Learning Action Models for Planning

Qiang Yang

Department of Computer Science,

Hong Kong University of Science and Technology,

Hong Kong, China

http://www.cse.ust.hk/~qyang

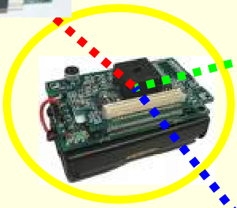
Acknowledgements

- My students
  - Junfeng Pan, Jie Yin, Dou Shen, Jialin Pan, Dr. Rong Pan, Kangheng Wu
- My sponsors
  - Hong Kong RGC
  - NEC China Labs
- My Colleagues
  - James Kwok, DY Yeung, Lionel Ni, Yiqiang Chen, Yunfei Jiang

Sensors provide a continuous stream of data

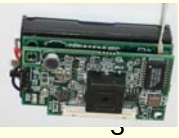
Node 1

Node 2



-30dBm	-70dBm	-40dBm

Node 7



南华早报
2006年3月12日

Sunday, March 12, 2006

Big Brother tracking device that will help to save lives

NIKI LAW

Prev. Story | Next Story

Spy technology right out of a James Bond film has been brought to life and will soon help save lives, or - to the horror of the inventors - supply Big Brother with its ultimate tool.

Artificial-intelligence experts at the University of Science and Technology have created a device of tiny sensors and a computer program that interprets data to track a person's every move, and even see what they are doing.

"It is the ultimate dream for any science fiction fan. Using these sensors we can tell the temperature of a person, the brightness of their surroundings, whether they have come into contact with any machinery, how fast they are moving, and hear what they are hearing," said Yang Qiang, a professor.

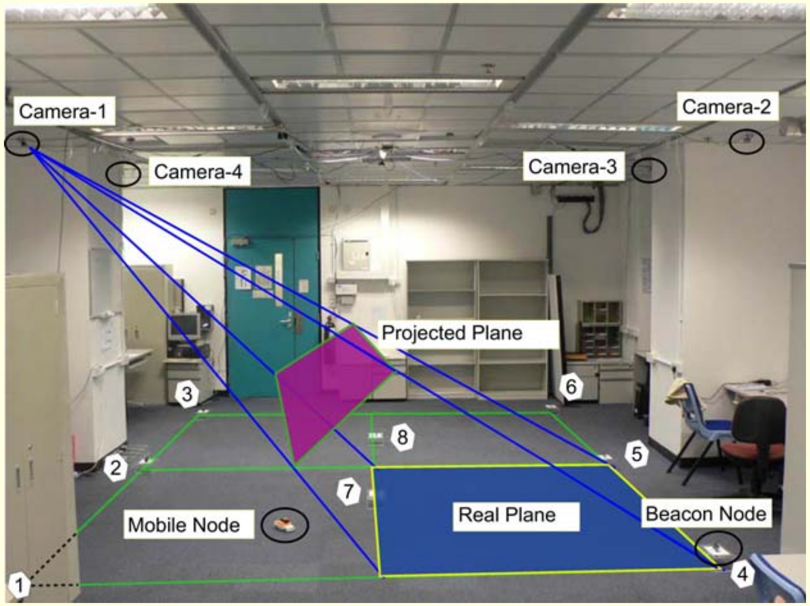
"It is like GPS for the indoor environment. All you have to do is put sensors on a person and along the path inside the building. Data is sent to a computer and within a second you know exactly what is

More Hong Kong stories

- Searches mean data may live on in cyberspace
- Mainland invitation considered for Zen
- Campaign to train 'fresh blood' for poll
- Hong Kong deputies call for Basic Law's review
- Conflict of interest concern for arts festival
- Boom in air traffic fuels rise in noise complaints
- Widow tells of mortuary ordeal
- Angry students

Our Research in Wireless Sensor Networks (WSN)

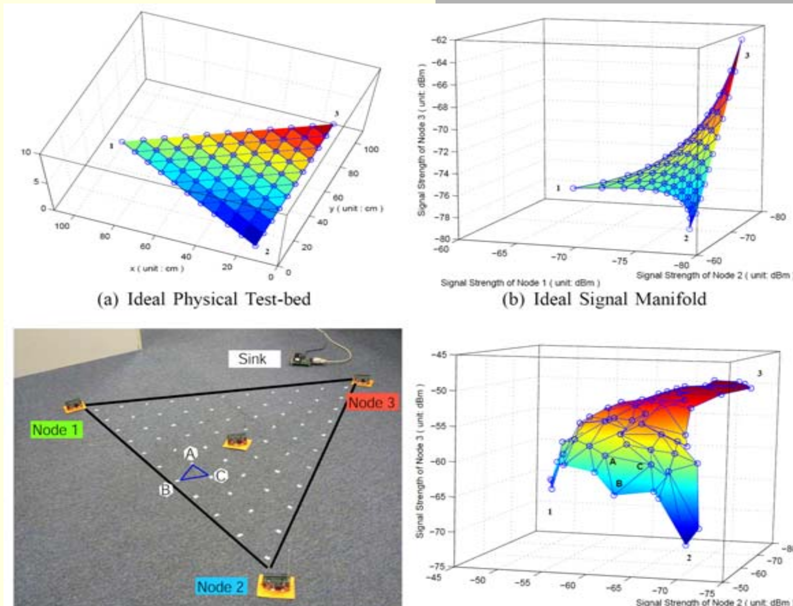
- Objective: develop new algorithms for learning user activities
  - To help users conduct planning
  - To recognize users' locations and actions
  - To detect abnormal user behavior
- Questions:
  - Can existing inductive learning framework help?
  - If not, can we develop new learning algorithms?



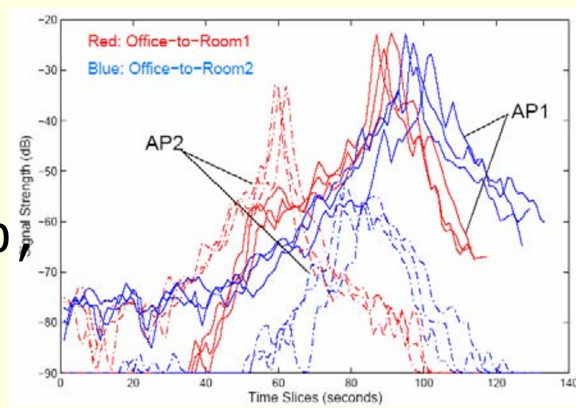
Three Learning Issues of AR in WSN

(AR = Activity Recognition; WSN = Wireless Sensor Net)

- From Action Sequences to Action Models
  - Learning to attach preconditions and post-condition to action sequences
  - Learning probabilistic and factored models of actions
- From Location and Signal Streams to
  - Learning Dynamic Models to Segment Actions
  - Learning Dynamic Bayesian Networks for Goals
- From Wireless Signals to Locations
  - Indoor Model-based Location Estimation
  - Learning Problems:
    - Manifold Learning
    - Semi-supervised Learning
    - Feature Selection and Active Learning
    - Conditional Random Fields Learning
    - Transfer Learning



(a) Ideal Physical Test-bed (b) Ideal Signal Manifold
(c) Experimental Physical Test-bed (d) Experimental Signal Manifold

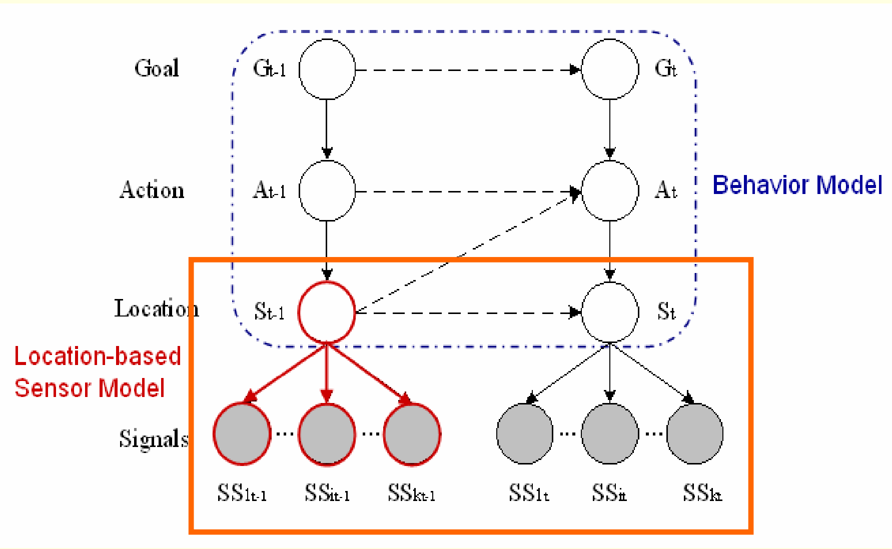


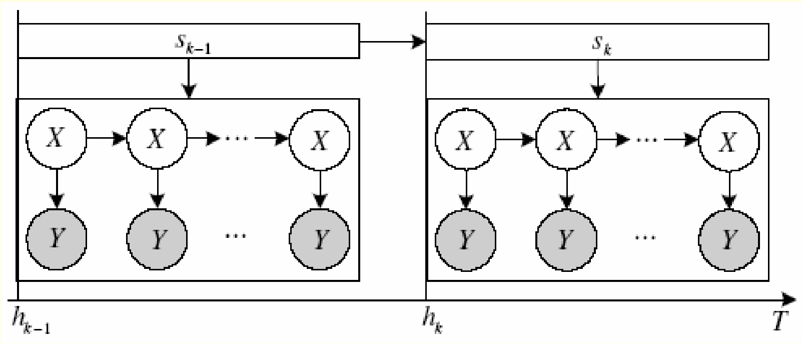
IJCAI05, IJCAI07, AAAI06,Percom05a,b; IEEE TKDE 06a, b; IEEE TMC 07]

Three Learning Issues of AR in WSN

(AR = Activity Recognition; WSN = Wireless Sensor Net)

- From Action Sequences to Action Models
  - Learning to attach preconditions and post-condition to action sequences
  - Learning probabilistic and factored models of actions
- From Location and Signal Streams to
  - Learning Dynamic Models to Segment Actions
  - Learning Dynamic Bayesian Networks for Goals
- From Wireless Signals to Locations
  - Indoor Model-based Location Estimation
  - Learning Problems:
    - Manifold Learning
    - Semi-supervised Learning
    - Feature Selection and Active Learning
    - Conditional Random Fields Learning
    - Transfer Learning



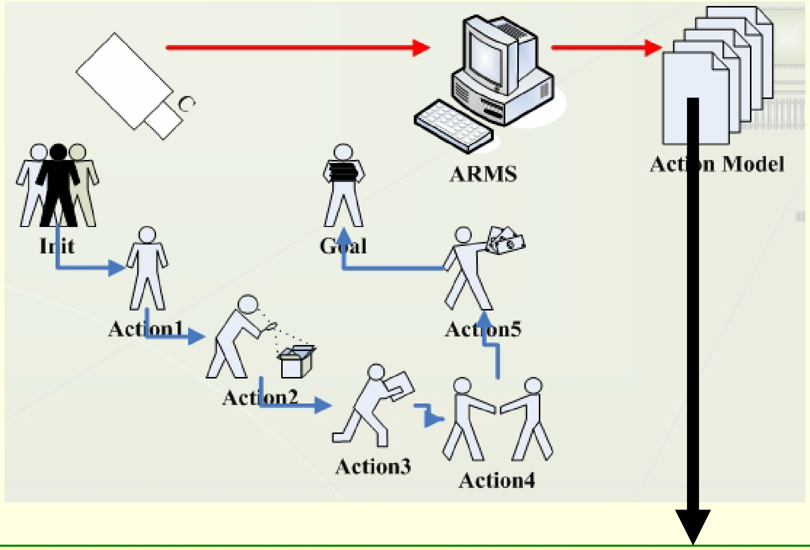


[AAAI04, AAAI05a,b, ICDM05, AIJ 06(c)]

Three Learning Issues of AR in WSN

(AR = Activity Recognition; WSN = Wireless Sensor Net)

- From Action Sequences to Action Models
  - Learning to attach preconditions and post-condition to action sequences
  - Learning probabilistic and factored models of actions
- From Location and Signal Streams to
  - Learning Dynamic Models to Segment Actions
  - Learning Dynamic Bayesian Networks for Goals
- From Wireless Signals to Locations
  - Indoor Model-based Location Estimation
  - Learning Problems:
    - Manifold Learning
    - Semi-supervised Learning
    - Feature Selection and Active Learning
    - Conditional Random Fields Learning
    - Transfer Learning



Load-truck *(?x - hoist ?y - box ?z - truck ?p - place)*

pre: (at x p), (at z p), (lifting x y)

del: (lifting x y)

add: (at y p), (in y z), (available x), (clear y)

[ICAPS 05, AIJ 06]

This Talk

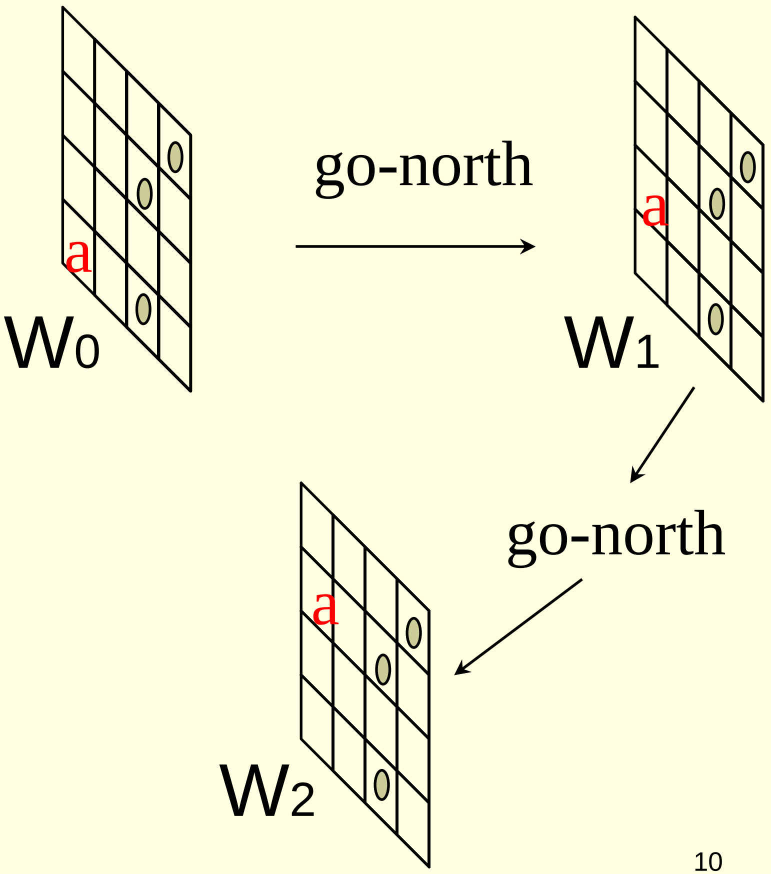
- Learn action models for planning
  - Statistical relational learning
- Learn user actions from sensor signals, and detect abnormal behavior
  - Outlier Detection: Abnormal User-Behavior Detection (if have time)

- Difference from Reinforcement Learning
  - Reinforcement learning acquires relations between actions
    - Pr(S1|S2,A3)
- We try to learn factored action models
  - More compact than RL
  - Can work with RL (later)

background on AI planning

- Before 1995
  - Forward and Backward Search based
  - Traditional Planning
    - Propositional world descriptions
    - Precondition/Post-conditions given for each action
      - Output: A sequence of actions
      - Performance: CPU Time – slow (5-6 actions in several minutes)
  - Probabilistic Planning
    - State transitions using Markov models
      - Markov decision processes (MDP)
      - Partially observable MDPs
      - Output: a policy (state-action-outcome probabilities)
      - Inefficient and impractical still

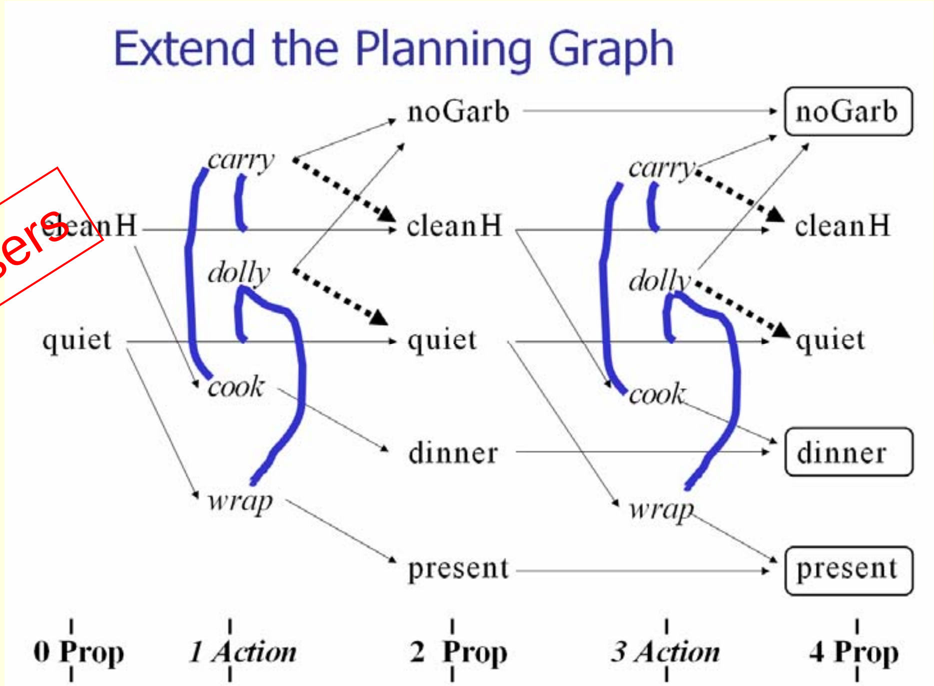
Action Descriptions are provided by Users



Background on AI Planning

- After 1995
  - Planning based on Operations Research Methods
    - Graph Optimization Techniques + Search Techniques
    - Much Faster: > 100 actions in a plan in less than one second on a PC

Actions still need to be provided by Users




學術搜尋

所有結果
A Blum
M Furst
J Langford

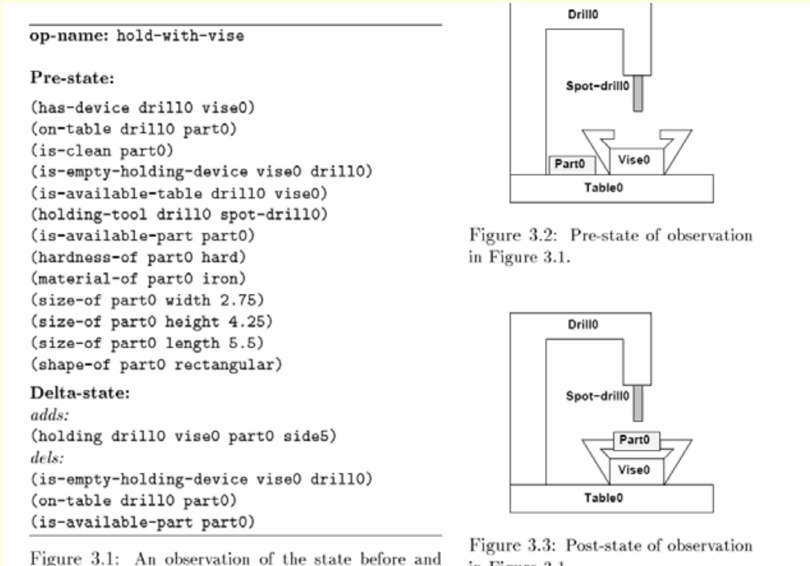
Fast Planning Through Planning Graph Analysis - 10 個群組 »
A Blum, ML Furst - Artificial Intelligence, 1997 - cs.cmu.edu
Abstract We introduce a new approach to planning in STRIPS-like domains bas
structure we call a Planning Graph. We describe a new planner, Graphplan, tha
returns a shortest- possible partial-order plan, or ...
被引用 851 次 - 相關文章 - HTML 版 - 網頁搜尋 - 圖書館搜尋

Learning in AI Planning

- Before 2000
  - Main effort in learning search control
  - Explanation based learning
    - Extend knowledge of search tree to learn concept of success and failure in search
    - Not inductive learning
  - Learning Actions:
    - Almost none
- After 2000
  - Search speed is not considered a problem anymore
    - Thus, search control learning → nil
  - Learning MDP Policies and POMDP Policies from example plans
    - Emphasize learning of relations between actions and states, but not on learning "factored action models"
- Observation
  - No learning on action models up until now
  - Why?

Related Work on Action Learning

- Inductive Logic Programming (e.g. Sablon & Bruynooghe 1994)
  - But cannot learn when intermediate states are not known in a long plan trace (~50 steps)
- Learning by observation and practice: An incremental approach for planning operator acquisition (Wang, X. 1995)
  - But requires full states to be observed
- Knowledge Editors, GIPO (McCluskey, Liu, & Simpson 2003)
  - But requires a human editor


op-name: hold-with-vise

Pre-state:

(has-device drill0 vise0)
(on-table drill0 part0)
(is-clean part0)
(is-empty-holding-device vise0 drill0)
(is-available-table drill0 vise0)
(holding-tool drill0 spot-drill0)
(is-available-part part0)
(hardness-of part0 hard)
(material-of part0 iron)
(size-of part0 width 2.75)
(size-of part0 height 4.25)
(size-of part0 length 5.5)
(shape-of part0 rectangular)

Delta-state:
adds:
(holding drill0 vise0 part0 side5)
dels:
(is-empty-holding-device vise0 drill0)
(on-table drill0 part0)
(is-available-part part0)

Figure 3.1: An observation of the state before and

Figure 3.2: Pre-state of observation in Figure 3.1.

Figure 3.3: Post-state of observation

(X. Wang thesis CMU, 1996)

Learning Actions' Effects and Preconditions [Wang, 1996, CMU]

- When states before and after actions are known in training
  - Use a *covering based algorithm*
  - *Each action A corresponds to a rule:*
    - *If fact1 and fact2 and .. Then effect1 and effect2..*
    - *…*
- Training Data:
  - pre-states = a set of facts before the action *A*
  - *post-states* = a set of facts after the action *A*
- *Data = {<pre-state, post-state, class = A, or not A>, i=1, 2, …}*
- *Note:*
  - ***class=A** means A can be executed in pre-state;*
  - ***class= not A** means A cannot be executed*

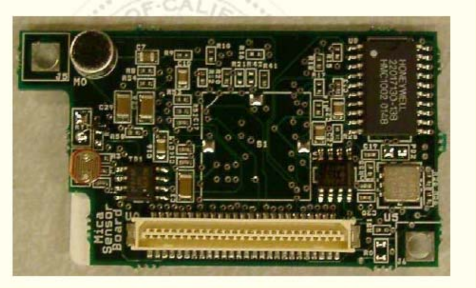
Learning Algorithm

- Initialize Precondition(A) to be first positive data(i)'s pre-state
- Initialize Effect(A) to be the first positive data(i)'s post-state
- For each data(i) in rest(Data), do
  - If class(data(i)=A) then
    - Remove unnecessary preconditions in A, and
    - Add new effects
  - If class(data(i)) is not A, then
    - Label unmet preconditions
    - Add more preconditions
  - …
- Comments:
  - Local search algorithm, Ad Hoc in nature
  - No inductive learning
  - No concept of testing

Relating Sensors to Actions: ready!

- RFID (radio frequency identification) is a technology that incorporates the use of electromagnetic or electrostatic coupling in the radio frequency (RF) portion of the electromagnetic spectrum to uniquely identify an object, animal, or person.
- An RFID system consists of three components: an antenna and transceiver (often combined into one reader) and a transponder (the tag).
- Range: 6 to 90 feet

- Wireless LAN
  - Everywhere today, coverage can be citywide
  - Indoor alternative to GPS
- Motes



• Jonathan Lester et al., "A Hybrid Discriminative-Generative Approach for Modeling Activities ," *IJCAI 2005*, Jul. 30, 2005

With WSN, data are becoming available (2000 hours of data; cf: Hai Leong Chieu1, et al. 2005)

7:30am armband goes on after shower
7:45am subject gets into car
7:48am timestamp for start of driving
8:15am timestamp for end of driving
…

6:24pm subject timestamps start of exercise_stationary_bike
6:44pm subject timestamps end of exercise_stationary_bike
6:50pm subject timestamps beginning of general_exercise (tae kwon do class, which isn't a possible annotation)
…

7:10pm subject timestamps end of general_exercise
9:25pm subject timestamps beginning of watching_tv
11:30pm subject timestamps end of watching_tv
12:01am subject timestamps beginning of lying_down
7:30am subject timestamps end of lying_down and removes armband

Observations

- Research on learning action models is few in the past
- Why
  - Data have not been available
    - With the WSN, the situation is changing
  - Learning actions from observations involves a new type of learning
    - Different from pure inductive learning
      - because data are not in normalized format
    - Nature of learning:
      - Sequences
      - Relational
      - One-class
      - Probabilistic
      - MDL: regularization is important because we try to minimize the model

- Using sensors in WSN,
  - Actions and parameters are given
  - Initial and goal facts are given
  - Sequences of actions are given, but
    - The intermediate states are only partially known
    - Lots of noise and uncertainty
  - to learn:
    - Action models!
  - Question:
    - What type of learning?

Our Main Idea: Action Model Learning [ICAPS 2006; AIJ, 2006, w/ KH Wu and Y. Jiang]

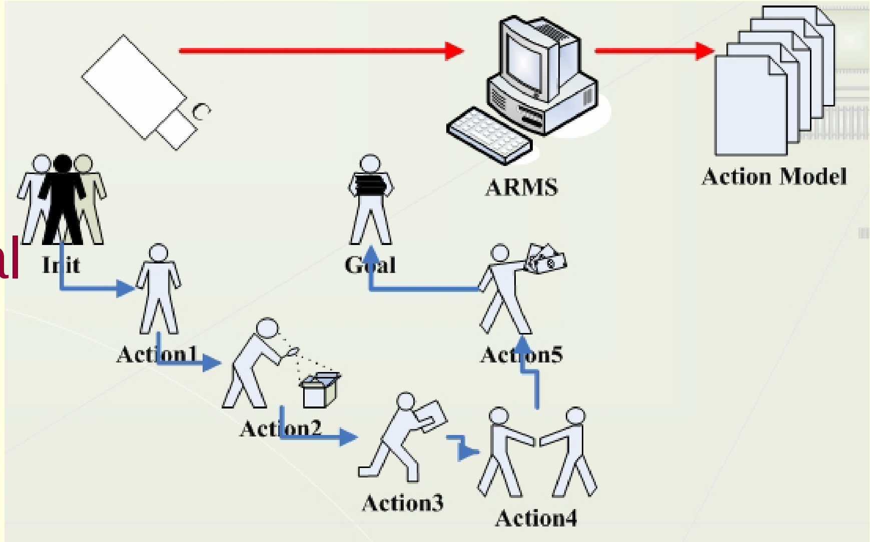
- Input: observed plans
  - $init_1, a_{11}, a_{12}, a_{13}, \ldots, a_{1n}, goal_1$
  - $init_2, a_{21}, a_{22}, a_{23}, \ldots, a_{2m}, goal_2$
  - ...
- Output: action models; e.g.

  load (x - hoist y - crate z - truck p - place)
  - pre: (at x p), (at z p), (lifting x y)
  - del: (lifting x y)
  - add: (at y p), (in y z),
    - (available x), (clear y)
- Main Issue:
  - Automatically guess an initial action model

    Then allow humans to edit these models

<u>Key contribution</u>:

- can learn action models even when no intermediate state observations are available



Action-Relation Modeling System (ARMS)

- Learning action models from observed plans with incomplete knowledge (0% →100%)
  - With or without intermediate states
  - Example plan traces given as input ➔ positive examples
  - Generates an **approximately correct and near-minimal** logical action model
    - STRIPS (this work)
    - ADL and PDDL (future work)

- Methodology:
  - Step 1: Building an action model from observed plans with ARMS
    - Build a clausal form➔ SAT formula, and
    - Solve it using a Weighted MAXSAT solver
      - A kind of one-class relational learning
  - Step 2: Hand editing to make the model more correct

Basic Idea of ARMS System

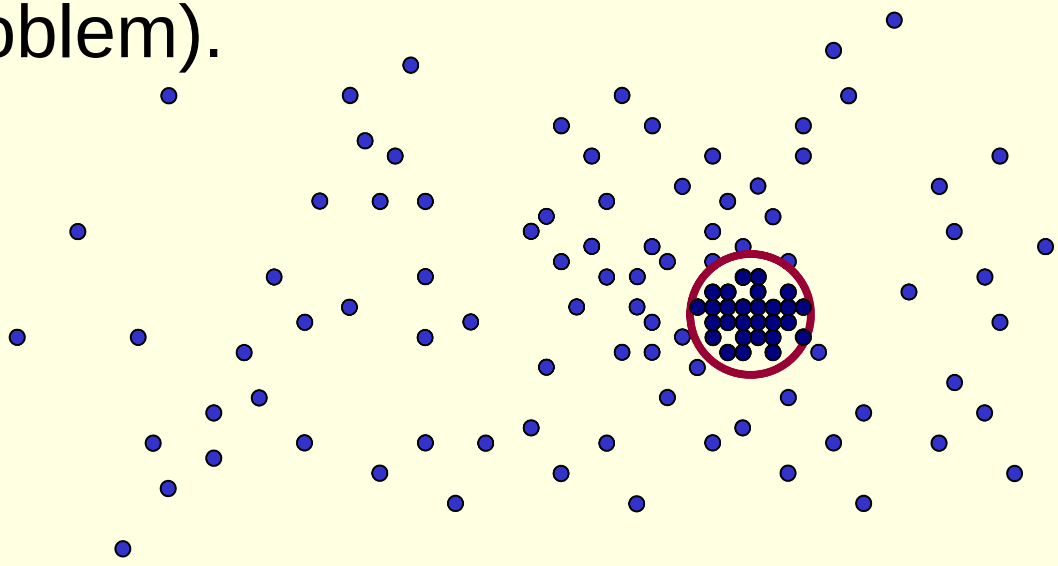
- Observed plans = constraints on actions
- Relations to be learned
  - Whether a relation should be in precondition of A, or effect of A, or not
- Constraints on relations can be integrated into a global optimization formula
  - Maximum Satisfiability Problem
  - One-class Relational Learning
- Testing
  - Correctness
  - Conciseness

- Constraints
  - Preconditions and effects must share parameters
  - Non-empty preconditions and effects
  - If (a1, a2, …an) is frequently co-occurring,
    - Each ai gives something for later actions
      - …

The one-class learning problem:
(ref. Gal Chechik, Stanford)

Find a subset of similar/typical samples

Formally: find a ball of a given radius (with some metric) that covers as many data points as possible (related to the set covering problem).



The MDL principle

- MDL stands for *minimum description length*
- The description length is defined as:

  space required to describe a theory (action model size)

  \+

  the theory's mistakes (constraint violations)
- In our case the theory is the classifier and the mistakes are the errors on the training data
- Aim: we want a classifier with minimal DL
- MDL principle is a *model selection criterion*

Input Data (Plans w/ action names)

domain	Depot
types	place locatable - object depot distributor - place truck hoist surface - locatable pallet crate - surface
predicates	(at x - locatable y - place) (on x - crate y - surface) (in x - crate y - truck) (lifting x - hoist y - crate) (available x - hoist) (clear x - surface)
actions	drive (x - truck y - place z - place) lift(x - hoist y - crate z - surface p - place) drop(x - hoist y - crate z - surface p - place) load(x - hoist y - crate z - truck p - place) unload(x - hoist y - crate z - truck p - place)

I_1 : (at p0 dp0), (clear p0), (available h0), (at h0 dp0), (at t0 dp0), (at p1 ds0), (clear c0), (on c0 p1), (available h1), (at h1 ds0)
I_2 : (at p0 dp0), (clear p0), (available h0), (at h0 dp0), (at t0 ds0), (at p1 ds0), (clear c1), (on c1 c0), (on c0 p1), (available h1), (at h1 ds0)
I_3 : (at p0 dp0), (clear p0), (available h0), (at h0 dp0), (at p1 dp1), (clear p1), (available h1), (at h1 dp1), (at p2 ds0), (clear c1), (on c1 c0), (on c0 p2), (available h2), (at h2 ds0), (at t0 ds0), (at t1 ds0)

Table 2: Three plan examples

	Plan1	Plan2	Plan3
Initial state	I_1	I_2	I_3
Step1	(lift h1 c0 p1 ds0), (drive t0 dp0 ds0)	(lift h1 c1 c0 ds0)	(lift h2 c1 c0 ds0)
Step2	(load h1 c0 t0 ds0)	(load h1 c1 t0 ds0)	(load h2 c1 t1 ds0)
Step3	(drive t0 ds0 dp0)	(lift h1 c0 p1 ds0)	(lift h2 c0 p2 ds0), (drive t1 ds0 dp1)
Step4	(unload h0 c0 t0 dp0)	(load h1 c0 t0 ds0)	(unload h1 c1 t1 dp1), (load h2 c0 t0 ds0)
Step5	(drop h0 c0 p0 dp0)	(drive t0 ds0 dp0)	(drop h1 c1 p1 dp1), (drive t0 ds0 dp0)
Step6		(unload h0 c1 t0 dp0)	(unload h0 c0 t0 dp0)
Step7		(drop h0 c1 p0 dp0)	(drop h0 c0 p0 dp0)
Step8		(unload h0 c0 t0 dp0)	
Step9		(drop h0 c0 c1 dp0)	
Goal State	(on c0 p0)	(on c1 p0) (on c0 c1)	(on c0 p0) (on c1 p1)

The ARMS Algorithm: Step1

- Step 1. Initialize by lifting and parameter matching
- Step 2. Frequent-set Mining
- Step 3. Weighted MAX-SAT problem
- Step 4. Output

Example

- load (gripper12 crate2 truck2 place3)…

↓

- load (?x ?y ?z ?p)
- …
- Lift(?x, ?y)

The ARMS Algorithm: Step 2

- Step 1. Initialize
- Step 2. Frequent-set Mining
- Step 3. Weighted MAX-SAT problem
- Step 4. Output

- $init_1, a_{11}, a_{12}, \ldots, a_{1(n-1)}, a_{1n}, goal_1$
- $init_2, a_{21}, a_{22}, \ldots, a_{2(n-1)}, a_{2n}, goal_2$
- …
- $init_n, a_{n1}, a_{n2}, \ldots, a_{n(n-1)}, a_{nn}, goal_n$

$\Downarrow$

- pre_i, a_i (support=60%, conf=..)
- a_i, a_j, (85%)
- a_i, a_j, a_k (90%)

The ARMS Algorithm: Step 3

- Step 1. Initialize
- Step 2. Frequent-set Mining
- Step 3. Encode and Solve a weighted MAX-SAT problem
- Step 4. Output

Input: clauses

…

- 1 17 0
- 1 18 0
- 795 -9 3 0
- 795 -10 4 0
- 795 1 2 3 4 0
- 795 5 6 7 8 0

Output: a SAT solution

- 1 1
- 2 0
- 3 1
- …

The ARMS Algorithm: Step 4

- Step 1. Initialize
- Step 2. Frequent-set Mining
- Step 3. Weighted MAX-SAT problem
- Step 4. Output,
  Test by X-validation
  Human Edit..

- action models:
- *Load* (?x - hoist ?y - crate ?z - truck ?p - place)
  - pre: (at ?x ?p), (at ?z ?p), (lifting ?x ?y)
  - add: (in ?y ?z), (available ?x)
  - del: (lifting ?x ?y)
- Unload …

Key Component: Weighted MAXSAT formulation

- Explain why an action a1 often appears before a2?
  - Perhaps because a1 produces a proposition that a2 consumes, or
  - a1 and a2 both require the same precondition, or
  - a2 produces a proposition that a1 deletes
  - Or
  - …
- Explain general requirements for a1’s preconditions, adds and deletes
  - Preconditions and adds don’t intersect
  - Effects are subsets of Preconditions
  - …

Imposing Constraints to find a plausible action model

- 1. Proposition Constraints
  - For a first or last action ***A*** in a plan, an initial or goal proposition ***P,***
  - ***(P,A)*** is a candidate of ***Pre(A)*** if frequency of ***(P,A)*** is greater than a [theta] value.

Table 3: Examples of All Supported Predicate Constraints

Label	Predicate Constraints	Support
{y}	(clear y) $\in pre_{lift}$	3
{x, p}	(at x p) $\in pre_{lift}$	3
{x }	(available x) $\in pre_{lift}$	3
{z, p}	(at z p) $\in pre_{lift}$	3
{y, p}	(at y p) $\in pre_{lift}$	3
{y, z}	(on y z) $\in pre_{lift}$	3
{x, y}	(at x y) $\in pre_{drive}$	1
{y, z}	(on y z) $\in add_{drop}$	3

2. Action Constraints
Precondition and Add-list :

- Intersection between add-list and precondition must be empty

$$pre_i \cap add_i = \phi.$$

- Example
  - $(lifting\ x\ y) \in add_i \Rightarrow (lifting\ x\ y) \notin pre_i$
  - $(at\ y\ p) \in add_i \Rightarrow (at\ y\ p) \notin pre_i$
  - $(in\ y\ z) \in add_i \Rightarrow (in\ y\ z) \notin pre_i$
  - $(clear\ y) \in add_i \Rightarrow (clear\ y) \notin pre_i$
  - $(at\ z\ p) \in add_i \Rightarrow (at\ z\ p) \notin pre_i$
  - $(lifting\ x\ y) \in pre_i \Rightarrow (lifting\ x\ y) \notin add_i$
  - $(at\ y\ p) \in pre_i \Rightarrow (at\ y\ p) \notin add_i$
  - $(in\ y\ z) \in pre_i \Rightarrow (in\ y\ z) \notin add_i$
  - $(clear\ y) \in pre_i \Rightarrow (clear\ y) \notin add_i$
  - $(at\ z\ p) \in pre_i \Rightarrow (at\ z\ p) \notin add_i$

2. Action Constraints
Precondition and Del-list

- Deletes are subsets of preconditions

$$del_i \subseteq pre_i.$$

- Example
  - $(lifting\ x\ y) \in del_i \Rightarrow (lifting\ x\ y) \in pre_i$
  - $(at\ y\ p) \in del_i \Rightarrow (at\ y\ p) \in pre_i$
  - $(in\ y\ z) \in del_i \Rightarrow (in\ y\ z) \in pre_i$
  - $(clear\ y) \in del_i \Rightarrow (clear\ y) \in pre_i$
  - $(at\ z\ p) \in del_i \Rightarrow (at\ z\ p) \in pre_i$

3. Plan Constraints

- Explain why two or more frequent nearby actions a_i and a_j in plans
  - Either they use the same precondition, or a_i adds p for a_j, or a_i deletes p, but a_j adds p again.

$$\exists p.(p \in (pre_i \cap pre_j) \;\wedge p1 \;\notin (del_i)) \vee \;(p \in (add_i \cap pre_j)) \vee \;(p \in (del_i \cap add_j))$$

- For example, *suppose that*
- *((lift x - hoist y - crate z – surface p - place),*
- *(load x - hoist y - crate z - truck p - place), 0)*
- *Is a frequent pair.*
- *The relevant parameters: x-x, y-y, p-p.*
- *Thus, these two actions are possibly connected by predicates (at x p), (available x), (lifting x y), (at y p), and (clear y)…*
- We can then formulate the plan constraints as stated…

ARMS Learning Example

- *Plan:*
  - *Initial → (lift obj truck place) → (load obj truck place) → Goal*
  - *Initial = {at(obj, place), at(truck, place)}*
  - *Goal={In(obj, truck)}*
- *Other propositions: { lifting(obj, truck)}*
- *Creating Clauses:*
  - *Proposition Constraint:*
    - *{ (at(obj,place)* ∈ PRE(lift) or at(truck,place) ∈ PRE(lift)},…
  - *Action Constraint*
    - *(at(obj,place)* ∈ PRE(lift) ➔ *(at(obj,place)* ∈ ADD (lift)
    - etc

Example: continued…

- *Creating Clauses:*
  - *From Plan Constraint:*

{lifting(obj, truck) $\in$ ADD(*(lift obj truck place)) and lifting(obj, truck)* $\in$ PRE*(load obj truck place)}*

or

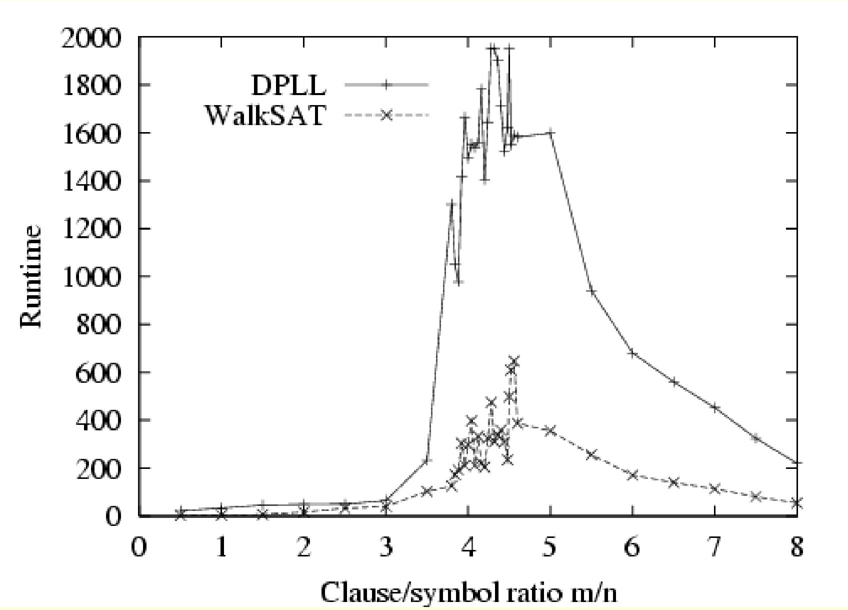
{at(truck, place) $\in$ ADD(*(lift obj truck place)) and at(truck, place)* $\in$ PRE*(load obj truck place)} or <etc>*

Final Clausal Form:

- 1: *lifting(obj, truck)* ∈PRE(*lift)*
- 2: *at(obj, place)* ∈PRE(*lift)*
- 3: *at(truck, place)* ∈PRE(*lift)*
- 4: *in(obj, truck)* ∈PRE(*lift)*
- 5: *lifting(obj, truck)* ∈ADD(*lift)*
- 6: *at(obj, place)* ∈ADD(*lift)*
- 7: *at(truck, place)* ∈ADD(*lift)*
- 8: *in(obj, truck)* ∈ADD(*lift)*
- 9: *lifting(obj, truck)* ∈DEL(*lift)*
- 10: *at(obj, place)* ∈DEL(*lift)*
- 11: *at(truck, place)* ∈DEL(*lift)*
- 12: *in(obj, truck)* ∈DEL(*lift)*
- 13: lifting(obj, truck) ∈PRE(load)
- 14: at(obj, place) ∈PRE(load)
- 15: at(truck, place) ∈PRE(load)
- 16: in(obj, truck) ∈PRE(load)
- 17: lifting(obj, truck) ∈ADD(load)
- 18: at(obj, place) ∈ADD(load)
- 19: at(truck, place) ∈ADD(load)
- 20: in(obj, truck) ∈ADD(load)
- 21: lifting(obj, truck) ∈DEL(load)
- 22: at(obj, place) ∈DEL(load)
- 23: at(truck, place) ∈DEL(load)
- 24: in(obj, truck) ∈DEL(load)

Final Clausal Form: A Satisfiability Formula

- Convert to 3-CNF sentences. e.g.,
  $(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$
  m = number of clauses
  n = number of symbols
  - Hard problems seem to cluster near *m/n* = 4.3 (critical point)



- *… can have up to 7000 clauses*

Weighted MAX-SAT

- SAT Solvers:
  - find an assignment of true values to variables that can satisfy a collection of clauses.
- Weighted MAX SAT Solvers:
  - Assign a weight to each clause and seeks an assignment that maximizes the sum of the weights of the satisfied clauses
  - In ARMS, weights = probability of appearing in the plan examples

- Weighted MAX Satisfiability Problem:
  - Given a collection C of m clauses, C1, . . .Cm involving n logical variables, with clause weights Wi, find a truth assignment that maximizes the total weight of the satisfied clauses in C.

Brian Borchers and Judith Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization*, 1999.
Henry Kautz, Bart Selman, and Yueyen Jiang. A general stochastic approach to solving problems with hard and soft constraints. *The Satisfiability Problem: Theory and Applications*, 35, 1997.

WalkSAT and MaxWalkSAT

The WalkSAT Algorithm

```
for i ← 1 to max-tries do
   solution = random truth assignment
   for j ← 1 to max-flips do
      if all clauses satisfied then
         return solution
      c ← random unsatisfied clause
      with probability p
         flip a random variable in c
      else
         flip variable in c that maximizes number of satisfied clauses
return failure
```

The MaxWalkSAT Algorithm

```
for i ← 1 to max-tries do
   solution = random truth assignment
   for j ← 1 to max-flips do
      if Σ weights(sat. clauses) > threshold then
         return solution
      c ← random unsatisfied clause
      with probability p
         flip a random variable in c
      else
         flip variable in c that mximizes
            Σ weights(sat. clauses)
return failure, best solution found
```

Source: Domingos, 2006

Cross Validation in ARMS

- Correctness
  - A plan is said to be **correct** if (1) all actions‘ preconditions hold in the state just before that action and (2) all goal propositions hold after the last action.
  - Error rate
- Redundancy
  - A **non-redundant** action model is when every effect of an action is useful later in a plan
  - Redundancy rate
- Demo

Experimental Metrics

$$\text{Error Rate} = \frac{\text{Number of all unsatisfied preconditions}}{\text{Number of all preconditions}}$$

$$\text{Redundancy Rate} = \frac{\text{Number of all unuseful predicates in add list}}{\text{Number of all predicates in add list}}$$

- Number of Clauses: how simple the ARMS encoding is
- CPU Time: Efficiency of learning

- The planning domains in International Planning Competition 2002
- Planner: MIPS, used to generate example plan traces
- Five-fold cross-validation
  - Training: 160 plans
  - Test: 40 plans
  - Repeat five times, take the average
- Plan lengths: average length=50 actions

Error, redundancy, CPU, Clauses vs. Theta

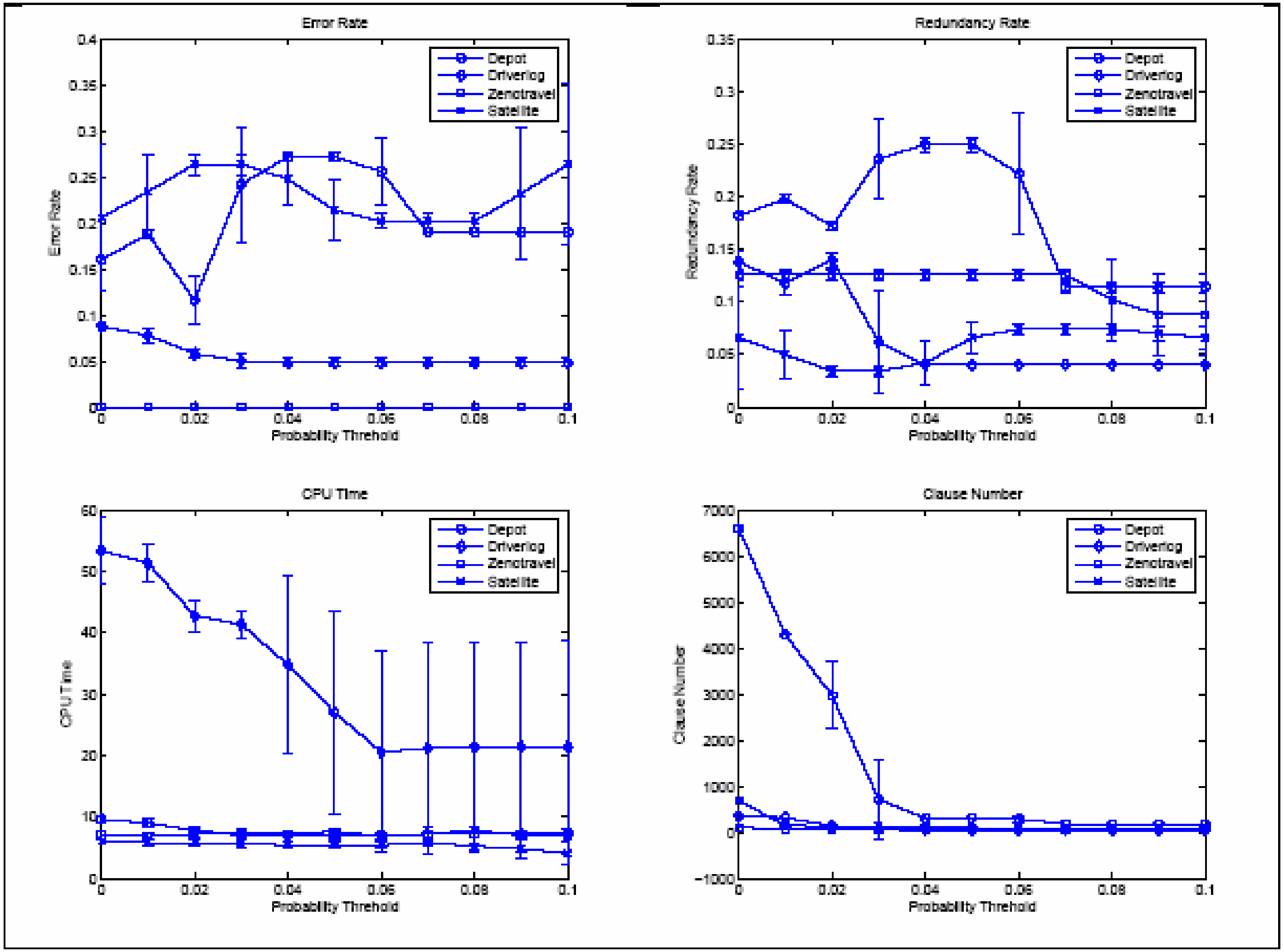


Fig. 1. Varying the probability threshold in four planning domains.

Error, redundancy, CPU, Clauses vs. # Plans in Training Data

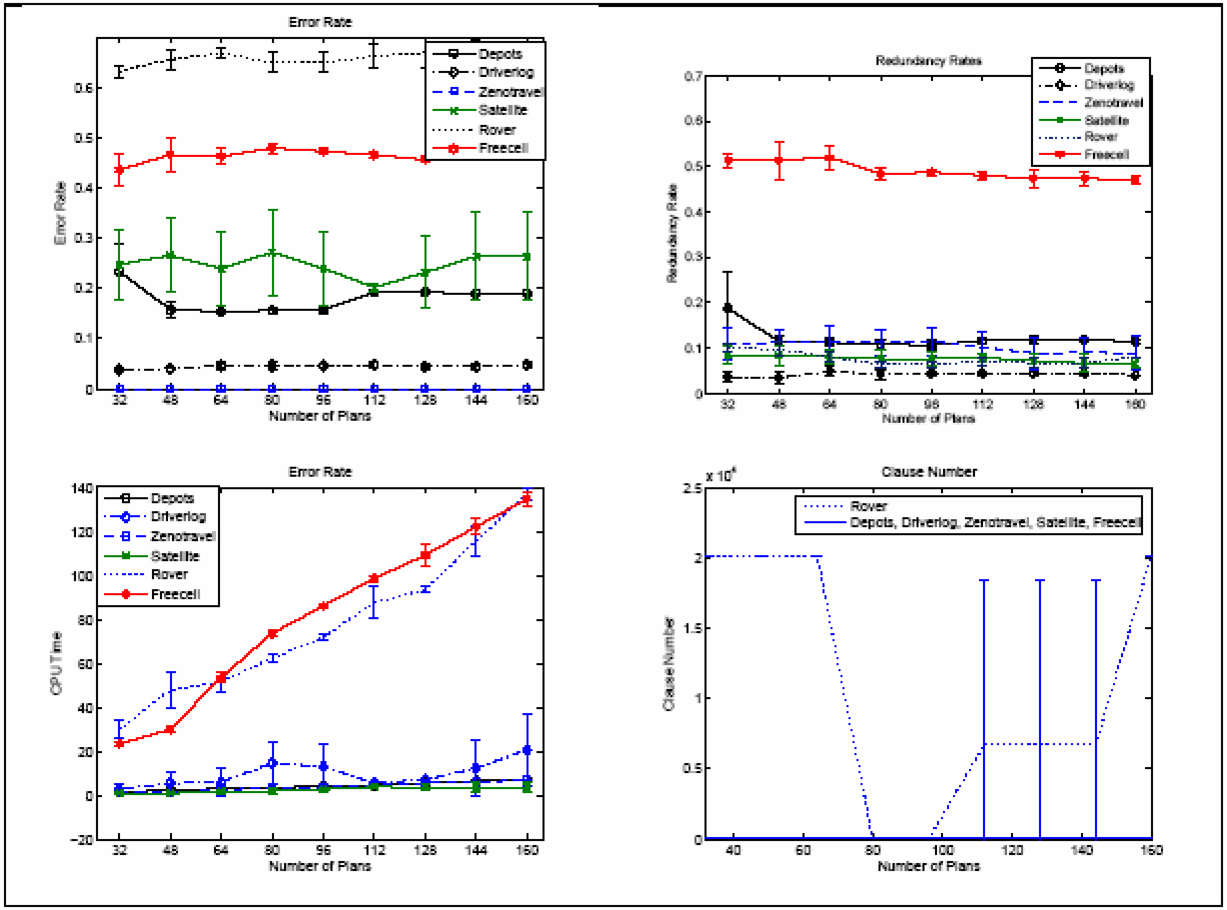


Fig. 3. Varying the number of plans

The Learned Action Model in Depot Domain

Table 5: The Learned Action Model(Depots Domain, $\theta = 80\%$)

ACTION	drive (x - truck y - place z - place)
PRE:	(at x y)
ADD:	(at x z)
DEL:	(at x y)
ACTION	lift(x - hoist y - crate z - surface p - place)
PRE:	(at x p),(available x),(at y p),(on y z), (clear y),**(at z p)**
ADD:	(lifting x y),(clear z)
DEL:	(at y p),(clear y),(available x),(on y z)
ACTION	drop(x - hoist y - crate z - surface p - place)
PRE:	(at x p),(at z p),(clear z),(lifting x y)
ADD:	(available x),(clear y),(on y z)
DEL:	(lifting x y),(clear z)
ACTION	load(x - hoist y - crate z - truck p - place)
PRE:	(at x p),*(at z p)*,(lifting x y)
ADD:	(in y z),(available x),**(at y p), (clear y)**
DEL:	(lifting x y)
ACTION	unload(x - hoist y - crate z - truck p - place)
PRE:	(at x p) (at z p) (available x) (in y z), **(clear y)**
ADD:	(lifting x y)
DEL:	(in y z),(available x),**(clear y)**

Bold: in learned model but not in hand-crafted model

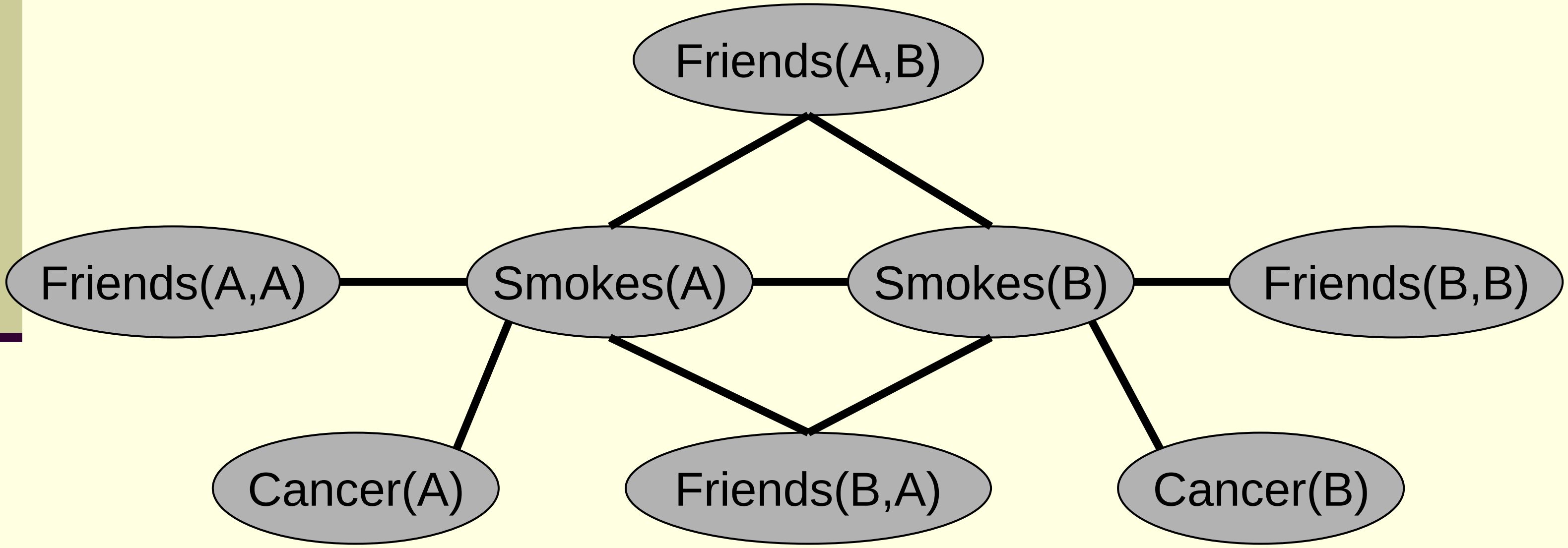
Italic: in hand-crafted model, but not in learned model

Model: can generate plans,
But plans may be different from learning examples

What type of Learning: Markov Logic Networks (ref Domingos, 2006) Smokers Example:

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)



Source: Domingos 2005.

Learning MLNs

- Data is a relational database
- Closed world assumption (if not: EM)
- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning structure

Source: (Domingos 2005)

Relation to Statistical Models

- Special cases:
  - Markov networks
  - Markov random fields
  - Bayesian networks
  - Log-linear models
  - Exponential models
  - Max. entropy models
  - Gibbs distributions
  - Boltzmann machines
  - Logistic regression
  - Hidden Markov models
  - Conditional random fields

- Obtained by making all predicates zero-arity
- Markov logic allows objects to be interdependent (non-i.i.d.)
- Discrete distributions

Source: (Domingos 2005)

Relating MLNs with ARMS

- The ARMS algorithm can be considered as a special case of the MLN algorithm,
  - We invent a new predicate InPrecond such that, for a literal P and action A,
    - InPrecond takes P and A as arguments in the form of InPrecond(P,A) todenote the fact that the literal P is assigned to the precondition of the action A
  - We invent a new predicate InAdd such that, for a literal E and action A, InAdd takes E and A as arguments in the form of InAdd(E,A) to denote the fact that the literal E is assigned to the effects of the action A.
  - Similarly, we define InDelete(E,A) for the delete list items.
  - Constraints: the action constraint (A.1) can be represented as a knowledge-base (KB) formula:
    - A 2 Actions.InPrecond(P,A)) ¬InAdd(P,A) which states that the intersection of preconditions and add list of all actions are empty.

ARMS: What type of learning?

- Statistical relational learning
  - Markov Logic Networks
- Also learning from one-class of data
  - Unlike traditional multi-class classification problems
  - Optimization used to ensure minimal model size and maximum coverage of training data
    - MDL principle

ARMS Learning as MLN:

- Nodes themselves are constructed using the domain specific predicates, variables and constants such as (on ?x,?y).
- Relations between nodes include logical axioms that encode specific constraints in a problem domain.
  - For example, in the blocks world, an axiom states that the predicate (clear ?x)=True precludes that there exists an object ?y, such that (on ?y,?x)=True.
  - These axioms as well as the constraints form the relations between the nodes in the MLN.
- The weights of the *hard* constraints, such as the above example, are set to be the highest. The action, information and plan constraints receive their weights accordingly

References on ARMS

- Qiang Yang, Kangheng Wu and Yunfei Jiang. Learning Action Models from Plan Traces using Weighted MAX-SAT. *Artificial Intelligence Journal* (**AIJ**). Accepted Oct 2006.
- Qiang Yang, Kangheng Wu and Yunfei Jiang, Learning Action Models from Plan Examples with Incomplete Knowledge. *In Proceedings of the 2005 International Conference on Automated Planning and Scheduling,* **(ICAPS 2005)** Monterey, CA USA June 2005. Pages 241-250.
- Co-Champion for the System ARMS (with Kangheng Wu and Yunfei Jiang) for learning planning models, 2005 First International Competition on Knowledge Engineering for Planning and Scheduling, ICAPS 2005, Monterey CA USA
- AAAI 2005 Conference Highlight:
  - Presented as a ICAPS 05 highlight
- Machine Learning Summer School 2006, Australia
  - Highlighted in Rao's Presentation

Next Topic: Abnormal Activity Detection in WSN

- Abnormal activities
  - They occur rarely
  - They are not expected in advance
- Challenges
  - Extremely scarce/no training data for abnormal activities
- Example
  - User Walks around
    - Normal
  - User takes elevator
    - Normal
  - User suddenly falls down
    - Abnormal
  - User runs fast
    - Abnormal

Previous Works

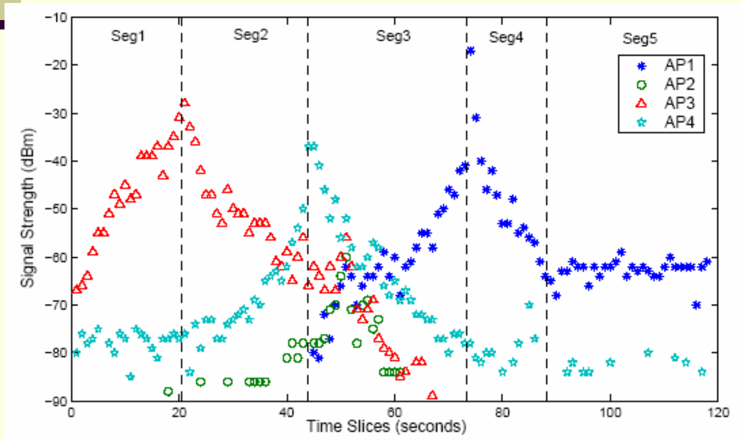
- Y. Yao, F.-Y. Wang, J. Wang, and D. D. Zeng, “Rule + exception strategies for security information analysis.” *IEEE Intelligent Systems*, vol. 20, no. 5, pp. 52–57, 2005.
  - Rule based knowledge base describe normal behavior
  - Exception rules describe abnormality
    - Our approach is probabilistic in nature, more suitable when not much data are available for training exception rules
- J. Lester, et al. “A hybrid discriminative/generative approach for modeling human activities,” in *IJCAI 05*
  - a hybrid discriminative/generative approach to recognizing human activities using an ensemble of static classifiers and HMMs
- P. Lukowicz, J. Ward, H. Junker, M. St¨ager, G. Tr¨oster, A. Atrash, and T. Starner, “Recognizing workshop activity using body worn microphones and accelerometers,” in Pervasive 04.
  - used body-worn microphones accelerometers track users' daily activities
  - Most of these works employ supervised learning to recognize users' normal activities, which requires a large amount of labeled training data.

Related Works

- Computer Vision Area
  - T. Xiang and S. Gong, “Video behaviour profiling and abnormality detection without manual labeling,” in ICCV *2005*
  - T. Duong, H. Bui, D. Phung, and S. Venkatesh, “Activity recognition and abnormality detection with the switching hidden semi-Markov model,” in *CVPR 2005*
    - switching hidden semi-Markov models were applied to represent user’s activities and perform abnormality detection.
    - only focused on detecting a more subtle form of abnormality,
    - Abnormalities only in the state duration, but not in the state order.
- Data Mining Area
  - S. D. Bay and M. Schwabacher, “Mining distance-based outliers in near linear time with randomization and a simple pruning rule,” In KDD 2003
  - [8] A. Lazarevic, L. Ert¨oz, A. Ozgur, J. Srivastava, and V. Kumar, “A comparative study of anomaly detection schemes in network intrusion detection,” in *SDM 2003*.
  - For similarity-based approaches, the main task is to define pair-wise distances between all the data points and identify outliers by examining the distance to an example’s nearest neighbors.
  - But, distances are hard to define

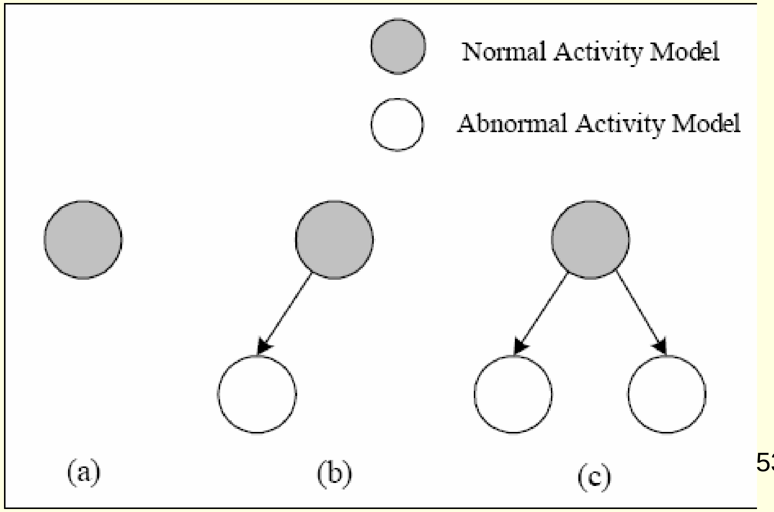
Idea of the approach

- Offline: Given a collection of normal user traces
  - Each being a sequence of signals
  - First filter all traces to build a one-class SVM model
  - Then, construct a normal activity model using the normal traces



- Online: For a given new trace, Use the normal activity model to tell if it is an abnormal trace.

  If so, derive its model from the trace and the normal trace models



Offline: Modeling Normal Activities

- We assume that there is a hidden mechanism to generate the user behavior
- This hidden state can be modeled using Hidden Markov Models
  - We use a set of *m* HMMs with Gaussian observation density to model the normal traces (*m* is fixed ahead of time)
  - The log likelihood of each pair <Trace, HMM>:

$$L(Y_i; \lambda_j) = \log P(Y_i|\lambda_j), 1 \le i \le N, 1 \le j \le M.$$

Offline: Feature vectors and one-class SVM

- for each training trace *Yi*, 1 <= *i* <= *N*, we can obtain an *m*-dimensional feature vector

$$\mathbf{x}_i = \langle L(Y_i; \lambda_1), \ldots, L(Y_i; \lambda_M) \rangle$$

- This feature vector can be used to train a one-class SVM:

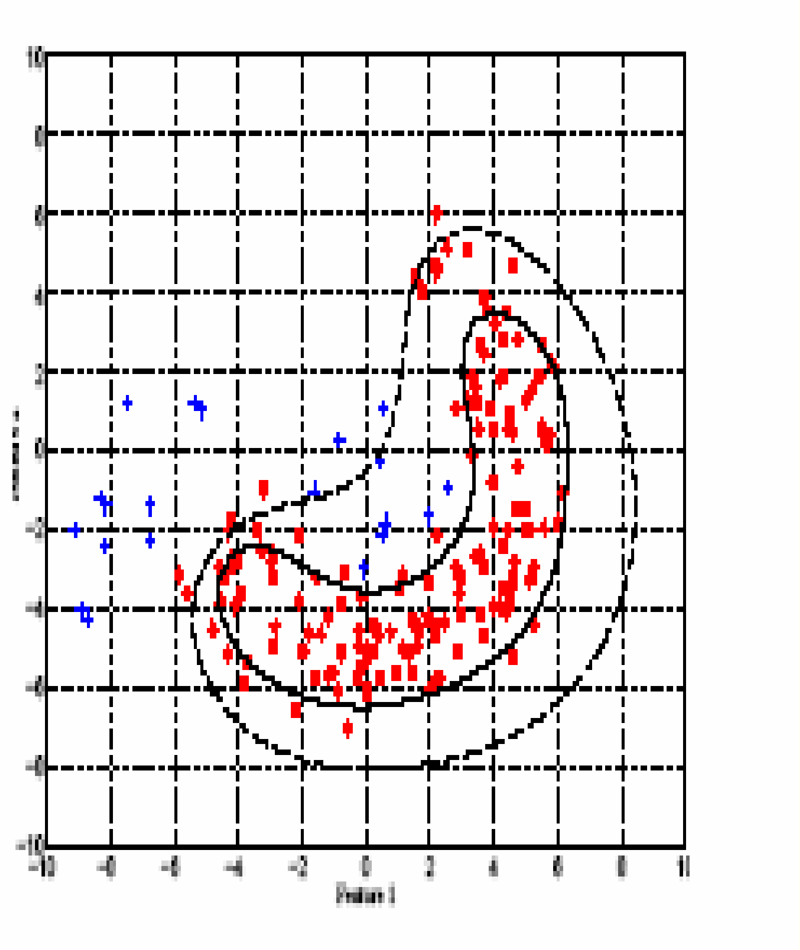
$$\begin{aligned} \min \quad & R^2 + C \sum_{i=1}^{n} \xi_i, \\ s.t. \quad & \|\mathbf{c} - \mathbf{x}_i\|^2 \le R^2 + \xi_i, \\ & \xi_i \ge 0. \end{aligned}$$

- The parameters

Here, the slack variables ξ_i are introduced to allow some data points to lie outside the sphere, and the parameter $C \ge 0$ controls the tradeoff between the volume of the sphere and the number

Offline: Build a Normal HMM

- One-class SVM is very sensitive to the boundary
  - Can generate many false negatives
    - Meaning: abnormal classified as normal
  - Thus, we restrict one-class SVM to bias towards low false negative rates→ normal traces with high confidence
  - We then use these normal traces to train a single normal HMM
- This HMM is used in online phase



Online Phase: classify a new trace

- We first apply the normal activity HMM
  - If Normal, return (0)
  - Else
    - Decide whether to derive an abnormal model
      - Fall down model
      - Running model
      - Crawling model
      - …
  - Issue: how to build an abnormal model based on only one trace?
  - Answer: adapt the normal HMM model

- *Kernel Nonlinear Regression (KNLR) Adaptation:*

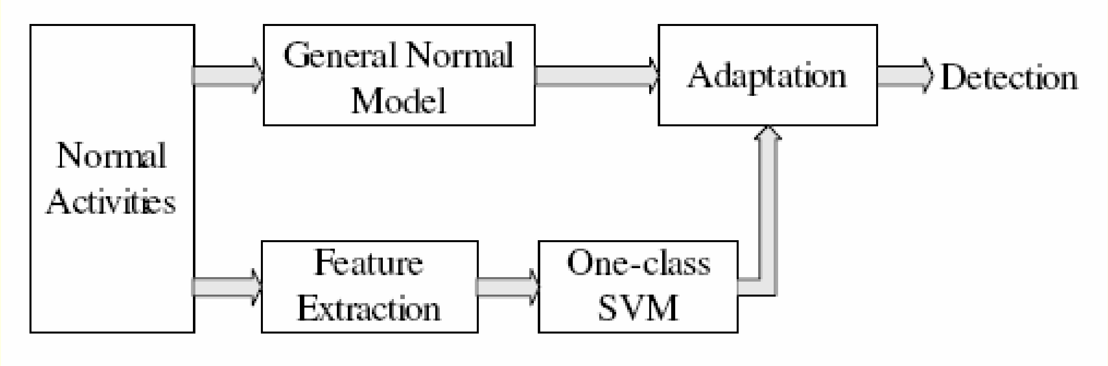
$$\mu_i = \alpha \cdot \mu_i^{old} + (1 - \alpha) \cdot \mu_i^{new},$$

$$\mu_i^* = (\mathbf{BK} + \beta \mathbf{AK}^{-1})(\mathbf{K}^2 + \epsilon \mathbf{I})^{-1}\mathbf{K}.$$

Summary: A Two-phase Solution

- Step 1: build a one-class SVM model based on normal data
- Step 2: derive abnormal activity models from a general normal model via model adaptation [YYP06]

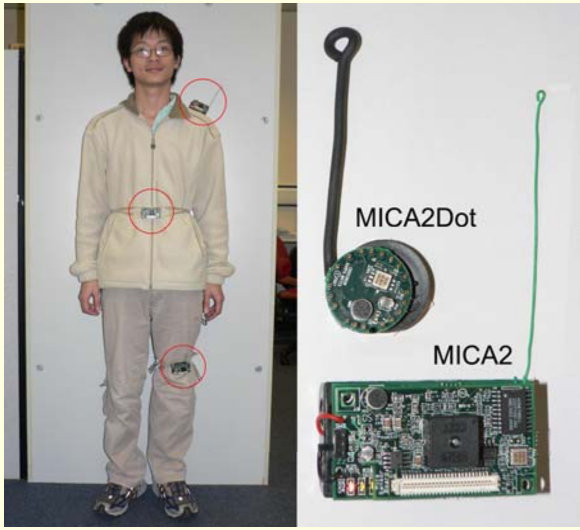
- Model Adaptation
  - Maximum Likelihood Linear Regression (MLLR)
  - Kernel Nonlinear Regression (KNLR)



Y6

Experiments

- Setup:
  - three Crossbow MICA2s (MPR400) with the sensor board MTS310CA
  - “slipping on a wet floor” and “falling down to a floor”
- Data set:
  - 216 normal traces for training
  - 215 normal traces and 112 abnormal traces for testing



Detection Rate

False Alarm Rate

Y6 This figure shows the ROC curve with respect to the detection rate and the false alarm rate. OneSVM gives the poorest performance because it achieves a high detection rate at the cost of a high false alarm rate. By applying model adaptation techniques, SVM+MLLR and SVM+KNLR can improve the performance of OneSVM. However, by using nonlinear transformation techniques, SVM+KNLR can outperform SVM+MLLR.

YJ, 6/9/2006

Experimental Setup

Normal Activities	Abnormal Activities
sitting down	slipping on the ground
walking	falling down backwards
walking downstairs	falling down forwards
walking upstairs	...
running	...

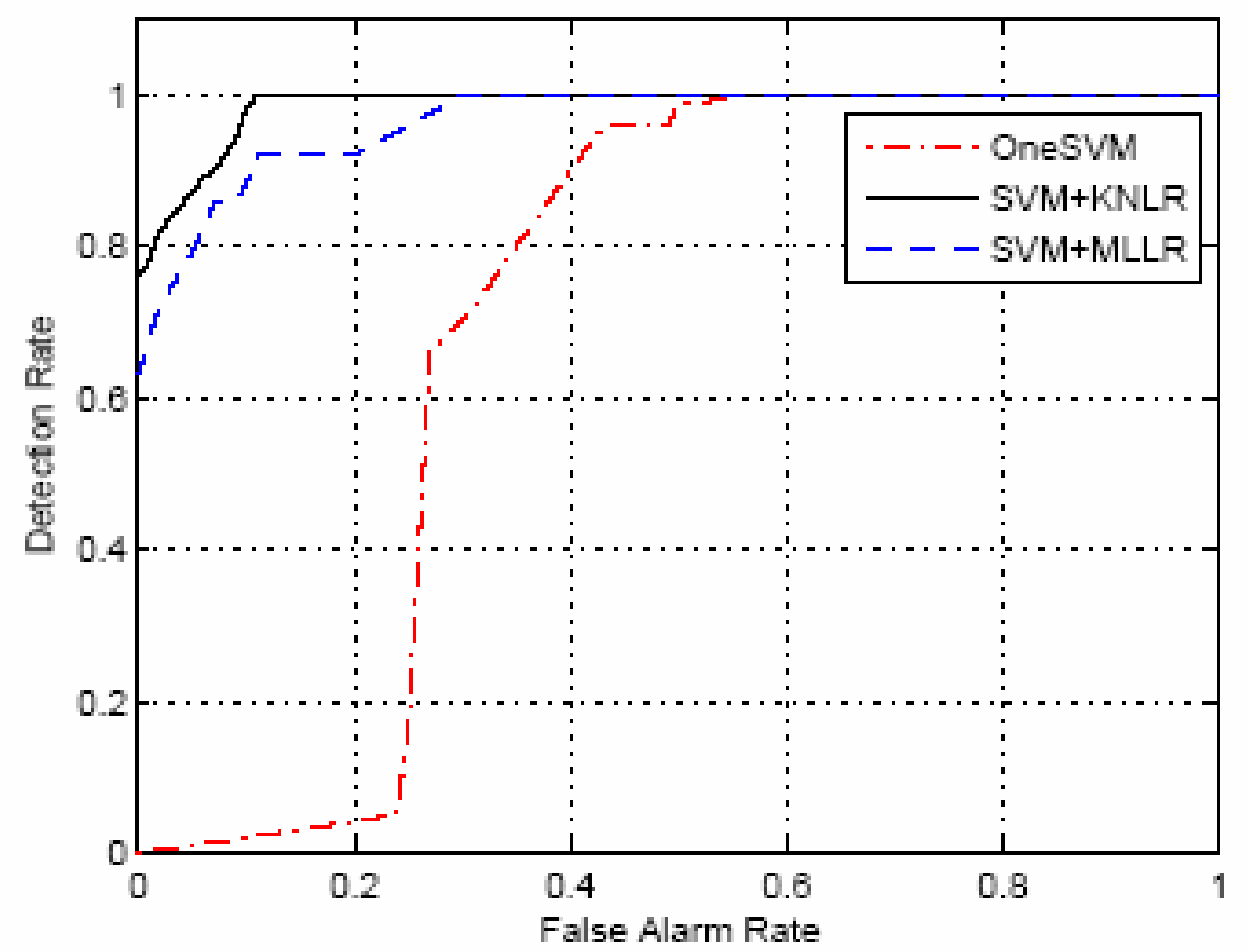
TABLE I

EXAMPLES OF NORMAL AND ABNORMAL ACTIVITIES

$$\text{Detection Rate} = \frac{TN}{TN + FN},$$

$$\text{False Alarm Rate} = \frac{FP}{FP + TP}.$$

Experimental Results



Conclusions

- Action Model Learning
  - Statistical Relational Learning
- Connects low level sensors with high level logics
- Significance
  - Acquires knowledge bases for planning systems
  - Towards realization of the dream of AI
- Abnormal Activity Detection
  - Normal activities
    - Lots of data
  - Abnormal activities
    - Very few data
  - Approach is to build a good reliable model, then adapt that model for abnormal activities

What does it take to do this research?

- Preparation
  - Knowledge of high level reasoning
  - Knowledge of statistical learning
- Impact
  - I believe the impact is huge for future of AI
  - Makes all work in first 30 years of AI research relevant
- Who are doing it?
  - Pedro Domingos (U Washington, Seattle)
  - ***Daphne*** Koller (Stanford)
  - Pad Langley (Stanford)
  - …

Students in this area…

- Familiar with logic and statistics
- Motivated to work in this new frontier
  - Not much work been done yet!

- Contact: Qiang Yang @ http://www.cse.ust.hk/~qyang