Modern Reinforcement Learning Theory

John Langford (with help from many others...)

Yahoo Research

Backign Material: http://hunch.net/~jl/tutorial/RL.html

MLSS 2006, Taiwan, July 25
Reinforcement Learning is Always Relevant

- Supervised Learning
  - Classification
- Semi-Supervised Learning
- Markov Decision Process (RL)
  - N–armed Bandits
- Reinforcement Learning
  - Active Learning
The answer to: “Is this an RL problem?” is always “yes”.

The implication: RL theory is broadly applicable.

The other implication: RL theory is often only weakly relevant. (breadth + relevance = hard.)

Understanding a problem as an RL problem is the beginning to solving it. Whenever possible, you want to understand how the problem is special.
Outline

1. Sample Complexity Results

2. Limitations of Sample Complexity

3. Reductions Results
What is a sample complexity guarantee?

“With high probability, given $m(\cdot)$ samples, guarantee $\phi$ holds.”

1. $S =$ the number of states in an MDP

2. $A =$ the number of actions/state in an MDP

3. $T =$ the horizon time you care about (or $\gamma =$ discount factor)

4. $O =$ number of observations

5. $\epsilon =$ precision parameter
Important Derived Quantities

\[ V_t^\pi(s) = E_{(s,a,r) \sim \text{MDP}_{s,\pi}} \sum_{t'=1}^{t} r_{t'} \] is the value of being in state \( s \) and acting according to \( \pi \) for \( t \) timesteps.

\[ Q_t^\pi(s, a) = E_{(s,a,r) \sim \text{MDP}_{sa,\pi}} \sum_{t'=1}^{t} r_{t'} \] is the value of being in state \( s \), acting with \( a \), and then acting according to \( \pi \) for \( t \) timesteps.

\[ \pi^*(s) = \arg \max_a Q_t^\pi^*(s, a) \] is the recursive definition of optimal policy.

\[ Q_t^*(s, a) = Q_t^\pi^*(s, a) \] is short hand for optimal policy \( Q \) values.
The $E^3$ Guarantee

Trace Model = ability to read current state $s$, take action $a$, observe next state $s'$ and reward $r$. Notation: $TM : A \rightarrow S \times [0, 1]$.

Assume $\text{MDP}(S, A, p(s' | s, a))$ with horizon $T$

1. Original: + assume mixing time $\tau \Rightarrow \text{Poly}(S, A, \tau, \frac{1}{\epsilon})$ samples implies ability to act $\epsilon$ optimal for $T > \tau$.

2. Modified: $\text{Poly}(S, A, T, \frac{1}{\epsilon})$ samples implies ability to act $\epsilon$ optimal for $T$ timesteps.

(2) + mixing assumption implies (1). (2) holds even for deterministic worlds. We’ll go through (2).
$E^3$ Theorem Statement

Theorem: There exist an algorithm $E^3$ such that for all MDP $(S, A, T, p(s'|a, s))$ with rewards $r \in [0, 1]$, with probability $1 - \delta$, for all except $\text{Poly}(S, A, T, \frac{1}{\epsilon}, \ln \frac{1}{\delta})$ steps $Q_{T-t}^{E^3} \mod T(s, E^3(h)) \geq V_{T-t}^{*} \mod T(s) - \epsilon$ where $h$ is the history of observations.

Suboutline:

1. The Algorithm

2. The Proof
The **Known**(\(h\)) MDP

A state \(s\), all actions \(a\) leaving \(s\) and the probability of their outcomes is known if all actions \(a\) leaving \(s\) have been executed at least \(n\) times.

An MDP

Initially: known MDP = nothing
The **Known($h$) MDP**

An MDP

Probability
Action
Reward

Then:

Complete dangling action(s) with one state that always has reward 0.
The Known($h$) MDP

An MDP

Probability
Action
Reward

Probability
Action
Reward

Then:
The **Known\((h)\) MDP**

An MDP

![Diagram](image)

<table>
<thead>
<tr>
<th>Probability</th>
<th>Action</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>a₁</td>
</tr>
<tr>
<td>0.6</td>
<td>a₂</td>
<td>0</td>
</tr>
<tr>
<td>0.4</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Finally:

![Diagram](image)

<table>
<thead>
<tr>
<th>Probability</th>
<th>Action</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>a₂</td>
</tr>
<tr>
<td>0.58</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>0.42</td>
<td>a₁</td>
<td>1</td>
</tr>
</tbody>
</table>

*(note: the probabilities are empirical counts)*
The **Unknown**(h) MDP

**Unknown**(h) = **Known**(h) except the reward is 1 for actions which leave the known states and 0 otherwise.
Dynamic Program

Fundamental operation: Given MDP $M$ and state $s$,

$$DP(M, s, t) = a, v$$

where $v =$ the maximum expected $T - (t \mod T)$ reward sum and $a =$ action achieving it.

Computation:

$$DP(M, s, t) = \max_a E_{s', r \sim M(s, a)} r + DP(M, s', t + 1)$$

$$DP(M, s, nT) = 0$$
$E^3(h)$ Explicit Explore or Exploit Algorithm

1. If last $s$ not in $\text{Known}(h)$: choose the least previously used action

2. Else:

   (a) If $\text{DP}(\text{Unknown}(h)) > \epsilon'$ then act according $\text{DP}(\text{Unknown}(h))$
       until state is unknown or $t \mod T = 0$ then go to (1).

   (b) else act according to $\text{DP}(\text{Known}(h))$. 
The proof uses 5(!) MDPs

1. MDP — the true MDP (Imposed by world)

2. Known\( (h) \) = known MDP (Known by \( E^3 \) algorithm)

3. Unknown\( (h) \) = unknown MDP (Known by \( E^3 \) algorithm)

4. \( MDP_{K(h)} \) = MDP restricted to the known states (exists only in proof)

5. \( MDP_{U(h)} \) = MDP restricted to the known states with rewards set to 0 except for escaping rewards. (exists only in proof)
Proof Sketch:

Simulation Lemma:

\[ |\text{DP}(\text{MDP}_{U/K(h)}) - \text{DP}((\text{Un})\text{Known}(h))| \leq \frac{1}{\text{Poly}(S, A, T, \ln \frac{1}{\delta})} \]

Explore/Exploit Lemma:

\[ \text{DP}(\text{MDP}_{K(h)}) + T \text{DP}(\text{MDP}_{U(h)}) \geq \text{DP}(\text{MDP}) \]

So \( n = \text{Poly}(S, A, T, \ln \frac{1}{\delta}) \) implies ability to simulate on known states to precision \( \frac{1}{\text{Poly}(S, A, T, \ln \frac{1}{\delta})} << \epsilon \). ⇒ Explore/Exploit Lemma implies \( \text{DP}(\text{MDP}) - \text{DP}(\text{MDP}_{K(h)}) > \epsilon \Rightarrow \text{DP}(\text{MDP}_{U(h)}) > \frac{\epsilon}{T} \)

⇒ probability about \( \frac{\epsilon}{T} \) of encountering new state if exploring. This can happen only \( O\left(\frac{nSAT}{\epsilon}\right) \) times (Using the Chernoff bound). Each exploration uses at most \( T \) steps ⇒ proof.
Simulation Lemma Proof sketch

Next state

\[
P(s'|a,s)
\]
\[
P(s'|a,s) \text{ estimate}
\]

Enough samples \(\Rightarrow\) small \(l_1\) distance
\[
d = \sum_{s'} |\hat{p}(s'|a,s) - p(s'|a,s)|.
\]

\(l_1\) distance \(d\) \(\Rightarrow\) with probability \(1 - \frac{d}{2}\) draw from \(p(s'|a,s)\).

\(\Rightarrow\) \(1 - T\frac{d}{2}\) draw from \(T\)-length sequence.

\(\Rightarrow\) Expectations correct up to \(T\frac{d}{2}\).
So how do we prove $l_1$ convergence?

Each next state $s' = \text{IID draw.}$

$n$ IID coin flips converge to mean like $O\left(\frac{1}{\sqrt{n}}\right) = \text{Chernoff bound.}$

$$\Rightarrow \Pr \left( \left| \hat{p}(s'|a, s) - p(s'|a, s) \right| \leq \sqrt{\frac{\ln \frac{2}{\delta}}{2n}} \right) \geq 1 - \delta$$

Union $\Rightarrow \Pr \left( \forall s', s, a : \left| \hat{p}(s'|a, s) - p(s'|a, s) \right| \leq \sqrt{\frac{\ln \frac{2S^2A}{\delta}}{2n}} \right) \geq 1 - \delta$

Only $S$ next states $s'$ so $1 - \gamma$ probability on $\{s' : p(s'|a, s) > \frac{\gamma}{S}\}$ so

$n = \tilde{O} \left( \frac{S^2}{\gamma^2} \right)$ implies a $1 - \gamma$ fraction of $\hat{p}(s'|a, s)$ indistinguishable from $p(s'|a, s)$. 
Explore or Exploit Lemma Proof Sketch

\[ DP(MDP) = E_{(s,a,r)^T \sim \pi^*, MDP}[\sum_t r_t] \]

Let \( v = (s, a, r)^T \in \text{Known}(h) \)

Let \( p = \text{Pr}_{\pi^*, MDP}(v) \)

\[ = pE_{(s,a,r)^T \sim \pi^*, MDP|v}[\sum_t r_t] + (1 - p)E_{(s,a,r)^T \sim \pi^*, MDP|\bar{v}}[\sum_t r_t] \]

\[ \leq pE_{(s,a,r)^T \sim \pi^*, MDP|K(h)}[\sum_t r_t] + \text{DP}(MDP|U(h))E_{(s,a,r)^T \sim \pi^*, MDP|\bar{v}}[\sum_t r_t] \]

\[ \leq \text{DP}(MDP|K(h)) + \text{DP}(MDP|U(h))T \]
R-Max(h) Modification

(Roughly) The same result holds for the following algorithm:

1. If last $s$ not in $\text{Known}(h)$: choose the least previously used action.

2. Else act according to $\text{DP(known}(h) + \text{Unknown}(h))$.

(Where “+” adds the rewards of each MDP on each edge.)
Delayed Q-learning

The theorem can be tightened from $\text{Poly}(S,A)$ to $\tilde{O}(SA)$ using the Delayed Q-learning algorithm.
Outline

1. Sample Complexity Results

2. Limitations of Sample Complexity

3. Reductions Results
The Limits of Sample Complexity: A lower bound

Theorem: Any algorithm $A$ satisfying the $E^3$ statement must use at least $\Omega(TSA)$ actions to explore.

(There are stronger lower bounds, but this is sufficient.)
Proof

A "Key lock" MDP

States in a chain. One action leads to next state, all the rest lead to the beginning. The final state has an action with reward 1.
Implications

Lower bound $\Rightarrow$ the really big problems can’t be solved.

But the problems are solvable: we solve them every day.

$\Rightarrow$ More or different assumptions are required.
Attempt 1: Factored-$E^3$

Assume state is represented by a set of bits $s = (b_1, b_2, ..., b_n)$.

Assume $P(s' | s, a) = \prod_i P(b_i | B_i, a)$ where $B_i \subseteq \{b_1, b_2, ..., b_n\}$.

Theorem: There exist an algorithm Factored-$E^3$ such that for all Factored MDP, with probability $1 - \delta$, for all except

$$\text{Poly} \left( |MDP|, T, \frac{1}{\epsilon}, \ln \frac{1}{\delta} \right)$$

steps $Q^\text{Factored-E3}_{T-t \mod T}(s, \text{Factored} - E^3(h)) \geq V^*_{T-t \mod T}(s) - \epsilon$

where $h$ is the history of observations and $|MDP| = \text{description length}$.

Problem 1: Factoring isn’t enough. Consider this room...

Problem 2: The computational complexity of planning is high.
Attempt 2: Metric-\(E^3\)

Assume there exists \(\text{Model}(\{(s, a, r, s')^*\}, s'')\) such that with \(m\) experiences satisfying \(d((s, a), (s'', a'')) \leq \alpha\), Model outputs \(s', r\) from almost the same distribution as \(P(s', r|s'', a'')\).

Theorem: There exist an algorithm Metric-\(E^3(h)\) such that for all Metric MDPs, with probability \(1 - \delta\), for all except

\[
\text{Poly} \left( \text{Cover}(MDP), T, \frac{1}{\epsilon}, \ln \frac{1}{\delta} \right)
\]

steps \(Q_{T-t \ mod \ T}(s, E^3(h)) \geq V_{T-t \ mod \ T}^*(s) - \epsilon\) where \(h\) is the history of observations.

Problems: Can we reasonably expect Model to be that accurate?
Do we really want the guarantee these algorithms provide?

See Abbeel & Ng ICML 2005 for a bit of discussion.
Outline

1. Sample Complexity Results

2. Limitations of Sample Complexity

3. Reductions Results
Reduction-style

Philosophy: Start with what we know we can do (even if we can’t prove it), then use it repeatedly to see what we can do.

In Machine Learning, we know how to classify. Can we (re)use this ability to solve Reinforcement Learning?

General form: “cost sensitive” classification $D$ on $X \times [0, \infty)^k$:

$$\arg \min_h E_{x, \bar{c} \sim D} [c_h(x)]$$

(Reducible to binary classification. Reduction suggests minimizing $\sum_i c_i$ is good.)
How do we think about a general RL problem?

\[ D(o', r|(o, a, r)^*, o) = \text{conditional probability table} \]

1. \( o', o \in O \) = observations

2. \( r \in [0, \infty) \) = reward

3. \( a \in A \) = action:

Find \( \pi((o, a, r)^*, o) \to a \) maximizing:

\[ \eta(\pi) = E_{(o, a, r) \sim \pi, D} \left[ \sum_{t=1}^{T} r_t \right] \]
An “Algorithm”

Let $\pi^* =$ optimal policy

1. Given empty history, predict optimal $a$ assuming $\pi^*$ followed for $T - 1$ timesteps afterwards.

2. Given first prediction, 1 step history, predict optimal $a$ assuming $\pi^*$ followed for $T - 2$ timesteps afterwards.

3. Given 2 step history, predict optimal $a$ assuming $\pi^*$ followed for $T - 3$ timesteps afterwards.

4. ...
The Algorithm: Pretty Pictures
The Algorithm: Pretty Pictures

Start

Goal

16 Steps
Cost for action $a = \text{steps}(a) - \min_{a'} \text{steps}(a') = 0, 1, 0$
The Algorithm: Pretty Pictures

cost = 0, 1, 0, 0, 1
The Algorithm: Pretty Pictures

Goal

Start

13 Steps
15 Steps

Cost = 0, 2
The Algorithm: Pretty Pictures

Cost = 0, 0 Note: condition on suboptimal outcome at previous step.
The Critical Lemma

\[ A_t^*(D, \pi, h_t) = E_{(a,o,r)^T D,(\pi,h_t,\pi^*)} \left[ \sum_{t'=t}^{T} r_{t'} \right] = \text{Expected reward sum when starting with } \pi, \text{ acting according to } h_t \text{ and completing with the optimal policy. (} A \text{ for “Advantage”.)} \]

Lemma: For all \( D, \pi = (h_1, ..., h_T), \)

\[ \eta(\pi^*) - \eta(\pi) = \sum_{t=1}^{T} A_t^*(D, \pi, \pi_{t}^*) - A_t^*(D, \pi, h_t) \]

= sum of cost sensitive losses.
Proof

$$\eta(\pi^*) - \eta(\pi) = E_{(o,a,r)^T \sim D, \pi^*}[\sum_{t=1}^{T} r_t] - E_{(o,a,r)^T \sim D, \pi}[\sum_{t=1}^{T} r_t]$$

$$= \sum_{t'=1}^{T} E_{(o,a,r)^T \sim D,(\pi,\pi_{t'},\pi^*)}[\sum_{t=1}^{T} r_t] - E_{(o,a,r)^T \sim D,(\pi,\pi_{t'},\pi^*)}[\sum_{t=1}^{T} r_t]$$

$$= \sum_{t'=1}^{T} E_{(o,a,r)^T \sim D,(\pi,\pi^*,\pi^*)}[\sum_{t=t'}^{T} r_t] - E_{(o,a,r)^T \sim D,(\pi,h_{t'},\pi^*)}[\sum_{t=t'}^{T} r_t]$$

$$= \sum_{t=1}^{T} A_t^*(D,\pi,\pi_t^*) - A_t^*(D,\pi,h_t)$$
Ways to think about the result

1. Reduction of RL to classification given oracle access to optimal policy. (A form of “apprenticeship learning”.) [Surprisingly useful.]

2. A nonconstructive reduction of RL to classification. There exists a set of classification problems such that if they are solved well, we solve RL. (But which problem is unclear.)
What can we do constructively?

Given side information, we can subvert the exploration problem.

1. **Generative model (GM) (high exponential)**

2. Training time Acces to Optimal Policy + GM

3. State Distribution of (near) optimal policy + GM
Sparse Sampling

1. For every first action, sample $k$ times

2. For every result, for every second action, sample $k$ times.

3. For every result, for every third action, sample $k$ times.

4. etc.
Sparse Sampling, the picture

First Action
Second Action
Third Action
Etc...

Picture is difficult, very high sample complexity.
Sparse Sampling Decisions

1. For each action, expectation over samples at step $T$

2. Max over actions.

3. For each action, expectation over $T$ maxes at step $T - 1$.

4. Etc...

$\Rightarrow$ Estimate of $l_t(D, \pi, h_t)$ accurate to $(T - t)\sqrt{\frac{\log A + \log \frac{1}{\delta}}{k}}$.

$\Rightarrow$ can apply reduction
What can we do constructively?

Given side information, we can subvert the exploration problem.

1. Generative model (GM) (high exponential)

2. Training time Acces to Optimal Policy + GM

3. State Distribution of (near) optimal policy + GM
Training time: Acces to Optimal Policy + GM

Algorithm from before applies.

But: Learning new classifier for every timestep is computationally difficult.

Solution: Learn one classifier for all timesteps simultaneously.

But: This breaks good reduction properties.

Solution: Use “Conservative Policy Iteration” trick.
The Conservative Policy Iteration trick

We have policy $\pi_{i-1}$ which we use to define examples for $h$ using GM:

\begin{align*}
\text{Iteration} & = \pi_i \leftarrow (1 - \beta) \pi_{i-1} + \beta h \quad \text{(stochastic interpolation)}
\end{align*}

If you start with $\pi^*$, after $O\left(\frac{1}{\beta}\right)$ iterations, the policy is entirely learned.
The Searn Theorem

Theorem: For all RL problems $D$, for all sequences of learned classifiers $c_1, c_2, \ldots$, for small $\beta$, after $2T^3 \ln T$ iterations,

$$\text{Loss(learned policy)} \leq \text{Loss(Initial Policy)} + \text{Average classifier loss} \times 2T \ln T + (1 + \ln T)^{\max \text{ loss}}$$

Note: This is actually worse than $O(T)$ steps. However, in practice, line search over $\beta$ implies $< T$ classifiers learned.
## Searn Experimental Results

### Spanish Named Entity Recognition (1517 test)

<table>
<thead>
<tr>
<th></th>
<th>CRF(300)</th>
<th>SVM(300)</th>
<th>SVMISO(300)</th>
<th>Searn(300)</th>
<th>Searn(8324)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc.</td>
<td>94.83</td>
<td>94.94</td>
<td>94.9</td>
<td>95.01</td>
<td><strong>97.67</strong></td>
</tr>
</tbody>
</table>

### (Multidigit) Handwriting Recognition (5500 or 600 test)

<table>
<thead>
<tr>
<th>Training set size</th>
<th>Perceptron</th>
<th>Logistic</th>
<th>SVM</th>
<th>$M^3N$</th>
<th>Searn</th>
</tr>
</thead>
<tbody>
<tr>
<td>600 train</td>
<td>65.56</td>
<td>68.65</td>
<td>82.63</td>
<td>87.5</td>
<td><strong>88.2</strong></td>
</tr>
<tr>
<td>5500 train</td>
<td>70.05</td>
<td>72.10</td>
<td>82.52</td>
<td>?</td>
<td><strong>90.91</strong></td>
</tr>
</tbody>
</table>

### Joint sequence labeling (8936 train, 2012 test)

<table>
<thead>
<tr>
<th></th>
<th>Inc. P.</th>
<th>LASO</th>
<th>Factored CRF</th>
<th>CRF</th>
<th>Searn</th>
</tr>
</thead>
<tbody>
<tr>
<td>Joint Accuracy</td>
<td>93.12</td>
<td>95.12</td>
<td>96.48</td>
<td>?</td>
<td><strong>96.81</strong></td>
</tr>
<tr>
<td>Chunk F-score</td>
<td>92.44</td>
<td>93.49</td>
<td>93.87</td>
<td><strong>94.77</strong></td>
<td>94.36</td>
</tr>
<tr>
<td>Parts of speech</td>
<td>98.05</td>
<td>98.90</td>
<td>98.92</td>
<td>?</td>
<td><strong>99.07</strong></td>
</tr>
</tbody>
</table>

### Document Summarization (fraction of max “Rouge-1+2”)

<table>
<thead>
<tr>
<th>word count</th>
<th>DUC 05 winner</th>
<th>BD03</th>
<th>BD05</th>
<th>Searn</th>
<th>Searn (word)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>?</td>
<td>0.803</td>
<td>0.900</td>
<td>0.971</td>
<td><strong>1.129</strong></td>
</tr>
<tr>
<td>250</td>
<td>0.749</td>
<td>0.696</td>
<td>0.883</td>
<td>0.906</td>
<td><strong>1.170</strong></td>
</tr>
</tbody>
</table>
What can we do constructively?

Given side information, we can subvert the exploration problem.

1. Generative model (GM) (high exponential)

2. Training time Acces to Optimal Policy + GM

3. State Distribution of (near) optimal policy + GM
State Distribution of (near) optimal policy + GM

What if (near) optimal policy is missing?

Sometimes we only know where an optimal policy visits. Is this enough?
The PSDP (“Policy Search by Dynamic Programming”) algorithm

Let $D_1, \ldots, D_T = \text{distribution over states at timestep } t$ induced by $\pi^*$. 

1. Repeatedly draw $s$ from $D_T$ and use GM to estimate rewards. Learn a $T$th step policy $\pi_T$ via cost sensitive classification.

2. Repeatedly draw $s$ from $D_{T-1}$ and use GM + $\pi_T$ to estimate rewards. Learn $(T - 1)$th step policy $\pi_{T-1}$ via cost sensitive classification.

3. etc...
The PSDP Theorem

\[ A_t^\pi(D_t, h_t) = E_{(a,o,r)T} D_t, (h_t, \pi) \left[ \sum_{t'=t}^{T} r_{t'} \right] \]

(Like \( A_t^\pi \) except we start by assuming optimal policy and end with learned policy.)

Theorem: For all \( \pi^* \) generating \( D_1, ..., D_T \), and all \( \pi = (h_1, ..., h_T) \),

\[ \eta(\pi^*) - \eta(\pi) = \sum_{t=1}^{T} A_t(D_t, \pi_t^*) - A_t(D_t, h_t) \]

= sum of cost sensitive losses/rewards.

- The same as given \( \pi^* + \text{GM} \) but Searn-style variant unknown.
The Future

1. Specializations of Reinforcement Learning to particular domains.
   (a) Robotics
   (b) Games
   (c) Stock Market

2. Exploration: the elephant in the closet.

3. New models and new methods of analysis needed and welcome.
Related Reading


Extra Discussion on many things, including RL, Reductions, etc...

http://hunch.net