Neural-Symbolic Integration
A self-contained introduction

Sebastian Bader   Pascal Hitzler

ICCL, Technische Universität Dresden, Germany
AIFB, Universität Karlsruhe, Germany
Outline of the Course

- Introduction and Motivation
- The Core Method for Propositional Logic
- Applications of the Propositional Core Method
- A New Approach to Pedagogical Extraction
- The Core Method for First-Order Logic
- More on First-Order & other Perspectives
Part

Pedagogical Extraction: the CoOp Approach
Outline: Pedagogical Extraction: the CoOp Approach

Introduction

Decomposition

Extraction of Coalitions and Oppositions

Composition

Conclusions
Outline: Pedagogical Extraction: the CoOp Approach

Introduction

Decomposition

Extraction of Coalitions and Oppositions

Composition

Conclusions
The CoOp Approach

▶ Joint work with:
  • Steffen Hölldobler (ICCL)
  • Valentin Meyer-Eichberger (TU Dresden)
▶ “Extracting Propositional Rules from Feed-forward Neural Networks — A New Decompositional Approach”
  (Bader, Hölldobler, ea., 2007)
A Small Example

(units compute the $-1/ + 1$ threshold function)
The Rule Extraction Problem

Definition
Let $\mathcal{N} = \langle U, U_{\text{inp}}, U_{\text{out}}, C, \omega, \theta, t^-, t^+ \rangle$ be a threshold feed-forward network. A propositional formula $F$ over $U_{\text{inp}}$ is called a *propositional representation* for some unit $A$ iff for all interpretations $I$ with $I \models F$ we find $A$ to be active under the corresponding input.

Example
$F_c = a \lor b$ is a propositional representation for unit $c$. 
Outline: Pedagogical Extraction: the CoOp Approach

Introduction

Decomposition

Extraction of Coalitions and Oppositions

Composition

Conclusions
Decomposition of the Example Network

\[
\begin{array}{cccccc}
\omega & c & d & e & f & g & h \\
\hline
a & 1 & -2 & 5 & 2 & & \\
b & 1 & 1 & -3 & 1 & & \\
c & & & & & 1 & 0 \\
d & & & & & 2 & -3 \\
e & & & & & 3 & 0 \\
f & & & & & 5 & -2 \\
\end{array}
\]
Decomposition of the Example Network
Input Patterns and their Minimal Input

**Definition (Input pattern)**

Let $\mathcal{P}_A = \langle t^-, t^+, \theta, \mathcal{I}, \omega \rangle$ be a perceptron. A subset $l \subseteq \mathcal{I}$ is called an *input pattern*. The minimal and maximal input wrt. the input pattern $l$ are defined as:

$$i_{\text{min}}(l) = t^+ \cdot \sum_{A \in l} \omega(A) + \sum_{A \in \mathcal{I} \setminus l} \min(t^- \cdot \omega(A), t^+ \cdot \omega(A))$$

and

$$i_{\text{max}}(l) = t^+ \cdot \sum_{A \in l} \omega(A) + \sum_{A \in \mathcal{I} \setminus l} \max(t^- \cdot \omega(A), t^+ \cdot \omega(A)).$$
Input Patterns and their Minimal Input

Example
For $I = \{c, d\}$ we have

\[ i_{\text{min}}(I) = (1.0 + 2.0) - (3.0 + 5.0) = -5.0 \]
\[ i_{\text{max}}(I) = (1.0 + 2.0) + (3.0 + 5.0) = 11.0. \]
Coalitions

**Definition (Coalition)**

Let $\mathcal{P}_A = \langle t^-, t^+, \theta, \mathcal{I}, \omega \rangle$ be a perceptron. Let $I \subseteq \mathcal{I}$ be some input pattern. $I$ is called a *coalition*, if $i_{\text{min}}(I) \geq \theta$. A coalition $I$ is called *minimal*, if none of its subset $I' \subset I$ is a coalition. We will use $C_A$ to denote the set of minimal coalitions for perceptron $\mathcal{P}_A$.

**Example**

For $\mathcal{P}_g$, we find $I = \{c, d, f\}$ to be a coalition, as $i_{\text{min}}(I) = (1.0 + 2.0 + 5.0) - (3.0) = 5.0 > 4.0$. 
Opposition

**Definition (Opposition)**

Let $\mathcal{P}_A = \langle t^-, t^+, \theta, I, \omega \rangle$ be a perceptron. Let $I \subseteq \mathcal{I}$ be some input pattern. $I$ is called an *opposition*, if $i_{\max}(I) < \theta$. An opposition $I$ is called *minimal*, if none of its subset $I' \subset I$ is an opposition. We will use $W_A$ to denote the set of minimal oppositions for perceptron $\mathcal{P}_A$.

**Example**

For $\mathcal{P}_e$, we find that $J = \{b\}$ is an opposition, as $i_{\max}(J) = (-3.0) + (5.0) = 2.0 < 4.0$. 
Definition (Positive form)

Let $\mathcal{P}_A = \langle t^-, t^+, \theta, \mathcal{I}, \omega \rangle$ be a perceptron. Its positive form $\mathcal{P}_A^+$ is defined as follows:

$\mathcal{P}_A^+ := \langle t^-, t^+, \theta, \mathcal{I}', \omega' \rangle$ with

$\mathcal{I}' = \{ \bar{U} \mid U \in \mathcal{I} \text{ and } \omega(U) < 0 \} \cup \{ U \mid U \in \mathcal{I} \text{ and } \omega(U) \geq 0 \}$

$\omega' : \mathcal{I}' \to \mathbb{R} : U \mapsto \begin{cases} \omega(U) & \text{if } U \in \mathcal{I} \\ -\omega(U) & \text{otherwise} \end{cases}$
Positive Perceptrons

Example

The positive forms $P^+_c, \ldots, P^+_h$ of the perceptrons from above:
Negative Perceptrons

Example

The negative forms $\mathcal{P}_c^-, \ldots, \mathcal{P}_h^-$ of the perceptrons from above:
Outline: Pedagogical Extraction: the CoOp Approach

Introduction

Decomposition

Extraction of Coalitions and Oppositions
  A Search Tree to Find Coalitions
  A Pruned Search Tree to Find Coalitions
  Symmetries in the Search Trees
  Binary Decision Diagrams
  From Pruned Search Trees to ROBDDs

Composition

Conclusions
Let $\mathcal{P} = \langle t^-, t^+, \theta, I, \omega \rangle$ be a perceptron.

- find all minimal coalitions and oppositions as fast as possible.
- represent them as compact as possible.
A Naive Search Tree
A Naive Search Tree

Definition (Search Tree Node)
Let $P = \langle t^-, t^+, \theta, \mathcal{I}, \omega \rangle$ be a positive perceptron. Let $\prec$ be a fixed linear order on $\mathcal{I}$ such that $A \prec B$ if $\omega(A) \geq \omega(B)$. A search tree node is a pair $\langle I, C \rangle$, with $I \subseteq \mathcal{I}$ and $C$ being a list of child nodes. For each child node $\langle I_i, C_i \rangle$ we find $I_i = I \cup \{A_i\}$ with $A \prec A_i$ for all $A \in I$. The list of children is sorted ascending wrt the minimal input of $I_i$.

Definition (Full Coalition Search Tree)
Let $P = \langle t^-, t^+, \theta, \mathcal{I}, \omega \rangle$ be a positive perceptron. The full coalition search tree $ST_P$ for $P$ is defined to be the fully expanded search tree node $\langle \emptyset, C \rangle$. 
Properties of the Search Tree

The full coalition search tree $ST_P$ ...

- ... is complete (i.e., for each input pattern there exists a corresponding node in the tree)
- ... is sorted wrt. the minimal inputs (ascending from right to left and from top to bottom)
A Pruned Search Tree
A Pruned Search Tree

**Definition (Irrelevant Node)**

Let \( \mathcal{P} = \langle t^-, t^+, \theta, \mathcal{I}, \omega \rangle \) be a perceptron and let \( \text{ST}_\mathcal{P} \) be the corresponding coalition search tree as defined above for the linear order \( \prec \). Let \( i(I) \) be the maximal element of some \( I \subseteq \mathcal{I} \) wrt. \( \prec \) and let \( J(I) \subseteq \mathcal{I} \) be the set of all inputs which are bigger than \( i \), i.e., \( J(I) := \{ j \mid j \in \mathcal{I} \text{ and } i(I) \not\prec j \} \). Let \( r(I) = (t^+ - t^-) \cdot \sum_{j \in J} \omega(j) \). A node \( \langle I, C \rangle \) in \( \text{ST}_\mathcal{P} \) is called irrelevant if \( I \) is a non-minimal coalition, or if \( i_{\min}(I) + r(I) < \theta \).

**Definition (Pruned Coalition Search Tree)**

The pruned coalition search tree \( \text{PT}_\mathcal{P} \) is obtained from \( \text{ST}_\mathcal{P} \) by removing all irrelevant nodes. Please note that if all nodes are irrelevant, the pruned search tree is empty.
Properties of the Pruned Search Tree

The pruned coalition search tree $PT_P$ ...

- ... is complete (i.e., for each input pattern corresponding to a minimal coalition there exists a node in the tree)
- ... is sorted wrt. the minimal inputs (ascending from right to left and from top to bottom)
- ... can easily be constructed by left depth first search
Why more?
Symmetries and Regularities in the Trees

▶ from \{e, f\} to \{e\}?
▶ from the right siblings of \{e, f\} to the childs of \{e\}?
## From Boolean Functions to Binary Decision Diagrams

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
From Boolean Functions to Binary Decision Diagrams

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
From Boolean Functions to Binary Decision Diagrams

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Remove duplicate terminals
From Boolean Functions to Binary Decision Diagrams

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Remove duplicate terminals
- Remove duplicate non-terminals
From Boolean Functions to Binary Decision Diagrams

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Remove duplicate terminals
- Remove duplicate non-terminals
- Remove redundant tests
Properties of ROBDDs

If a BDD is reduced and ordered, then we call it *reduced ordered BDD* (ROBDD).

- BDDs represent propositional formulae in if-then-else normal form.
- ROBDDs are unique for each variable order.
- Easy to compose and to manipulate.
- Efficient implementations available.
- Successfully applied in many areas (chip design and verification, etc)
- It is an NP-complete problem to find the optimal variable order.
Extraction of Coalition ROBDDs
Extraction of Coalition ROBDDs
Extraction of Coalition ROBDDs
Extraction of Coalition ROBDDs
From Pruned Trees to ROBDDs

**Definition**

Let \( \mathcal{P} = \langle t^-, t^+, \theta, I, \omega \rangle \) be a positive perceptron, let \( \text{PT}_\mathcal{P} \) be the corresponding pruned search tree for \( \prec \). We define the sets \( N \) and \( R \) as follows:

- If \( \text{PT}_\mathcal{P} \) is empty, then \( R = 0 \) and \( N = \{\} \).
- If \( \text{PT}_\mathcal{P} \) contains only the root node, then \( R = 1 \) and \( N = \{\} \).
- ...

\( \text{robdd}(\prec, 0, 1, R, N) \) is called coalition ROBDD for \( \mathcal{P} \).
From Pruned Trees to ROBDDs

Definition

- Otherwise: \( R = \text{id}(r_l) \) for the leftmost child \( r_l \) of the root node and \( N \) be defined as follows:

1. For each leaf \( n \) in \( PT_\mathcal{P} \), add \( \langle \text{id}(n), \text{var}(n), 1, l \rangle \) to \( N \) with 
   \( l = 0 \) if \( n \) has no right sibling or \( l = \text{id}(\text{rs}(n)) \).

2. Let \( n = \langle l, C \rangle \) be a tree node and let \( l \) be its left sibling 
   (BDD node \( \langle \text{id}(l), \text{var}(l), h_l, l_l \rangle \)). Let \( \langle l_l, \text{var}(n), h_l_1, l_l_1 \rangle \) be the 
   BDD node corresponding to the leftmost child of \( l \). If 
   \( \text{mci}(l) - \theta + (t^- - t^+) \cdot w(l) + (t^+ - t^-) \cdot w(n) > 0 \) add 
   \( \langle \text{id}(n), \text{var}(n), l_l_1, l_n \rangle \) to \( N \) with \( l_n = 0 \) if \( n \) has no right sibling 
   or \( l_n = \text{id}(\text{rs}(n)) \).

3. Otherwise, add a node \( \langle \text{id}(n), \text{var}(n), \text{id}(c), l \rangle \) to \( N \) with \( c \) 
   being the left-most child of \( n \) and \( l = 0 \) if \( n \) has no right 
   sibling or \( l = \text{id}(\text{rs}(n)) \).
A bigger Example

Neural-Symbolic Integration (Sebastian Bader, Pascal Hitzler)
A bigger Example
A bigger Example
Extracting Oppositions

Oppositions can be extracted by “reversing some inequalities” and starting from negative perceptrons.
Outline: Pedagogical Extraction: the CoOp Approach

Introduction

Decomposition

Extraction of Coalitions and Oppositions

Composition
  Composition based on Propositional Formulae
  Naive Composition of ROBDDs
  Online Composition of ROBDDs
  Integrity Constraints

Conclusions
From ROBDDs to Propositional Formulae

ROBDDs can easily be transformed into propositional formulae.

\[
\begin{align*}
c & \leftrightarrow (a \lor b) \\
d & \leftrightarrow (\bar{a} \land b) \\
e & \leftrightarrow (a \land \bar{b}) \\
f & \leftrightarrow \text{true} \\
g & \leftrightarrow ((e \land f) \lor (c \land d \land f)) \\
h & \leftrightarrow (\bar{d} \lor \bar{f}) \\
\bar{c} & \leftrightarrow (\bar{a} \land \bar{b}) \\
\bar{d} & \leftrightarrow (a \lor \bar{b}) \\
\bar{e} & \leftrightarrow (\bar{a} \lor b) \\
\bar{f} & \leftrightarrow \text{false} \\
\bar{g} & \leftrightarrow (\bar{f} \lor (\bar{d} \land \bar{e}) \lor (\bar{c} \land \bar{e})) \\
\bar{h} & \leftrightarrow (d \land f)
\end{align*}
\]
Naive Composition of Formulae

Using those formulae and the usual laws of logic, we obtain

\[ g \leftrightarrow ((e \land f) \lor (c \land d \land f)) \]
\[ \leftrightarrow (((a \land \overline{b}) \land \text{true}) \lor ((a \lor b) \land (\overline{a} \land b) \land \text{true})) \]
\[ \leftrightarrow ((a \land \overline{b}) \lor ((a \lor b) \land \overline{a} \land b)) \]
\[ \leftrightarrow ((a \land \overline{b}) \lor (\overline{a} \land b)) \]

\[ h \leftrightarrow (\overline{d} \lor \overline{f}) \]
\[ \leftrightarrow ((a \lor \overline{b}) \lor \text{false}) \]
\[ \leftrightarrow (a \lor \overline{b}) \]
Naive Composition of Formulae

... works, but the propositional formulae are usually much bigger than the BDDs and we do not even need all of them.
Naive Composition of ROBDDs
Naive Composition of ROBDDs
Online Composition of ROBDDs

- We can compose the ROBDD online, while extracting the perceptrons.
- All units can be extracted into the same ROBDD.
A Small Example

\[
g \leftrightarrow ((a \land \neg b) \lor (\neg a \land b))
\]

\[
h \leftrightarrow (a \lor \neg b)
\]
Integrity Constraints

- Usually only a (small) subset of all possible input combinations is interesting.
- Even more important: all training samples are usually taken from this subset.
- The network will learn to solve the task under the implicit conditions imposed by this selection.
- Integrity constraints are a method to make those conditions explicit during the extraction.
- Integrity constraints can be arbitrary formulae, describing valid input combinations, where the definition of “valid” is application dependent.
The Encoder–Decoder Example

- Task: learn a one-to-one mapping from inputs to outputs.
- In all training pattern, exactly one input unit is active, while all others are inactive.
- Using less hidden units forces the network to learn a compressed representation of the inputs.

The training samples:

<table>
<thead>
<tr>
<th>input pattern</th>
<th>output pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>(+1, −1, −1, −1, −1, −1, −1, −1)</td>
<td>(−1, +1, −1, −1, −1, −1, −1, −1)</td>
</tr>
<tr>
<td>(−1, +1, −1, −1, −1, −1, −1, −1)</td>
<td>(−1, +1, −1, −1, −1, −1, −1, −1)</td>
</tr>
<tr>
<td>(−1, −1, +1, −1, −1, −1, −1, −1)</td>
<td>(−1, −1, +1, −1, −1, −1, −1, −1)</td>
</tr>
<tr>
<td>(−1, +1, −1, −1, +1, −1, −1, −1)</td>
<td>(−1, −1, +1, −1, −1, −1, −1, −1)</td>
</tr>
<tr>
<td>(−1, −1, −1, −1, +1, −1, −1, −1)</td>
<td>(−1, −1, −1, −1, +1, −1, −1, −1)</td>
</tr>
<tr>
<td>(−1, −1, −1, −1, −1, +1, −1, −1)</td>
<td>(−1, −1, −1, −1, −1, +1, −1, −1)</td>
</tr>
<tr>
<td>(−1, −1, −1, −1, −1, +1, −1, −1)</td>
<td>(−1, −1, −1, −1, −1, +1, −1, −1)</td>
</tr>
<tr>
<td>(−1, −1, −1, −1, −1, −1, −1, +1)</td>
<td>(−1, −1, −1, −1, −1, −1, −1, +1)</td>
</tr>
</tbody>
</table>
The Encoder–Decoder Network
Extraction without and with Integrity Constraints
Extraction without and with Integrity Constraints
Extraction with Integrity Constraints
Extraction with Integrity Constraints
<table>
<thead>
<tr>
<th>Introduction</th>
<th>Decomposition</th>
<th>Extraction</th>
<th>Composition</th>
<th>Conclusions</th>
</tr>
</thead>
</table>
| **Outline: Pedagogical Extraction: the CoOp Approach**

- Introduction
- Decomposition
- Extraction of Coalitions and Oppositions
- Composition
- Conclusions
## Evaluation

<table>
<thead>
<tr>
<th>inputs</th>
<th>size(PT)</th>
<th>size(BDD)</th>
<th>node/sol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>2.58</td>
<td>4.44828</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>3.26</td>
<td>2.78632</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>4.52</td>
<td>1.74517</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>5.77</td>
<td>1.10115</td>
</tr>
<tr>
<td>5</td>
<td>32</td>
<td>8.49</td>
<td>0.75939</td>
</tr>
<tr>
<td>6</td>
<td>64</td>
<td>12.04</td>
<td>0.51081</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>16.91</td>
<td>0.33491</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>24.41</td>
<td>0.24842</td>
</tr>
<tr>
<td>9</td>
<td>512</td>
<td>35.52</td>
<td>0.18422</td>
</tr>
<tr>
<td>10</td>
<td>1024</td>
<td>49.97</td>
<td>0.12258</td>
</tr>
<tr>
<td>11</td>
<td>2048</td>
<td>72.70</td>
<td>0.08773</td>
</tr>
<tr>
<td>12</td>
<td>4096</td>
<td>105.39</td>
<td>0.06597</td>
</tr>
<tr>
<td>13</td>
<td>8192</td>
<td>152.96</td>
<td>0.04508</td>
</tr>
<tr>
<td>14</td>
<td>16384</td>
<td>223.91</td>
<td>0.03323</td>
</tr>
<tr>
<td>15</td>
<td>32768</td>
<td>313.25</td>
<td>0.02282</td>
</tr>
<tr>
<td>16</td>
<td>65536</td>
<td>445.35</td>
<td>0.01619</td>
</tr>
<tr>
<td>17</td>
<td>131072</td>
<td>633.63</td>
<td>0.01171</td>
</tr>
<tr>
<td>18</td>
<td>262144</td>
<td>918.80</td>
<td>0.00835</td>
</tr>
</tbody>
</table>
Conclusions

The CoOp approach ...

- is a new decompositional approach to extract rules from threshold networks.
- representation based on ROBDDs.
- easy composition of intermediate results.
- easy integration of integrity constraints.
- first experiments show a good performance.
Outline of the Course

- Introduction and Motivation
- The Core Method for Propositional Logic
- Applications of the Propositional Core Method
- A New Approach to Pedagogical Extraction
- The Core Method for First-Order Logic
- More on First-Order & other Perspectives