Cross-Domain Transfer of Constraints for Inductive Process Modeling

Will Bridewell and Ljupco Todorovski

Computational Learning Laboratory
CSLI, Stanford University

with acknowledgements to Pat Langley and Tolga Könik.

willb@csli.stanford.edu
Ross-96 to Ross-97

best performance: 0.94 (no transfer), 0.94 (transfer)
Outline

• Inductive process modeling
• Constraining model structures
• Learning constraints for process modeling
• Measures for evaluating transfer
• Cross domain transfer
Inductive Process Modeling

Model Objectives: Explanation and Prediction

Bridewell et al. 2008, MLj
Quantitative Process Models

• **Ordinary Differential Equations**
  
  \[
  \frac{dhare.density}{dt} = 2.5 \times hare.density + -0.1 \times hare.density \times wolf.density
  \]
  
  \[
  \frac{dwolf.density}{dt} = -1.2 \times wolf.density + 0.3 \times 0.1 \times hare.density \times wolf.density
  \]

• **Processes**

  process exponential_growth
  
  equations \[d[hare.density, t, 1] = 2.5 \times hare.density\]

  process exponential_loss
  
  equations \[d[wolf.density, t, 1] = -1.2 \times wolf.density\]

  process predation_holling_type_1
  
  equations \[d[hare.density, t, 1] = -0.1 \times hare.density \times wolf.density\]
  
  \[d[wolf.density, t, 1] = 0.3 \times 0.1 \times hare.density \times wolf.density\]
Quantitative Process Models

• Ordinary Differential Equations
  \[ \frac{dhare.density}{dt} = 2.5 \times hare.density \]
  \[ \frac{dwolf.density}{dt} = -1.2 \times wolf.density \]

• Processes
  
  process exponential_growth
  equations \[ d[hare.density, t, 1] = 2.5 \times hare.density \]

  process exponential_loss
  equations \[ d[wolf.density, t, 1] = -1.2 \times wolf.density \]
Quantitative Process Models

• Ordinary Differential Equations
  \[
  \begin{align*}
  \frac{dhare.density}{dt} &= 2.5 \times hare.density + \frac{-0.1 \times hare.density \times wolf.density}{(1 + 0.2 \times -0.1 \times hare.density)} \\
  \frac{dwolf.density}{dt} &= -1.2 \times wolf.density + \frac{0.3 \times 0.1 \times hare.density \times wolf.density}{(1 + 0.2 \times -0.1 \times hare.density)}
  \end{align*}
  \]

• Processes
  
  process exponential_growth
  \[
  \begin{align*}
  \text{equations} \ d[hare.density, t, 1] &= 2.5 \times hare.density
  \end{align*}
  \]

  process exponential_loss
  \[
  \begin{align*}
  \text{equations} \ d[wolf.density, t, 1] &= -1.2 \times wolf.density
  \end{align*}
  \]

  process predation_holling_type_2
  \[
  \begin{align*}
  \text{equations} \ d[hare.density, t, 1] &= \frac{-0.1 \times hare.density \times wolf.density}{(1 + 0.2 \times -0.1 \times hare.density)} \\
  \frac{dwolf.density}{dt} &= \frac{0.3 \times 0.1 \times hare.density \times wolf.density}{(1 + 0.2 \times -0.1 \times hare.density)}
  \end{align*}
  \]
Model Components

generic process predation_Holling_1{predation}
  entities $P_1${prey}, $P_2${predator}
  parameters $r[0, \infty], e[0, \infty]$
  equations 
  $d[P_1.density, t, 1] = -1 \cdot r \cdot P_1.density \cdot P_2.density$
  $d[P_2.density, t, 1] = e \cdot r \cdot P_1.density \cdot P_2.density$
Model Components

generic process predation_Holling_1{predation}

entities P1{prey}, P2{predator}

parameters r[0, infinity], e[0, infinity]

equations
\[ d[P1.density, t, 1] = -1 \times r \times P1.density \times P2.density \]
\[ d[P2.density, t, 1] = e \times r \times P1.density \times P2.density \]

P1: hare       P2: wolf
r: 0.1          e: 0.3

Instantiation
Model Components

generic process predation_Holling_1{predation}
  
entites P1{prey}, P2{predator}
  
parameters r[0, infinity], e[0, infinity]
  
equations d[P1.density, t, 1] = \(-1 \times r \times P1.density \times P2.density\)
    d[P2.density, t, 1] = \(e \times r \times P1.density \times P2.density\)

<table>
<thead>
<tr>
<th>P1: hare</th>
<th>P2: wolf</th>
</tr>
</thead>
<tbody>
<tr>
<td>r: 0.1</td>
<td>e: 0.3</td>
</tr>
</tbody>
</table>

process wolves_eat_hares
  
equations d[hare.density, t, 1] = \(-1 \times 0.1 \times hare.density \times wolf.density\)
    d[wolf.density, t, 1] = \(0.3 \times 0.1 \times hare.density \times wolf.density\)
The IPM System
(a naive approach)

- **Given:**
  - Data
  - A library of generic entities and processes
  - Instantiated entities

- Ground the generic processes with instantiated entities
- Generate all combinations of the instantiated processes
- Fit the numeric parameters of each structure

- **Output:** The best model based on its fit to the data

- **Problem:** Many structures are implausible
Structural Constraints

• Eliminate implausible structures
• Reduce the search space
• Make complex domains tractable
• Improve model accuracy during incomplete search

HIPM, Todorovski et al. AAAI-05
Hierarchical Constraints

HIPM, Todorovski et al. AAAI-05
Learning Constraints

Goal:
Identify implicit or unknown constraints to use in future modeling tasks

Plan:
Analyze the space of model structures
Use machine learning techniques to help

Key Idea:
Don’t throw away any models
Even the bad ones contain valuable information

Bridewell & Todorovski (2007), ILP and KCAP
Inspiration for Constraint Induction

1996–1997 Ross Sea
Learning Constraints

1. Build and parameterize process models
2. Store the models for analysis
3. Formally describe the model structures
4. Identify good and bad models
5. Use ILP to generate descriptions of accurate and inaccurate models
6. Convert descriptions into constraints for IPM
Good and Bad Models

1996–1997 Ross Sea
Descriptive Rules

Run 1: Generate a theory for accurate models.

accurate_model(A) :-
  does_not_include_process(A,death_exp2),
  includes_process_entity(A,monod_lim,iron).

Run 2: Generate a theory for inaccurate models.

inaccurate_model(A) :-
  does_not_include_process_entity(A,monod_2nd,iron),
  does_not_include_process_entity(A,monod_lim,iron).

inaccurate_model(A) :-
  includes_process(A,death_exp2),
  includes_process_entity(A,deangelis_beddington,phytoplankton).

We chose Aleph by Ashwin Srinivasan due to its ready availability and capabilities.
Step 6: Convert Theories into Structural Constraints

Rules for accurate models become sufficient conditions for retaining model structures.

The negation of rules for inaccurate models become necessary conditions for retaining the model structures.
Type of Transfer

• Subsuming Domain: Aquatic Ecosystems
  – Marine Ecosystem -- Southern Ocean
  – Freshwater Ecosystem -- Bled Lake

• Shared Processes

• Different Environment
  – nutrients, light patterns, ice, temperature

• Different Agents
  – phytoplankton, zooplankton
Measures

• Worst Case Modeling scenario

• performance loss (in $r^2$) due to transfer
  – remember “exhaustive” search

• how much faster do we see the most accurate model in the transfer case vs. no transfer?
Ross-96 to Ross-97

best performance: 0.94 (no transfer), 0.94 (transfer)
Ross-96 to Bled-02

best performance: 0.97 (no transfer), 0.92 (transfer)
Bled-02 to Ross-96

best performance: 0.98 (no transfer), 0.97 (transfer)
Related Work

• Inductive Process Modeling and other quantitative modelers
  — IPM (Langley et al., see ICML 2002)
  — HIPM (Todorovski et al., see AAAI 2005)
  — Lagramge (Todorovski & Dzeroski)
  — PRET (Bradley & Stolle)

• Some similar work
  — Learning Constraint Networks (Bessiere et al.)
  — Relational Clichés (Silverstein & Pazzani; Morin & Matwin)
  — Mode Declarations (McCreath & Sharma)
  — Rule Reliability from Prior Performance (Mark Reid)
Future Directions

• Extended analysis of cross-domain transfer

• Incorporating learned constraints into IPM
  — See Bravo et al. in the ICML workshop IPM-07

• Transferring this approach to other tasks
  — Planning, classifier learning, etc.
  — (See me if your work could benefit.)

• Building an integrated creative system