Practical kernel-based reinforcement learning (and other stories)

Joelle Pineau

*Reasoning and Learning Lab / Center for Intelligent Machines*
School of Computer Science, McGill University

Joint work with: Andre Barreto, Mahdi Milani Fard, William Hamilton, Doina Precup

Workshop on Big Data and Statistical Machine Learning
Fields Institute
January 29, 2015
1. Learning agent tries a sequence of actions \((a_t)\).

2. Observes outcomes (state \(s_{t+1}\), rewards \(r_t\)) of those actions.

3. Statistically estimates relationship between action choice and outcomes, \(Pr(s_t|s_{t-1}, a_t)\).

After some time... learn action selection policy, \(\pi(s)\), that optimizes selected outcomes.

\[
\arg\max_{\pi} E_{\pi} [ r_0 + r_1 + \ldots + r_T ]
\]

[Bellman, 1957; Sutton, 1988; Sutton&Barto, 1998.]
Recent advances in Reinforcement Learning

Recent applications of RL

• Robotics
• Medicine
• Advertisement
• Resource management
• Game playing …
Goal: To create an **adaptive** neuro-stimulation system that can maximally reduce the incidence of epileptiform activity.
Adaptive deep-brain stimulation

Closed-loop, non-periodic, stimulation strategy learned for an in vitro model of epilepsy.
Speech-based control of smart wheelchair

<table>
<thead>
<tr>
<th>User command</th>
<th>Speech recognition output</th>
<th>Robot action</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Turn right.&quot;</td>
<td>turn right</td>
<td>Turns right 90 deg.</td>
</tr>
<tr>
<td>&quot;Stop.&quot;</td>
<td>this</td>
<td>Please repeat your command.</td>
</tr>
<tr>
<td>&quot;Turn left a little.&quot;</td>
<td>turn left little</td>
<td>Turn left 22.5 deg.</td>
</tr>
<tr>
<td>&quot;Drive forward.&quot;</td>
<td>drive forward</td>
<td>Drive forward (max. 100m).</td>
</tr>
<tr>
<td>&quot;Follow right wall.&quot;</td>
<td>right wall</td>
<td>Drive along the right wall (distance to wall = 0.65m).</td>
</tr>
<tr>
<td>&quot;Turn around.&quot;</td>
<td>turn around</td>
<td>Turn to the left 180 deg.</td>
</tr>
<tr>
<td>&quot;Follow left wall.&quot;</td>
<td>left all</td>
<td>Please repeat your command.</td>
</tr>
<tr>
<td>&quot;Follow left wall.&quot;</td>
<td>fall left wall</td>
<td>Drive along the left wall (distance to wall = 0.65m).</td>
</tr>
<tr>
<td>&quot;Stop.&quot;</td>
<td>stop</td>
<td>Stop</td>
</tr>
</tbody>
</table>

Recent advances in Reinforcement Learning
**RL vs supervised learning**

**Training signal = desired (target outputs), e.g. class**

**Inputs** → Supervised Learning → **Outputs**

**Learning from i.i.d samples**

**Training signal = “rewards”**

**Inputs** → Reinforcement Learning → **Environment** → Outputs ("actions")

**Jointly learning AND planning from correlated samples**
Learning problem

Learn the Q function, defining state-action values of the policy $\pi$:

$$Q^\pi(s,a) = r(s,a) + \gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^\pi(s',a')$$

Immediate reward

Future expected sum of rewards

Error to adjust the Q-function, given a sample $<s,a,r,s',a'>$:

$$\delta = Q(s,a) - (r + \gamma \max_{a'} Q(s',a'))$$
In large state spaces: Need approximation

\[ \hat{Q}^\pi(s, a) = \sum_{i=1}^{d} \theta_i \phi_i(s, a) \]

Challenge: finding good features
Batch reinforcement learning

Use **regression analysis** to estimate the long-term cost of different actions from the training data.

Regression with linear function, kernel function, random forests, neural networks, …

Important!
Target function, $Q$, is **sum of future expected rewards**.
Algorithms for large-scale RL

- Algorithms with **sample complexity** guarantees
  \[\text{Fard et al., NIPS'13, AAAI'12, UAI'11, NIPS'10}\]

- Algorithms with **convergence** guarantees
  \[\text{Barreto et al., JAIR'14, NIPS'12, NIPS'11}\]

# features

# examples

---

Recent advances in Reinforcement Learning

Joelle Pineau
Kernel-based RL (Ormoneit & Sen, 2002)

Sample transitions associated with action $a$
Kernel-based RL (Ormoneit & Sen, 2002)

\[ \hat{p}^a(\hat{s}|s) := \kappa(s, s) = \frac{k_\tau(s, s)}{\sum_s k_\tau(s, s)}, \]

with \[ k_\tau(s, s) = \phi(||s - s||/\tau) \]
Kernel-based RL (Ormoneit & Sen, 2002)

\[
\hat{Q}(s, a) = \sum_{i=1}^{n_a} \kappa(s, s_i) \left[ r_i + \gamma \hat{V}^*(\hat{s}_i) \right],
\]
Kernel-based RL (Ormoneit & Sen, 2002)

- KBRL has good theoretical guarantees:
  - Converges to a solution.
  - Consistent, in the statistical sense.
    » Additional data improves quality of approximation.
  - If kernel widths are decreased at an appropriate rate, the probability of selecting suboptimal actions converges to zero.

**Problem:** One application of the Q-function update is $O(n^2)$. 
Stochastic factorization trick

Stochastic matrix

- \( p_{ij} \geq 0 \)
- \( \sum_j p_{ij} = 1 \)
Stochastic factorization trick

Stochastic factorization

\[ \text{D} \quad \text{K} \quad = \quad \text{P} \]
Stochastic-factorization trick

\[ K \times D = \bar{P} \]
Kernel-based stochastic factorization

KBRL

Stochastic-factorization trick

KBSF
Kernel-based stochastic factorization

Select a set of representative states.
Define $D \in \mathbb{R}^{n \times m}$, $K \in \mathbb{R}^{m \times n}$
KBSF algorithm

1. For each $a \in \mathcal{A}$ do
   1. Compute matrix $\mathbf{D}^a$: $d^a_{ij} = \kappa(s^a_i, \bar{s}_j)$
   2. Compute matrix $\mathbf{K}^a$: $k^a_{ij} = \kappa(\bar{s}_i, \hat{s}_j)$
   3. Compute $\bar{r}^a$: $\bar{r}^a_i = \sum_j k^a_{ij} \bar{r}^a_j$
   4. $\bar{\mathbf{P}}^a = \mathbf{K}^a \mathbf{D}^a$

2. Solve the MDP described by $\bar{\mathbf{P}}^a$ and $\bar{r}^a$
### Computational requirements

<table>
<thead>
<tr>
<th></th>
<th>KBRL</th>
<th>KBSF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model generation</td>
<td>$O(n^2</td>
<td>A</td>
</tr>
<tr>
<td>Policy evaluation</td>
<td>$O(n^3)$</td>
<td>$O(m^3)$</td>
</tr>
<tr>
<td>Policy improvement</td>
<td>$O(n^2</td>
<td>A</td>
</tr>
</tbody>
</table>

$n = \text{number of sample transitions}$  
$m = \text{number of representative states}$
Advantages of KBSF

• It always converges to a solution.

• It has good theoretical guarantees.

• It is fast.

• It is simple.

• It has a predictive behavior, generally improving its performance as $m \to n$ or $n \to \infty$. 
Epilepsy suppression in silico

Recent advances in Reinforcement Learning

Joelle Pineau
Online / Parallel KBSF

Incremental KBSF
Incremental KBSF

Helicopter task (from the 2013 RL competition)

2013 RL competition results for KBSF: 1st for the polyathlon task; 2nd for the helicopter task (best value-based, online learning method).

High-dimensional input data?
Learning representations for RL

Original state \( s \)

\[ Q(\phi(s), a) \]

\[ \phi(s)? \]
Partially Observable MDP

- POMDP defined by n-tuple \(<S, A, Z, T, O, R>\),
  where \(<S, A, T, R>\) are same as in an MDP.
- States are hidden \(\Rightarrow\) Observations (Z)
- Observation function \(O(s,a,z) := \Pr(z | s, a)\)
POMDP planning in high-dimensions


http://jveness.info/publications/default.html
Learning representations for RL

[Hamilton, Fard & Pineau, 2013]

\[ Q(\phi(s), a) \]

Original state \( \rightarrow \phi(s) \rightarrow \) Random projection \( \rightarrow Q(\phi(s), a) \)
Latent-state approaches to learning

- Expectation maximization
  - HMM learning (Rabiner, 1990)
  - Online nested EM (Liu, Liao & Carin, 2013)
  - Model-free RL as mixture learning (Vlassis & Toussaint, 2009)

- Bayesian learning
  - Kalman filtering
  - Bayes-Adaptive POMDP (Ross, Chaib-draa & Pineau, 2007)
  - Infinite POMDP (Doshi-Velez, 2009)
Event-based approaches to learning

• An alternative is to work directly with observable events.
  
  o Merge-split histories
    » U-Tree (McCallum, 1996)
    » MC-AIXI (Veness, Ng, Hutter, Uther & Silver, 2011)
  
  o Predictive state representations
    » Observable operator models (Jaeger, 2000)
    » PSRs (Littman, Sutton, Singh, 2002)
  
  o Methods of moments
    » Spectral HMMs (Hsu et al., 2008)
    » TPSRs (Boots & Gordon, 2010)
    » Tensor decomposition (Anandkumar, 2012)

Estimate probabilities of the form \( p(o_{t:t+k} | o_{1:t-1}) \) and represent compactly.
The PSR systems dynamics matrix

\[ D \]

\[
\begin{array}{cccc}
\tau_1 = o^1 \\
\tau_2 = o^2 \\
\vdots \\
\tau_j = o^1 o^2 \\
\tau_{j+1} = o^2 o^1 \\
\vdots \\
P(\tau_j | h_i) \quad \text{history (i.e. previous observations)}
\end{array}
\]

\[
\begin{array}{cccc}
h_1 = o^1 \\
h_2 = o^2 \\
\vdots \\
h_i = o^1 o^2 \\
h_{i+1} = o^2 o^1 \\
\vdots \\
\end{array}
\]

\[ \text{test (i.e. possible future)} \]
Learning PSRs from data

- **Spectral approach** [Rosencrantz et al., 2004; Boots et al., 2009]:
  - Estimate large sub-matrix of $D$.
  - Project to low-dimensions subspace using SVD.
  - Globally optimal, if you know $\text{rank}(D)$.

- Still computationally expensive, $O(|T|^2|H|)$. 
Regularity in large observation spaces

- We say that $D$ is $k$-sparse if only $k$ tests are possible for any given history $h_i$.

- Valid assumption? In Poc-Man domain, large sub-matrix estimates of $D$ empirically observed to have an average of 99.9% column sparsity.
Exploiting sparsity

- Sparse structure can be exploited using random projections.

Compress $m \times n$ matrix $Y$ to a $d \times n$ matrix $X$, where $d << m$.

Let $X = \phi Y$,

where $\phi$ is a $d \times m$ projection matrix with entries from $N \sim (0, 1/d)$. 
The CPSR Algorithm

**Algorithm**

- Obtain compressed estimates for sub-matrices of $D$, $\Phi P_T H$, $\Phi P_{T,o,H}$, and $P_H$ by sampling time series data.
  - Estimate $\Phi P_T H$ in compressed space by adding $\phi_i$ to column $j$ each time $t_i$ observed after $h_i$ (Likewise for $\Phi P_{T,o,H}$).
- Compute CPSR model:
  - $c_0 = \Phi \hat{P}(T|\emptyset)$
  - $C_o = \Phi P_{T,o,H}(\Phi P_T H)^+$
  - $c_\infty = (\Phi P_T H)^+ \hat{P}_H$

**Note:** Multiplication done “online” and $D$ matrix never held in memory.
Using the compact representation

State definition and necessary equations

- $c_0$ serves as initial prediction vector (i.e. state vector).
- Update state vector after seeing observation with
  - $c_{t+1} = \frac{c_c c_t}{c_c c_c}$
- Predict k-steps into the future using
  - $P(o^j_{t+k} | h_t) = b_c C_o^j (C_*)^{k-1} C_t$ where $C_* = \sum_{o_i \in O} C_{o_i}$. 

Efficiently Modeling Sparse Dynamical Systems

William Hamilton, Mahdi Fard, Joelle Pineau
Properties of CPSR

• Computationally efficient (projection has low cost).
  • A projection size of $d = O(k \log |Q|)$ suffices in most systems.

• Allows projection to subspaces of dimension $d < \text{rank}(D)$.
  • If $d \geq \text{rank}(D)$ then model is trivially consistent.

• Regularizes the solution (i.e. the learned model parameters).

• Total propagated error for $T$ steps is bounded.
Poc-Man prediction error

Partially observable variant of Pac-Man video-game with $|S| = 10^56$ and $|O| = 2^{10}$ [Silver and Veness, 2010].

![Pac-Man game screen](image)

![Prediction error graph](image)
Planning in CPSRs

**Point-based value iteration** (traditional POMDP approach):

CPSR state vector = POMDP belief state

**Point-based value iteration** (common MDP approach):

CPSR state vector = input to value function approximation

Selecting belief points

- Earlier strategies:
  - Fully observable states
  - Points at regular intervals
  - Random samples

- Can we use theoretical properties to pick important beliefs?
Poc-Man results

Observation space too large for building TPSR model!
Poc-Man results

- **S-PocMan**: Drop parts of the observation vector that allow agent to sense in what direction food lies.
Adaptive migratory bird management

**IJCAI 2013 Challenge problem** (Nicol et al. 2013):
Observe bird population level at different breeding sites over time + protection level on intertidal sites on their migratory path.

---

**Figure 8:** Average discounted reward per episode (i.e., average return per episode) achieved in the AMM domain using different methods over 100000 test episodes (each of length 50). The numbers beside the CPSR method names denote the projected dimension size. 95% confidence intervals are too small to be visible.

- **Hashed**
- **Memoryless**
- **Rademacher**
- **Random**
- **Spherical**

Importantly, in S-Pocman where part of the observation vector is dropped and the rewards are sparsified, we see that the top-performer is again a CPSR based model (which in this case uses spherical projections). This matches expectations since the food-rewards are no longer fully discernible from the observation vector, and thus the domain is significantly less observable. It is also worth noting that building naive TPSRs (without compression or domain-specific feature selection) is infeasible computationally in these PacMan-inspired domains, and thus the use of a PSR-based reinforcement learning agent (via the compression techniques used) in these domains is a considerable advancement.

A final observation is that the performance is quite sensitive to the choice of projection matrices in these results. For example, in the S-PocMan domain, the Rademacher projections perform no better than the memoryless baseline, whereas for PocMan the Rademacher outperforms the other projection methods. The exact cause of this performance change is unclear. Nevertheless, this highlights the importance of evaluating different projection techniques when applying this algorithm in practice.
Learning (more!) representations for RL

$Q(\phi(s), a)$

$\phi(s)$?

Original state
Deep Q-learning

\[(s, a) \rightarrow F \rightarrow \hat{Q}(s, a)\]

convolutional neural network
Deep Q-learning

Reinforcement learning + deep learning in the Arcade Learning Environment
[Mnih et al., 2013; Guo et al., 2014].

Many interesting open questions:
• Efficient exploration
• Temporally extended actions
• Transfer learning between games
• …

Figure 1: Screen shots from five Atari 2600 Games: (Left-to-right) Pong, Breakout, Space Invaders, Seaquest, Beam Rider
Research team @ McGill

Funding: Natural Sciences and Engineering Research Council Canada (NSERC), Canadian Institutes for Health Research (CIHR), National Institutes of Health (NIH), Fonds Québécois de Recherche Nature et Technologie (FQRNT)