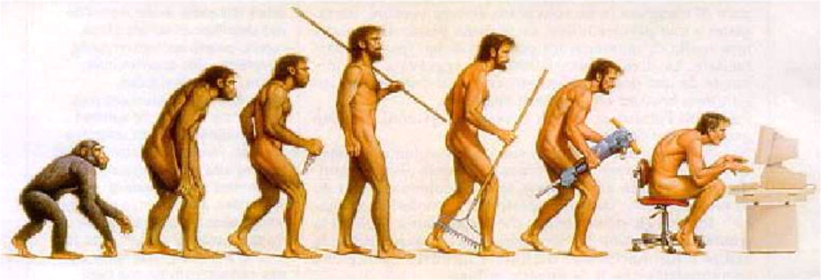
Which Parameters/Algorithm/System Should I Use ?

Autonomic Computing with Data Mining

Michele Sebag
CNRS – INRIA – Université Paris-Sud
http://www.lri.fr/∼sebag

IEEE DM Forum
Hong Kong May 29th, 2008

Autonomic Computing



Considering current technologies, we expect that the total number of device administrators will exceed 220 millions by 2010

Gartner 6/2001

in Autonomic Computing Wshop

Irina Rish & Gerry Tesauro, ECML / PKDD 2006

Autonomic Computing

The need

- Main bottleneck of the deployment of complex systems: shortage of skilled administrators

Vision

- Computing systems take care of the mundane elements of management by themselves.
- Inspiration: central nervous system (regulating temperature, breathing, and heart rate without conscious thought)

Goal

Computing systems that manage themselves in accordance with high-level objectives from humans

Kephart & Chess, IEEE Computer 2003

Dream Algorithms, Two Visions

The Computer Scientist View

- My program can do anything
- Just tell me what you want – what your problem is
  There's a default option; but you can do so much better...

The Software Editor View

- Just press the **GO** button
  the pristine simplicity of the Google screen...

Autonomic Software is badly needed too...

General Goal: Crossing the Chasm

Marketing & Selling High-Tech Products to Mainstream Customer

Geoffrey A. Moore, 1991

User's question:

Which algorithm/system is best suited to MY problem ?

The obvious answer..

Just take the best one !

... does not work

No Free Lunch Theorem

Forget about the killer algorithm/system, period

Growing needs & Growing Field

- IBM Manifesto for Autonomic Computing 2001
  http://www.research.ibm.com/autonomic
- ECML/PKDD Wshop on Autonomic Computing 2006
  http://www.ecmlpkdd2006.org/workshops.html
- JIC. on Measurement and Performance of Systems 2006
  http://www.cs.wm.edu/sigm06/
- NIPS Wshop on Machine Learning for Systems 2007
  http://radlab.cs.berkeley.edu/MLSys/
- Networked System Design and Implementation 2008
  http://www.usenix.org/events/nsdi08/

Overview

1. Motivations
2. Autonomic Computing: a Killer Application for DM
   - Optimization
   - Meta-Learning
   - Competence Maps
3. Autonomic Computing: Lessons for Data Mining

Autonomic Computing

1. Optimization
2. Meta-Learning
3. Competence Maps

Autonomic Computing with Optimization

Find the best parameter configuration for a single algorithm

- Define an objective function

  e.g. computational cost or quality of the solution,...

- Define a suite of representative problem instances

  ... use benchmarks...ask experts...

- Search space: defined by the algorithm parameters

  discrete & continuous

Autonomic Computing with Optimization

Examples

- R. Kohavi & G. John ICML 1995

  33 pb, best-first search in parameter space
- M. Birattari & al. GECCO 2003

  a racing alg. to filter out bad parameter settings
- B. Srivastava & A. Mediratta AAAI 2005

  apply decision tree in parameter space
- B. Adenso-Daz & M. Laguna Operations Research, 2006

  fractional experimental design in parameter space
- ...

Autonomic Computing with Meta-Learning

Find the best algorithm for a given problem instance

Specification

Given

algorithm $\mathcal{L}$, dataset D

Predict

whether $\mathcal{L}$ is the best alg. on D

the predictive accuracy of $\mathcal{L}$ on D

Autonomic Computing with Meta-Learning

Find the best algorithm for a given problem instance

Specification

Given

algorithm $\mathcal{L}$, dataset D

Predict

whether $\mathcal{L}$ is the best alg. on D — *binary classification*

the predictive accuracy of $\mathcal{L}$ on D — *regression*

Autonomic Computing with Meta-Learning

Find the best algorithm for a given problem instance

Specification

Given

algorithm $\mathcal{L}$, dataset D

Predict

whether $\mathcal{L}$ is the best alg. on D — *binary classification*

the predictive accuracy of $\mathcal{L}$ on D — *regression*

Resolution

A Discriminant Learning pb : Meta-Learning

The EU METAL project

Kalousis PhD, 2002

Janssen Furnkranz, 2007

Input

gather meta-examples: $\mathcal{E} = \{(x_i = (D_\ell, \mathcal{L}_k), y_i = \mathcal{L}_k(D_\ell))\}$

Output

Construct $\hat{y}(D, \mathcal{L})$ from $\mathcal{E}$

Use

For every D, use $\mathcal{L}^* = argmax_k\{\hat{y}(D, \mathcal{L}_k)\}$

The EU METAL project, 2

The formulation is brilliant ! Now let us gather meta-examples:

Find representative problems

Irvine repository ?

Find good features

Some are obvious

(number of examples, features, values, classes,...)

Some are useful but data are not representative

(missing information rate,...)

Some are as expensive as solving the problem

(distribution of the data)

Data gathering/preparation is 80% of the task

Build compound heuristics

Janssen and Furnkranz 2007

The context: Separate and Conquer

- Greedy optimization of a (non-monotonic) heuristics
- Remove covered examples
- Assess on test set

The Metal space

- Heuristics: precision, Laplace, accuracy, WRA, correlation
- Descriptors: True/false positive rate, prior, length,...

The result

The induced heuristics improves on the previous best

Limited scope

- Selection of representative problems
- Descriptive features

Autonomic Computing

1. Optimization

   The best default for an algorithm

2. Meta-Learning

   The best algorithm for a problem instance

3. Competence Maps

   Modeling the behaviour of an algorithm

   - Relational domains
   - Propositional domains

Taking a cue from CSP community

Cheeseman, IJCAI 91

Where are the really hard problems ?

- CSP are NP hard   worst case
- Still, algorithms often behave well...

An engineer's view:

- 80% of the problems are easy to solve
- we spend 80% of our time on the other 20%

Constraint Satisfaction Problems

Given

variables: $X_1, ...X_n$	n
domains: X_j in $\Omega = \{a_1, ..a_L\}$	L
constraints: $r_i(X_j, X_k))$	m
relations: $Rel(r_i) = \{r_i(a_2, a_3), r_i(a_4, a_7), ...\}$	N

Find

assignment θ: $X_j \mapsto \{a_1, ..a_L\}$, $j = 1..n$

such that

$$r_i(\theta(X_j), \theta(X_k)) \in Rel(r_i), \;\; i = 1..m$$

Constraint Satisfaction Problems, 2

Order parameters

constraint density $p_1 = 2\frac{m}{n(n-1)}$

constraint tightness $p_2 = 1 - \frac{N}{L^2}$

Any measure f on csp instances

f = satisfiability *[whether csp admits a solution]*

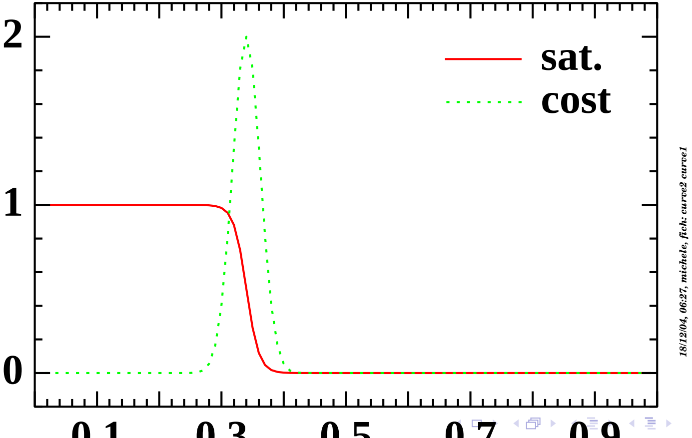
f = computational cost *[for finding one or proving there isn't any]*

$\rightarrow$ Random variable $F(p_1, p_2)$

The Phase Transition

Experiments based on CSP sampling — Fix p_1, increase p_2

YES region	underconstrained CSPs
NO region	overconstrained CSPs
Phase transition	where the real hard pbs are



Autonomic Computing

1. Optimization

   The best default for an algorithm

2. Meta-Learning

   The best algorithm for a problem instance

3. Competence Maps

   Modeling the behaviour of an algorithm

   - Relational domains
   - Propositional domains

Phase Transition: Impacts on Relational Learning

WHY ?

In Relational ML/DM, Covering test = Θ-subsumption $\equiv$ CSP

Example

$$\begin{array}{ll} h: & atm(X), atm(Y), atm(Z), bond(X,Y), bond(X,Z) \\ Ex: & atm(a), atm(b), atm(c), atm(d), \ldots \\ & bond(a,b), bond(b,c), bond(b,d), \ldots \end{array}$$

$$h \text{ covers } Ex \quad iff\ \exists\theta \ / \ h\theta \ \subseteq \ Ex$$

Here:

$$\theta = \{X/b, Y/c, Z/d\}$$

Order parameters for θ-subsumption

Giordana Saitta MLJ 00

m	nb constraints	N	relation size
n	nb variables	L	domain size

Hypothesis space $\mathcal{H}_{n,m}$

Clauses with n variables and m predicate symbols.

$$h(X_1, .., X_n) = p_1(X_{j(1)}, X_{k(1)}) \wedge ... \wedge p_m(X_{j(m)}, X_{k(m)})$$

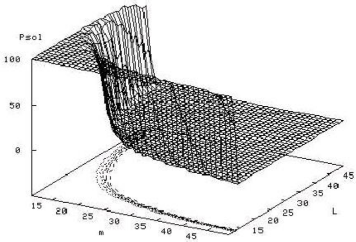
Example space $\mathcal{E}_{N,L}$

Examples with N literals per predicate symbol p_i,
involving L distinct constants

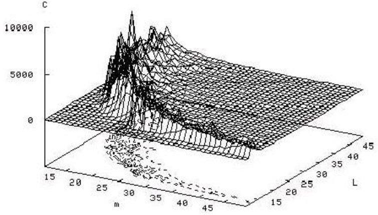
$$\begin{aligned} Ex = \quad & p_1(a_{j(1,1)}, a_{k(1,1)}) \wedge ... \wedge p_1(a_{j(1,N)}, a_{k(1,N)}) \wedge \\ & \ldots \\ & p_m(a_{j(m,1)}, a_{k(m,1)}) \wedge ... \wedge p_1(a_{j(m,N)}, a_{k(m,N)}) \end{aligned}$$

Phase Transition for Θ-subsumption

In plane m, L with $n = 10, N = 100$



Coverage



Cost

Impact of Phase Transition on ILP

Botta et al, JMLR 2003

Artificial ILP Problems

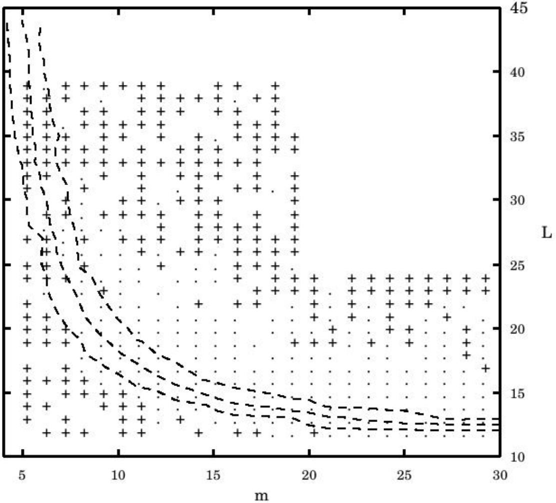
n=4, N=100

Problem (m, L):

1. Draw *tc* in $\mathcal{L}_{4,m}$
2. Draw examples in $\mathcal{E}_{100,L}$
3. Label examples according to *tc*
4. Gather *balanced* training and test sets.
5. Use FOIL : success if accuracy on test set $> 80\%$

Available :
http://www.di.unito.it/~mluser/challenge/index.html

FOIL Competence MAP



$+$ Success ($>$ 80% on test set) $\cdot$ Failure

Experimental evidences

- FOIL favors hypotheses in the PT

  the most relevant region
- But gets lost on the path for medium size *tc*

  search criteria misleading in the YES region
- Discovering long *tc* is much easier

  (any $gen(tc)$ in the PT will do)

Note

Localizing FOIL failure region leads to new algorithms

Ales-Bianchetti et al., ICML 2002

Autonomic Computing

1. Optimization

   The best default for an algorithm

2. Meta-Learning

   The best algorithm for a problem instance

3. Competence Maps

   Modeling the behaviour of an algorithm

   - Relational domains

     JMLR 2003, MLJ 2004, IJCAI 2005, ILP 2007, JIIS 2008

   - Propositional domains

     ICML 2004

Assessing / Understanding systems

Principle natural and physical sciences

Hypothesize *order parameters*

Use these parameters to observe systems/entities/algs

Find regularities

- functioning modes
- localization of the transitions

[or refine order parameters]

Quality

ML: predictive accuracy

DM: Type I & Type II errors

Competence Maps in Machine Learning

According to order parameters

- Draw ML problems (training set, test set)
  - learn h on training set
  - compute $Err(h)$ on test set
- Average Err over all pbs with same order parameters
- Competence map: Err(order parameters)

Criteria

- Readable competence map
- Low variance of error

A case study: C4.5R

Baskiotis-Sebag, ICML 04

Order parameters

- m: nb of features — instance space $\{0,1\}^m$
- k, ℓ: target concept $= k - \ell$ DNF

$$tc = C_1 \vee .. \vee C_k \text{ where } C_i \text{ involves } \ell \text{ literals}$$

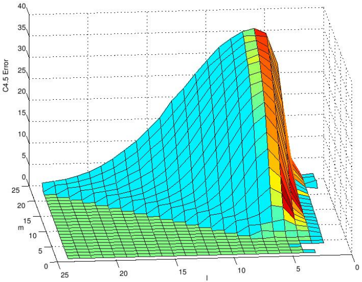
- r: fraction of positive examples
- ε: label noise

First experimental setting

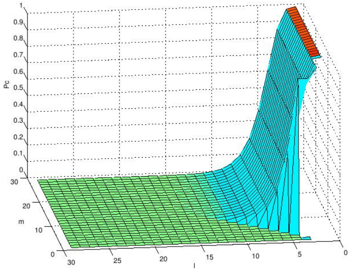
- $m = 5..30$
- $k = 1..20$
- $\ell = 1..m$
- Fix: $r = 1/2, \varepsilon = 0$

Compute $Err(m, k, \ell)$ — averaged over 100 pbs (m, k, ℓ)

Competence Map(C4.5R ;m, k, ℓ)



Error



Coverage

The error peak coincides with the coverage transition.

Discussion

Pro and Cons

$+$ readable

$-$ high variance of error

$-$ $k - \ell$-DNF very limited language

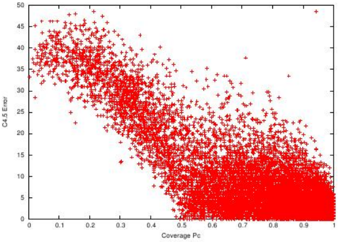
$\Rightarrow$ Competence Map not usable...

Order parameters, revisited

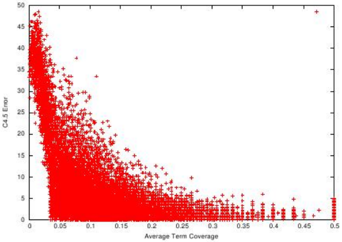
m: nb of features — instance space $\{0,1\}^m$

P_c: coverage of tc

P_{ac}: average coverage of conjuncts in tc



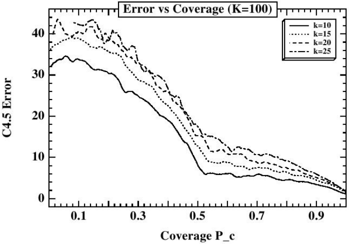
Err vs P_c



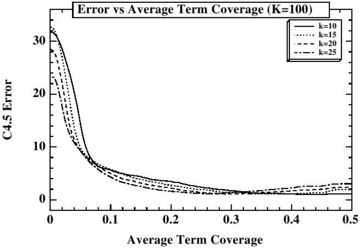
Err vs P_{ac}

C4.5R Results

(1 point per pb instance)



Err vs P_c



Err vs P_{ac}

C4.5R Results

(convolution with Gaussian kernel)

C4.5R Competence map

It works

Competence map == Lookup table

Predict C4.5R error with good precision from P_c, P_{ac}

Limitation

P_c, P_{ac} must be guessed by the expert

About C4.5

tells nothing totally new Holte 89

but tells it precisely

Conclusion
Behavioural modelling of algorithms/systems

- Towards Autonomic Computing
- A Killer Application for Data Mining
- Allows for Certification
- Allows for Improving Algorithms and Systems
  based on identifying their failure region

Perspectives
Competence Maps for Data Mining

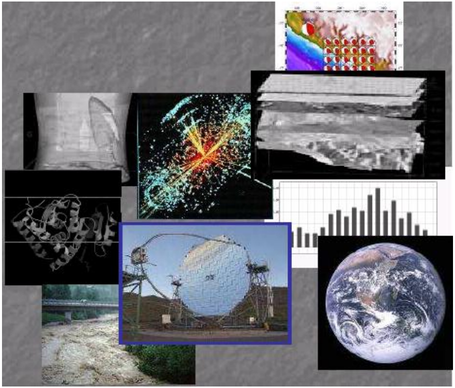
- Identification of the PT for Relational DM
- Identification of order parameters for DM
  - Density, Feature correlation, Rotation...
- Beyond computational cost: Type I and Type II Errors

Challenge to come

Modelling the EGEE Grid

Enabling Grids for e-Science in Europe, http://www.eu-egee.org

FP6 and FP7
Large scale grid for e-Science
91 partners, 32 countries
20K CPUs, 5PB
20K jobs 24 x 7



Goal: Grid modelling

Heterogeneous systems: processors, storage, network, services.

State can at most be estimated

Mutualisation paradigm: load depends on collective behavior

... must be estimated on the fly

Needed: a grid model, in order to

- Control and maintain the system — detect ill-configured units
- Predict the application performances — dimension the capacities for jobs
- Optimize the system — refine the scheduler

Modelling the grid: a DM problem

Input data

Traces of the jobs:

- 800 Ko per job, including specifications and all events
- some hundred thousands jobs per trace
- spatio-temporal (redundant) structure

Goals

- Classification: jobs are *done, aborted*, or *lost*
- Early detection: predict as early as possible
- Clustering: provide the user with model chunks and/or outliers

Call to Arms !