




**Learning based Directions  
in Collective Adaptive Systems**

Borbala Benko ([bbenko@hit.bme.hu](mailto:bbenko@hit.bme.hu))  
Budapest University of Technology and Economics



## Agenda

- Intro: learn, adaptation, ...
- Two case studies for online learning:
  - Pervasive learner gaming agent
  - Autonomously evolving communication protocols
- A few interesting tools from the world of data mining
  - Dynamic Time Warping
  - The idea of collaborative filtering and the kNN method
  - Feature handling tricks

PerAda-ASSYST Summer School, 2010 2



## Adaptive systems – the challenge of adaptation

- Adaptive system: Ability to adjust itself to the changes of the environment
  - Fine-tunes operational parameters or alters its behavior
  - According to the observed situation
- How to achieve adaptivity?



## How to achieve adaptativity?

### Knowledge based approach

- Idea: know the world and utilize this knowledge (semantics, rules & reasoning)
- Operational model:
  - Understand what you see (semantic categorization)
  - Deduce (rule engine)
  - Act
- Prerequisites:
  - Data model (situation detection)
  - Extensive world model (rules, taxonomies)

### Learning based approach

- Idea: learn from what you see
- Operational model:
  - Abstract from what you see (features, historic similarity)
  - Induce (use your experience)
  - Act
- Prerequisites:
  - Data model (feature extraction)
  - Knowledge extraction mechanism



## They're so similar, aren't they?

- Theoretic difference
  - Deductive vs Inductive, A priori vs A posteriori knowledge:
    - The scientist tells how to adapt
    - The system learns how to adapt from its experience
- Different operational properties
  - Knowledge based approach
    - Guarantees optimal action. (within model)
    - Deterministic (foreseeable) response. Can be formally analyzed offline.
    - ??? when unforeseen situation (incomplete world model)
    - ??? when the world reacts in a different way (improper world model)
  - Learning based approach:
    - Optimality: short term NOT, long term YES (global optimum).
    - Not deterministic, no formal analysis (probabilistic analysis is possible)
    - Adapts to unforeseen situations (with time)
    - Adapts to changing world rules (with time)



## Which approach is better?

- Both are good, but in different aspects.

	Injected Knowledge	Offline Learning	Online Learning
Phases	Operation	Training Operation	Operation
Input	Complete world model Data model	Examples Data model	Data model
Guarantees	Yes	Maybe	No (Long-term or probabilistic: maybe)
Robustness (unforeseen situations)	No	Maybe	Yes



## Does learning really work?

- Two case studies:
  - A pervasive learner gaming agent
  - Autonomously evolving communication protocols



## Pervasive learner gaming agent

*Credits: Dávid Lányi*



## The challenge

---

- Basics:
  - Connect-5 game, human vs. agent
  - No initial knowledge
- Distributed parallel setting
  - Multitudes of agents and humans playing games in parallel
- Dynamics: changing game rules (connect-4, connect-6, draw-rectangle, ...)
- Pervasive-adaptive setting
  - Agents form an ad-hoc network
  - Agents can communicate with their neighbors (share expertise)



## Problem specification

---

- Agents with limited observation capacity
- No initial knowledge
- Interactions
- Time series model
- Dynamic goals (changing game rules)
- Feedback at certain states (reinforcement)
  
- Collectiveness
  
- Need for self-optimization



## How to answer this challenge?

- Solution model:
  - Reinforcement learning with incremental Least Square Temporal Difference
  - + Collective dimension: Knowledge sharing
  - + Self-optimization: Knowledge optimization



## Preface to LSTD: Markov Decision Process

- Markov Decision Process: (S,A,R,P, $\gamma$ )
  - S: states
  - A: actions
  - R:  $S \times S \rightarrow [-1,1]$ : reward function
  - P:  $S \times A \times S \rightarrow [0,1]$ : state transition probability function
  - $\gamma$ : discount factor
- If we know P and R, it is possible to analytically
  - find a policy ( $\pi: S \rightarrow A$ ) that maximizes the discounted total reward
  - where the value of a state ( $V: S \rightarrow [0,1]$ ) is modeled with the expected rewards received when following that policy
- Problem 1: P and R are not fully observable.
  - Solution: use what we know
- Problem 2: S is too large and ineffective
  - Solution: linear approximator  $V^\pi(s) \approx w^T \phi(s)$
  - Where  $\phi$  is the feature vector of the observed state



## Least Squares Temporal Difference

- Follows the previous line (State transitions, Value functions, Rewards
- But
  - Uses observations only (observed state transitions, rewards)
  - Builds the matrices/vectors incrementally
- Two data structures:
  - **Vector  $b$** : Perceived goodness of each 'state' (discounted total reward experienced there)
  - **Matrix  $A$** : Probability of 'state' transitions
- Feature based instead of state based! ('state'=features)
- Value of the next state:  $A^{-1}b$



## Side effects

- Matrix  $A$  = effect of the agent's action + the opponent's reaction
  - No need to explicitly model the opponent (prejudice)
  - Is applicable for  $n > 1$  opponents too
- Influence of the agent's action choice strategy
  - Short-term objective (reward) vs. Long-term objective (general knowledge)
- Influence of the opponent's behaviour
  - Convergence speed
- Sparsity prevention strategies



## Collective dimension

- Collective factor:
  - Agents can share their knowledge
  - Why? When? How?
  - Methodology: sharing A and b
    - $A' = w_1 * A + w_2 * A_{rec}$
    - $b' = w_1 * b + w_2 * b_{rec}$
    - where  $w_1 + w_2 = 1$
  - Extremities
    - if  $w_1 = 0 \rightarrow$  knowledge replacement (cloning)
    - if  $w_2 = 0 \rightarrow$  no import
  - How to determine the weight?
    - Self-confidence based measure
    - Sparsity based measure
    - Random numbers



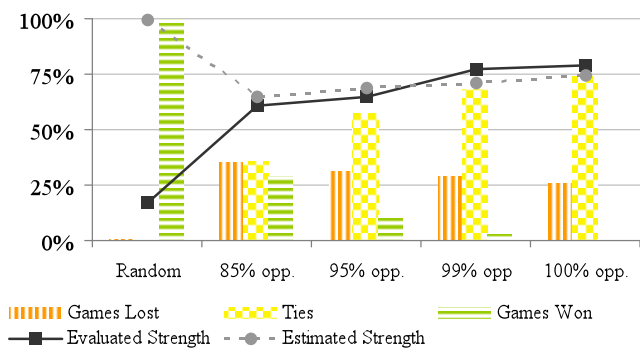
## Adaptation to changing game rules

- The agent senses the change implicitly, through the reward
- Seamless adaptation:
  - Discount factor  $\rightarrow$  the old knowledge flows out
  - Matrix A and vector b follow the changes automatically
- Knowledge purge
  - Low self-confidence  $\rightarrow$  the agent decides to purge its knowledge and start from scratch
- Knowledge import
  - Collective learning



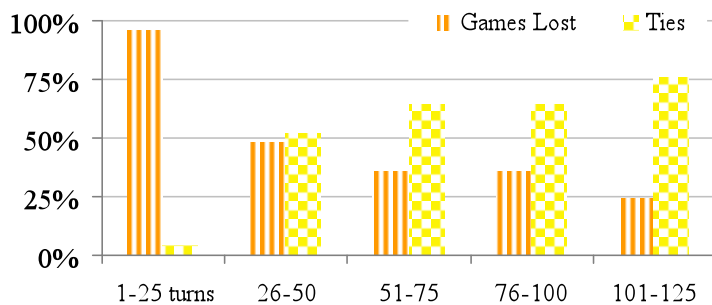
## Effect of the opponent's strategy

- 5 identical agents, 5 opponents (random, mathematically optimal strategies with 15,5,1,0% error probability)
- Phase 1: Online learning against the opponent
- Phase 2: Offline evaluation against the 100% opponent



## Effect of changing game rules

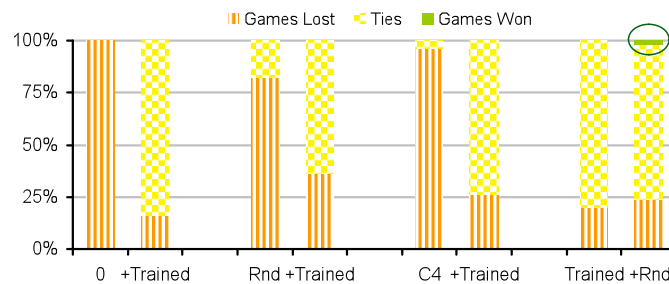
- Seamless adaptation: Connect-5-trained agent has to play Connect-4





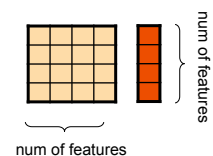
## Effect of knowledge sharing

- Receptor + imported knowledge
  - Empty agent imports 100%-trained knowledge
  - Random trained agent imports 100%-trained knowledge
  - Connect-4 trained agent imports 100%-trained knowledge
  - 100% trained agent imports Random-trained knowledge
- Chart: offline evaluation before and after the import



## Feature optimization

- Features have very high impact on the learning
  - Good feature selection = fast convergence
  - Bad feature selection = slow learning (no learning?)
  - Size and meaning!
- Feature set optimization:
  - Eliminate unworthy features
  - Eliminate redundancy
  - Create better features
- When to optimize the knowledge?
- Two of the possible optimization methodologies:
  - Singular value decomposition (SVD)
  - Correlation groups





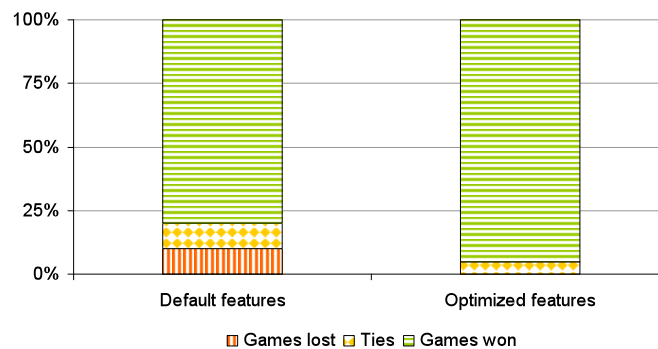
## Greedy feature optimization

- Identify unworthy features:
  - If unfilled  $\rightarrow$  unworthy
- Identify correlation groups:
  - Correlation group:
    - $G = \{G_i\}$ , where  $G_i = \{\varphi_1, \varphi_2, \dots, \varphi_{n_i}\}$ ,
    - so that for all  $j, k$  in  $G_i$ :  $\text{corr}(\varphi_j, \varphi_k) \approx 1$
- For each correlation group:
  - Remove the original features
  - Add a new feature (average of the original constituents' values)



## Effect of feature optimization

- 40 online learning rounds, then offline evaluation against a 50% opponent
- Case1: default features, Case2: optimized features





## Results

---

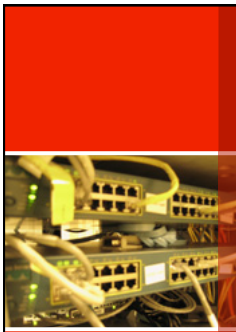
- What did we achieve?
  - Learning with no initial (pre-injected) knowledge
    - Initial features: yes, but they get optimized
  - Autonomous decisions in a very interactive environment
  - Collective learning
  - Self-optimization
  - Promising evaluation results



## Outlook: Applicability, Limitations

---

- Is this useful in real life?
  - Many real-world problems can be abstracted in a similar way
  - Some examples:
    - Intelligent user interfaces, gadgets
      - Intentions of the human user may change from time to time
    - Intelligent pervasive services
  - Generally fits well with the vision of socio-technical cities



## Autonomously Evolving Communication Protocols

*Credits: Endre S. Varga, Bernát Wiandt, Vilmos Simon*



### The challenge

- Communication protocol = crucial point in all networks
  - What and when and whom to disseminate
  - Too 'talkative', too 'tight-lipped'
- Today:
  - Manual protocol design
- Vision: different levels of autonomy:
  - Level1: Parameter optimization with offline/online simulations
  - Level2: Protocol selection
  - Level3: Autonomous evolution of protocols
    - There are basic building blocks
    - Protocol instance = combination built of these blocks



## Problem specification

---

- Emergence of protocols
  - Autonomous protocol evolution
    - Generation of new protocol instances
    - Non-obstructive performance evaluation
    - Natural selection
- Tool: Genetic programming



## Basics: GA, GP

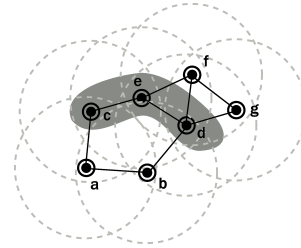
---

- Genetic algorithm:
  - Global random search (guarantees optimum)
  - Basics:
    - Generation, Individuals
    - Fitness function
    - Genetic operators (crossover and mutation)
    - Natural selection
  - Diversity of the population
- Genetic programming
  - Individuals are programs



## Multi-hop broadcast in ad-hoc networks

- Multi-hop broadcast in ad-hoc networks:
  - Distribute messages globally to all nodes
  - Local broadcast (radio) channel
- Formal solution: Minimal Connected Dominating Set
- Finding MCDS is NP-complete
- The topology may change faster than we can observe
  - MCDS is not the way to go
- Manually engineered protocols



## Stage 1: Natural selection of protocols

- A self-tuning system that selects the optimal protocol from a predefined set
  - Input: a set of manually engineered protocols
  - A fitness function to evaluate the performance
  - Natural selection
- Problem: instable choice
  - The diversity of protocols is too low, "Rock-Paper-Scissors" effect



## Stage 2: Evolution of protocols

- Use GP to generate and select protocols
- How to describe a protocol?
  - Classical programming language: NO
  - Genetic programming language: YES
    - No variables
    - Instead: Typed stacks
    - Instructions take their arguments from the stack
    - Extensions:
      - Even handlers (data, neighbor, timer, ...)
      - Relations
      - NOP

```

relation.dup // duplicate for Step1
relation.dup // duplicate for Step2

node.self // nodeId to the node stack

// Step1: look up my direct neighbors
relation.filter_key eq // params: top
of node stack, top of relation stack

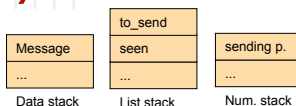
// Step2: look up neighbors of the
neighbors
relation.join // params: two top
elements of the relation stack

```

Finding my 2-hop neighbors from the list of all neighbor pairs



## Sample: simple algorithm



```

On receive (data: DataMessage) {
  delta := 2
  if contains(seen, data) then
    period := period + delta // Increases broadcast period
  else
    to_send := to_send + data // Adds the message to the to_send list
    seen := seen + data // Saves the message to the seen list
}

```

```

handler data { // If a new message arrives
  number.popglobal // Gets the sending period from the global stack
  list.popglobal // Gets the to_send list from the global stack
  list.popglobal // Gets the seen list from the global stack
  data.dup // Makes a copy of the data message
  list.inlist // Checks if the data message (top of the data stack) is in the seen
list (top of the list stack), and puts the result to the bool stack. (Note: the lists
are typed.)
  // Condition comes from the bool stack
  if // If we already saw this message
    number(2) // Instantiates a new number with value=2 (variable delta)
    number.add // Increases sending period with delta
  else // If the message is new
    data.dup // Duplicates the message so that it can be added to two lists
    list.additem // Adds the message to the seen list
    list.swap // Swaps the to_send and seen lists
    list.additem // Add the message to to_send list
    list.swap
  endif
  number.pushglobal // Puts the sending period to the global stack
  list.pushglobal // Puts the to_send list to the global stack
  list.pushglobal // Puts the seen list to the global stack
}

```



## Genetic operators on the protocol code

- Combination:
  - Combines the bodies of two event handlers
- Mutation:
  - Changes the body of an event handler



## Measurability and Decision inversion

- What to measure?
  - New info = good, Duplicate info = bad
- How to measure it effectively?
  - Only the receiver can determine whether the message is useful or a duplicate
  - Messages can be lost
  - Control messages must use the same channel as payload messages do
  - Control messages can be lost, too
  - Acknowledgements to control messages can be lost, as well
- Idea: inverted decision making
  - What if
    - Results are not sent back to the sender
    - But the receiver uses them directly



## Evaluation by simulation

- Test setting: a realistic problem case
  - 100 nodes in a 50x50 area, Random Direction Mobility Model (movement pattern), 1m/s speed, New message: every 20s, Simulated time: 7000s
- At each node:
  - Own generation of protocols, Round-robin
  - Quota
- Measured metrics:
  - Global: Score, Duplicates, Useful messages
  - Local: Score, Duplicates, Useful messages,
    - Instruction count, handler counts, quota remainder
    - Neighbors, Wins, Winrate, Minscore, Maxscore
    - Epoch



## Some interesting results

- Dependence between the global and local measures
  - Question: how does the local score relate to the global score
  - Result: Global and local measures are linearly dependent and strongly associated
- Information content of various data sets
  - Question: what measure sets hold the same information as the global score
    - Local scores
    - Raw local data
    - Local score + raw local data
    - Local score + raw local data + quasi-local data
  - Tool: predictive (regression) model



## Information content

	Local scores	Raw local	Local scores +Raw local	Local scores +Raw local + Quasi local
Mean actual value	-6.68	-6.68	-6.68	-6.68
Mean predicted value	-6.69	-6.23	-7.24	-7
Mean absolute error	1.63	5.62	3.98	1.21
Root mean square error	4.67	8.41	2.51	1.78
Predictive confidence	Naive+85.49%	Naive+73.89%	Naive+87.6%	Naive+94.44%

Residual plot: actual vs error



## Some interesting results

- The evolution works
  - NOPs:
    - Epoch 1: NOPs tend to occur in bad protocols (weak correlation with the negative indicators such as remaining quota)
    - Epoch 2: No clear trend
    - Epoch 3: the trend turns, NOPs often occur in well-performing protocols (weak negative correlation with the negative indicators)
  - Why does this mean that the evolution works?
    - NOPs can ONLY occur in evolutionary protocols
    - If NOPs occur in successful instances this means that the evolution results in well-performing instances, so it works



## Summary

---

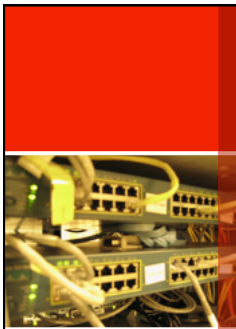
- A self-optimizing system that evolves (creates and selects) communication protocols
- Protocol = Real protocol, not just operational parameters
- Bio-inspired ☺



## Outlook: Applicability, Limitations

---

- Applicability:
  - The multi-Hop broadcast problem is an especially beneficial application area
  - There are be similar challenges in other areas (e.g. sensor networks)
  - We don't say that evolution is a generally applicable effective tool - but there may be other good areas for it to be used
- Limitations:
  - Guarantees...

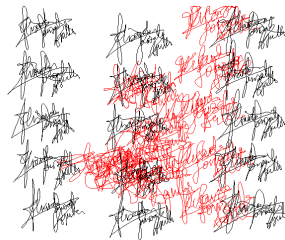


## A few useful but less known Data Mining tools



## Data Mining in general

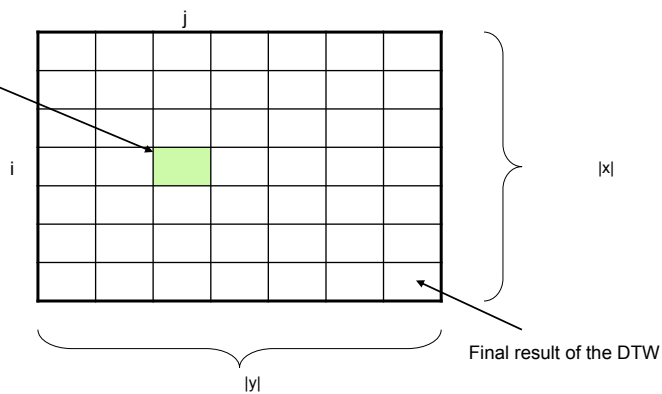
- Challenge: find the minimal distance between two time series so that
  - There may be insertions
  - There may be deletions
- E.g. signature, movement pattern





## DTW – Distance matrix (1)

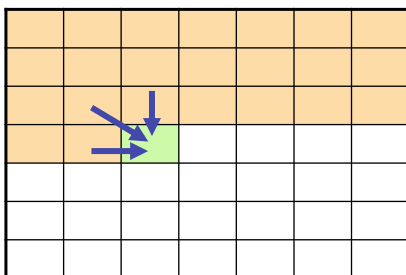
The smallest distance between  $x[0..i]$  and  $y[0..j]$ , besides all possible insertions/deletion



## DTW – Distance matrix (2)

$$\text{val}(i, j) = \text{distance}(x[i], y[j]) + \text{Min}(\text{val}(i-1, j-1), \text{val}(i-1, j), \text{val}(i, j-1))$$

normal way    insertion    deletion





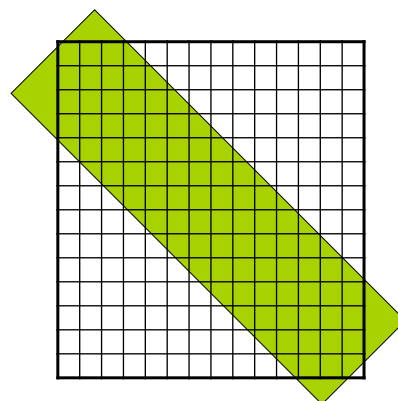
## DTW – Distance matrix (3)

0	INF	INF	INF	INF	INF	INF	INF
INF							
INF							
INF							
INF							
INF							
INF							
INF							



## Efficient implementation

- Naïve implementation:
  - $n \times m$  matrix
- Memory-efficient implementation:
  - Two rows: previous and current
  - Fill, then switch
- Runtime efficiency: limit





## DTW – Pro and Contra

- Pro:
  - Straightforward, works well in practical problems
- Contra:
  - n-square algorithm
  - Sensitive for outliers and large values
    - Data normalization
    - Band-pass filter
  - Not applicable for periodic time series (e.g. voice)



## Collaborative filtering

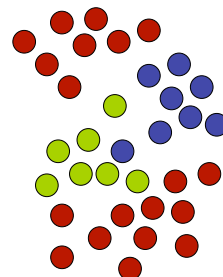
- Sample challenge: recommend me books I like
- Approaches:
  - Classic: Identify features, Look for books with similar features
  - Collaborative: look for people who have similar book reading history, and recommend what they bought
- Collaborative filtering: use the information coming from other agents to predict the behavior of the current agent
- A common method:
  - Locate the k nearest neighbors (kNN)
  - Extract their preference





## K nearest neighbors (kNN)

- kNN: original purpose
  - Given: a set of labeled items
  - Find the label of the new item
- Idea:
  - Find the nearest neighbors of the new item
  - Majority decision for the label
- Problems:
  - Size of  $K = ?$
  - Distance metrics
  - How to find neighbors effectively?
- Ideas: hashing



## Tricks with features

- Problem of redundant features
- Feature optimization methods:
  - Preprocessing: norms, discretization, smoothing
  - Feature reduction, Feature selection
    - Iterative methods: top-down and bottom-up
  - Feature injection
    - Common sense, Brute force
  - Feature extraction
    - SVD



## Summary

---

- Learning
- Two case studies for online learning:
  - A collective-adaptive game agent (connect-5)
  - Autonomously evolving protocols for multi-hop broadcast in ad-hoc networks
- A few tools from the Data Mining world



**Thanks! Questions?**

*Borbala Benko*  
*[bbenko@hit.bme.hu](mailto:bbenko@hit.bme.hu)*