Monte Carlo Tree Search:
Learning for Games and Learning as a Game

Michèle Sebag
joint work: Olivier Teytaud, Sylvain Gelly, Philippe Rolet,
Romain Gaudel

TAO, Univ. Paris-Sud
KAUST — Feb. 26th, 2011
Why do we need cognitive systems?

Artificial Intelligence
From *Nice-to-have* to *Must-have*

Why?
- The world is partially observable
- Technology and users evolve continuously
- The art of programming hardly scales up
Cognitive Systems

What remains to be done

- Reasoning 10%
- Dialogue 60%
- Perception 90%
Reasoning

Learning

Games

Games are

- Closed microworlds
- Simple rules, yet deep complexity
- Challenges for human players
- *Intimately related to human learning*
The message to take home

- A grand AI Challenge: the game of Go
- Significant advances have been made since 2006 MoGo
- The approach is general
- Two examples of application:
  - How to select informative examples Active Learning
  - How to select informative features Feature Selection
Features

- Number of games $2 \cdot 10^{170} \sim$ number of atoms in universe.
- Branching factor: 200 ($\sim$ 30 for chess)
- Local and global features (symmetries, freedom, ...)

Therefore

- Brute force will not make it
- Smart pruning forcluded
Features

- Explore the game tree
- Gradually focusing on most promising moves
- A weak but unbiased assessment function

Principles of MoGo

- While in the tree, select the best son node
- Otherwise play random... a weak but unbiased assessment.

Allows the machine to play against itself and build its own strategy

Y. Wang and S. Gelly 2006
O. Teytaud, R. Coulomb, J.Y. Audibert, ...
Monte-Carlo Tree Search

- **Upper Confidence Tree (UCT)** [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

---

[1] L. Kocsis, and C. Szepesvári ECML’06
- **Upper Confidence Tree (UCT)** [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

---

[1] L. Kocsis, and C. Szepesvári ECML'06
Upper Confidence Tree (UCT) [1]
- Gradually grow the search tree
- Building Blocks
  - Select next action (bandit-based phase)
  - Add a node (leaf of the search tree)
  - Select next action bis (random phase)
  - Compute instant reward
  - Update information in visited nodes
- Returned solution:
  - Path visited most often

[1] L. Kocsis, and C. Szepesvári ECML'06
Monte-Carlo Tree Search

- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

[1] L. Kocsis and C. Szepesvári ECML’06
Monte-Carlo Tree Search

- **Upper Confidence Tree (UCT) [1]**
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

---

[1] L. Kocsis, and C. Szepesvári ECML'06
Monte-Carlo Tree Search

- **Upper Confidence Tree (UCT) [1]**
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

---

[1] L. Kocsis, and C. Szepesvári ECML’06
Monte-Carlo Tree Search

- **Upper Confidence Tree (UCT) [1]**
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

[1] L. Kocsis, and C. Szepesvári ECML’06
Monte-Carlo Tree Search

- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

[1] L. Kocsis, and C. Szepesvári ECML’06
Monte-Carlo Tree Search

- **Upper Confidence Tree (UCT) [1]**
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
- Returned solution:
  - Path visited most often

---

[1] L. Kocsis, and C. Szepesvári ECML'06
Monte-Carlo Tree Search

- **Upper Confidence Tree (UCT) [1]**
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

---

[1] L. Kocsis, and C. Szepesvári ECML'06
Monte-Carlo Tree Search

- **Upper Confidence Tree (UCT)** [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

---

[1] L. Kocsis, and C. Szepesvári ECML’06
Monte-Carlo Tree Search

- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

[1] L. Kocsis, and C. Szepesvári ECML'06
Monte-Carlo Tree Search

- Upper Confidence Tree (UCT) [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
  - Returned solution:
    - Path visited most often

[1] L. Kocsis, and C. Szepesvári ECML’06
Monte-Carlo Tree Search

- **Upper Confidence Tree (UCT)** [1]
  - Gradually grow the search tree
  - Building Blocks
    - Select next action (bandit-based phase)
    - Add a node (leaf of the search tree)
    - Select next action bis (random phase)
    - Compute instant reward
    - Update information in visited nodes
- Returned solution:
  - Path visited most often

[1] L. Kocsis, and C. Szepesvári ECML'06
Exploration vs Exploitation Dilemma

Multi-Armed Bandits

- In a casino, one wants to maximize one’s gains while playing.
- Play the best arms so far?
- But there might exist better arms...

Lai, Robbins 1985

Exploitation

Exploration
Stochastic multi-armed bandit problem

Setting
- A set of $K$ arms; when pulled, each provides a reward in $[0, 1]$ with distribution $X_k$
- Find a policy: time $t \rightarrow k_t$, receives $r_{k_t}$ (iid $\sim X_{k_t}$)
- Goal: maximize the expected cumulative reward

$$\sum_{t=1}^{T} r_t$$

Many applications
- Select news or ads to put on a Web page
- Select medications for patients
Stochastic multi-armed bandit problem, 2

Definitions

- Let $\mu_k$ the expected value of arm $k$
- Let $\mu^*$ the maximal value and $k^*$ an arm with maximal value
- The regret is:

$$R_T = \sum_{t=1}^{T} (\mu^* - \mu_{k_t}) = \sum_{k=1}^{K} n_k \Delta_k$$

where $n_k$ is the number of times arm $k$ has been played up to time $T$ and $\Delta_k = \mu^* - \mu_k$.

Goal Find a policy minimizing $R_T$
Stochastic multi-armed bandit problem, 3

Upper Confidence Bound

- For each arm $k$, compute $\hat{\mu}_k$ its empirical value;
- Select

$$
k^* = \arg\max_k \hat{\mu}_k + C \sqrt{\frac{\log(\sum n_i)}{n_k}}
$$

Decision : Optimism in front of unknown !

- Exploitation : $\hat{\mu}_k$ : favors the best so far
- Exploration : $\sqrt{\frac{\log(\sum n_i)}{n_k}}$ : tends toward uniform sampling

Variants

- Take into account standard deviation of $\hat{\mu}_k$
- Trade-off controlled by $C$
- Progressive widening

Auer et al. 2001, 2002
Stochastic multi-armed bandit problem, 3

Upper Confidence Bound

- For each arm $k$, compute $\hat{\mu}_k$ its empirical value;
- Select

$$k^* = \arg\max \hat{\mu}_k + C\sqrt{\frac{\log(\sum n_i)}{n_k}}$$

Decision: Optimism in front of unknown!

- Exploitation: $\hat{\mu}_k$: favors the best so far
- Exploration: $\sqrt{\frac{\log(\sum n_i)}{n_k}}$: tends toward uniform sampling

Variants

- Take into account standard deviation of $\hat{\mu}_k$
- Trade-off controlled by $C$
- Progressive widening
Second Building Block: Weak unbiased assessment

Monte-Carlo-based Brügman (1993)

1. While possible, add a stone (white, black)
2. Compute Win(black)
3. Repeat 1-2 and average

Remark: The point is to be unbiased if there exists situations where you (wrongly) think you’re in good shape then you go there and you’re in bad shape...
Monte-Carlo Tree Search

Comments: MCTS grows an asymmetrical tree

- Most promising branches are more explored
- thus their assessment becomes more precise
- Needs heuristics to deal with many arms...
- Share information among branches

First to win over a 7th Dan player in $19 \times 19$
The message to take home

- A grand AI Challenge: the game of Go
- Significant advances have been made since 2006
- The approach is general

- Application to Machine Learning:
  Optimize accuracy of the learned hypothesis through
  - Example set
    Rolet et al., ECML 2009
  - Feature set
    Gaudel et al., ICML 2010
  - Active learning
Active Learning, position of the problem

Supervised learning, the setting

- Target hypothesis $h^*$
- Training set $\mathcal{E} = \{(x_i, y_i), i = 1 \ldots n\}$
- Learn $h_n$ from $\mathcal{E}$

Criteria

- Consistency: $h_n \rightarrow h^*$ when $n \rightarrow \infty$.
- Sample complexity: number of examples needed to reach the target with precision $\epsilon$

\[ \epsilon \rightarrow n_\epsilon \text{ s.t. } ||h_n - h^*|| < \epsilon \]
Motivations

- Given $x$, obtaining $h^*(x)$ is costly
- Goal: reduce *sample complexity* while keeping *generalization error* low
- Motivating application: numerical engineering

=> Learn simplified models with only ~ 100 examples
Active Learning, definition

Passive learning

\[ E = \{(x_i, y_i), i = 1 \ldots n\} \]

Active learning

\( x_{n+1} \) selected depending on \( \{(x_i, y_i), i = 1 \ldots n\} \)

In the best case, exponential improvement:
Active Learning as a Game

Optimization problem

Find \( F^* = \text{argmin} \ Err(\mathcal{A}, \mathcal{E}, \sigma, T) \sigma : \mathcal{E} \mapsto \mathcal{Z} \) sampling strategy

\( \mathcal{E} \) : Training data set

\( \mathcal{A} \) : Machine Learning algorithm

\( \mathcal{Z} \) : Set of instances

\( \sigma \) : \( \mathcal{E} \mapsto \mathcal{Z} \) sampling strategy

\( T \) : Time horizon

\( Err \) : Generalization error

Bottlenecks

- Combinatorial optimization problem
- Generalization error unknown
Optimal Strategy for AL

- Learning algorithm $\mathcal{A}$
- Finite Horizon $T$
- Sampling strategy $S_T$
- Target concept $h^*$

**Goal:** $\arg\min E[\ Err(\mathcal{A}(S_T(h^*)), h^*)]$
\textbf{BAAL}(P_H, s_0, T, N)
for \( i = 1 \) to \( N \) do
\hspace{1cm} \( h = \text{DrawSurrogateHypothesis}(s_0) \)
\hspace{1cm} \( \text{Tree-Walk}(s_0, T, h) \)
end for
Return \( x = \arg \max_{x' \in X} \{ n(s \cup \{x'\}) \} \)
\text{Tree-Walk}(s, t, h)
Increment \( n(s) \)
if \( t > 0 \) then
\hspace{1cm} \( X(s) = \text{ArmSet}(s, n(s)) \)
\hspace{1cm} Select \( x^* = \text{UCB}(s, X(s)) \)
\hspace{1cm} Get label \( h(x^*) \) from surrogate
\hspace{1cm} \( r = \text{Tree-Walk}(s \cup \{(x^*, h(x^*))\}, t - 1, h) \)
else
Compute \( r = \text{Err}(A(s), h) \)
end if
\( r(s) \leftarrow (1 - \frac{1}{n(s)})r(s) + \frac{1}{n(s)} r \)
Return \( r \)
The message to take home

- A grand AI Challenge: the game of Go
- Significant advances have been made since 2006
- The approach is general

Application to Machine Learning:
Optimize accuracy of the learned hypothesis through
- Example set
  - Active learning
    - Rolet et al., ECML 2009
- Feature set
  - Feature selection
    - Gaudel et al., ICML 2010
Feature Selection

Optimization problem

\[ F^* = \arg\min_{\mathcal{F}} \text{Err}(\mathcal{A}, F, \mathcal{E}) \]

\( \mathcal{F} \): Set of features

\( F \): Feature subset

\( \mathcal{E} \): Training data set

\( \mathcal{A} \): Machine Learning algorithm

\text{Err} \): Generalization error

Feature Selection Goals

- Reduced Generalization Error
- More cost-effective models
- More understandable models

Bottlenecks

- Combinatorial optimization problem: find \( F \subseteq \mathcal{F} \)
- Generalization error unknown
FUSE: bandit-based phase
The many arms problem

- Bottleneck
  - A many-armed problem (hundreds of features)
    $\Rightarrow$ need to guide UCT

- How to control the number of arms?
  - Continuous heuristics [1]
    - Use a small exploration constant $c_e$
  - Discrete heuristics [2,3]: Progressive Widening
    - Consider only $\lfloor T^b \rfloor$ actions ($b < 1$)

[1] S. Gelly, and D. Silver ICML '07
Results on Madelon after 200,000 iterations

- **Remark**: FUSE$^R = \text{best of both worlds}
  - Removes redundancy (like CFS)
  - Keeps conditionally relevant features (like Random Forest)
### NIPS 2003 Feature Selection challenge

- **Test error on a disjoint test set**

<table>
<thead>
<tr>
<th>database</th>
<th>algorithm</th>
<th>challenge error</th>
<th>submitted features</th>
<th>irrelevant features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madelon</td>
<td>FSPP2 [1]</td>
<td>6.22% (1&lt;sup&gt;st&lt;/sup&gt;)</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>D-FUSE&lt;sup&gt;R&lt;/sup&gt;</td>
<td>6.50% (24&lt;sup&gt;th&lt;/sup&gt;)</td>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>Arcene</td>
<td>Bayes-nn-red [2]</td>
<td>7.20% (1&lt;sup&gt;st&lt;/sup&gt;)</td>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>D-FUSE&lt;sup&gt;R&lt;/sup&gt;(on all)</td>
<td>8.42% (3&lt;sup&gt;rd&lt;/sup&gt;)</td>
<td>500</td>
<td>34</td>
</tr>
<tr>
<td></td>
<td>D-FUSE&lt;sup&gt;R&lt;/sup&gt;</td>
<td>9.42% 500 (8&lt;sup&gt;th&lt;/sup&gt;)</td>
<td>500</td>
<td>0</td>
</tr>
</tbody>
</table>


Optimization under uncertainty
- many problems, ranging from optimal energy policy to robotics
- can be formalized as games

Lessons learned from MoGo
- training (acquiring expertise) > plugging-in expertise
- playing many games >> playing more complex games

Perspective in Machine Learning
- Coupling Active Learning and Feature Selection