Abduction in Classification Tasks

AI*IA 2003

M. Atzori, P. Mancarella, F. Turini
{atzori, paolo, turini}@di.unipi.it

Dipartimento di Informatica
Università di Pisa, Italy
Goal

In *Data Mining* we want to get more information from raw data:
Goal

In *Data Mining* we want to get more information from raw data:

- generalizing data
Goal

In *Data Mining* we want to get more information from raw data:

- generalizing data
- using aggregated data
Goal

In *Data Mining* we want to get more information from raw data:

- generalizing data
- using aggregated data

The framework we are going to present is a *postprocessing* step useful to:
Goal

In *Data Mining* we want to get more information from raw data:

- generalizing data
- using aggregated data

The framework we are going to present is a *postprocessing* step useful to:

- obtain new information from aggregated data
Goal

In *Data Mining* we want to get more information from raw data:

- generalizing data
- using aggregated data

The framework we are going to present is a *postprocessing* step useful to:

- obtain new information from aggregated data
- query aggregated data
Goal

In *Data Mining* we want to get more information from raw data:

- generalizing data
- using aggregated data

The framework we are going to present is a *postprocessing* step useful to:

- obtain new information from aggregated data
- query aggregated data
- explain aggregated data
Summary

- Abduction in Logic Programming
Summary

- Abduction in Logic Programming
- Abductive Interpretation of Decision Trees
Summary

- Abduction in Logic Programming
- Abductive Interpretation of Decision Trees
  - Definition
Summary

- Abduction in Logic Programming
- Abductive Interpretation of Decision Trees
  - Definition
  - Examples of Applications
Summary

• Abduction in Logic Programming
• Abductive Interpretation of Decision Trees
  • Definition
  • Examples of Applications
  • Theoretical Results
Summary

- Abduction in Logic Programming
- Abductive Interpretation of Decision Trees
  - Definition
  - Examples of Applications
  - Theoretical Results
- Implementation
Summary

- Abduction in Logic Programming
- Abductive Interpretation of Decision Trees
  - Definition
  - Examples of Applications
  - Theoretical Results
- Implementation
- Conclusions
What is Abduction?

*Abduction* is a form of synthetic reasoning which infers the case from a rule and a result, i.e.

\[
\begin{array}{c}
B, A \Rightarrow B \\
\hline
A
\end{array}
\]
What is Abduction?

Abduction is a form of synthetic reasoning which infers the case from a rule and a result, i.e.

\[ B, A \Rightarrow B \]

\[ \underline{A} \]

In Logic Programming:

Let \( \langle P, A, Ic \rangle \) be an abductive framework and let \( G \) be a goal. Then an abductive explanation for \( G \) is a set \( \Delta \subseteq A \) of ground abducible atoms such that:
What is Abduction?

Abduction is a form of synthetic reasoning which infers the case from a rule and a result, i.e.

\[ B, A \Rightarrow B \]
\[ A \]

In Logic Programming:

Let \( \langle P, A, Ic \rangle \) be an abductive framework and let \( G \) be a goal. Then an abductive explanation for \( G \) is a set \( \Delta \subseteq A \) of ground abducible atoms such that:

- \( P \cup \Delta \models G \)
What is Abduction?

Abduction is a form of synthetic reasoning which infers the case from a rule and a result, i.e.

\[
B, A \Rightarrow B \\
\hline 
A
\]

In Logic Programming:

Let \( \langle P, A, Ic \rangle \) be an abductive framework and let \( G \) be a goal. Then an abductive explanation for \( G \) is a set \( \Delta \subseteq A \) of ground abducible atoms such that:

- \( P \cup \Delta \models G \)
- \( P \cup \Delta \cup Ic \) is consistent.
Classification as an Abductive Problem

- Knowledge Base
- Observations
- Integrity Constraints
Classification as an Abductive Problem

- Knowledge Base
  - *Set of rules corresponding to all tree paths*
- Observations
- Integrity Constraints
Classification as an Abductive Problem

- Knowledge Base
  - *Set of rules corresponding to all tree paths*
- Observations
  - *One of the leaves*
- Integrity Constraints
Classification as an Abductive Problem

- **Knowledge Base**
  - *Set of rules corresponding to all tree paths*
- **Observations**
  - *One of the leaves*
- **Integrity Constraints**
  - *Extra information about the domain*
Classification as an Abductive Problem

- Knowledge Base
  - *Set of rules corresponding to all tree paths*
- Observations
  - *One of the leaves*
- Integrity Constraints
  - *Extra information about the domain*

We obtain a framework able to answer *abductive queries* starting from the induced data.
The Process

Induced Tree
The Process

Induced Tree

Transformation into rules

Abductive Framework
The Process

Induced Tree
Transformation into rules

Abductive Framework

Abductive Queries ↔ Abductive Answers

User

Abduction in Classification Tasks  A*IA 2003 – p.6
Applications

Abductive Logic Programming frameworks can be profitably used in order to query induced decision trees (*representing generalized data*) in an abductive way, obtaining for example:

- better classification (by adding domain specific knowledge as integrity constraints)
- the reason why an instance belongs to a particular class (by adding knowledge about the instance and then a simple abductive query)
- a set of attributes whose values should be changed in order to obtain a different class (by finding differences between two similar results of different goals)
Applications

Abductive Logic Programming frameworks can be profitably used in order to query induced decision trees (representing generalized data) in an abductive way, obtaining for example:

- better classification (by adding domain specific knowledge as integrity constraints)
Applications

Abductive Logic Programming frameworks can be profitably used in order to query induced decision trees (representing generalized data) in an abductive way, obtaining for example:

- **better classification** *(by adding domain specific knowledge as integrity constraints)*

- **the reason why an instance belongs to a particular class** *(by adding knowledge about the instance and then a simple abductive query)*
Applications

Abductive Logic Programming frameworks can be profitably used in order to query induced decision trees (representing generalized data) in an abductive way, obtaining for example:

- **better classification** (*by adding domain specific knowledge as integrity constraints*)
- **the reason why an instance belongs to a particular class** (*by adding knowledge about the instance and then a simple abductive query*)
- **a set of attributes whose values should be changed in order to obtain a different class** (*by finding differences between two similar results of different goals*)
### An Example: Training Set

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Low</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Low</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Low</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>Low</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Low</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>Low</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Low</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
### An Example: Training Set

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Low</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Low</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Low</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>Low</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>Low</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>Low</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

*Abduction in Classification Tasks: AI*IA 2003 – p.8*
### An Example: Training Set

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Low</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cool</td>
<td>Low</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Low</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
</tbody>
</table>
An Example: Tree

- Overlook
  - Sunny
    - Humidity
      - High
        - No
      - Low
        - Yes
  - Rainy
    - Overcast
    - Wind
      - Yes
        - No
      - Strong
      - Weak
An Example: Extra Knowledge

Let’s imagine that whenever there is strong wind the humidity is not high:

\[ I_c = \neg (\text{Humidity}(\text{High}), \text{Wind}(\text{Strong})) \]
An Example: Extra Knowledge

Let’s imagine that whenever there is strong wind the humidity is not high:

\[ I_c = \neg(Humidity(High), Wind(Strong)) \]

Possible reasons:
An Example: Extra Knowledge

Let’s imagine that whenever there is strong wind the humidity is not high:

\[ I_c = \neg(Humidity(High), Wind(Strong)) \]

Possible reasons:
- we are interested only in that kind of days
An Example: Extra Knowledge

Let’s imagine that whenever there is strong wind the humidity is not high:

\[ I_c = \neg (\text{Humidity}(\text{High}), \text{Wind}(\text{Strong})) \]

Possible reasons:

- we are interested only in that kind of days
- our extra knowledge arises from knowledge sources different from the ones which provide the training set
An Example: Abduction

We want to classify the instance $e = \{\text{Overlook}(\text{Sunny}), \text{Wind}(\text{Strong})\}$. 
An Example: Abduction

We want to classify the instance 
\( e = \{Overlook(Sunny), Wind(Strong)\} \).

Abduction in Classification Tasks  AI*IA 2003 – p.11
An Example: Abduction

We want to classify the instance
\[ e = \{\text{Overlook}(\text{Sunny}), \text{Wind}(\text{Strong})\} \].

In the corresponding abductive framework:
An Example: Abduction

We want to classify the instance

\[ e = \{Overlook(Sunny), Wind(Strong)\} . \]

In the corresponding abductive framework:

- \[ \Delta_e = \{Overlook(Sunny), Wind(Strong)\} \]
An Example: Abduction

We want to classify the instance
\[ e = \{Overlook(Sunny), Wind(Strong)\}. \]

In the corresponding abductive framework:

- \[ \Delta_e = \{Overlook(Sunny), Wind(Strong)\} \]
- \[ Ic = \neg(Humidity(High), Wind(Strong)) \]
An Example: Abduction

We want to classify the instance
\( e = \{\text{Overlook}(\text{Sunny}), \text{Wind}(\text{Strong})\} \).

In the corresponding abductive framework:

- \( \Delta_e = \{\text{Overlook}(\text{Sunny}), \text{Wind}(\text{Strong})\} \)
- \( Ic = \neg(\text{Humidity}(\text{High}), \text{Wind}(\text{Strong})) \)
- \( \Delta_1 = \{\text{Humidity}(\text{Low})\} \) for Yes
An Example: Abduction

We want to classify the instance
e = \{Overlook(Sunny), Wind(Strong)\}.

In the corresponding abductive framework:

- \(\Delta_e = \{Overlook(Sunny), Wind(Strong)\}\)
- \(Ic = \neg(Humidity(\text{High}), Wind(\text{Strong}))\)
- \(\Delta_1 = \{Humidity(\text{Low})\}\) for Yes
- \(\Delta_2 = \{Humidity(\text{High})\}\) for No
An Example: Abduction

We want to classify the instance
\( e = \{\text{Overlook}(\text{Sunny}), \text{Wind}(\text{Strong})\} \).

In the corresponding abductive framework:

- \( \Delta_e = \{\text{Overlook}(\text{Sunny}), \text{Wind}(\text{Strong})\} \)
- \( Ic = \neg(\text{Humidity}(\text{High}), \text{Wind}(\text{Strong})) \)
- \( \Delta_1 = \{\text{Humidity}(\text{Low})\} \) for Yes
- \( \Delta_2 = \{\text{Humidity}(\text{High})\} \) for No

\( \Delta_e \cup \Delta_2 = \{\text{Overlook}(\text{Sunny}), \text{Wind}(\text{Strong}), \text{Humidity}(\text{High})\} \) is inconsistent \( \Rightarrow \) \( \Delta_2 \) is ruled out.
Soundness and Completeness

We have an instance (even with missing attribute values) $E$ with class $C$, and $AB_T$ (the abductive framework obtained from the tree $T$)
Soundness and Completeness

We have an instance (even with missing attribute values) $E$ with class $C$, and $AB_T$ (the abductive framework obtained from the tree $T$)

- Soundness

- Completeness
Soundness and Completeness

We have an instance (even with missing attribute values) $E$ with class $C$, and $AB_T$ (the abductive framework obtained from the tree $T$)

- **Soundness**
  - If $\Delta$ is a minimal solution for $AB_T$ then using $T$ with $E \cup \Delta$ we reach a leaf $C$

- **Completeness**
Soundness and Completeness

We have an instance (even with missing attribute values) $E$ with class $C$, and $AB_T$ (the abductive framework obtained from the tree $T$)

- **Soundness**
  - If $\Delta$ is a minimal solution for $AB_T$ then using $T$ with $E \cup \Delta$ we reach a leaf $C$

- **Completeness**
  - If given $\Delta$ we can reach a leaf $C$ in the tree then $\Delta$ is a minimal solution for $AB_T$
Soundness and Completeness

We have an instance (even with missing attribute values) $E$ with class $C$, and $AB_T$ (the abductive framework obtained from the tree $T$)

- **Soundness**
  - If $\Delta$ is a minimal solution for $AB_T$ then using $T$ with $E \cup \Delta$ we reach a leaf $C$

- **Completeness**
  - If given $\Delta$ we can reach a leaf $C$ in the tree then $\Delta$ is a minimal solution for $AB_T$

$\Rightarrow$ *The abductive framework is at least as powerful as decision trees*
Implementation

We are testing the *Abduction framework* on decision trees obtained from *web log datasets*.

The abductive framework is automatically generated from the decision trees by a parser written in *Java*.

The abductive answers are obtained using *ACLP* (University of Cyprus) within the *Eclipse* Prolog.

http://www.di.unipi.it/~atzori/DTAabduction
Conclusions

- Abductive reasoning can be useful in the context of Classification, as a postprocessing step, for:
  - Improving effectiveness, when we deal with incomplete data and with external domain knowledge
  - Explaining results in order to get the reason of a classification
  - Answer abductive queries finding out how attribute values should be changed in order to get a different classification
Future Works

- We still need to insert abductive interpretation of decision trees into a probabilistic abductive framework, in order to get, for example, support and confidence of abductive answers.

- Join together different data mining paradigms:
  - **Classification**, as already showed
  - **Association Rules**, as a way to automatically generate constraints from the training set
  - **Clustering**, finding similarities between rules and then, through abduction, showing the differences between rules in the same cluster