TOP-DOWN ABSTRACTION LEARNING USING PREDICTION AS A SUPERVISORY SIGNAL

AAAI REPLEARN

Jonathan Mugan

July 15, 2013
Imagine a rat in a Skinner box.

The rat can see a screen of images, and a dot in the lower-right corner determines if there will be a shock.

Bottom-up methods may not find this dot, but a top-down approach requires a supervisory signal.

Our supervisory signal comes from predictions.
Agenda

• The importance of top-down abstraction learning
• Autonomous development with top-down abstraction learning
• Application of autonomous development to cyber security
• Top-down abstraction learning for cyber security
Agenda

• The importance of top-down abstraction learning
• Autonomous development with top-down abstraction learning
• Application of autonomous development to cyber security
• Top-down abstraction learning for cyber security
The Qualitative Learner of Actions and Perception, QLAP

1. Begin with a very broad discretization of the environment.

2. Simultaneously learn a discretization and a set of predictive models of the environment.

3. Convert the models into plans, and form the plans into a set of hierarchical actions.

4. Use learned actions to explore the environment.
Perception in QLAP

Images →

Continuous variables → Discrete variables → Models of the environment

Feedback from model to discretization
Perception in QLAP

Images

time

Continuous variables

...
Perception in QLAP

Images

Continuous variables

Discrete variables

...
Perception in QLAP

Images

Continuous variables

Discrete variables

Models of the environment
Perception in QLAP

Images

Continuous variables

Discrete variables

Models of the environment

Feedback from model to discretization
Convert Models to Plans

Model

Model in the form of a dynamic Bayesian network

[Dean and Kanazawa, 1989]

Plan

Plan in the form of a Q-function

[Dean and Kanazawa, 1989]

\[ Q(s, a) \]

\[ \pi(s) = \arg \max_a Q(s, a) \]
The result is a useful abstract state representation and a hierarchy of effective higher-level actions.
Autonomous Learning

- Qualitative Representation
- Learning Predictive Models
- From Models to Actions and Plans
- Exploration
A Qualitative Representation

A qualitative representation encodes the values of variables relative to known landmarks [Kuipers, 1994].

Landmarks bridge the gap between the continuous and the discrete.

The variable value is either less than, greater than, or equal to that landmark.
A Qualitative Representation

X

landmarks
A Qualitative Representation

\[ X = x \]
A Qualitative Representation

\[ X = x \]
Landmarks and Qualitative Values

Variable $X$ with landmarks $l_1, l_2$ has qualitative values

$$Q(X) = \{(-\infty, l_1), l_1, (l_1, l_2), l_2, (l_2, +\infty)\}$$
Initially, variables have no landmarks

\[ Q(X) = \{(-\infty, +\infty)\} \]
Initial Landmarks

Initially, variables have no landmarks

\[ Q(X) = \{(-\infty, +\infty)\} \]

But for each variable \( X \) we define

\[ \dot{X}_t = X_t - X_{t-1} \]

\[ Q(\dot{X}) = \{(-\infty, 0), 0, (0, +\infty)\} = \{[-], [0], [+]\} \]
Advantages of a Qualitative Representation

1. Generalization: different real values map to the same qualitative value.

2. Focus: the learner can focus on important events.
QLAP uses a qualitative representation to model the continuous with special predicates

Consider \( x \in Q(X) = \{(-\infty, l_1), l_1, (l_1, l_2), l_2, (l_2, +\infty)\} \)

\( \text{event}(t, X, x) \)
QLAP uses a qualitative representation to model the continuous with special predicates

Consider $x \in Q(X) = \{(-\infty, l_1), l_1, (l_1, l_2), l_2, (l_2, +\infty)\}$

**Example:**

$\text{event}(t, X, l_2) \quad \implies \quad X_{t-1} \neq l_2 \quad \text{and} \quad X_t = l_2$

$X_t \rightarrow l_2$
QLAP uses a qualitative representation to model the continuous with special predicates

Consider \( x \in Q(X) = \{(-\infty, l_1), l_1, (l_1, l_2), l_2, (l_2, +\infty)\} \)

\[ \text{event}(t, X, l_2) \quad \text{such that} \quad X_{t-1} \neq l_2 \text{ and } X_t = l_2 \]

\[ X_t \rightarrow l_2 \]

\[ \text{soon}(t, X, x) \equiv \exists t'[t \leq t' \leq t+k \text{ and } \text{event}(t', X, x)] \]

\( \text{soon} \) is a time window for an event to occur
Autonomous Learning

- Qualitative Representation
- Learning Predictive Models
- From Models to Actions and Plans
- Exploration
Predictive models are learned by identifying contingencies

A contingency is a pair of events that occur together in time. E.g., flip switch and light goes on.

Humans have an innate contingency detection module [Gergely and Watson, 1999].

Human infants can detect contingencies shortly after birth [DeCasper and Carstens, 1981].

Contingencies are:
1. Easy to learn; they only require looking at pairs of events.
2. A natural representation for planning.
Contingencies

Learn contingency \( \langle E_1 \Rightarrow E_2 \rangle \) when

\( E_2 \) is more likely to soon occur given that \( E_1 \) has occurred than otherwise

\[
P(\text{soon}(E_2) \mid E_1) > P(\text{soon}(E_2))
\]

We look at all pairs of events.
Each model is based on a contingency

Extracted contingences become dynamic Bayesian networks.
DBN with context variables

Context variables learned through marginal attribution [Drescher, 1991]

antecedent event

event\((t, X, x)\)

consequent event

soon\((t, Y, y)\)

context variables

\(V_1\)

\(V_n\)

\(V_i\)

\(V_2\)

Conditional Probability Table
Top-down abstraction learning

\[ \text{event}(t, X, x) \rightarrow \text{soon}(t, Y, y) \]
Top-down abstraction learning

\[ \text{event}(t, X, x) \rightarrow \text{soon}(t, Y, y) \]

Environment responds with a “yes” or “no.”

<table>
<thead>
<tr>
<th>Time</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>92</td>
<td>Yes</td>
</tr>
<tr>
<td>93</td>
<td>Yes</td>
</tr>
<tr>
<td>96</td>
<td>Yes</td>
</tr>
<tr>
<td>100</td>
<td>No</td>
</tr>
<tr>
<td>103</td>
<td>No</td>
</tr>
<tr>
<td>105</td>
<td>No</td>
</tr>
</tbody>
</table>
Top-down abstraction learning

$event(t, X, x) \rightarrow soon(t, Y, y)$

Environment responds with a “yes” or “no.”

| $V(t)$ | 92 | Yes |
| 93 | Yes |
| 96 | Yes |
| 100 | No |
| 103 | No |
| 105 | No |

Fayyad and Irani [1993]

Entropy:

$$H(S) = -\sum_j P(S = s_j) \log_2 P(S = s_j)$$

Information gain:

$$I_g = H(S) - \frac{|S^-|}{|S|} H(S^-) - \frac{|S^+|}{|S|} H(S^+)$$
Autonomous Learning

• Qualitative Representation
• Learning Predictive Models
• From Models to Actions and Plans
• Exploration
Two Types of Planning

Planning in QLAP combines symbolic and MDP planning

Symbolic Planning

Useful when only some states and variables are relevant.

QLAP uses symbolic planning to link models together.

\[ d \leftarrow c \leftarrow b \leftarrow a \]

MDP Planning

Useful when you need to model uncertainty.

QLAP uses reinforcement learning within models.

\[ Q(s, a) \leftarrow \sum_{s'} P(s' | s, a) \left[ R(s') + \gamma \max_{a'} Q(s', a') \right] \]
Motor actions directly set effectors.

There is a plan and action for each discrete motor value.
Hierarchy of Actions and Plans

Easy to build on top of existing pieces.
The end product of development.
Movie of learning structure
Autonomous Learning

- Qualitative Representation
- Learning Predictive Models
- From Models to Actions and Plans
- Exploration
Motor Babbling
Exploration at 50,000 timesteps

Learns abstractions:
1. The force needed to move the hand.
2. The limits of movement.
3. Having its hand be the left or right of the block.
Exploration at 100,000 timesteps
Task: hit block off table
Task: grasp block
The importance of top-down abstraction learning

Autonomous development with top-down abstraction learning

Application of autonomous development to cyber security

Top-down abstraction learning for cyber security
The extension of QLAP to cyber security is called Cy-QLAP
**Generalized Cy-QLAP Algorithm**

We expanded the QLAP developmental learning algorithm into a domain-general system protection algorithm.

Generalized Cy-QLAP Algorithm

1. human SME specifies a set of states and actions
2. human SME specifies a set of undesirable events that should be avoided
3. Cy-QLAP actively explores to learn the dynamics of the environment
4. do forever:
   a. Cy-QLAP monitors the system to see if it is possible to formulate a plan to bring about an undesirable event
   b. If such a plan is found, Cy-QLAP takes a proportional action to break a link in that plan
Experimental Environment

Windows 7 protected end node

- Windows Registry
- Windows logs
- packet sniffer

Cy-QLAP

- sensitive files

Cy-QLAP remote process

Linux Ubuntu remote virtual machine
Windows 7 protected end node
Experimental Environment

Windows 7 protected end node

Cy-QLAP

Sensitive files
Experimental Environment

Windows Registry

Windows logs

packet sniffer

Cy-QLAP

sensitive files

Windows 7 protected end node
Experimental Environment

Windows Registry
Windows logs
packet sniffer
Cy-QLAP
sensitive files
Cy-QLAP remote process
Linux Ubuntu remote virtual machine

Windows 7 protected end node
Experimental results on autonomous exploration and learning

<table>
<thead>
<tr>
<th>Generalized Cy-QLAP Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. human SME specifies a set of states and actions</td>
</tr>
<tr>
<td>2. human SME specifies a set of undesirable events that should be avoided</td>
</tr>
<tr>
<td>3. perform <strong>Autonomous Exploration and Learning</strong></td>
</tr>
<tr>
<td>4. do forever:</td>
</tr>
<tr>
<td>a. activate the <strong>Threat Monitoring Module</strong></td>
</tr>
<tr>
<td>b. If such a plan is found, activate the <strong>Threat Intervention Module</strong></td>
</tr>
</tbody>
</table>
Experimental results on autonomous exploration and learning

• Cy-QLAP learned the important dynamics of the environment

• Cy-QLAP learned how to
  – open a file remotely
  – exfiltrate a file
  – open and close a file share
Experimental results on protecting the system

Generalized Cy-QLAP Algorithm

1. human SME specifies a set of states and actions
2. human SME specifies a set of undesirable events that should be avoided
3. perform **Autonomous Exploration and Learning**
4. do forever:
   a. activate the **Threat Monitoring Module**
   b. If such a plan is found, activate the **Threat Intervention Module**
Experimental results on protecting the system

• Cy-QLAP:
  – learned that if a file share was open, a sensitive file could be exfiltrated.
  – also learned how to close file shares.
• Cy-QLAP therefore would close a file share as soon as it was opened.

Cy-QLAP learned to protect the system without being told how.

We know of no other cyber defense system that learns through exploration.
Agenda

• The importance of top-down abstraction learning
• Autonomous development with top-down abstraction learning
• Application of autonomous development to cyber security
• Top-down abstraction learning for cyber security
Abstractions allow the controller to see each aspect of the system at the right level of detail.

- high-level abstractions:
  - Installed programs
  - System logs
  - Processes
  - System calls
  - Register values

- low-level abstractions
Landmarks are an instance of an abstraction hierarchy

QLAP noted the real value of all variables each time a model was applied.
Abstracting the landmark process in QLAP to find level-of-detail abstractions

• Approach: define a set of abstraction hierarchies and note the value of the current and next level of each hierarchy each time a model is applied
  – Keep going down until the next level is not more reliable than the current level

Example: configuration file abstraction hierarchy
Thanks for listening. Any questions?

Jonathan Mugan
jmugan@21ct.com
www.jonathanmugan.com
@jmugan