# KD Ubiq Summer School 2008
# Behavioural Modelling of a Grid System

Michele Sebag
CNRS − INRIA − Université Paris-Sud
http://tao.lri.fr

March 8th, 2008

# What is this about?

## Grids for Machine Learning/Data Mining — NO

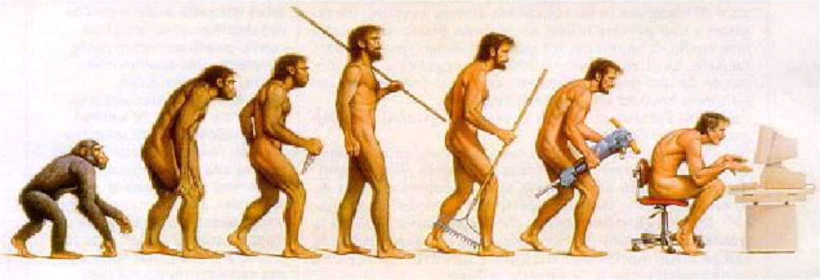
Distributed-*Everything*: — marginally

- Feature construction/selection
- Model selection
- Reinforcement learning
- Optimization

## Machine Learning/Data Mining for Grids — YES

- Grids are Complex Systems
- They work. Why ? How ?
- How: First principles $\vee$ Behavioural modelling
- Self-aware systems

# 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, ECML / PKDD 2006

Irina Rish & Gerry Tesauro.

# 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

# Autonomic Computing

## Activity: A 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 of the Tutorial

## Autonomic Computing

- ML & DM for Systems:
  Introduction, motivations, applications
- Zoom on an application: Performance management

## Autonomic Grid

- EGEE: Enabling Grids for e-Science in Europe
- Data acquisition, Logging and Bookkeeping files
- (change of) Representation, Dimensionality reduction

## Modelling Jobs

- Exploratory Analysis and Clustering
- Standard approaches, stability, affinity propagation

# ML & DM for Systems

## Some applications

- Cohen et al., OSDI 2004, Performance management

  detailed next

- Palatin-Wolf-Schuster, KDD06. Find misconfigured CPUs in a grid system

  find outliers

- Xiao et al. AAAI05, Active learning for game player modeling

  situations where it's too easy

- Zheng et al. NIPS03-ICML06, Use traces to identify bugs

  put probes, suggest causes for failures

- Baskiotis et al., IJCAI07, ILP07, Statistical Structural Software Testing

  construct test cases for software testing

# Advocated Attitude: *Bounded rationality*

H. Simon, 1958

*In complex real-world situations, optimization becomes approximate optimization since the description of the real-world is radically simplified until reduced to a degree of complication that the decision maker can handle.*

*Satisficing seeks simplification in a somewhat different direction, retaining more of the detail of the real-world situation, but settling for a satisfactory, rather than approximate-best, decision.*

# Performance management

## The goal

Ensure that the system complies with performance level objectives

## The problem: System Modelling

Large-scale system complex behavior depends on:

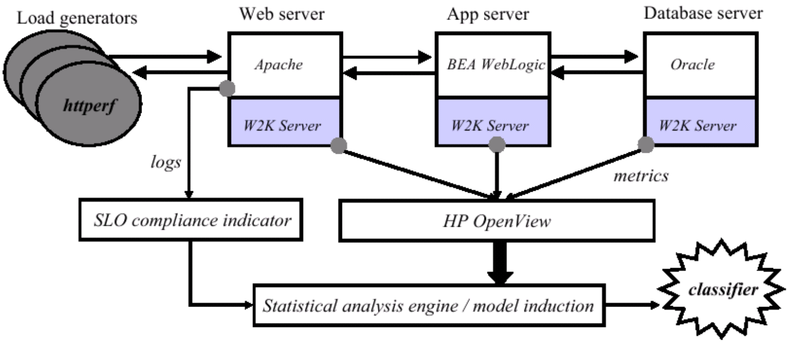
- Workload
- Software structure
- Hardware
- Traffic
- System goals

## The approaches

- Prior knowledge — set of (event - condition - action) rules
- Statistical learning

exploiting pervasive instrumentation / query facilities

# Example: a 3-tier Web application with a Java middleware component, backed by a DB




*Correlating instrumentation data to system states: A building block for automated diagnosis and control, Cohen et al. OSDI 2004*

# Supervised Learning, Notations

## Training set, set of examples, data base

(iid sample $\sim P(\mathbf{x}, y)$)

$$\mathcal{E} = \{(\mathbf{x}_i, y_i), \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1 \ldots N\}$$

- $\mathcal{X}$: Instance space
  - propositional (examples described after $D$ attributes) $\mathbb{R}^D$

    $$\mathbf{x} = (X_1(\mathbf{x}), \ldots X_D(\mathbf{x}))$$

  - relational (examples described after objects in relation, e.g. events - see later on)
- $\mathcal{Y}$: Label space
  - Discrete: classification (compliant, not-compliant)
  - Continuous: regression (average response time)

# Example

## Instance space, set of attributes

| Metric | Description |
|---|---|
| **mean_AS_CPU_1_USERTIME** | CPU time spent in user mode on the application server. |
| **var_AS_CPU_1_USERTIME** | Variance of user CPU time on the application server. |
| **mean_AS_DISK_1_PHYSREAD** | Number of physical disk reads for disk 1 on the application server, includes file system reads, raw I/O and virtual memory I/O. |
| **mean_AS_DISK_1_BUSYTIME** | Time in seconds that disk 1 was busy with pending I/O on the application server. |
| **var_AS_DISK_1_BUSYTIME** | Variance of time that disk 1 was busy with pending I/O on the application server. |
| **mean_DB_DISK_1_PHYSWRITEBYTE** | Number of kilobytes written to disk 1 on the database server, includes file system reads, raw I/O and virtual memory I/O. |
| **var_DB_GBL_SWAPSPACEUSED** | Variance of swap space allocated on the database server. |
| **var_DB_NETIF_2_INPACKET** | Variance of the number of successful (no errors or collisions) physical packets received through network interface #2 on the database server. |
| **mean_DB_GBL_SWAPSPACEUSED** | Amount of swap space, in MB, allocated on the database server. |
| **mean_DB_GBL_RUNQUEUE** | Approximate average queue length for CPU on the database server. |
| **var_DB_NETIF_2_INBYTE** | Variance of the number of KBs received from the network via network interface #2 on the database server. Only bytes in packets that carry data are included. |
| **var_DB_DISK_1_PHYSREAD** | Variance of physical disk reads for disk 1 on the database server. |
| **var_AS_GBL_MEMUTIL** | Variance of the percentage of physical memory in use on the application server, including system memory (occupied by the kernel), buffer cache, and user memory. |
| numReqs | Number of requests the system has served. |
| **var_DB_DISK_1_PHYSWRITE** | Variance of the number of writes to disk 1 on the database server. |
| **var_DB_NETIF_2_OUTPACKET** | Variance of the number of successful (no errors or collisions) physical packets sent through network interface #2 on the database server. |

## Label space

Compliance with Service Level Objectives (SLO) YES / NO

# Learning a model

## Desiderata

- Efficient — few prediction errors
- Compact — fast to use on further cases
- Easy/Fast to train — no expertise needed to use
- Interpretable — guide design/improvement

# Learning – Hypothesis search space

## Learning = finding $h$ with good quality

$$h \in \mathcal{H} : \mathcal{X} \mapsto \mathcal{Y}$$

## Loss function

$$\ell(y, y') = \text{ Cost of predicting } y' \text{ instead of } y$$

- $\ell(y, y') = 1_{[y=y']}$ — classification
- $\ell(y, y') = (y - y')^2$ — regression

# Learning – Hypothesis search space, 2

## Learning criterion

- Generalization error (ideal, alas $P(x, y)$ is unknown)

$$Err_{gen}(h) = E[\ell(y, h(\mathbf{x}))] = \int \ell(y, h(\mathbf{x}))dP(\mathbf{x}, y)$$

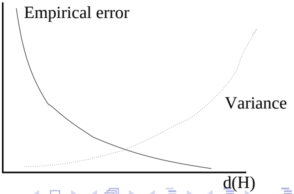
- Empirical error (known)

$$Err_{emp}(h) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, h(x_i))$$

## The bias/variance tradeoff

$d(\mathcal{H})$: dimension of Vapnik Cervonenkis

$$Err_{gen}(h) \leq Err_{emp}(h) + \mathcal{F}(n, d(\mathcal{H}))$$



# Bayesian Learning

## Bayes theorem

$$
\begin{aligned}
P(Y = y|X = \mathbf{x}) &= P(X = \mathbf{x}|Y = y).P(Y = y) \;/\; P(X = \mathbf{x}) \\
&\propto P(X = \mathbf{x}|Y = y).P(Y = y)
\end{aligned}
$$

Let $\mathbf{x} = (X_1(\mathbf{x}), \ldots, X_D(\mathbf{x})) \in \mathbb{R}^D$.
Assuming attributes are independent,

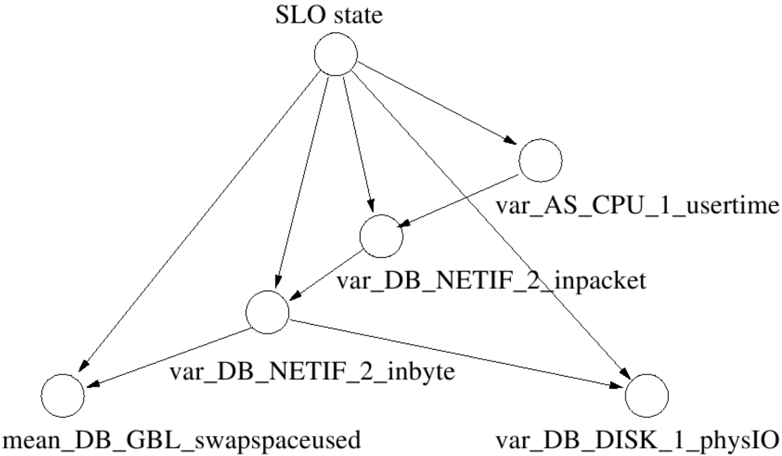
$$
P(X = \mathbf{x}|Y = y) = \prod_{i=1}^{d} P(X_i = X_i(\mathbf{x})|Y = y)
$$

Prediction: select class that maximizes the probability of $\mathbf{x}$

$$
\hat{y}(\mathbf{x}) = argmax\{\prod_{i=1}^{d} P(X_i = X_i(\mathbf{x})|Y = y_j).P(Y = y_j), y_j \in \mathcal{Y}\}
$$

# Tree-Augmented Naive Bayes

Learn probability of attribute $X_i$ conditionally to
* label $Y$;
* at most one other attribute $X_j$.



# Tree-Augmented Naive Bayes, 2

Friedman, Geiger, Goldszmidt, MLJ 1997

## Algorithm

- For each pair of attributes $(X_i, X_j)$, compute $I(X_i, X_j) =$

$$\sum_{v_i, v_j, y} P(X_i = v_i, X_j = v_j, Y = y) \ln \frac{P(X_i = v_i, X_j = v_j | Y = y)}{P(X_i = v_i | Y = y)\, P(X_j = v_j | Y = y)}$$

- Define the complete graph $\mathcal{G}$ with $I(X_i, X_j)$ on edge $(X_i, X_j)$
- Define the maximum weight spanning tree from $\mathcal{G}$

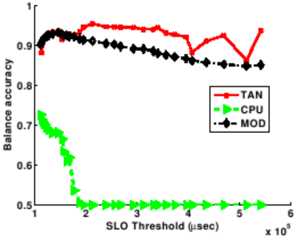
## Complexity

$D$ : number of attributes
$N$ : number of examples
Complexity: $\mathcal{O}(D^2 N)$

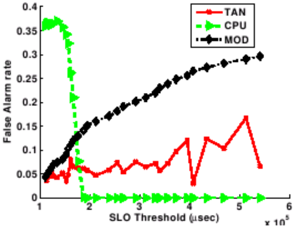
# Results: 1. Accuracy

Balanced accuracy $= \frac{1}{2}$ (True Pos. rate + True Neg rate ).
Measured by 10 fold CV

## Depending on performance threshold




Balanced accuracy




False alarm rate

- CPU: baseline predictor, use the CPU level only
- MOD: TAN trained with highest performance threshold
- TAN: TAN trained for each performance threshold

# Results: 2. Using the model

## Forecasting the failures

$$\ln \frac{P(X_{i,t+1} = v | X_{i,t} = v', Y = 0) P(Y = 0)}{P(X_{i,t+1} = v | X_{i,t} = v', Y = 1) P(Y = 1)} > 0$$

## Interpreting the causes of failures

- Direct interpretation might be hindered by limited description.
- Learning would select an effect for a (missing) cause.
- Example: *minute-average-load* used as *disk queue* is missing.

# Going ubiquitous – 1. What can be distributed

## The phenomenons

- Several instances of the process
- Confidentiality issues

## The examples

- For scalability
- Sampling with prior knowledge: e.g. periodicity
- Sampling with posterior knowledge: e.g. boosting, anomalies

## The attributes

- For scalability
- Feature selection
- Hidden causes
- Feature construction

# Going ubiquitous – 2. How to fuse/integrate partial results

## Migrating the examples

- Distinguishing outliers from novelties
- False discovery rate

## Migrating the models

- Claim: learning multiple models is GOOD.
- Exploration/Exploitation tradeoff.
  Island-model for Evolutionary Computation
- Confidentiality issues

## Cascading the models

- Pattern: If (Condition) Then Conclusion
- Throw the Conclusion, keep the Condition
- Turn it into a new Feature

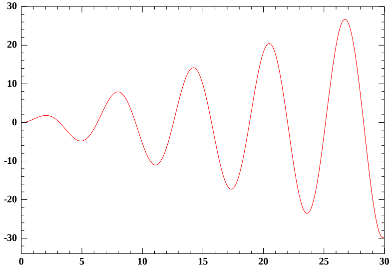
# Exploration vs Exploitation

## Exploitation

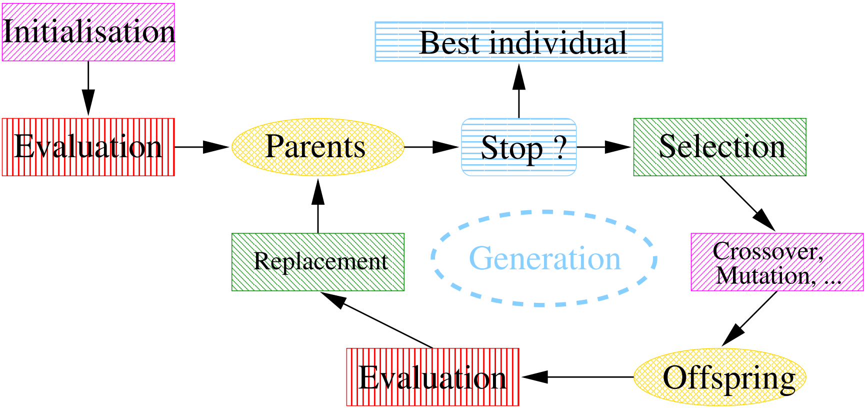
- Greedy optimization
- \+ Fast
- − Local optima

## Exploration

- Random walk
- \+ Finds global optimum with high probability
- − Very slow



# Evolutionary Computation




Stochastic operators (Representation-dependent)

"Darwinism" (stochastic or determinist)

Main CPU cost

Stopping and checkpointing

# Preserving diversity

## To be avoided

- Cloning: the best individual invades the population
- Diversity is lost, premature convergence

## Heuristics

- Restricted mating
- Control selective pressure
- Island model