LEARNING IN RICH REPRESENTATIONS:

INDUCTIVE LOGIC PROGRAMMING AND COMPUTATIONAL SCIENTIFIC DISCOVERY

Sašo Džeroski

Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia

Rich representations

- Allow for modelling a wide range of real-world problems/situations/systems
- Allow for the use of domain knowledge
- Are communicable (at least to a degree)
- Exemplified by logic programs and equations
- Are so expressive that even deductive reasoning is non-trivial, much less inductive reasoning
- Within the same representation, various restrictions can mean completely different computational complexity classes

The goals of this talk

- Review some key ideas from
  Inductive Logic Programming (ILP)

- Show how these ideas can be used
  in other learning settings, and in particular in
  Computational Scientific Discovery
  (equation discovery, to be more specific)

- Encourage more research on
  learning in rich representations and
  using domain knowledge

Outline

- Some ILP history
- Some key ILP approaches and ideas
- Equation discovery lines of research
- Some equation discovery systems
- Using domain knowledge therein
- Summary and discussion

ILP - The early 1990s

Learning logic programs from

- Examples
- Background knowledge

A typical task: learning list reversal

- Positive examples: `rev([],[])`, `rev([1],[1])`, `rev([2,1],[1,2])`
- Negative examples: `rev([1],[2])`, `rev([2,1],[2,1])`, `rev([1,3,2],[1,2,3])`
- Background knowledge:
  `list(List,Head,Tail)`
  `insertAtEnd(Element,List,NewList)`

The hypothesis:

```
rev(L,RL) :- list(L,H,T), rev(T,RT),
             insertAtEnd(H,RT,RL).
```

ILP - The late 1990s

Addresses a variety of learning tasks
in relational representations

- Relational examples
  (sets of facts/tuples, possibly spread across multiple relations)
- Relational background knowledge
  (defined extensionally, as sets of tuples,
  or intensionally as logic programs)
- Relational hypothesis language
  (based on/related to logic programs)

Note the connection of logic programs to relational databases
(predicates/relations, predicate definitions/views)

An example: ILP for QSAR

An example is a molecule, described by a set of ground facts


```
atom(ID,Element,AtomType,Charge)
bond(Atom1,Atom2,BondType).
```

```
atom(1,c,10,0.388).
atom(2,c,10,0.388).
atom(3,cl,93,-0.212).
atom(4,cl,93,-0.212).
atom(5,h,3,0.037).
atom(6,cl,93,-0.213).
atom(7,cl,93,-0.213).
atom(8,h,3,0.037).
```

```
bond(1,2,1).
bond(1,3,1).
bond(1,4,1).
bond(1,5,1).
bond(2,6,1).
bond(2,7,1).
bond(2,8,1).
```

An example: ILP for QSAR - continued

- The task is to predict the activity
  (mutagenicity, carcinogenicity, biodegradability ...) of a molecule.

- This can be a classification or a regression task

- Background knowledge typically includes
  definitions of functional groups and substructures (such as rings)

- Different hypothesis laguages can be used (rules, trees, ...)
  or none at all (as in distance-based approaches)

Relational Data Mining

Addressing data mining tasks in relational representations

- Discovering frequent Datalog patterns and relational association rules
- Relational distance-based prediction/clustering
- Relational classification and regression trees
- Relational rules for classification and regression

S. Džeroski and N. Lavrač, editors (2001)
Relational Data Mining. Springer, Berlin.

(Multi)Relational Data Mining Workshop @ KDD-2002

Relational Data Mining Summerschool @ ECML/PKDD-2002

Relational Reinforcement Learning

Reinforcement Learning focuses on an agent that learns a policy, i.e., how to act in an environment. The agent

- Perceives the state of the environment
- Selects and performs an action
- Receives reward/punishment and state of environment changes

Relational Reinforcement Learning uses relational representations for

- States: { *clear(c)*, *clear(d)*, *on(d,a)*, *on(a,b)*, *on(b,floor)*, *on(c, floor)* }
- Actions: *move(d,c)*
- Policies: *optimal(move(A,floor))* $\leftarrow$ *on(A,B)*, *on(B,C)*

K. Driessens and S. Džeroski. Integrating Experimentation and Guidance in Relational Reinforcement Learning. Talk @ ICML-2002.

Hot topics in ML and ILP

Topics in ML (in alphabetical order)

- Ensembles of classifiers
- Kernel methods
- Probabilistic models
- Reinforcement learning
- Scaling-up

Analogous topics in ILP

- Ensembles of relational classifiers (boosting, bagging)
- Kernel methods for structured data
- Relational probabilistic models (FOBNs, PRMs, SLPs)
- Relational reinforcement learning
- Scaling-up ILP/RDM algorithms

ILP - Some key approaches

- Transforming ILP problems
  to propositional form (propositionalization)

- Inverting deductive reasoning
  (e.g., inverting resolution/entailment)

- Adapting/upgrading approaches
  from propositional/attribute-value learning

Transforming ILP problems to propositional form

Original problem:

Training examples		Background knowledge	
daughter(sue,eve).	+	parent(eve,sue).	female(ann).
daughter(ann,pat).	+	parent(ann,tom).	female(sue).
daughter(tom,ann).	–	parent(pat,ann).	female(eve).
daughter(eve,ann).	–	parent(tom,sue).	

Propositional form:

	Variables		Propositional features						
Class	X	Y	female(X)	female(Y)	parent(X,X)	parent(X,Y)	parent(Y,X)	parent(Y,Y)	X = Y
+	sue	eve	true	true	false	false	true	false	false
+	ann	pat	true	false	false	false	true	false	false
–	tom	ann	false	true	false	false	true	false	false
–	eve	ann	true	true	false	false	false	false	false

Result of attribute-value rule induction:
Class = + IF ’female(X)’ = true AND ’parent(Y,X)’ = true

Transformation to program clause form:
daughter(X,Y) ← *female(X),parent(Y,X).*

ILP - Some key concepts

- Background knowledge (BK)
- ILP as search: Refinement operators
- Declarative bias
- Theory revision

Background knowledge in ILP

An example in ILP is, e.g., a molecule or a chess-board configuration (which may need multiple facts to be represented)

Background knowledge captures laws/regularities/rules generally valid in the given domain

- Definitions of functional groups

```
alcohol :- atom(X,c,_,_), atom(Y,o,_,_),
    atom(Z,h,_,_), bond(X,Y,1), bond(Y,Z,1).
```

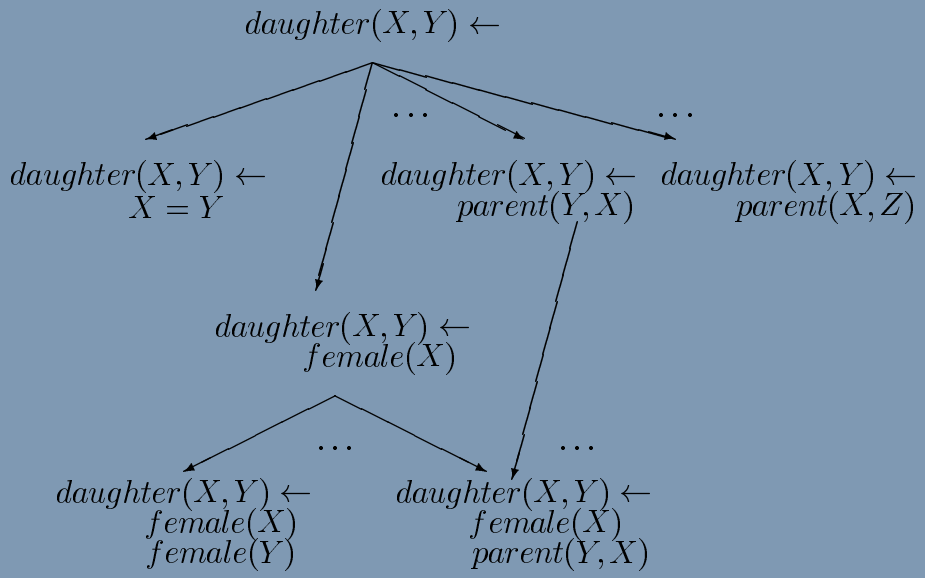
- Definition of how different pieces move on the chessboard

```
rook_move(X,Y,A,B) :- X=A.
rook_move(X,Y,A,B) :- Y=B.
```

Refinement operators

Given a clause (rule), a refinement operator

- Adds a literal (condition) to the body of the clause
  or substitutes a variable with a term (Specialization)
- Deletes a literal (condition) from the body of the clause
  or turns a term into a variable (Generalization)



A refinement operator structures the space of clauses/theories

Declarative bias

Many machine learning systems have the hypothesis language hard-wired

Declarative bias explicitly specifies the language of hypotheses considered
This is input to the learning system

- Parametrized language bias (e.g., maximum number of variables, literals, depth of variables, arity, etc,)

- Clause templates: (e.g., $P(X,Y) \leftarrow Q(X,Z), R(Z,Y)$; grandmother $\leftarrow$ mother, parent; grandfather $\leftarrow$ father, parent)

- Grammars (context-free)
  - Specify the language of clauses:
    Body $\leftarrow$ Literal | Literal Body; Literal $\leftarrow$ X=Y | adj(X,Y)
  - Implicitly define a refinement operator

Theory revision

The task of theory revision can be defined as follows

Given

- An existing (imperfect) domain theory
- A set of observations (examples)
  (some of which are not explained correctly by the theory)

Find a revised theory

- That explains the observations correctly
- Is as similar as possible to the original theory
  (minimality of change; mostly syntactic)

Rule: $daughter(X, Y) \leftarrow parent(Y, X)$
Counterexample: $daughter(tom, ann)$.
Revision: $daughter(X, Y) \leftarrow parent(Y, X), female(X)$

Computational scientific discovery

Applies computational methods to automate scientific activities

Early research reconstructed episodes from the history of science

Recent efforts in this area have focussed on individual scientific activities
(such as formulating quantitative laws)
and have led to several new discoveries

Emphasis on formalisms used to communicate among scientists

- Numeric equations
- Reaction pathways

S. Džeroski and L. Todorovski, editors (2002)
Computational Discovery of Communicable Knowledge.
Springer, Berlin. In preparation.

Equations as a representation

Equations are widely used to represent knowledge in science

(Ordinary and Partial) Differential equations widely used to model systems whose behavior changes over time (and other dimensions, e.g., space)

Newton's second law

- Higschool physics: $F = m \times a$
- College physics: $F = m \times \frac{d^2x}{dt^2}$

F - force, m-mass, x-position, v-velocity, a-acceleration, $a = \frac{dv}{dt}, v = \frac{dx}{dt}$

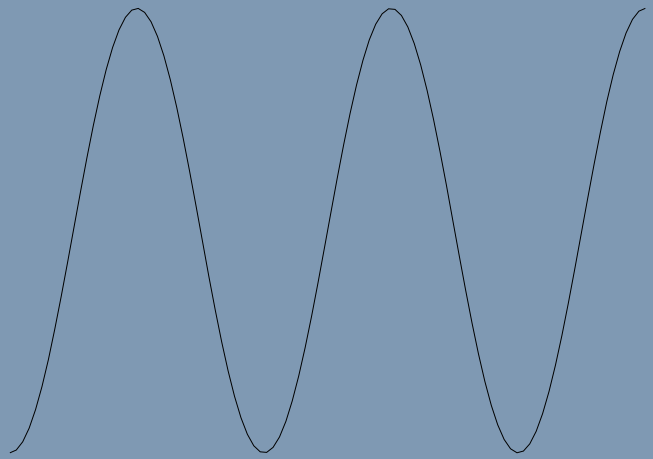
Equation Discovery Lines of Research

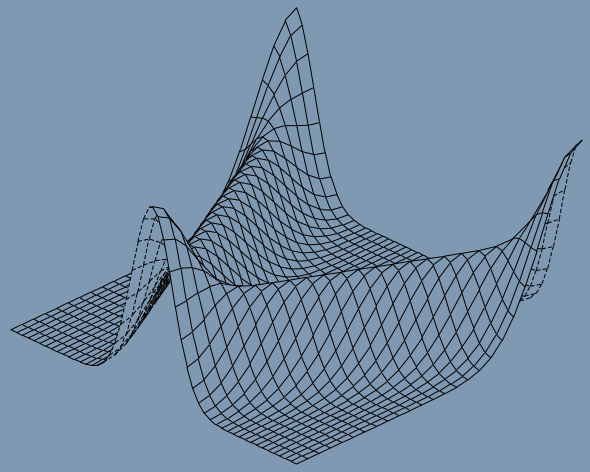
Followed by S. Džeroski and L. Todorovski
over the last decade (1993-2002)

- Different types of equations
- Language bias in equation discovery
- Using domain knowledge
  in the process of equation discovery

Different Types of Equations

- Algebraic (LAGRANGE): $F = m \times a$ (Newton)
- Ordinary differential equations (LAGRANGE/LAGRAMGE): $F = m \times \frac{d^2x}{dt^2}$ (Newton), $\frac{d^2u}{dt^2} + u = 0$ (1D wave)
- Partial differential equations (PADLES): $\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0$ (2D wave)





Discovering Algebraic Equations - The task

GIVEN: A set of measurements (*observations*)

- Variables: $X_1, \ldots X_n$
- Outcomes of experiments: measured values

X_1	X_2	$\ldots$	X_n
x_{10}	x_{20}	$\ldots$	x_{n0}
x_{11}	x_{21}	$\ldots$	x_{n1}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
x_{1m}	x_{2m}	$\ldots$	x_{nm}

FIND: Quantitative laws that summarize the observations

Laws = A set of algebraic equations

Discovering Ordinary Differential Equations – The task

GIVEN: Example behavior(s) of a dynamic system

- System variables: $v_1, \ldots v_n$
- System state measurements at equidistant time points

Time	System variables			
	v_1	v_2	$\ldots$	v_n
t_0	$v_{1,0}$	$v_{2,0}$	$\ldots$	$v_{n,0}$
t_1	$v_{1,1}$	$v_{2,1}$	$\ldots$	$v_{n,1}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_m	$v_{1,m}$	$v_{2,m}$	$\ldots$	$v_{n,m}$

FIND: Laws (model) describing the dynamics of the system

Model = Set of differential and algebraic equations

Discovering Partial Differential Equations - The task

GIVEN: Example behavior(s) of a dynamic system

- Dimension variables: $x_1, \ldots x_k$
- System variables: $v_1, \ldots v_n$
- System state measurements

Dimensions			System variables			
			v_1	v_2	$\ldots$	v_n
$x_{1,0}$	$\ldots$	$x_{k,0}$	$v_{1,0}$	$v_{2,0}$	$\ldots$	$v_{n,0}$
$x_{1,1}$	$\ldots$	$x_{k,1}$	$v_{1,1}$	$v_{2,1}$	$\ldots$	$v_{n,1}$
$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
$x_{1,m}$	$\ldots$	$x_{k,m}$	$v_{1,m}$	$v_{2,m}$	$\ldots$	$v_{n,m}$

FIND: Laws (model) describing the dynamics of the system

Model = Set of partial differential equations

The Language Bias of Equation Discovery

- Parametrized language bias
  (LAGRANGE - polynomials of degree up to d with at most r terms)

- Declarative language bias in the form of context-free grammars
  (LAGRAMGE)

- Using higher level concepts such as processes
  (e.g., in population dynamics)
  to specify language bias; transform these to grammars
  (possibly context dependent - LAGRAMGE2.0)

Using Domain Knowledge in Equation Discovery

In LAGRAMGE and LAGRAMGE2.0, domain knowledge is used

- Through declarative language bias
- Through background functions/predicates
- By using existing domain theories/models in the form of (sets/systems of) equations

LAGRANGE (1993)

- Input parameters:
  - o - the order of the dynamic system
  - d is the maximum depth of new terms
  - r is the maximum number of independent regression variables
  - t_R and t_S are tolerances used in testing equations

- Three main stages of the algorithm:
  1. Introduce time derivatives up to order o (numerically)
  2. Introduce new terms with:
     - sin and cos for variables measured in radians
     - multiplication (up to depth d)
  3. Generate and test linear equations with up to r dependent variables using linear regression (report those that fit data well, cf. t_R and t_S)

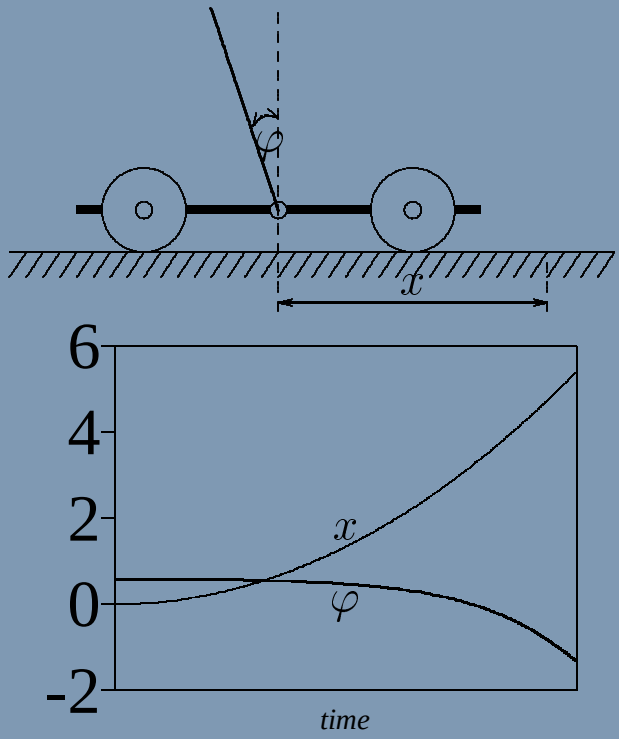
LAGRANGE Example – Pole and cart

Model equation

$$(M+m)\ddot{x} + \tfrac{1}{2}ml(\ddot{\varphi}\cos\varphi - \dot{\varphi}^2\sin\varphi) = F$$

$$\ddot{x}\cos\varphi + \tfrac{2}{3}l\ddot{\varphi} = g\sin\varphi$$

$$x_0 = 0,\ \varphi_0 = 3\pi/16,\ \dot{x}_0 = 0,\ \dot{\varphi}_0 = 0$$
$$M = 1,\ m = 0.1,\ l = 1,\ F = 7.5$$



LAGRANGE output

$$\begin{aligned}
\cos^2\varphi &= 1 - \sin^2\varphi \\
\ddot{x}\cos\varphi &= -0.67\ddot{\varphi} + 9.81\sin\varphi \\
\dot{\varphi}^2\sin\varphi &= -150 + 22\ddot{x} + \ddot{\varphi}\cos\varphi \\
\ddot{x}\cos\varphi\sin\varphi &= 9.81 - 0.67\ddot{\varphi}\sin\varphi - 9.81\cos^2\varphi
\end{aligned}$$

ILP ideas in LAGRANGE

- Parametrized language bias
  (order of differential equations at most o,
  polynomials of degree up to d with at most $r + 1$ terms)
- Transform the problem of learning ODEs
  to the problem of learning algebraic equations
  (by introducing time derivatives numerically)
- Transform the problem of learning polynomial equations
  to the problem of learning linear equations
  (by introducing products)
- ILP formulation of LAGRANGE:
  Descriptive setting (find all constraints satisfied by examples);
  Examples are behaviors;
  Background knowledge consists of
  numerical derivation, multiplication, linear regression

LAGRAMGE (1997) - Grammar based equation discovery

Given

- Set of system variables $V = \{v_1, v_2, \ldots v_n\}$
- Target system variable $v_d \in V$
- Context free grammar G
- Table of measurements (observations) of the system variables

Find an equation E that fits the observations

- E defines v_d in terms of variables in V
- E can be derived using the grammar G
- E minimizes the discrepancy between
  the measured and calculated (simulated) values of v_d

LAGRAMGE - The declarative bias formalism and the algorithm

The context free grammar G

- Prescribes the syntax of expressions on right hand side of equations
- Has to generate expressions legal in the C programming language
  Can use C built-in operators and C library functions, as well as
  user defined background functions
- Terminals with special meanings:
  $const$ – constant parameter to be fitted, $v_i \in V$ – system variables

The algorithm

- Heuristically (or exhaustively) searches the space of equations
- Refinement operator used with beam search
- Combination of error and equation complexity used as heuristic

LAGRAMGE - Lake Glumsoe experiment

Location and environment

- sub-glacial valley in Denmark
- average depth 2 m
- surface area 266,000 m^2
- recieves mechanically-biologically treated waste water from mainly agricultural community
- high nitrogen and phosphorus concentration in waste water caused hypereutrophication

Variables relevant for modelling are the concentrations of phytoplankton ($phyt$), zooplankton (zoo), soluble nitrogen ($nitro$), soluble phosphorus ($phosp$), and water temperature ($temp$)

LAGRAMGE - Lake Glumsoe context free grammar

Definition of the background knowledge function `monod`

```
double monod(double c, double v) {
  return(v / (v + c)); }
```

Context free grammar provided to LAGRAMGE

$$PRC \rightarrow FeedingPhyto \cdot phyt - DecayPhyto - FeedingZoo \cdot phyt$$

$$FeedingZoo \rightarrow const \cdot phyt \cdot zoo \;\mid\; 0$$

$$DecayPhyto \rightarrow const \cdot phyt$$

$$FeedingPhyto \rightarrow const \cdot Y \cdot Y \;\mid\; const \cdot Y + const \cdot Y \;\mid\; const \cdot Y$$

$$Y \rightarrow \mathrm{monod}(const, v_Y) \;\mid\; v_Y$$

$$v_Y \rightarrow phosp \mid nitro \mid temp$$

LAGRAMGE - Lake Glumsoe model learned

- Equation discovered by LAGRAMGE

$$\dot{phyt} = 0.553 \cdot temp \cdot \frac{phosp}{0.0264 + phosp} \cdot phyt - 4.35 \cdot phyt - 8.67 \cdot zoo \cdot phyt$$

- Human expert's comments on the equation
  - phosphorus is a limiting factor for phytoplankton growth
  - feeding of phytoplankton on phosphorus saturates
  - growth is temperature dependent
  - feeding of zooplankton on phytoplankton is non-negligible
  - suitable for long term prediction of phytoplankton growth
  - better predictions than a linear model

ILP ideas in LAGRAMGE

- Grammar-based declarative bias
- The grammar implicitly defines a refinement operator
  (actually used by LAGRAMGE
  to heuristically search the space of equations)
- Background knowledge
  in the form of function definitions
  can be used

Revising quantitative models (2001)

Given

- An existing quantitative model in the form of equations
- Additional observations of the variables in the model

Find a revised model

- That will fit the additional observations better
- That is as similar as possible to the original model

The basic procedure of revising quantitative models

- Take an existing model / equation (or set thereof)
- Construct a grammar that derives the model
  (this can be done automatically)
- Identify the most unreliable parts of the model (expert)
- Modify the grammar to specify alternatives for these parts (expert)
- Start equation discovery (LAGRAMGE)
  on the observations and grammar;
  during the search for equations prefer those
  that are most similar to the original one (minimal change)

Experimental evaluation: the CASA model

Developed at NASA Ames research center

We considered only the NPPc part of CASA
Predicts monthly net primary production of carbon at a given location

Dataset consists of the measurements of all variables
involved in the NPPc part of CASA for 303 locations

Two kinds of revisions: values of the constant parameters and
structure of equations defining the system variables

Submodels chosen for revision by environmental scientists:
the e_{max} constant; T_1; T_2; SR_FAS

Excerpt from original equations and corresponding grammar

$$NPPc = \max(0, E \cdot IPAR)$$
$$E = 0.389 \cdot T1 \cdot T2 \cdot W$$
$$T1 = 0.8 + 0.02 \cdot topt - 0.0005 \cdot topt^2$$
$$T2 = 1.1814/((1 + e^{0.2 \cdot (TDIFF - 10)}) \cdot (1 + e^{0.3 \cdot (-TDIFF - 10)}))$$
$$TDIFF = topt - tempc$$

NPPc → max(const[0:0], E * IPAR)
E → const[0.389:0.389] * T1 * T2 * W
T1 → const[0.8:0.8] + const[0.02:0.02] * topt
- const[0.0005:0.0005] * topt * topt
T2 → const[1.1814:1.1814] / ((const[1:1] + exp(const[0.2:0.2]
* (TDIFF - const[10:10]))) * (const[1:1]
+ exp(const[0.3:0.3] * (-TDIFF - const[10:10]))))
TDIFF → topt - tempc

Experimental evaluation

- E $\rightarrow$ const[0:0.778] * T1 * T2 * W
  Revised e_{max}=0.6423; 10cv error reduction: 12.59%
- T1 $\rightarrow$ const | const + (T1) * topt
  Revised T1 = $0.000142 * topt^5 - 0.0057 * topt^4 + 0.11 * topt^3 - 1.08 * topt^2 + 4.75 * topt - 5.31$
  10cv error reduction: 13.05%
- T2 replaced by a constant value, 10cv error reduction: 13.25%
- Best individual revision: 20% relative change of the constant parameter in the SR_FAS submodel (10cv error reduction: 14.93%)
- Best combination of four revisions has error reduction of 16.19%, but minimality of change would favour the SR_FAS revision
- Expert comments:
  Empirical improvements wellcome, given the uncertainty in the SR_FAS submodel (due to limited terrestrial coverage of NPPc measurements)

Using higher level domain knowledge in LAGRAMGE2.0 (2001/2002)

Based on the notions of variables and processes

Domain-specific modeling knowledge

- Taxonomy of (generic) processes in the domain
- Alternative models (templates) for each process
- How to combine the influences of processes on a variable

Task specific knowledge (i.e., concrete variables or processes)

Transform into a (context-dependent) grammar, use LAGRAMGE2.0

Exemplified on population dynamics modelling

The taxonomy of population dynamics processes

Single population processes

- Growth
  - Unlimited (Exponential)
  - Limited by a carrying capacity (Logistic)
- Decay

Interactions between pairs of populations

- Predator-prey
  - Unsaturated
  - Different types of saturation
- Symbiosis
- Parasitism

An example of domain and task specific knowledge

template(saturation, X, (X)).
template(saturation, X, (X / (X + const[0:]))).
template(saturation, X, (X * X / (X * X + const[0:]))).
template(saturation, X, (1 - exp(-const[0:] * X))).

template(growth, X, (const * X)).
template(growth, X, (const * X * (1 - X / const[0:]))).

$$\dot{N} = growth\_rate(N) - feeds\_on(P, N)$$
$$\dot{P} = feeds\_on(P, N) - decay\_rate(P)$$

```
task(aquatic).

population(aquatic, nut).
population(aquatic, phyto).     feeds_on(aquatic, phyto, nut).
population(aquatic, zoo).       feeds_on(aquatic, zoo, phyto).
```

The resulting grammar

```
aquatic ->
  time_deriv(nut) = - feeds_on(phyto,nut);
  time_deriv(phyto) = growth(phyto) + feeds_on(phyto,nut) - feeds_on(zoo,phyto);
  time_deriv(zoo) = const * zoo + feeds_on(zoo,phyto)

feeds_on(phyto,nut) -> const[0:]  * phyto * nutrient(nut)
feeds_on(zoo,phyto) -> const[0:]  * zoo * nutrient(phyto)

nutrient(nut) -> nut
nutrient(nut) -> nut/(nut+const[0:])
nutrient(nut) -> nut*nut/(nut*nut+const[0:])
nutrient(nut) -> 1-exp(-const[0:]*nut)

nutrient(phyto) -> phyto
nutrient(phyto) -> phyto/(phyto+const[0:])
nutrient(phyto) -> phyto*phyto/(phyto*phyto+const[0:])
nutrient(phyto) -> 1-exp(-const[0:]*phyto)

growth(phyto) -> const*phyto
growth(phyto) -> const*phyto*(1-phyto/const[0:])
```

P. Langley, J. Sanchez, L. Todorovski and S. Džeroski.
Inducing Process Models from Continuous Data. Talk @ ICML-2002.

Summary

- ILP focus has shifted away from logic program synthesis
  (to topics such as relational data mining, kernel methods for structured data and relational reinforcement learning)

- Key ideas from ILP concern the use of domain knowledge
  - Background knowledge
  - Declarative bias
  - Theory revision

- We have used these ideas (and other from ILP) in a number of equation discovery systems we have developed

- The use of domain knowledge has proved useful in applying equation discovery to several environmental problems

Advantages of learning in rich representations

Allows for solving more complex problems

Gets closer to human learning

- Difficult problems are not solved from scratch
- Existing domain knowledge is used
- Communicability

Trades off between (quality and quantity of) data and domain knowledge

- Little data, lots of knowledge
- Lots of data, little knowledge
- A reasonable amount of both data and knowledge

Disadvantages of learning in rich representations

Increased computational complexity

- Even deductive reasoning likely to be expensive
- Large hypothesis spaces
- Adding background knowledge enlarges the hypothesis space, this has to be compensated for (to an extent) by declarative bias and/or initial models

The learning process is less likely to be completely automated

- Data are not enough: domain knowledge needed
- Domain knowledge not trivial to encode (need to have domain specific or customizable formalisms)
- User interaction gains importance

Langley's new lessons for scientific discovery

1. Traditional ML notations are not easily communicable to scientists

2. Scientists often have initial models
   that should influence the discovery process

3. Scientific data are often rare and difficult to obtain rather than plentiful

4. Scientists want models that move beyond description to
   provide explanations of data

5. Scientists want computational assistance
   rather than automated discovery systems

Relation to the advantages/disadvantages

- Ability to trade-off data/domain knowledge needed (2,3)
- Hypotheses much more likely to be communicable, explanatory and accepted by users/scientists if formulated in terms of/consistent with domain knowledge (1,4)
- This is very likely if they are similar to existing theories as required by theory revision (2)
- Complete automation undesirable (5)

Further research directions

- Learning in other rich representations,
  logic-based or otherwise (e.g., graphs, pathways)

- Domain-specific libraries of models (constructed by humans)
  - To be used as background knowledge and/or for revision
  - This should be feasible in scientific domains
  - cf. the ECOBAS initiative in ecological modelling

- Inductive databases
  - Learned models/patterns stored in IDB
  - Thus available for further consideration/generalization/revision
  - Finding models/patterns that satisfy given constraints

Further research directions

- Using domain knowledge in other (than predictive modelling) learning settings, such as clustering and reinforcement learning

- We have taken a first stab at this with relational reinforcement learning
  - Background knowledge and declarative bias in learning a relational representation of the Q-function (e.g. *above(A,B)* in the blocks world)
  - Indirect theory/policy revision: Generate traces from given policy, use them to bootstrap the Q-learning process (K. Driessens, p.m.)

- But much more room remains for improvement, e.g.,
  - Declarative bias (specify the space of possible policies)
  - Policy revision (change policy to perform better, but keep it similar to initial policy)
  - Both can apply to Q-function representations or direct policy representations

Take-home message

- Background knowledge, declarative bias and theory revision can be used for different representations and learning tasks
- Using domain knowledge and rich representations gives you a lot of power to solve complex problems
- Consider using them for the learning tasks you are studying
- Questions to think about given a learning task
  - What domain knowledge is available?
  - What representation is suitable for representing domain knowledge and hypotheses? Consider formalisms established in the domain of use!
  - What learning algorithms can handle this representation?
  - Can existing algorithms be modified to learn in this representation?