., Ramon, J.: A lower bound on the probability that a binomial random variable is exceeding its mean. Stat. Probab. Lett. **119**, 305–309 (2016)
19. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)

20. Shih, A., Choi, A., Darwiche, A.: A symbolic approach to explaining Bayesian network classifiers. In: Lang, J. (ed.) IJCAI, pp. 5103–5111 (2018)
21. Sigillito, V.G., Wing, S.P., Hutton, L.V., Baker, K.B.: Classification of radar returns from the ionosphere using neural networks. J. Hopkins APL Tech. Dig. **10**, 262–266 (1989)
22. Šter, B., Dobnikar, A.: Neural networks in medical diagnosis: comparison with other methods. In: Proceedings of the International Conference on Engineering Applications of Neural Networks (1996)
23. Valiant, L.G.: A theory of the learnable. Commun. ACM **27**(11), 1134–1142 (1984)
24. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (2013). https://doi.org/10.1007/978-1-4757-3264-1
25. Yang, Y., Webb, G.I.: Proportional k-interval discretization for Naive-Bayes classifiers. In: De Raedt, L., Flach, P. (eds.) ECML 2001. LNCS (LNAI), vol. 2167, pp. 564–575. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44795-4_48