Explaining classifiers and data via propositional logic

Reijo Jaakkola
reijo.jaakkola@tuni.fi

Tampere University

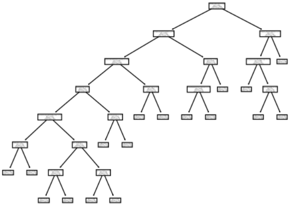
November 23, 2023

Joint work with Tomi Janhunen, Antti Kuusisto, Masood Feyzbakhsh Rankooh and Miikka Vilander.

Overview

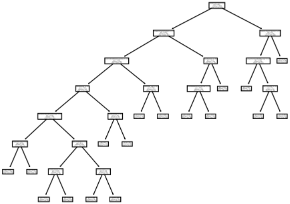
1. Background
2. Explaining classifiers with propositional logic
3. Explaining data with propositional logic

Explainability and interpretability in machine learning



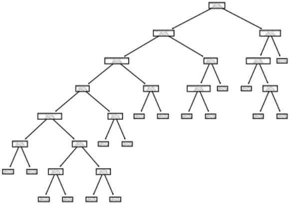
- In machine learning the goal is to learn algorithms/models/classifiers from data.

Explainability and interpretability in machine learning



- In machine learning the goal is to learn algorithms/models/classifiers from data.
- A practical challenge is that these classifiers are often black boxes.

Explainability and interpretability in machine learning



- In machine learning the goal is to learn algorithms/models/classifiers from data.
- A practical challenge is that these classifiers are often black boxes.
- Essentially two options for overcoming these challenges: develop methods for either **explaining** classifiers or for producing classifiers that are inherently **interpretable**.

Propositional logic

- In two recent works — one published in RCRA 2022 and the second one in JELIA 2023 — we investigated both approaches using propositional logic.

Propositional logic

- In two recent works — one published in RCRA 2022 and the second one in JELIA 2023 — we investigated both approaches using propositional logic.
- Given a finite set Φ of proposition symbols, we use $\mathrm{PL}[\Phi]$ to denote the set of formulas generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi,$$

where $p \in \Phi$.

Propositional logic

- In two recent works — one published in RCRA 2022 and the second one in JELIA 2023 — we investigated both approaches using propositional logic.
- Given a finite set Φ of proposition symbols, we use $\mathrm{PL}[\Phi]$ to denote the set of formulas generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi,$$

  where $p \in \Phi$. The **size** $\mathrm{size}(\varphi)$ of φ is defined as the number of proposition symbols + the number of connectives that occur in φ.

Propositional logic

- In two recent works — one published in RCRA 2022 and the second one in JELIA 2023 — we investigated both approaches using propositional logic.
- Given a finite set Φ of proposition symbols, we use $\mathrm{PL}[\Phi]$ to denote the set of formulas generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi,$$

  where $p \in \Phi$. The **size** $\mathrm{size}(\varphi)$ of φ is defined as the number of proposition symbols + the number of connectives that occur in φ.
- Formulas of $\mathrm{PL}[\Phi]$ are evaluated on Φ-valuations v, i.e., mappings $v : \Phi \to \{0, 1\}$.

Propositional logic

- In two recent works — one published in RCRA 2022 and the second one in JELIA 2023 — we investigated both approaches using propositional logic.
- Given a finite set Φ of proposition symbols, we use $\mathrm{PL}[\Phi]$ to denote the set of formulas generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi,$$

  where $p \in \Phi$. The **size** $\mathrm{size}(\varphi)$ of φ is defined as the number of proposition symbols + the number of connectives that occur in φ.
- Formulas of $\mathrm{PL}[\Phi]$ are evaluated on Φ-valuations v, i.e., mappings $v : \Phi \to \{0, 1\}$.
- Formulas $\sim$ classifiers and valuations $\sim$ data points.

Explaining classifiers: motivating examples

Example

1. Consider the following valuation $v : \{p, q, r\} \to \{0, 1\}$:

$$v(p) = 1,\ v(q) = 1,\ v(r) = 0$$

Let $\varphi := p \land (q \lor \neg r)$. Clearly $v(\varphi) = 1$, i.e., φ **accepts** the valuation v. But **why**? One possible **explanation** for this is to note that φ accepts **any** valuation which maps p and q to one.

Explaining classifiers: motivating examples

Example

1. Consider the following valuation $v : \{p, q, r\} \to \{0, 1\}$:

   $$v(p) = 1,\ v(q) = 1,\ v(r) = 0$$

   Let $\varphi := p \wedge (q \vee \neg r)$. Clearly $v(\varphi) = 1$, i.e., φ **accepts** the valuation v. But **why**? One possible **explanation** for this is to note that φ accepts **any** valuation which maps p and q to one.

2. On the other hand, consider the following valuation $v : \{p, q, r\} \to \{0, 1\}$:

   $$v(p) = 1,\ v(q) = 0,\ v(r) = 1$$

   If φ is the same formula as above, then $v(\varphi) = 0$. This time this could be explained by observing that φ rejects **any** assignment that maps q to zero and r to one.

Special explainability problem

Input: a tuple (v, φ, b, k), where

Special explainability problem

Input: a tuple (v, φ, b, k), where

1. v is a Φ-valuation

Special explainability problem

Input: a tuple (v, φ, b, k), where

1. v is a Φ-valuation
2. φ is a formula of $\mathrm{PL}[\Phi]$ such that $v(\varphi) = b$

Special explainability problem

Input: a tuple (v, φ, b, k), where

1. v is a Φ-valuation
2. φ is a formula of $\mathrm{PL}[\Phi]$ such that $v(\varphi) = b$
3. $k \in \mathbb{N}$

Special explainability problem

Input: a tuple (v, φ, b, k), where

1. v is a Φ-valuation
2. φ is a formula of $\mathrm{PL}[\Phi]$ such that $v(\varphi) = b$
3. $k \in \mathbb{N}$

Output: **Yes**, if there exists a formula (= an explanation) ψ of $\mathrm{PL}[\Phi]$ with the following properties.

Special explainability problem

Input: a tuple (v, φ, b, k), where

1. v is a Φ-valuation
2. φ is a formula of $\mathrm{PL}[\Phi]$ such that $v(\varphi) = b$
3. $k \in \mathbb{N}$

Output: **Yes**, if there exists a formula (= an explanation) ψ of $\mathrm{PL}[\Phi]$ with the following properties.

1. $\mathrm{size}(\psi) \leq k$

Special explainability problem

Input: a tuple (v, φ, b, k), where

1. v is a Φ-valuation
2. φ is a formula of $\mathrm{PL}[\Phi]$ such that $v(\varphi) = b$
3. $k \in \mathbb{N}$

Output: **Yes**, if there exists a formula (= an explanation) ψ of $\mathrm{PL}[\Phi]$ with the following properties.

1. $\mathrm{size}(\psi) \leq k$
2. $v(\psi) = b$

Special explainability problem

Input: a tuple (v, φ, b, k), where

1. v is a Φ-valuation
2. φ is a formula of $\mathrm{PL}[\Phi]$ such that $v(\varphi) = b$
3. $k \in \mathbb{N}$

Output: **Yes**, if there exists a formula (= an explanation) ψ of $\mathrm{PL}[\Phi]$ with the following properties.

1. $\mathrm{size}(\psi) \leq k$
2. $v(\psi) = b$
3. for every Φ-assignment u we have the following implication

$$u(\psi) = b \Rightarrow u(\varphi) = b$$

Special explainability problem

Input: a tuple (v, φ, b, k), where

1. v is a Φ-valuation
2. φ is a formula of $\mathrm{PL}[\Phi]$ such that $v(\varphi) = b$
3. $k \in \mathbb{N}$

Output: **Yes**, if there exists a formula (= an explanation) ψ of $\mathrm{PL}[\Phi]$ with the following properties.

1. $\mathrm{size}(\psi) \leq k$
2. $v(\psi) = b$
3. for every Φ-assignment u we have the following implication

$$u(\psi) = b \Rightarrow u(\varphi) = b$$

Otherwise the answer is **No**.

Special explainability problem: examples

Example

1. Consider the following valuation $v : \{p, q, r\} \rightarrow \{0, 1\}$:

$$v(p) = 1,\ v(q) = 1,\ v(r) = 0$$

Let $\varphi := p \wedge (q \vee \neg r)$. On input $(v, \varphi, 1, 3)$ the answer is **Yes**, as witnessed by $p \wedge q$.

Special explainability problem: examples

Example

1. Consider the following valuation $v : \{p, q, r\} \to \{0, 1\}$:

$$v(p) = 1, \ v(q) = 1, \ v(r) = 0$$

   Let $\varphi := p \wedge (q \vee \neg r)$. On input $(v, \varphi, 1, 3)$ the answer is **Yes**, as witnessed by $p \wedge q$.

2. Consider the following valuation $v : \{p, q, r\} \to \{0, 1\}$:

$$v(p) = 1, \ v(q) = 0, \ v(r) = 1$$

   Let φ be as above. On input $(v, \varphi, 0, 4)$ the answer is **Yes**, as witnessed by $\neg q \wedge r$.

Useful observation

Lemma (Jaakkola et al., RCRA 2022)

Let (v, φ, b, k) be an input to the special explainability problem. Let ψ_v denote the conjunction that corresponds to v:

$$\bigwedge_{v(p)=1} p \wedge \bigwedge_{v(p)=0} \neg p$$

*Now, if there is an explanation of size at most k for $v(\varphi) = b$, then there is one which is **essentially** a subconjunction of ψ_v.*

Special explainability problem for monotone formulas

Theorem

For monotone formulas the special explainability problem is NP*-complete. In fact, the lower bound holds already for monotone* CNF*-formulas.*

Special explainability problem for monotone formulas

Theorem

For monotone formulas the special explainability problem is NP*-complete. In fact, the lower bound holds already for monotone* CNF*-formulas.*

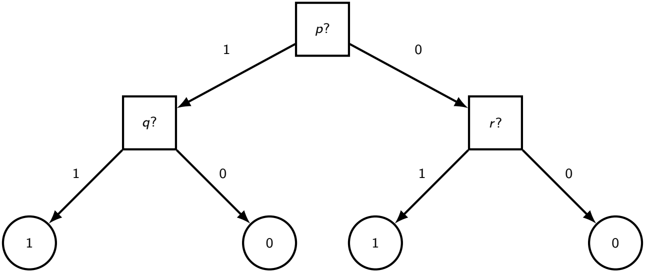
Proof.

Lower bound from the dominating set problem. Let $G = (V, E)$ be a graph. For each $x \in V$ we associate a propositional symbol p_x. Consider now the formula

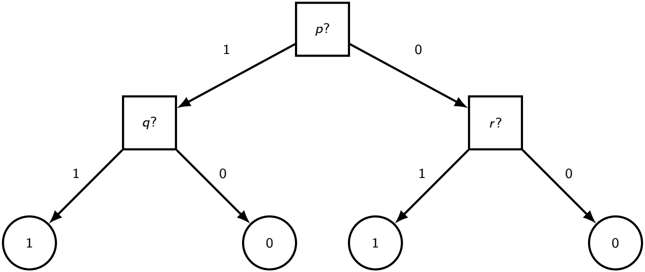
$$\varphi := \bigwedge_{x \in V} \Big(p_x \vee \bigvee_{(x,y) \in E} p_y \Big)$$

and set $v(p_x) = 1$, for every $x \in V$. Now, on input $(v, \varphi, 1, 2k-1)$ the answer is **Yes** iff G has a dominating set of size at most k. □

Special explainability problem for decision trees



Special explainability problem for decision trees



- In [Barceló et al., NeurIPS 2020] it was essentially established that the special explainability problem for Boolean decision trees is NP-complete. Lower bound is again from the dominating set problem, but the reduction is quite fancy.

Special explainability problem for full PL

Theorem (Jaakkola et al., RCRA 2022)

The special explainability problem for PL *is* Σ_2^p*-complete.*

Special explainability problem for full PL

Theorem (Jaakkola et al., RCRA 2022)

The special explainability problem for PL *is* Σ_2^p*-complete.*

Proof.

For the lower bound a reduction from $\Sigma_2\mathrm{SAT}$. Fix a quantified Boolean formula

$$\exists p_1 \ldots \exists p_n \forall q_1 \ldots \forall q_m \theta$$

and consider the following propositional formula:

$$\varphi \ := \ \bigwedge_{i=1}^{n} (p_i \vee \overline{p}_i) \wedge \left(\bigvee_{i=1}^{n} (p_i \wedge \overline{p}_i) \vee \theta \right).$$

Set v to be a valuation which maps all the propositional symbols to one. Now, on input $(v, \varphi, 1, 2n-1)$ the answer is **Yes** iff the original instance of $\Sigma_2\mathrm{SAT}$ is true. $\square$

Special explainability problem for DNF-formulas

Theorem

The special explainability problem for DNF*-formulas and* CNF*-formulas is* Σ_2^p*-hard.*

Special explainability problem for DNF-formulas

Theorem

The special explainability problem for DNF*-formulas and* CNF*-formulas is* Σ_2^p*-hard.*

- Can be proved via a reduction from the **shortest implicant problem**: on input (π, φ, k), where
  1. φ is a DNF-formula
  2. π is a conjunction of literals and $k \in \mathbb{N}$,

  decide whether there exists a subconjunction $\pi' \subseteq \pi$ with at most k literals such that $\pi' \models \varphi$. Proved in [Umans, 2001] to be Σ_2^p-complete.

Special explainability problem for DNF-formulas

Theorem

The special explainability problem for DNF-formulas and CNF-formulas is Σ_2^p-hard.

- Can be proved via a reduction from the **shortest implicant problem**: on input (π, φ, k), where
  1. φ is a DNF-formula
  2. π is a conjunction of literals and $k \in \mathbb{N}$,

  decide whether there exists a subconjunction $\pi' \subseteq \pi$ with at most k literals such that $\pi' \models \varphi$. Proved in [Umans, 2001] to be Σ_2^p-complete.
- To get the Σ_2^p-hardness for DNF-formulas, one essentially needs to show that the shortest implicant problem remains Σ_2^p-hard even if π is a **maximal** conjunction of literals.

Special cases of the special explainability problem

- For CNF-formulas the validity problem is solvable in polynomial time, because we can just check whether all of the clauses contain p and $\neg p$. Using this, given a conjunction χ and a CNF-formula φ we can also check $\chi \models \varphi$ in polynomial time.

Special cases of the special explainability problem

- For CNF-formulas the validity problem is solvable in polynomial time, because we can just check whether all of the clauses contain p and $\neg p$. Using this, given a conjunction χ and a CNF-formula φ we can also check $\chi \models \varphi$ in polynomial time.

Theorem (Jaakkola et al., RCRA 2022)

Restrict attention to those instances of the special explainability problem where $b = 1$. Then the problem is NP-complete for CNF-formulas.

Special cases of the special explainability problem

- For CNF-formulas the validity problem is solvable in polynomial time, because we can just check whether all of the clauses contain p and $\neg p$. Using this, given a conjunction χ and a CNF-formula φ we can also check $\chi \models \varphi$ in polynomial time.

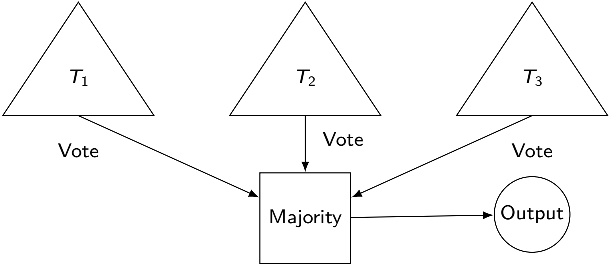
Theorem (Jaakkola et al., RCRA 2022)

Restrict attention to those instances of the special explainability problem where $b = 1$. Then the problem is NP-complete for CNF-formulas.

Corollary (Jaakkola et al., RCRA 2022)

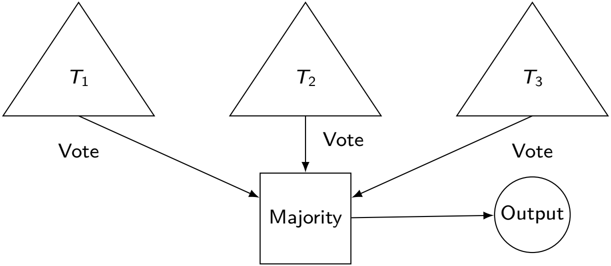
Restrict attention to those instances of the special explainability problem where $b = 0$. Then the problem is NP-complete for DNF-formulas.

Application: How hard it is to explain a random forest?



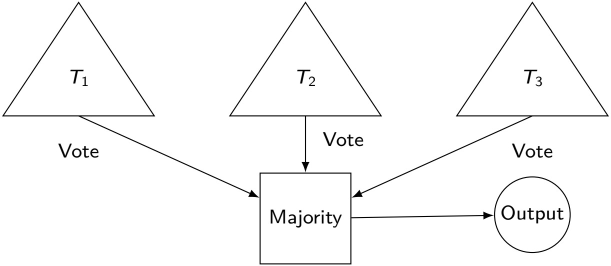
- A **random forest** is essentially a classifier that consists of a set of decision trees. Given an instance, the trees "vote" on what label the instance should receive.

Application: How hard it is to explain a random forest?



- A **random forest** is essentially a classifier that consists of a set of decision trees. Given an instance, the trees "vote" on what label the instance should receive.
- A very popular machine learning model (e.g., easy to train).

Application: How hard it is to explain a random forest?



- A **random forest** is essentially a classifier that consists of a set of decision trees. Given an instance, the trees "vote" on what label the instance should receive.
- A very popular machine learning model (e.g., easy to train).
- **Caveat:** much harder to interpretate than decision trees.

Application: How hard it is to explain a random forest?

Theorem

The special explainability problem for random forests is Σ_2^p*-complete.*

Application: How hard it is to explain a random forest?

Theorem

The special explainability problem for random forests is Σ_2^p-complete.

Proof.

A reduction from the case of DNF-formulas. Let $\varphi := t_1 \vee \cdots \vee t_m$ be an arbitrary DNF-formula. Each t_i is just a conjunction of literals and hence can be implemented via a linear size decision tree T_i. Now, the following is a random forest that is equivalent with φ:

$$F := \{T_1, \ldots, T_m, \underbrace{\top, \ldots, \top}_{m\text{-times}}\}$$

Thus we can reduce efficiently the SE problem of DNF formulas to that of random forests. $\square$

Related work: PI-explanations

- A conjunction π of literals is a **prime implicant** of an another formula φ, if $\pi \models \varphi$ and for every $\pi' \subset \pi$ we have that $\pi' \not\models \varphi$.

Related work: PI-explanations

- A conjunction π of literals is a **prime implicant** of an another formula φ, if $\pi \models \varphi$ and for every $\pi' \subset \pi$ we have that $\pi' \not\models \varphi$.
- Lot of recent implementation work on computing prime implicants for Boolean classifiers, see e.g. [Darwiche, LICS 2023] for references. The related algorithms are not trying to find prime implicants that are **globally** optimal.

Related work: PI-explanations

- A conjunction π of literals is a **prime implicant** of an another formula φ, if $\pi \models \varphi$ and for every $\pi' \subset \pi$ we have that $\pi' \not\models \varphi$.
- Lot of recent implementation work on computing prime implicants for Boolean classifiers, see e.g. [Darwiche, LICS 2023] for references. The related algorithms are not trying to find prime implicants that are **globally** optimal.
- Related literature also contains results on the computational complexity of determining whether a given conjunction is a prime implicant. E.g. for arbitrary formulas of propositional logic it is D^{P}-complete.

Explaining data: motivating example

Example

Consider the following (very small) Boolean data set over a vocabulary $\{p_1, p_2, p_3, q\}$:

p_1	p_2	p_3	q
1	1	1	1
1	1	0	1
1	0	0	0
0	1	0	0

Using the symbols p_1, p_2, p_3 we would like to **predict** the value of q.

Explaining data: motivating example

Example

Consider the following (very small) Boolean data set over a vocabulary $\{p_1, p_2, p_3, q\}$:

p_1	p_2	p_3	q
1	1	1	1
1	1	0	1
1	0	0	0
0	1	0	0

Using the symbols p_1, p_2, p_3 we would like to **predict** the value of q. Based on this data, one formula that seems to work well is $(p_1 \wedge p_2)$.

Theoretical and empirical error

- Fix a vocabulary $\Phi = \{p_1, \ldots, p_k\}$ and $q \notin \Phi$. Set $\Phi_q := \Phi \cup \{q\}$. We assume an underlying but **unknown** probability distribution

$$\mu : \{\Phi_q\text{-assignments}\} \to [0, 1]$$

μ encodes how q depends on the propositions from Φ.

Theoretical and empirical error

- Fix a vocabulary $\Phi = \{p_1, \ldots, p_k\}$ and $q \notin \Phi$. Set $\Phi_q := \Phi \cup \{q\}$. We assume an underlying but **unknown** probability distribution

$$\mu : \{\Phi_q\text{-assignments}\} \to [0, 1]$$

  μ encodes how q depends on the propositions from Φ.

- **Goal** is to find a formula φ of $\mathrm{PL}[\Phi]$ which minimizes the **theoretical (ideal) error**:

$$\mathrm{err}_\mu(\varphi) := \Pr_{v \sim \mu}[v(\varphi) \neq v(q)]$$

Theoretical and empirical error

- Fix a vocabulary $\Phi = \{p_1, \ldots, p_k\}$ and $q \notin \Phi$. Set $\Phi_q := \Phi \cup \{q\}$. We assume an underlying but **unknown** probability distribution

$$\mu : \{\Phi_q\text{-assignments}\} \to [0,1]$$

  μ encodes how q depends on the propositions from Φ.

- **Goal** is to find a formula φ of $\mathrm{PL}[\Phi]$ which minimizes the **theoretical (ideal) error**:

$$\mathrm{err}_\mu(\varphi) := \Pr_{v \sim \mu}[v(\varphi) \neq v(q)]$$

- Since μ is unknown, we can not calculate $\mathrm{err}_\mu(\varphi)$ directly. Instead, we approximate it by taking samples S ($=$ data sets) from μ. We then evaluate formulas based on their **empirical error**:

$$\mathrm{err}_S(\varphi) := \frac{1}{|S|} \sum_{\substack{v \in S \\ v(\varphi) \neq v(q)}} 1$$

Overfitting

- No guarantee that $\mathrm{err}_S(\varphi)$ is close to $\mathrm{err}_\mu(\varphi)$, unless the sample S is sufficiently large.

Overfitting

- No guarantee that $\mathrm{err}_S(\varphi)$ is close to $\mathrm{err}_\mu(\varphi)$, unless the sample S is sufficiently large.

Example

Consider the following sample:

p_1	p_2	q
1	1	1
1	1	1
0	1	0

Overfitting

- No guarantee that $\mathrm{err}_S(\varphi)$ is close to $\mathrm{err}_\mu(\varphi)$, unless the sample S is sufficiently large.

Example

Consider the following sample:

p_1	p_2	q
1	1	1
1	1	1
0	1	0

Both p_1 and

$$(p_1 \wedge p_2) \vee (\neg p_1 \wedge \neg p_2)$$

obtain good accuracy on this data set. However, we might expect the first formula to perform better outside this particular sample.

Overfitting

- No guarantee that $\mathrm{err}_S(\varphi)$ is close to $\mathrm{err}_\mu(\varphi)$, unless the sample S is sufficiently large.

Example

Consider the following sample:

p_1	p_2	q
1	1	1
1	1	1
0	1	0

Both p_1 and

$$(p_1 \wedge p_2) \vee (\neg p_1 \wedge \neg p_2)$$

obtain good accuracy on this data set. However, we might expect the first formula to perform better outside this particular sample.

- Intuitively, a formula **overfits** to a sample if it focuses too much on the particular features of that sample.

Sample bounds

- In principle, overfitting can be avoided by using a large enough sample.

Sample bounds

- In principle, overfitting can be avoided by using a large enough sample. One can even try to establish formally bounds on how large samples are needed for avoiding overfitting, see statistical learning theory.

Sample bounds

- In principle, overfitting can be avoided by using a large enough sample. One can even try to establish formally bounds on how large samples are needed for avoiding overfitting, see statistical learning theory.

Theorem (Uniform convergence for PL)

Let $k \in \mathbb{Z}_+$ and $\delta > 0$. If a sample S of size at least

$$\frac{1}{2\varepsilon^2}(5k\log(k) + \ln(2/\delta))$$

is sampled from μ, then with probability greater than or equal to $1-\delta$, every formula φ of PL of size at most k satisfies $|\mathrm{err}_\mu(\varphi) - \mathrm{err}_S(\varphi)| < \varepsilon$.

Sample bounds

- In principle, overfitting can be avoided by using a large enough sample. One can even try to establish formally bounds on how large samples are needed for avoiding overfitting, see statistical learning theory.

Theorem (Uniform convergence for PL)

Let $k \in \mathbb{Z}_+$ and $\delta > 0$. If a sample S of size at least

$$\frac{1}{2\varepsilon^2}(5k\log(k) + \ln(2/\delta))$$

is sampled from μ, then with probability greater than or equal to $1-\delta$, every formula φ of PL of size at most k satisfies $|\mathrm{err}_\mu(\varphi) - \mathrm{err}_S(\varphi)| < \varepsilon$.

- Most of the time these type of bounds are not good enough to be practically relevant.

Cross-validation

- **Idea:** split the original sample into two parts — the **training set** and the **test set** — and do the following for increasing values of k:

Cross-validation

- **Idea:** split the original sample into two parts — the **training set** and the **test set** — and do the following for increasing values of k:

  1. Find/learn a formula φ of size at most k that has small error over the training set.

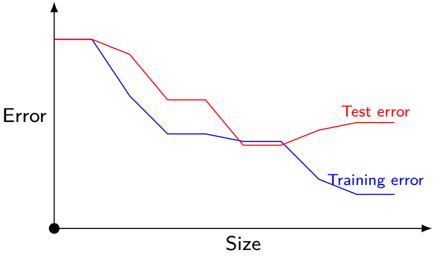
Cross-validation

- **Idea:** split the original sample into two parts — the **training set** and the **test set** — and do the following for increasing values of k:

  1. Find/learn a formula φ of size at most k that has small error over the training set.
  2. Check that φ also has a small error over the test set.

Cross-validation

- **Idea:** split the original sample into two parts — the **training set** and the **test set** — and do the following for increasing values of k:
  1. Find/learn a formula φ of size at most k that has small error over the training set.
  2. Check that φ also has a small error over the test set.



How to learn a Boolean formula?

- In our JELIA 2023 paper we simply did the following: for increasing values of k, we found a formula of size at most k which had the smallest possible training error.

How to learn a Boolean formula?

- In our JELIA 2023 paper we simply did the following: for increasing values of k, we found a formula of size at most k which had the smallest possible training error.
- Cross-validation was used for determining when the formulas started to overfit.

How to learn a Boolean formula?

- In our JELIA 2023 paper we simply did the following: for increasing values of k, we found a formula of size at most k which had the smallest possible training error.
- Cross-validation was used for determining when the formulas started to overfit.
- Computationally very difficult, because of the massive search space.

How to learn a Boolean formula?

- In our JELIA 2023 paper we simply did the following: for increasing values of k, we found a formula of size at most k which had the smallest possible training error.
- Cross-validation was used for determining when the formulas started to overfit.
- Computationally very difficult, because of the massive search space.
- We tested our Answer Set Programming based implementation on three data sets from the UCI machine learning repository.

How to learn a Boolean formula?

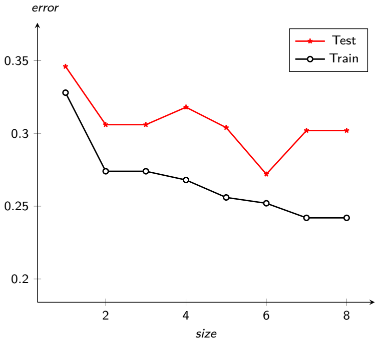
- In our JELIA 2023 paper we simply did the following: for increasing values of k, we found a formula of size at most k which had the smallest possible training error.
- Cross-validation was used for determining when the formulas started to overfit.
- Computationally very difficult, because of the massive search space.
- We tested our Answer Set Programming based implementation on three data sets from the UCI machine learning repository.
- Each data set contained several non-Boolean attributes, so Booleanization was needed.

Empirical results

- The first data set was the **Statlog-German credit** data set, classifies persons based on whether or not it is "risky" to give them a loan.
- 1000 data points and 68 attributes.
- The following formula of size six

  $$\neg(\text{negative\_balance} \wedge \text{above\_median\_loan\_duration}) \\ \vee \text{employment\_on\_managerial\_level}$$

  had 0.27 as its test error.
- Naive Bayes classifiers had test error 0.25 and neural networks 0.24.

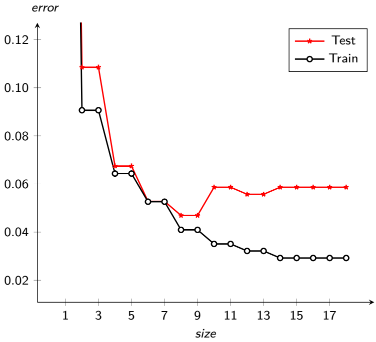


Empirical results

- The second data set was **Breast cancer Wisconsin** data set, classifies tumors based on whether or not they are benign.
- 683 data points ja 9 attributes.
- The following formula of size eight

$$\neg(((p \wedge q) \vee r) \wedge s)$$

  had 0.047 as its test error.
- Naive Bayes classifiers had test error 0.026 and neural networks 0.032.

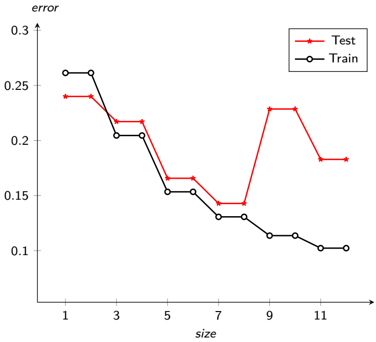


Empirical results

- The final data set was the **Ionosphere** data set, classifies radar signals based on whether or not they are “good”.
- 351 data points and 34 attributes.
- The following formula of size seven

$$((p \wedge q) \vee r) \wedge s$$

  had 0.14 as its test error.
- Naive Bayes classifiers had test error 0.1 and neural networks 0.04.



When do we need black box models?

Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead

When do we need black box models?

Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead

- In [Rudin, 2019] it is argued — based on empirical evidence — that for most problems with "meaningful structured covariates" different machine learning algorithms tend to perform similarly.

When do we need black box models?

Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead

- In [Rudin, 2019] it is argued — based on empirical evidence — that for most problems with "meaningful structured covariates" different machine learning algorithms tend to perform similarly.
- In particular, we might expect that interpretable models achieve similar accuracies as black box models (such as large neural networks). Our empirical results certainly support this claim.

When do we need black box models?

Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead

- In [Rudin, 2019] it is argued — based on empirical evidence — that for most problems with "meaningful structured covariates" different machine learning algorithms tend to perform similarly.
- In particular, we might expect that interpretable models achieve similar accuracies as black box models (such as large neural networks). Our empirical results certainly support this claim.

Thanks! :)