COMPUTATIONAL SCIENTIFIC DISCOVERY AND INDUCTIVE DATABASES

Sašo Džeroski

Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia

Outline

- Computational scientific discovery
- Equation discovery in Ljubljana
  - Lines of research
  - Equation discovery systems
  - Using domain knowledge
- Inductive databases
- Inductive databases and equation discovery
- Summary / discussion / further work

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

A sample of recent work in this area will be provided in

S. Džeroski and L. Todorovski, editors (2003)
Computational Discovery of Communicable Knowledge.
Springer, Berlin. In preparation.

Computational scientific discovery vs. Scientific data mining

Scientific data mining = Mining scientific data

- Uses data mining methods to find patterns in scientific data

- Patterns represented as trees/rules/probabilistic models/etc., in formalisms used in data mining (by data miners)

Computational scientific discovery
emphasizes formalisms used to communicate among scientists

- Numeric equations

- Reaction pathways

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

Language bias

Defines the language of patterns or models
considered by a machine learning or data mining systems

The definition of the language considered is most often
implicit in the design of the system

In many cases, data mining systems allows only a minimal
(and possibly indirect) influence of the user
on the language of patterns considered (e.g., in decision trees)
by setting some parameters

Declarative language bias explicitly specifies
the language of patterns considered - in an understandable fashion
This is then an input to the system, in addition to the data

Domain knowledge

Background knowledge specifies laws/regularities/rules generally valid in the given domain. It specifies concepts/functions/model fragments that can be used in formulating new patterns/models

- In chemistry, BK comprises definitions of functional groups (such as alcohol, benzen ring, etc.)
- In chess, BK defines how different pieces move on the chessboard
- In combinatorics, BK includes a definition of factorial function

Language bias can be used to specify how building blocks (model fragments) can be put together into models/patterns

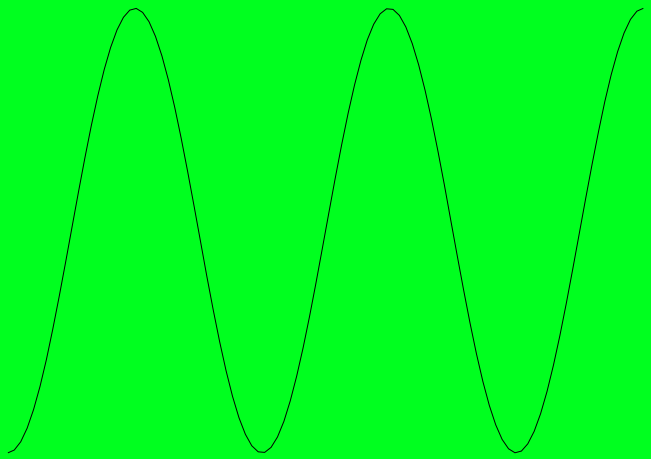
Existing models can be used as a starting point in the search for new models that explain new data better

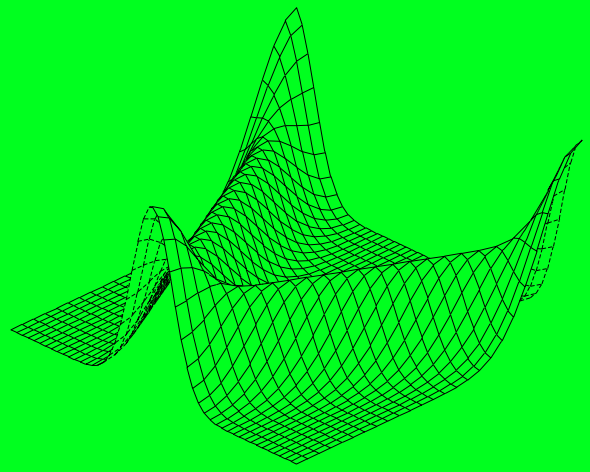
Different types of equations

- 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)

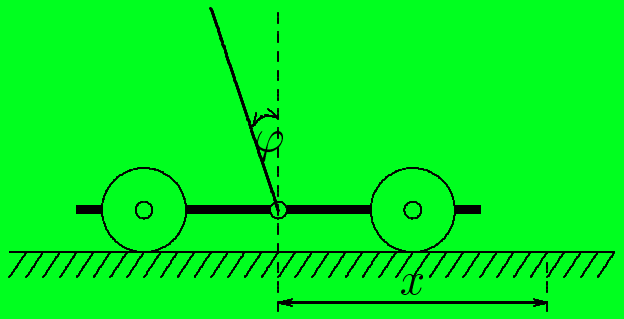
LAGRANGE example – Pole and cart

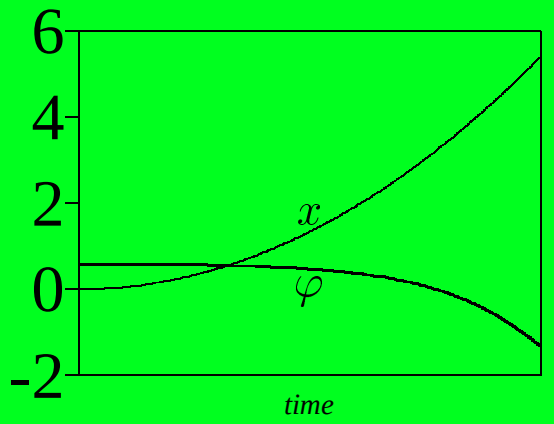
Model equation

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

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

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





LAGRANGE output

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

LAGRAMGE (1997) - Grammar based equation discovery

Given

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

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

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

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

$$DecayPhyto \rightarrow const \cdot phyt$$

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

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

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

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

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:])
```

Equation discovery summary

- Computational scientific discovery focusses on representations used by scientists

- Key ideas concerning the use of domain knowledge
  - Background knowledge
  - Declarative bias
  - Theory revision

- We have used these ideas 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

Inductive databases (Imielinski and Mannila 1996)

A database perspective on knowledge discovery:
Knowledge discovery processes are query processes

”There is no discovery in KDD,
it’s all a matter of the expressive power of the query language”

Inductive database = Database + Patterns

Sets of patterns can be materialized or defined as views

Data mining operations = Inductive queries

In addition to patterns (which are of local nature),
models (which are of global nature) can also be considered

A general formulation of data mining (Mannila and Toivonen 1997)

Given a language $\mathcal{L}$ of patterns to be considered,
a database $\mathbf{r}$ and a selection predicate q,
the aim is then to find the theory $Th(\mathcal{L}, q, \mathbf{r}) = \{\phi \in \mathcal{L} \mid q(\phi, \mathbf{r}) \text{ is true}\}$

This formulation involves the specification of

- A language of patterns
- A set of constraints that a pattern has to satisfy
  - Language constraints (concern only the pattern itself)
  - Evaluation constraints
    (concern the validity of the pattern with respect to a database)

Constraints play a central role in data mining
Constraint-based data mining - a recognized research topic (Bayardo 2002)

An approach to inductive databases

Types of patterns = Pattern domains

- Frequent itemsets
- Sequential patterns
- Molecular fragments
- ...
- Equations!

Inductive queries = Constraints (defined in terms of constraint primitives)

Design inductive query languages and solvers
for the different pattern domains

Following a philosophy akin to Constraint Logic Programming (CLP)
(De Raedt 2002)

Inductive queries

An inductive database instance can contain three components:
data (r_d), **patterns** (r_p), and **background knowledge** (r_b)

Types of (inductive) queries

- **Data retrieval**: the query uses $r_d \cup r_b$ and its result is in r_d;
  no pattern is involved in the query; this is a "classical" database query
- **Cross over**: the query uses $r_d \cup r_b \cup r_p$ and its result is in r_d;
  the query crosses over patterns and data in order to obtain new data
- **Processing patterns**: the query uses $r_p \cup r_b$ and its result is in r_p;
  patterns are queried without access to the data;
  this is what is usually done in post-processing
- **Data mining**: the query uses $r_d \cup r_b \cup r_p$ and its result is in r_p; new patterns are generated on the basis of the data and the existing patterns

Inductive query language primitives

- **Evaluation primitives**: functions that return some information about the semantics/validity of patterns in the data

- **Constraints**: boolean functions that specify whether their argument (pattern) should be returned as a result of the query
  - Evaluation constraints
  - Language constraints

- **Optimization primitives**: allow us to find patterns that optimize the value of an evaluation function

- **Connectives**: typically boolean operations, most often conjunction, used to compose primitive constraints; it is very common to have conjunctions of syntactic constraints and evaluation constraints

Inductive databases and equation discovery - A pattern domain of equations

Patterns (Models) = Equations

Background knowledge = Function definitions

Evaluation primitives for equations:
calculate the degree of fit of an equation to the data in a table

Evaluation constraints (use evaluation function)

Language constraints

- Parametric
- Subsumption for polynomials
- Similarity

Evaluation primitives for equations

Let D be a table with real valued attributes; i an index of records in D;
e an equation of the form LHS = RHS;
y_i the value of the LHS of a given equation on record i;
$\hat{y_i}$ the value of the RHS on the same record;
N the number of records in the table;
and $\overline{y}$ the mean value of y_i over the table records.

- The multiple correlation coefficient $R(e, D)$:

$$R^2 = 1 - \frac{\Sigma_{i=1}^{N}(y_i - \hat{y}_i)^2}{\Sigma_{i=1}^{N}(y_i - \overline{y})^2}$$

- The normalized standard deviation $S(e, D)$:

$$S^2 = \frac{\frac{1}{N}\Sigma_{i=1}^{N}(y_i - \hat{y}_i)^2}{\overline{y}^2 + e^{-\overline{y}^2}}$$

Evaluation primitives for equations (continued)

- The mean absolute error $MeanAE(e, D)$:

$$MeanAE = \frac{1}{N} \sum_{i=1}^{N} |\hat{y}_i - y_i|$$

- The maximum absolute error $MaxAE(e, D)$:

$$MaxAE = \max_{i=1}^{N} |\hat{y}_i - y_i|$$

- The mean square error $MSE(e, D)$):

$$MSE = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$

- The root mean square error $RMSE(e, D)$:

$$RMSE = \sqrt{MSE}$$

The language of polynomial equations

Given a set of variables V (set of real-valued attributes in a RDB table)

A polynomial equation has the form $T = P$,
where T is a term over V and P is a polynomial over V

A term is a finite product of variables, multiplied by a real-valued constant.
The degree of a variable is the number of times it appears in a term,
the degree of a term the sum of the degrees of its variables.

A polynomial is a sum of terms.
Its degree is the maximum of the degrees of its terms.

For a set of variables $V = \{X, Y, Z\}$, X^2Y is a term of degree 3,
XYZ^3 is a term of degree 5. A polynomial of degree 4 is $X^2Y + 2Y^3Z$.
An example polynomial equation is $XYZ^2 = 2X^2Y + 5Y^3Z$.

Subsumption constraints on polynomial equations

We ignore the real-valued constants and focus on equation structures

Subsumption constraints
introduce a lattice on the space of equation structures

- We can represent a polynomial as a set of terms.
  A term can be represented as a multiset of variables.
  For example, $XYZ^2 = \{X, Y, Z, Z\}$.
- T_1 subterm of T_2 iff $M_1 \subset M_2$ for the corresponding multisets.
  For example, XY^2 is subterm of X^2Y^4Z
  since $\{X, Y, Y\} \subset \{X, X, Y, Y, Y, Y, Z\}$.
- p_1 is a subpolynomial of p_2
  iff each term in p_1 is a subterm of some term in p_2.
  For example, $XY + Z$ is a subpolynomial of $2X^2Y^2 + 4Z^3$.

Language constraints on polynomial equations

Consider equation e of the form $T = P$

Parametric constraints:
set bounds for the degree of T and P and the number of terms in P

Subsumption constraints (t is a term, p polynomial)

- T is a subterm of t, t is a subterm of T
- P is a subpolynomial of p, p is a subpolynomial of P

Similarity constraints
Find equation as similar as possible to e that also satisfies other constraints
Similarity measures can be defined for equation structures,
based on similarity measures for sets and multisets

Inductive queries on polynomial equations

Consist of conjunctions of

Evaluation constraints: $M(e, D) < v$; $M(e, D) > v$,
where v is a positive constant and M one of the measures defined above

Optimization constraints: find e so that $M(e, d)$ is maximal / minimal

Language constraints: parametric and/or subsumption constraints

Similarity constraints: find equations most similar to e

Typical combinations of constraints:
Language + evaluation; Language + optimization; Evaluation + similarity

Inductive databases - summary

Inductive databases

- Contain data, background knowledge, and patterns/models
- View the process of knowledge discovery as
  an interactive querying process, thus supporting active mining
- Inductive queries consist of constraints
  These comprise evaluation, optimization, and language constraints
- The use of domain knowlede in the form of language bias and
  existing models (similarity constraints/theory revision) is supported
- The pattern domain of equations naturally fits the IDB framework

Advantages of learning with domain knowledge

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 with domain knowledge

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

- Capability of using domain knowledge
  essential for computational scientific discovery

- 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)
  The interactive inductive databases/queries approach is better!

Summary

We have developed a number of systems for discovering equations, a formalism widely used by scientists, but also elsewhere

- Different types of equations can be considered
- Different types of domain knowledge can be used

We are working on integrating equation discovery into the inductive database framework

- To facilitate an interactive process of computational scientific discovery (the domain scientist wants to be in control)
- While preserving the capability of using domain knowledge

Further work

Developing libraries of models and model fragments
in individual domains of application

- To be used as background knowledge and/or for revision

- Work in progress on a library for ecological modelling
  of aquatic ecosystems (and in particular lakes)

Integrating different pattern domains in inductive databases

- This would allow the combination of different representation languages

- As is often needed in practice, e.g. in QSAR modelling

Goseicho arigato gozaimasu

Nani ka shitsumon wa arimasu ka?

Eigo de shitsumon shite kudasai!