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

Reinforcement learning

1. Learning agent tries a sequence of actions (a_t).

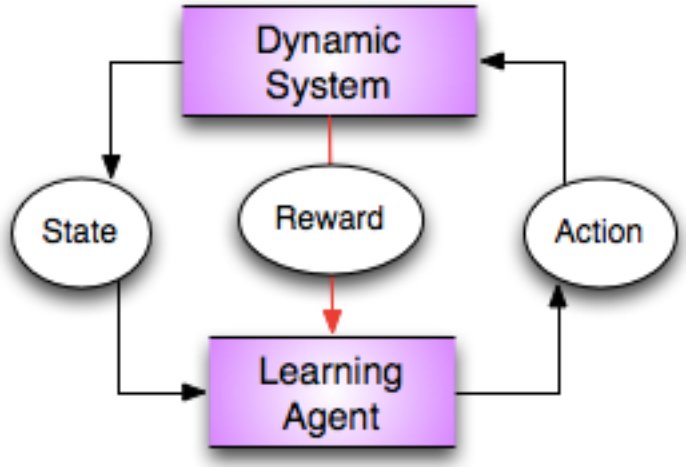
2. Observes outcomes (state s_{t+1}, rewards r_t) of those actions.

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.

$$argmax_{\pi}\, E_{\pi}\, [\, r_0 + r_1 + \ldots + r_T \,]$$

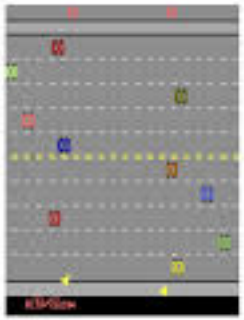
[Bellman, 1957; Sutton, 1988; Sutton&Barto, 1998.]

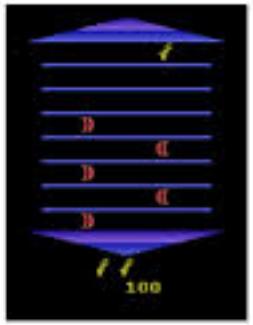


http://en.wikipedia.org/wiki/Animal_training

Recent applications of RL

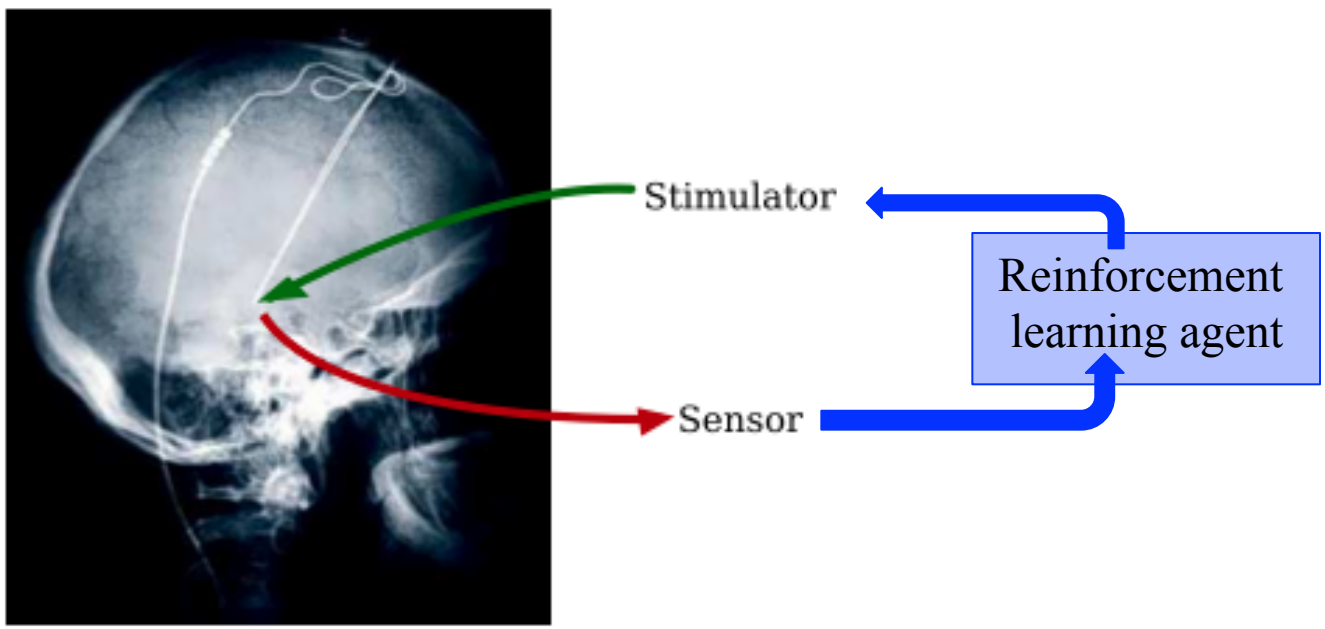
- Robotics
- Medicine
- Advertisement
- Resource management
- Game playing …





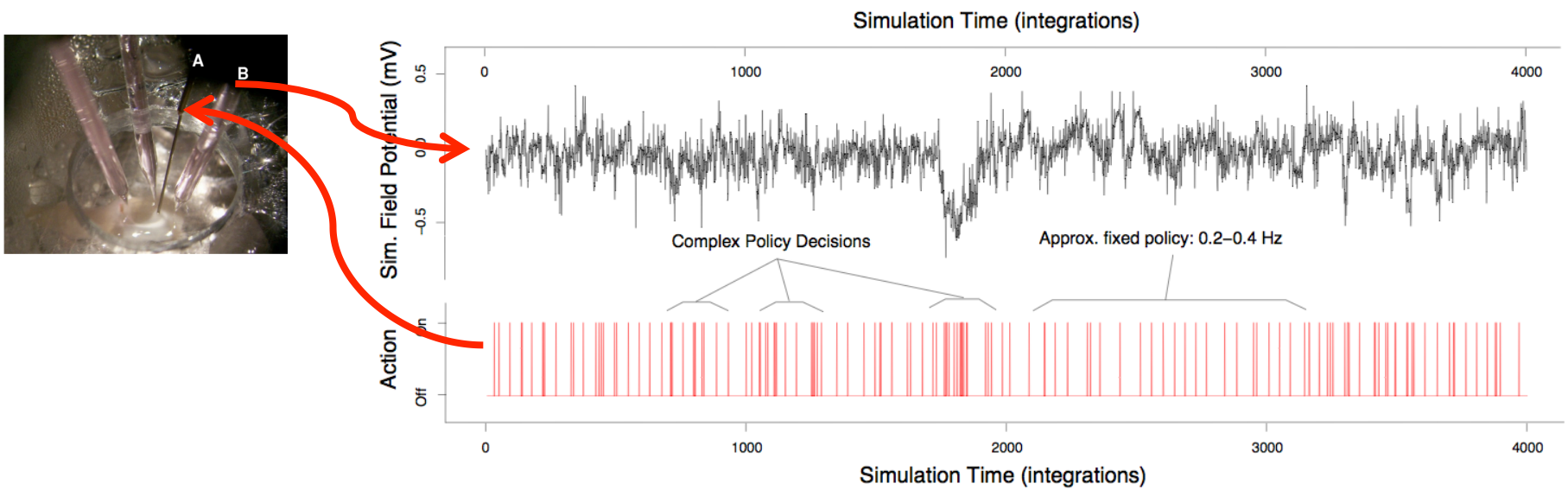
Adaptive deep-brain stimulation

Goal: To create an **adaptive** neuro-stimulation system that can maximally reduce the incidence of epileptiform activity.



Adaptive deep-brain stimulation

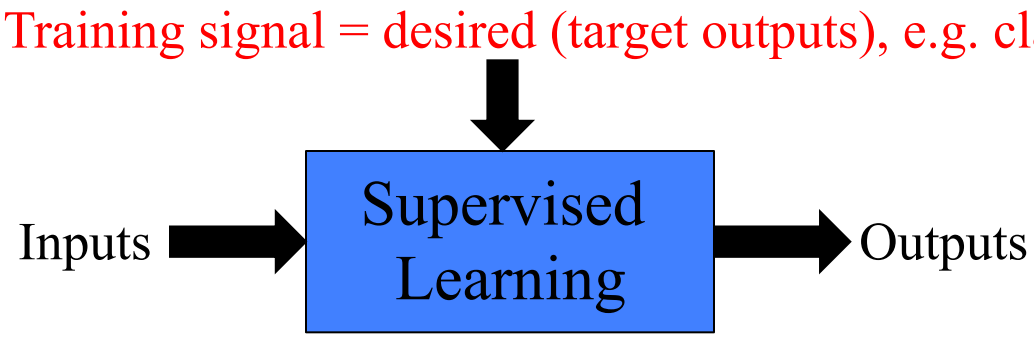
Closed-loop, non-periodic, stimulation strategy learned for an in vitro model of epilepsy.



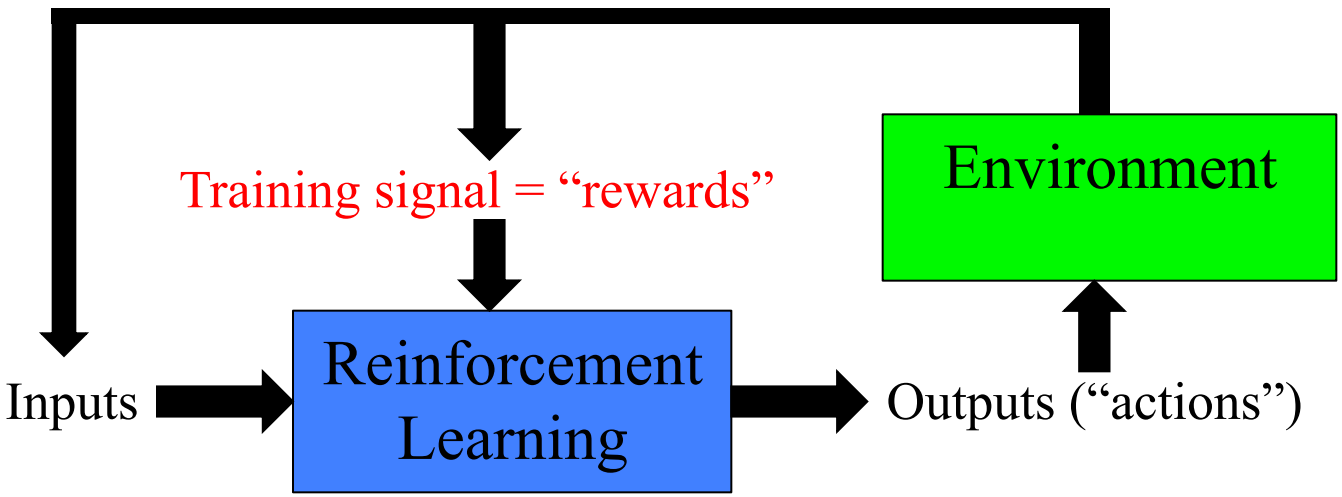
Speech-based control of smart wheelchair

User command	Speech recognition output	Robot action
"Turn right."	`turn right`	Turns right 90 deg.
"Stop."	`this`	*Please repeat your command.*
"Turn left a little."	`turn left little`	Turn left 22.5 deg.
"Drive forward."	`drive forward`	Drive forward (max. 100m).
"Follow right wall."	`right wall`	Drive along the right wall (distance to wall = 0.65m).
"Turn around."	`turn around`	Turn to the left 180 deg.
"Follow left wall."	`left all`	*Please repeat your command.*
"Follow left wall."	`fall left wall`	Drive along the left wall (distance to wall = 0.65m).
"Stop."	`stop`	Stop

RL vs supervised learning



Learning from i.i.d samples



Jointly learning AND planning from correlated samples

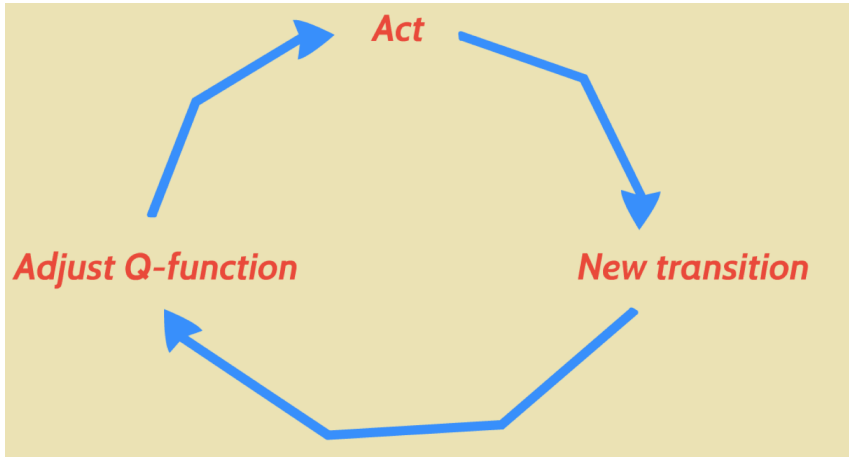
Learning problem

Learn the Q function, defining state-action values of the policy π :

$$Q^{\pi}(s,a) = \underbrace{r(s,a)}_{\text{Immediate reward}} + \underbrace{\gamma \sum_{s'} P(s'|s,a) \max_{a'} Q^{\pi}(s',a')}_{\text{Future expected sum of rewards}}$$

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

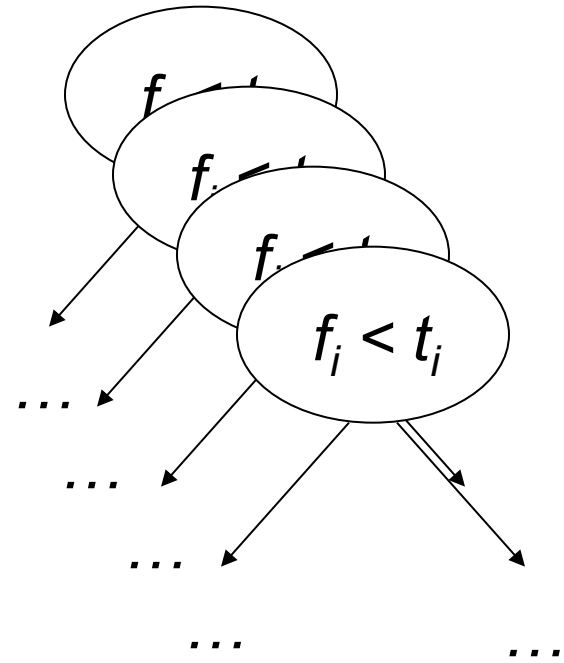
Challenge: finding good features

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

feature vector

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, …..

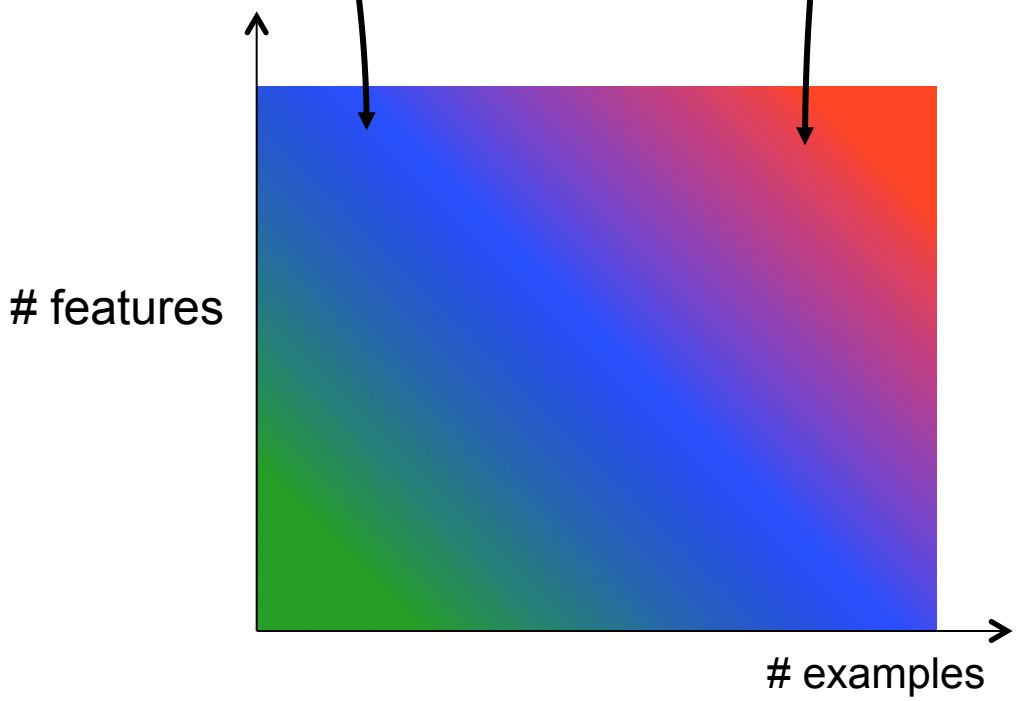
Important!

Target function, Q, is **<u>sum of future expected rewards</u>**.

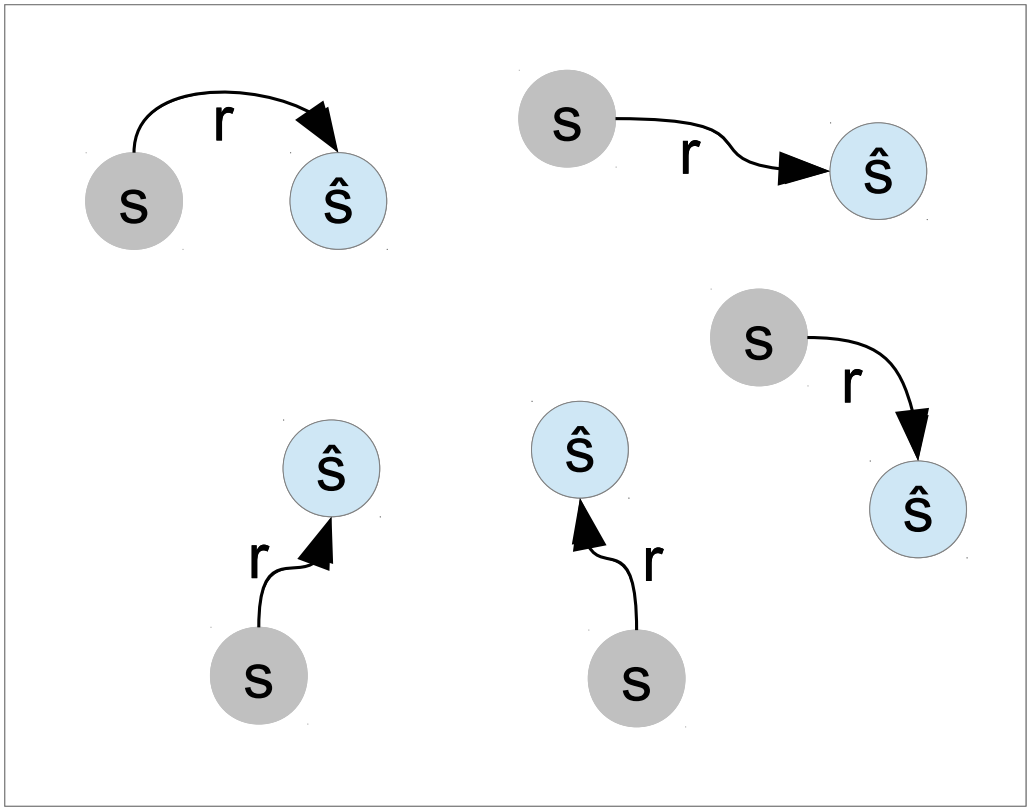
Algorithms for large-scale RL

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

Algorithms with **convergence** guarantees
[Barreto et al., JAIR'14, NIPS'12, NIPS'11]

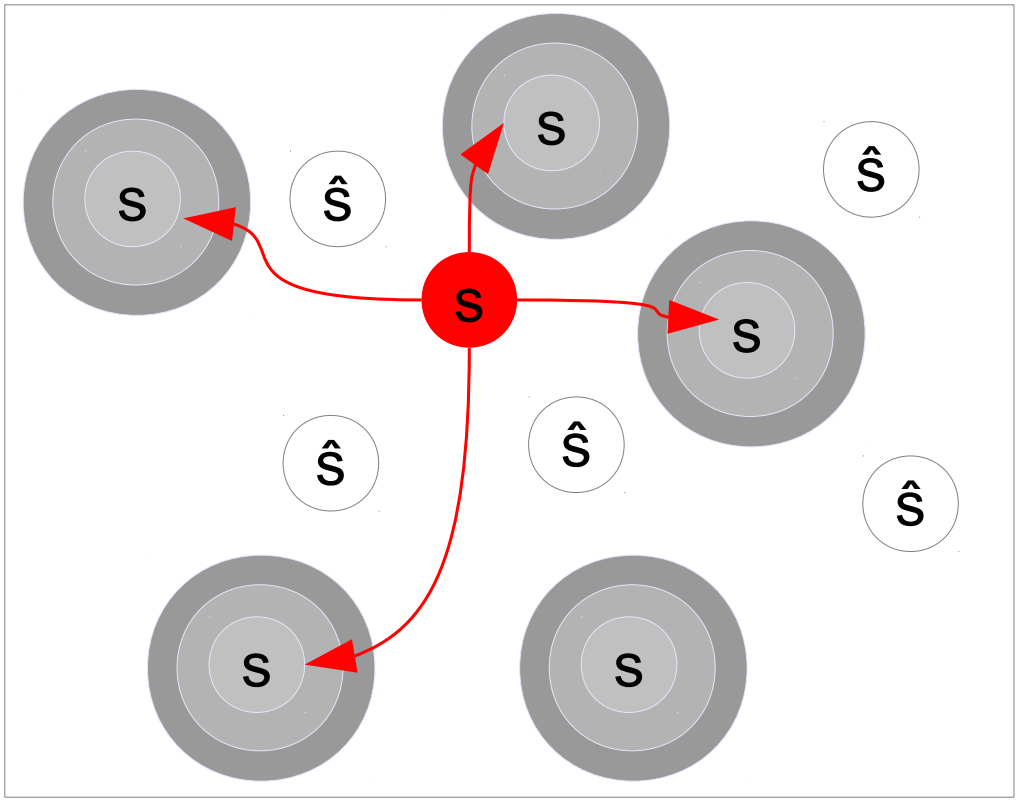


Kernel-based RL (Ormoneit & Sen, 2002)



Sample transitions associated with action a

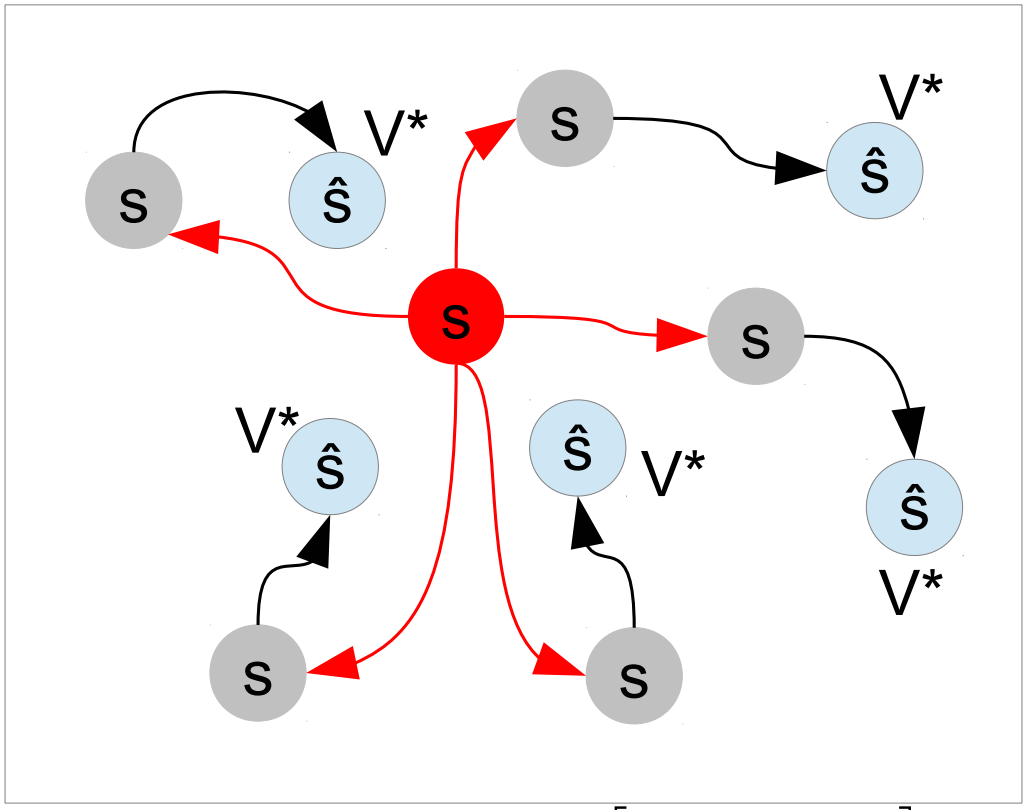
Kernel-based RL (Ormoneit & Sen, 2002)



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

with $k_\tau(s, s) = \phi(||s - s||/\tau)$

Kernel-based RL (Ormoneit & Sen, 2002)



$$\hat{Q}(s,a) = \sum_{i=1}^{n_a} \kappa(s, s_i) \left[r_i + \gamma \hat{V}^*(\hat{s}_i) \right],$$

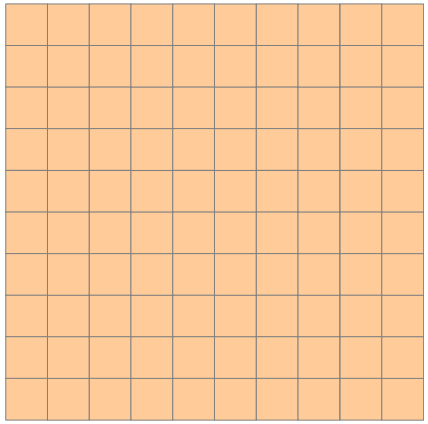
Kernel-based RL (Ormoneit & Sen, 2002)

- 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

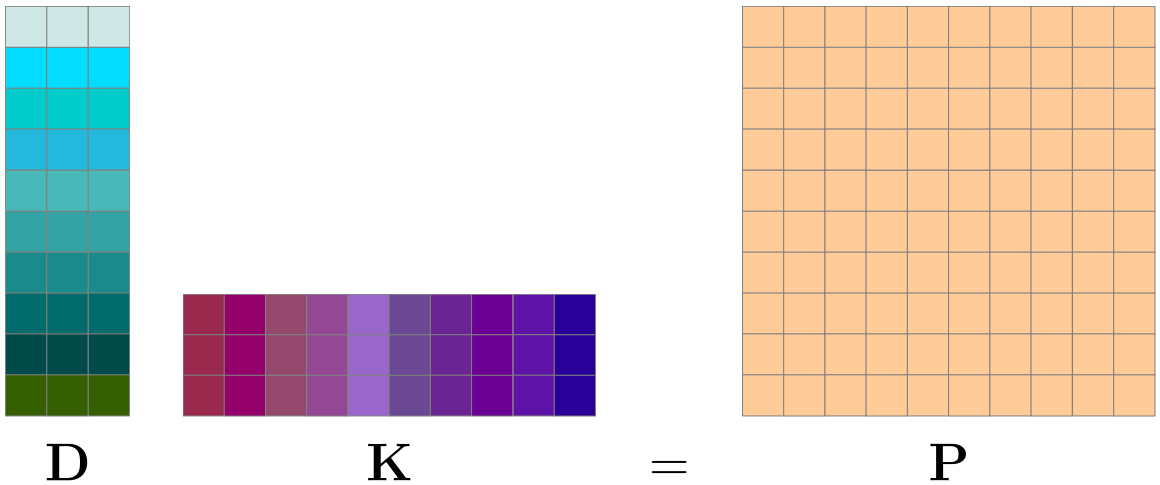


$\mathbf{P}$

- $p_{ij} \geq 0$
- $\sum_j p_{ij} = 1$

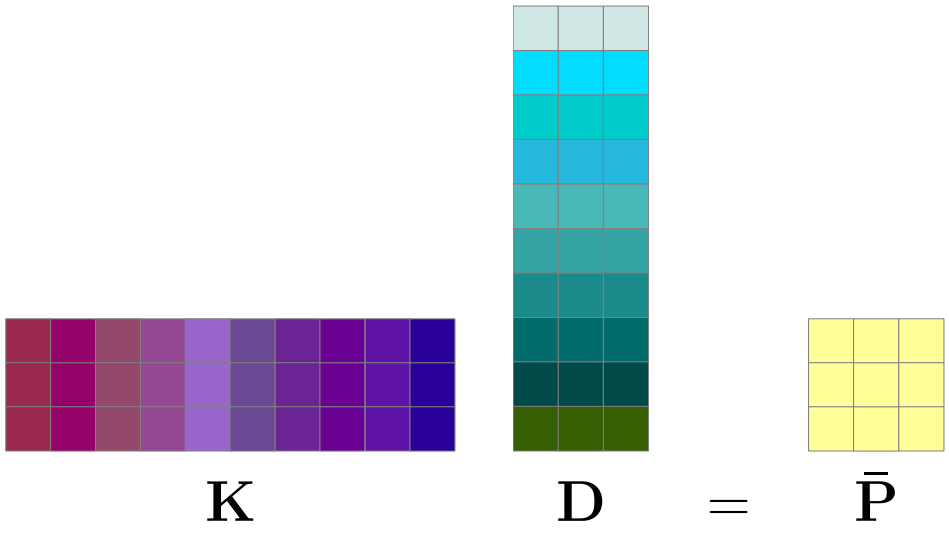
Stochastic factorization trick

Stochastic factorization

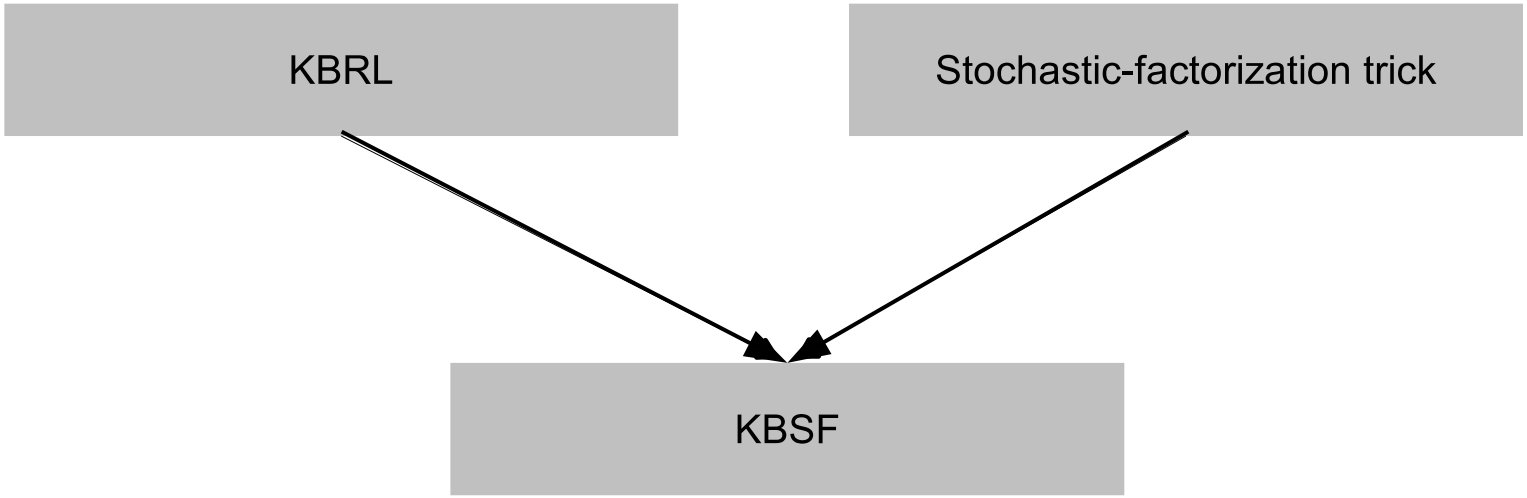


Stochastic factorization trick

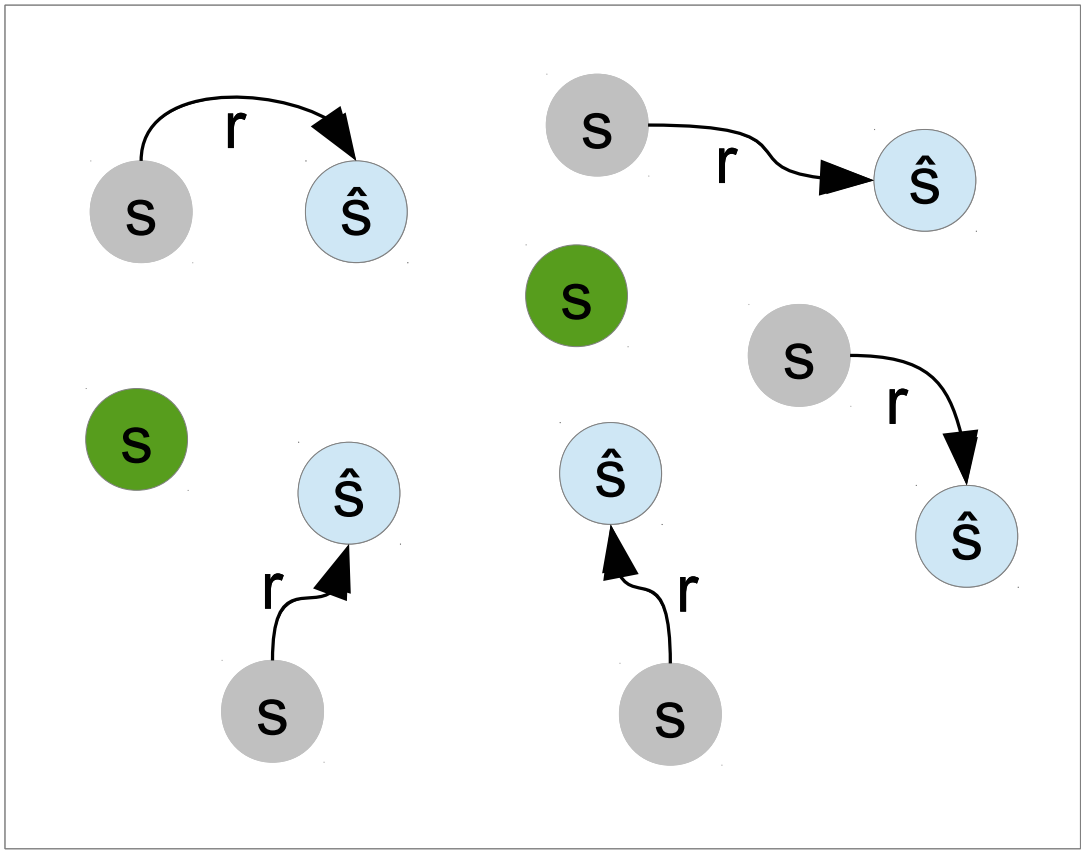
Stochastic-factorization trick



Kernel-based stochastic factorization



Kernel-based stochastic factorization



Select a set of representative states.
Define $\mathbf{D} \in \Re^{nxm}$, $\mathbf{K} \in \Re^{mxn}$

KBSF algorithm

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

Properties

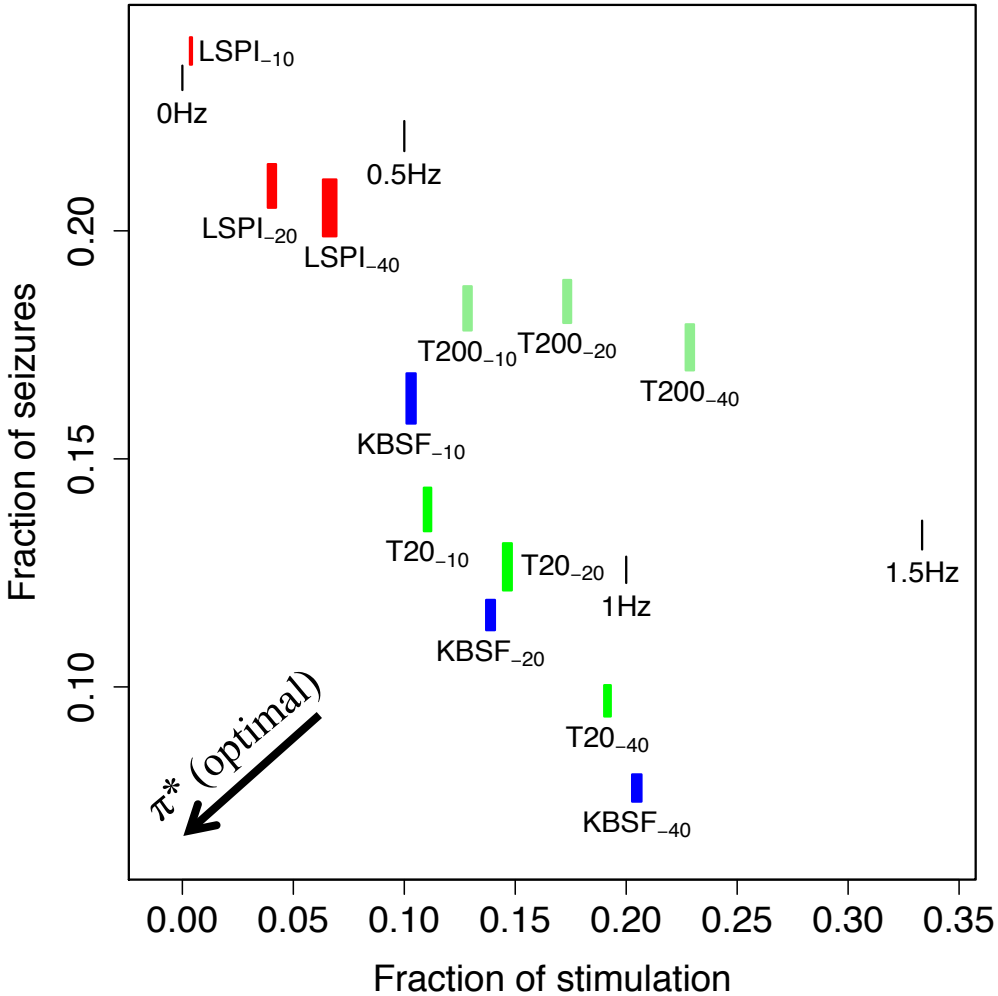
Computational requirements		
	KBRL	KBSF
Model generation	$O(n^2\|A\|)$	$O(nm^2\|A\|)$
Policy evaluation	$O(n^3)$	$O(m^3)$
Policy improvement	$O(n^2\|A\|)$	$O(m^2\|A\|)$

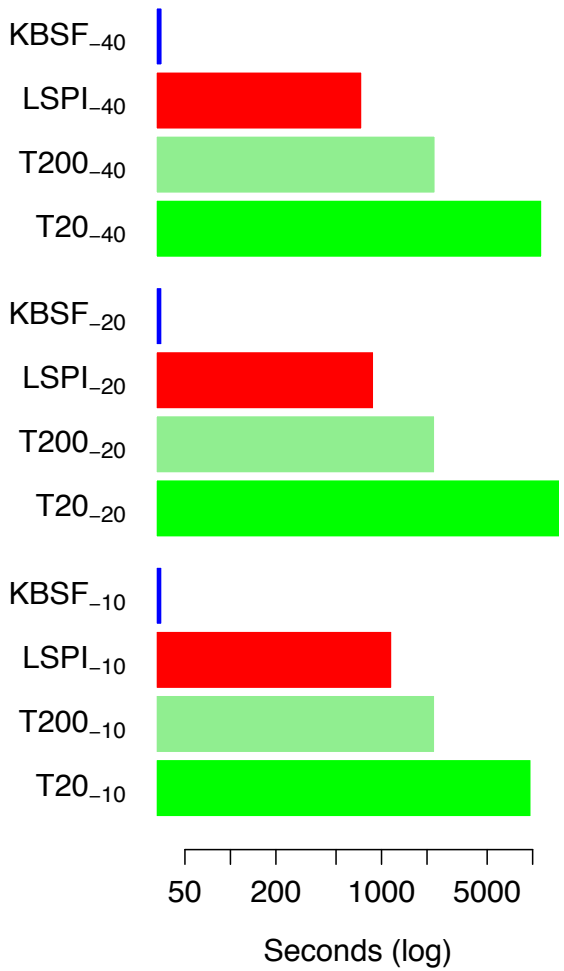
$n =$ number of sample transitions
$m =$ number of representative states

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 $m \rightarrow n$ or $n \rightarrow \infty$.

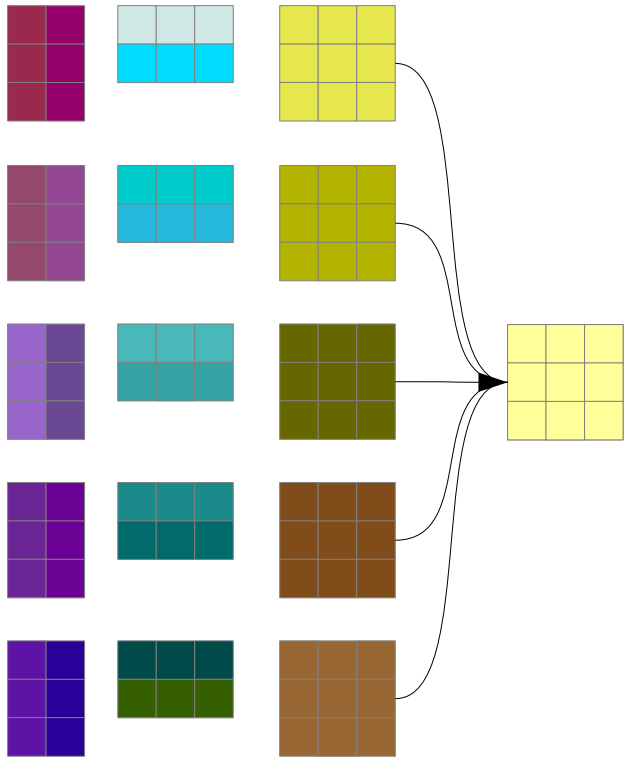
Epilepsy suppression in silico





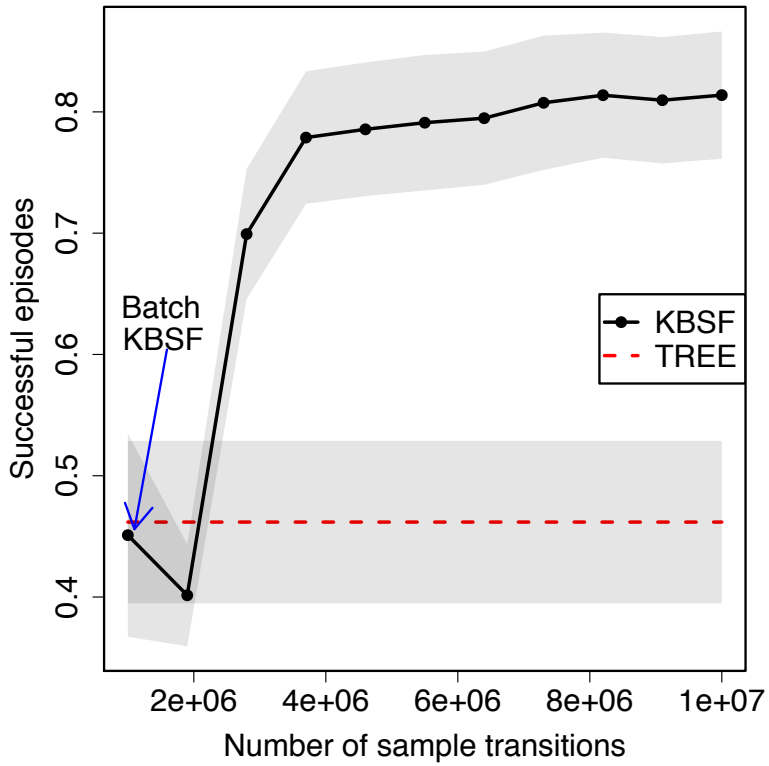
Online / Parallel KBSF

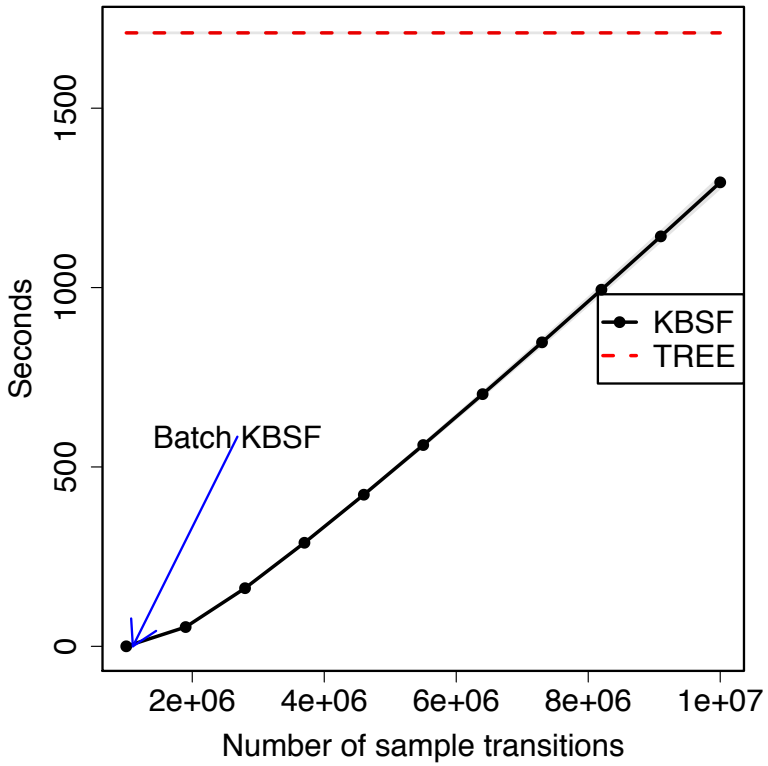
Incremental KBSF



Incremental KBSF

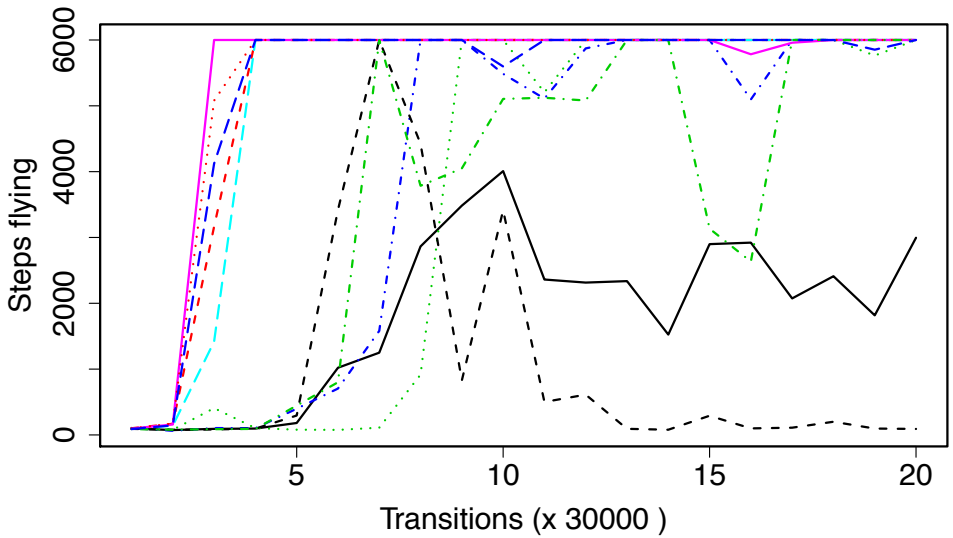
Triple Pole-Balancing





Incremental KBSF

Helicopter task (from the 2013 RL competition)

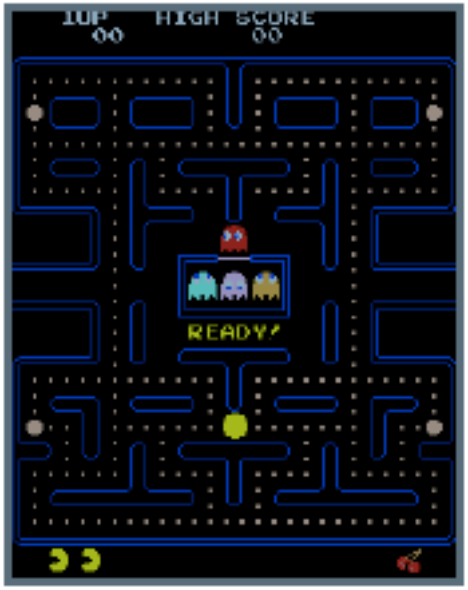


2013 RL competition results for KBSF: 1st for the polyathlon task; 2nd for the helicopter task (best value-based, online learning method).

High-dimensional input data?

Learning representations for RL

s



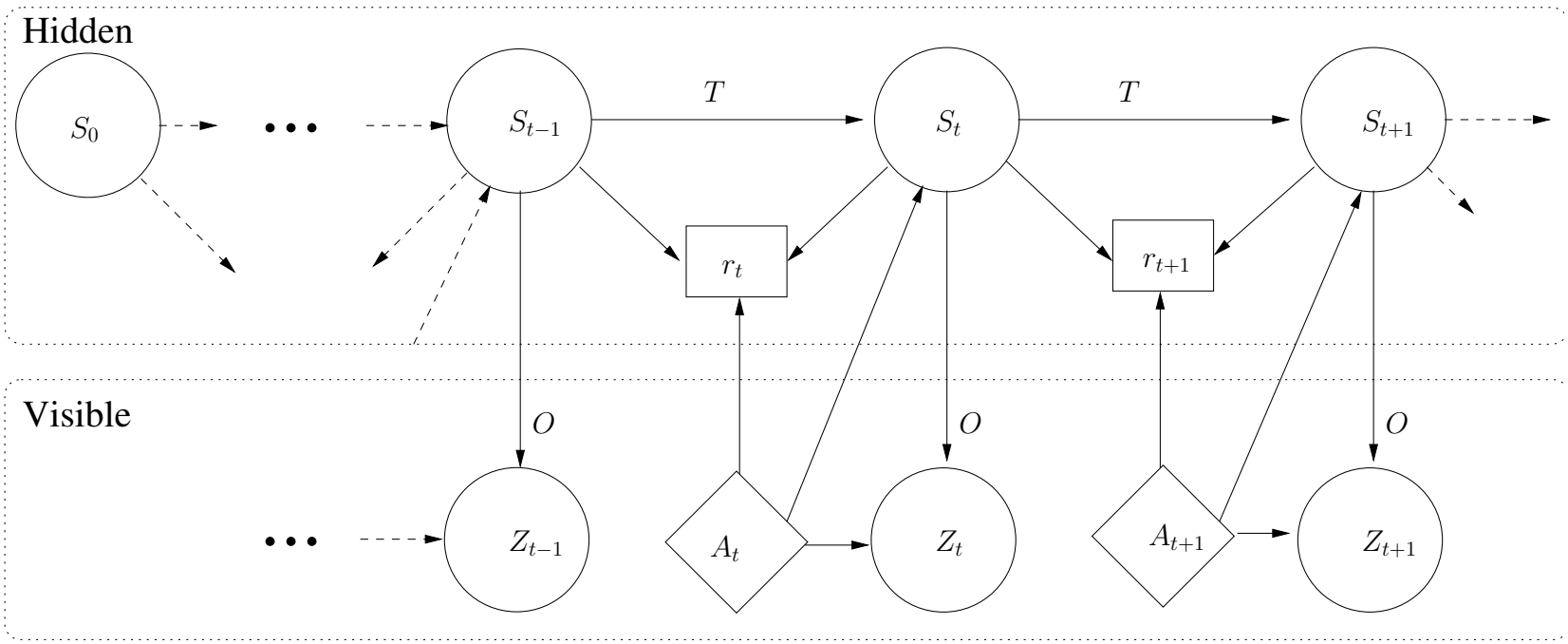
Original state

$\phi(s)?$

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

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 ➔ Observations (Z)
- Observation function O(s,a,z) := Pr(z | s, a)



POMDP planning in high-dimensions

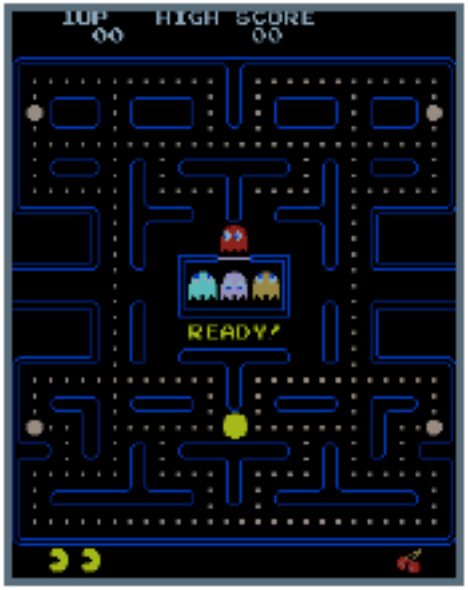
- Poc-Man problem: $|S|=10^{56}$, $|A|=4$, $|Z|=1024$.

http://jveness.info/publications/default.html

Learning representations for RL

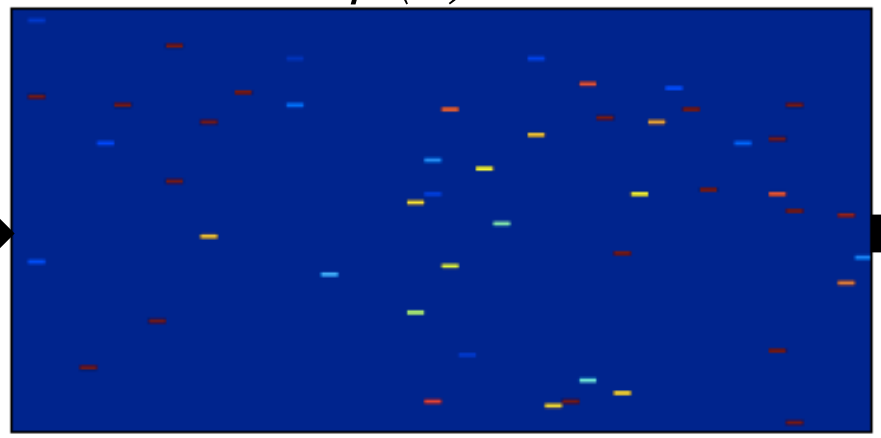
[Hamilton, Fard & Pineau, 2013]

s



Original state

$\phi(s)$



Random projection

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

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)

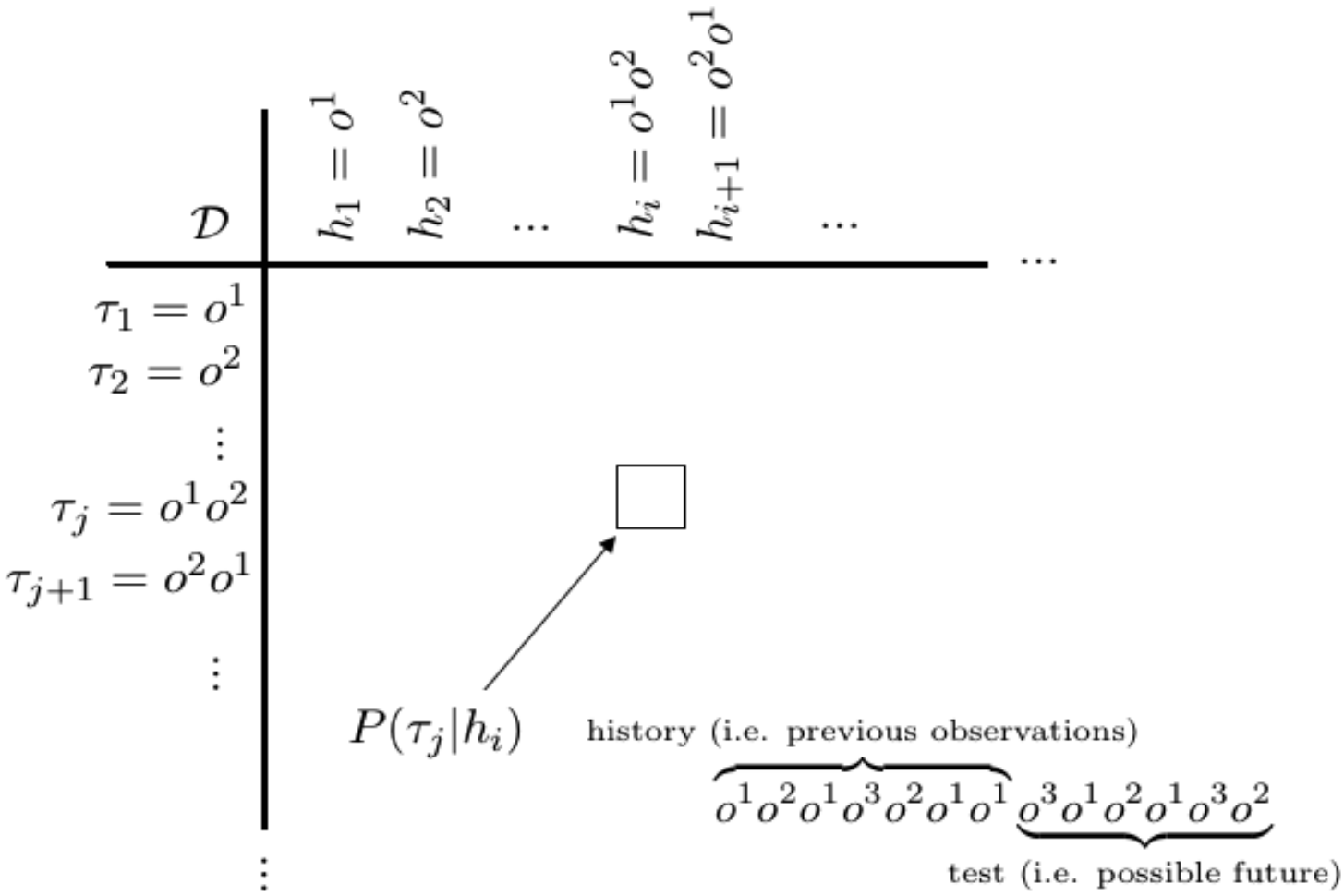
- Bayesian learning
  - » Kalman filtering
  - » Bayes-Adaptive POMDP (Ross, Chaib-draa & Pineau, 2007)
  - » Infinite POMDP (Doshi-Velez, 2009)

Event-based approaches to learning

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

Estimate probabilities of the form $p(o_{t:t+k} \mid o_{1:t-1})$ and represent compactly.

The PSR systems dynamics matrix



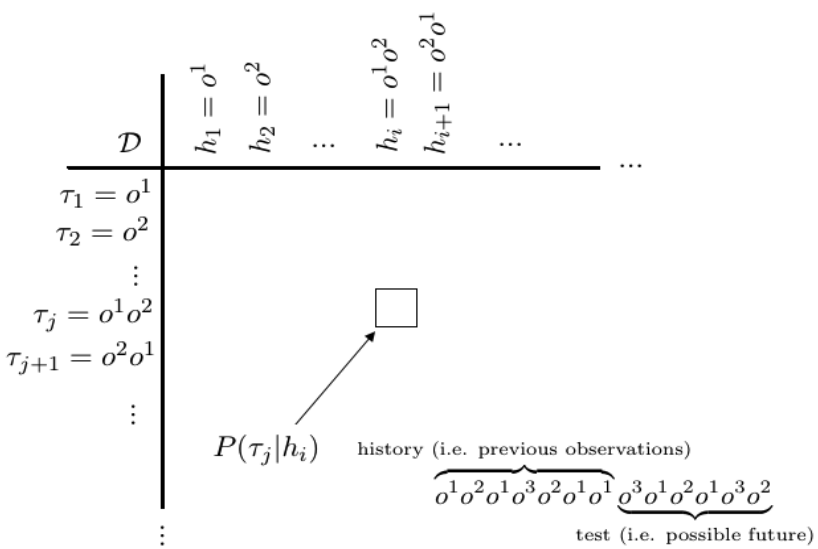
Learning PSRs from data

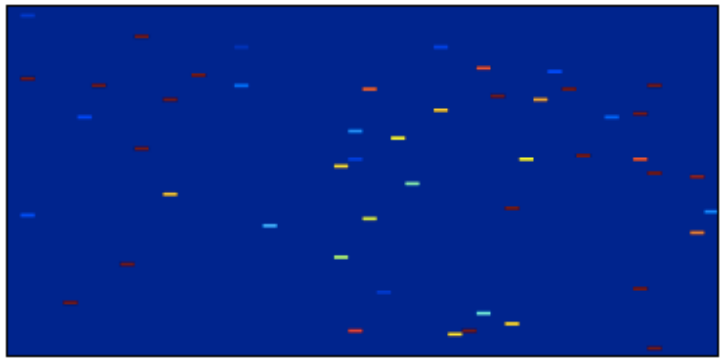
- **Spectral approach** [Rosencrantz et al., 2004; Boots et al., 2009]:
  - Estimate large sub-matrix of *D*.
  - Project to low-dimensions subspace using SVD.
  - Globally optimal, if you know *rank(D)*.

- 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.

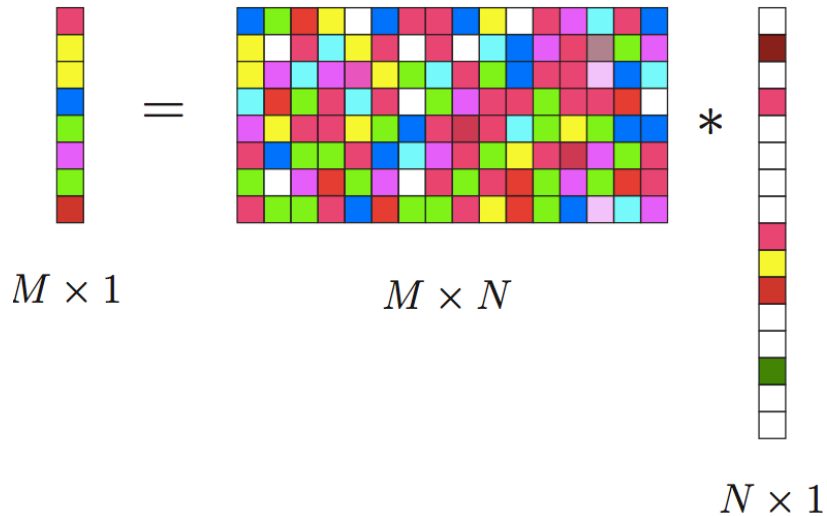




- Valid assumption? In Poc-Man domain, large sub-matrix estimates of *D* empirically observed to have an average of 99.9% column sparsity.

Exploiting sparsity

- Sparse structure can be exploited using **random projections.**



Compress *m* x *n* matrix **Y** to a *d* x *n* matrix **X**, where *d* << *m*.

Let **X** = ϕ**Y**,

where ϕ is a *d* x *m* projection matrix with entries from *N~(0, 1/d)*.

The CPSR Algorithm

Algorithm

- Obtain compressed estimates for sub-matrices of $\mathcal{D}$, $\Phi\mathcal{P}_{\mathcal{T},\mathcal{H}}$, $\Phi\mathcal{P}_{\mathcal{T},o',\mathcal{H}}$s, and $\mathcal{P}_{\mathcal{H}}$ by sampling time series data.
  - Estimate $\Phi\mathcal{P}_{\mathcal{T},\mathcal{H}}$ in compressed space by adding ϕ_i to column j each time t_i observed after h_i (Likewise for $\Phi\mathcal{P}_{\mathcal{T},o',\mathcal{H}}$s).
- Compute CPSR model:
  - $\mathbf{c}_0 = \Phi\hat{\mathcal{P}}(\tau|\emptyset)$
  - $\mathbf{C}_o = \Phi\mathcal{P}_{\mathcal{T},o',\mathcal{H}}(\Phi\mathcal{P}_{\mathcal{T},\mathcal{H}})^+$
  - $\mathbf{c}_\infty = (\Phi\mathcal{P}_{\mathcal{T},\mathcal{H}})^+\hat{\mathcal{P}}_H$

Note: Multiplication done "online" and **D** matrix never held in memory.

Using the compact representation

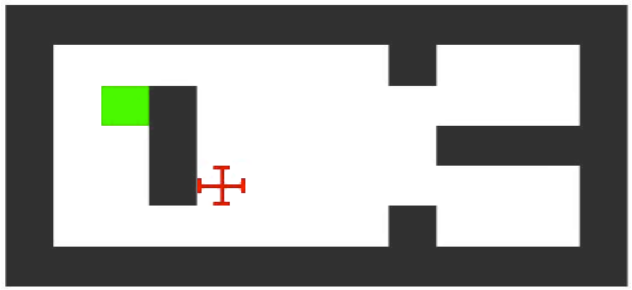
State definition and necessary equations

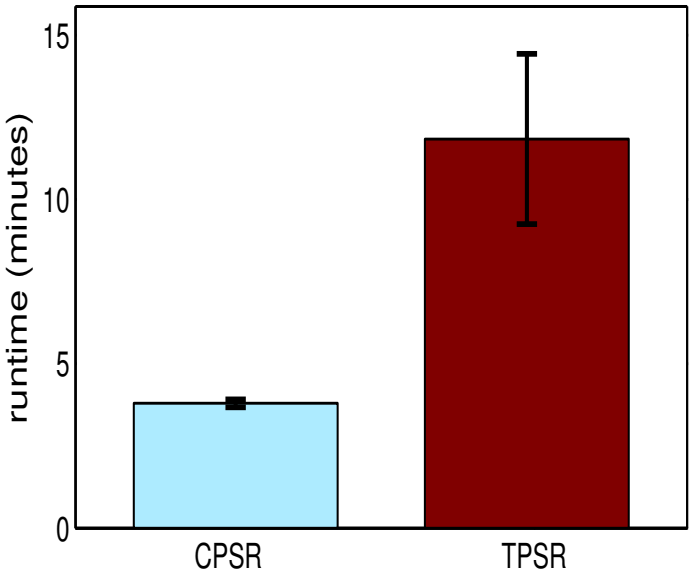
- $\mathbf{c}_0$ serves as initial prediction vector (i.e. state vector).
- Update state vector after seeing observation with
  - $\mathbf{c}_{t+1} = \frac{\mathbf{C}_o \mathbf{C}_t}{\mathbf{C}_\infty \mathbf{C}_o \mathbf{c}_t}$
- Predict k-steps into the future using
  - $P(o^j_{t+k}|h_t) = \mathbf{b}_\infty \mathbf{C}_{o^j} (\mathbf{C}_\star)^{k-1} \mathbf{C}_t$ where $\mathbf{C}_\star = \sum_{o^i \in \mathcal{O}} \mathbf{C}_{o_i}$.

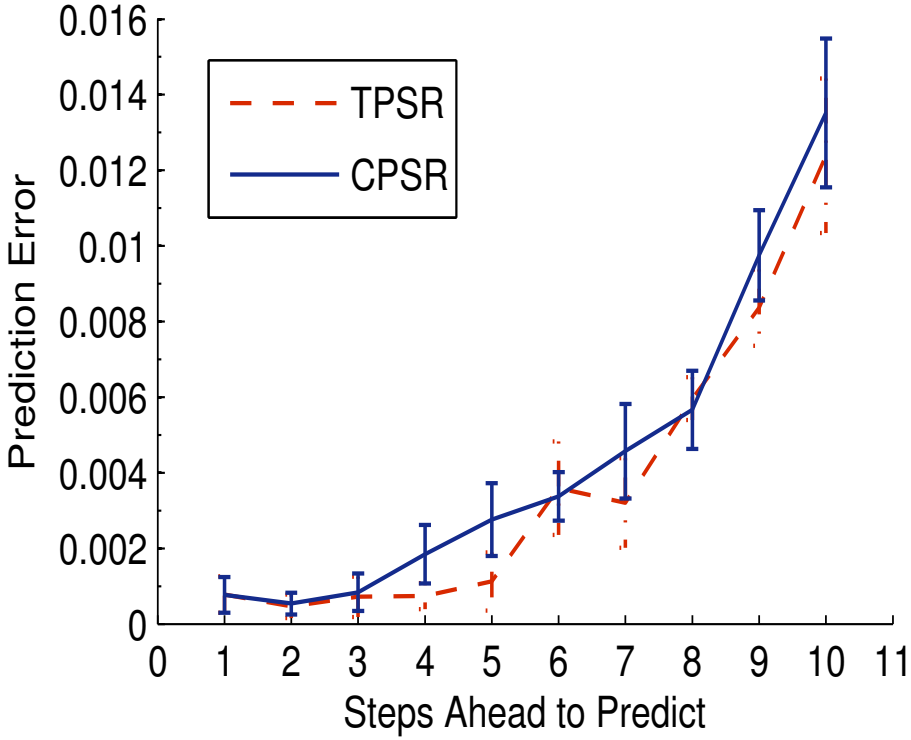
Properties of CPSR

- 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 $d < rank(D)$.
  - If $d \geq rank(D)$ then model is trivially consistent.
- Regularizes the solution (i.e. the learned model parameters).
- Total propagated error for T steps is bounded.

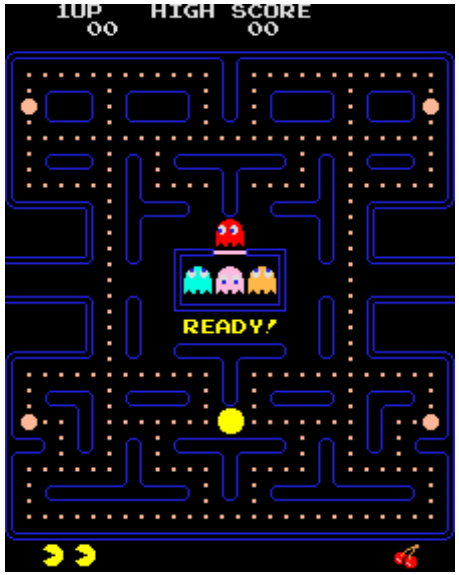
GridWorld prediction error

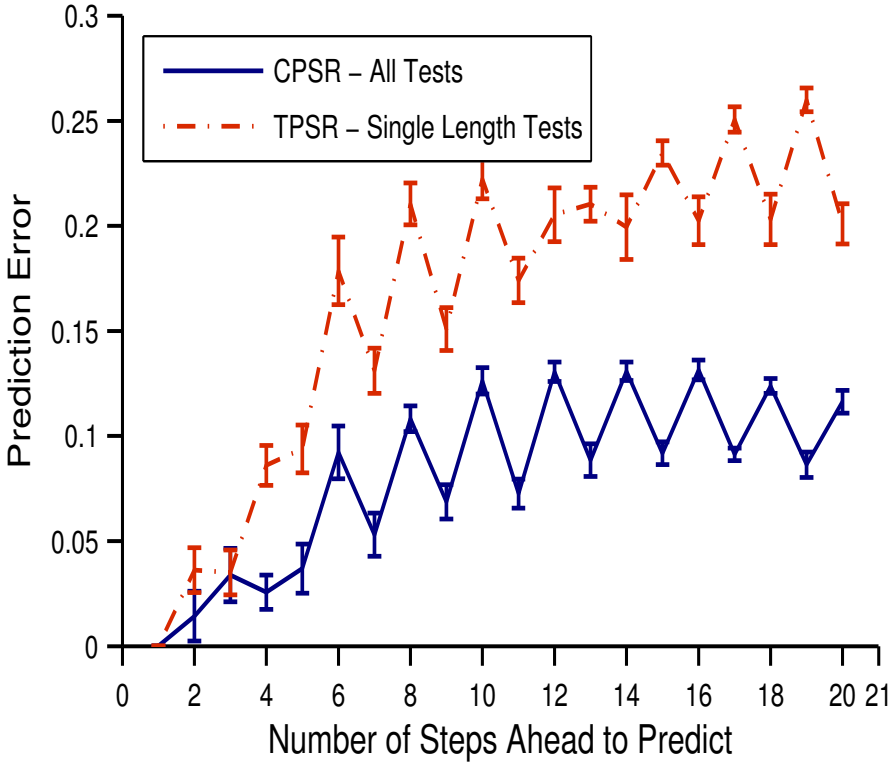






Poc-Man prediction error

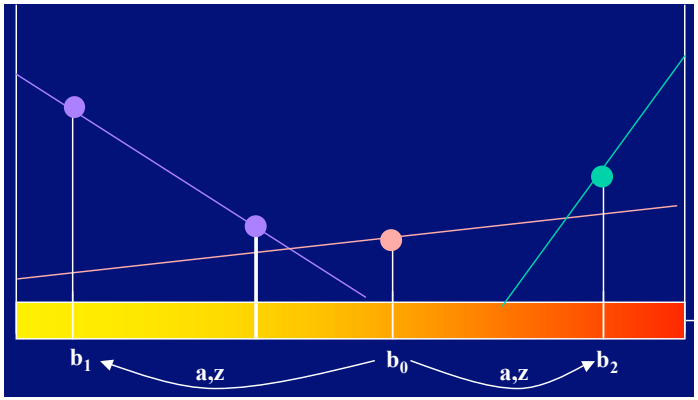




Planning in CPSRs

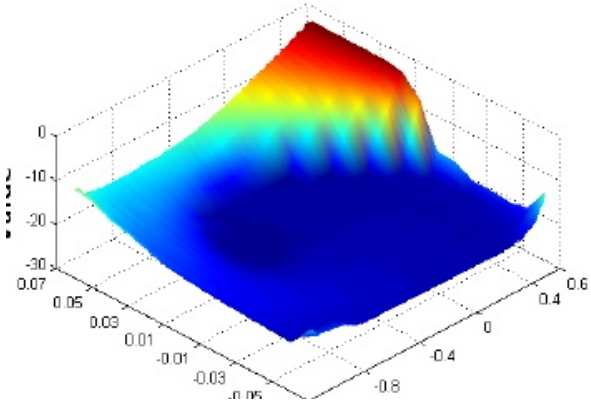
Point-based value iteration
(traditional POMDP approach)**:**

CPSR state vector =
POMDP belief state



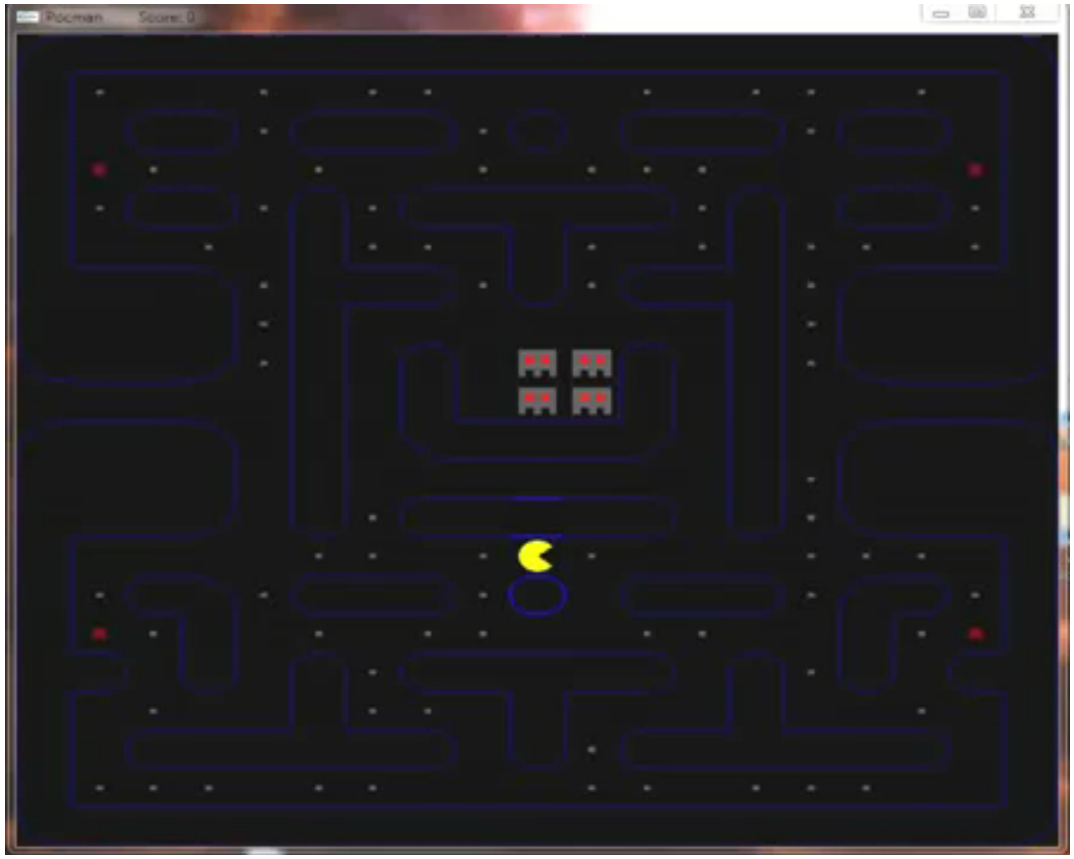
Point-based value iteration
(common MDP approach)**:**

CPSR state vector = input to
value function approximation



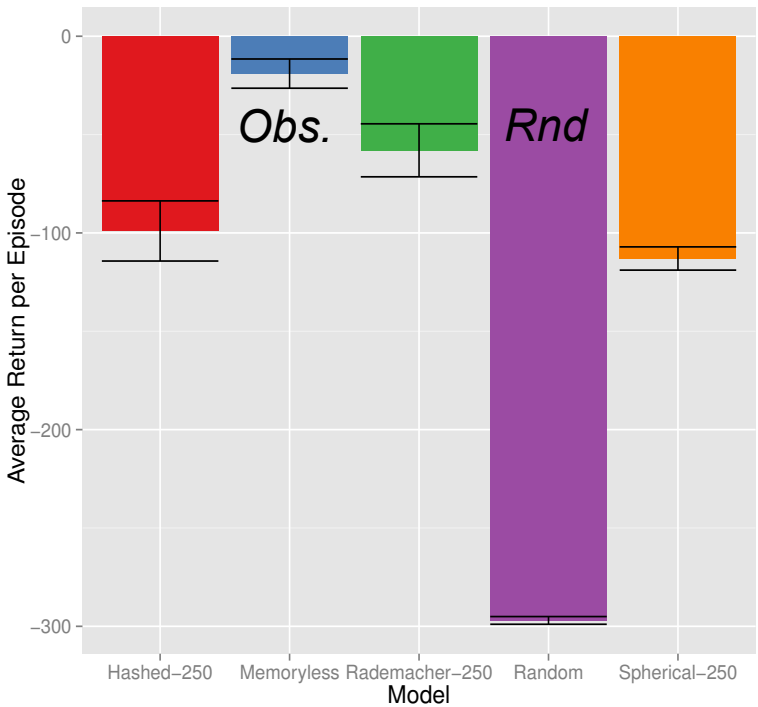
Poc-Man results

Observation space too large for building TPSR model!

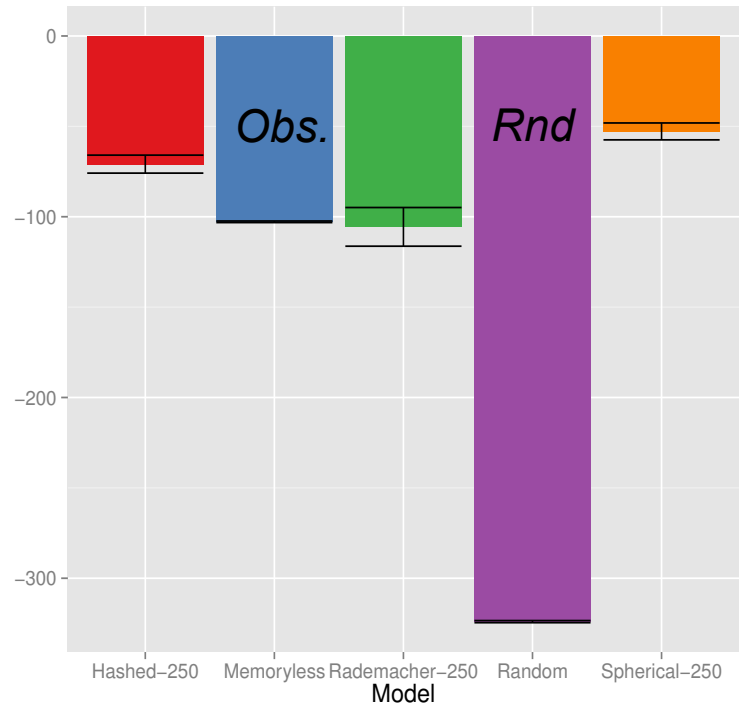


Poc-Man results

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



(a) *PocMan*

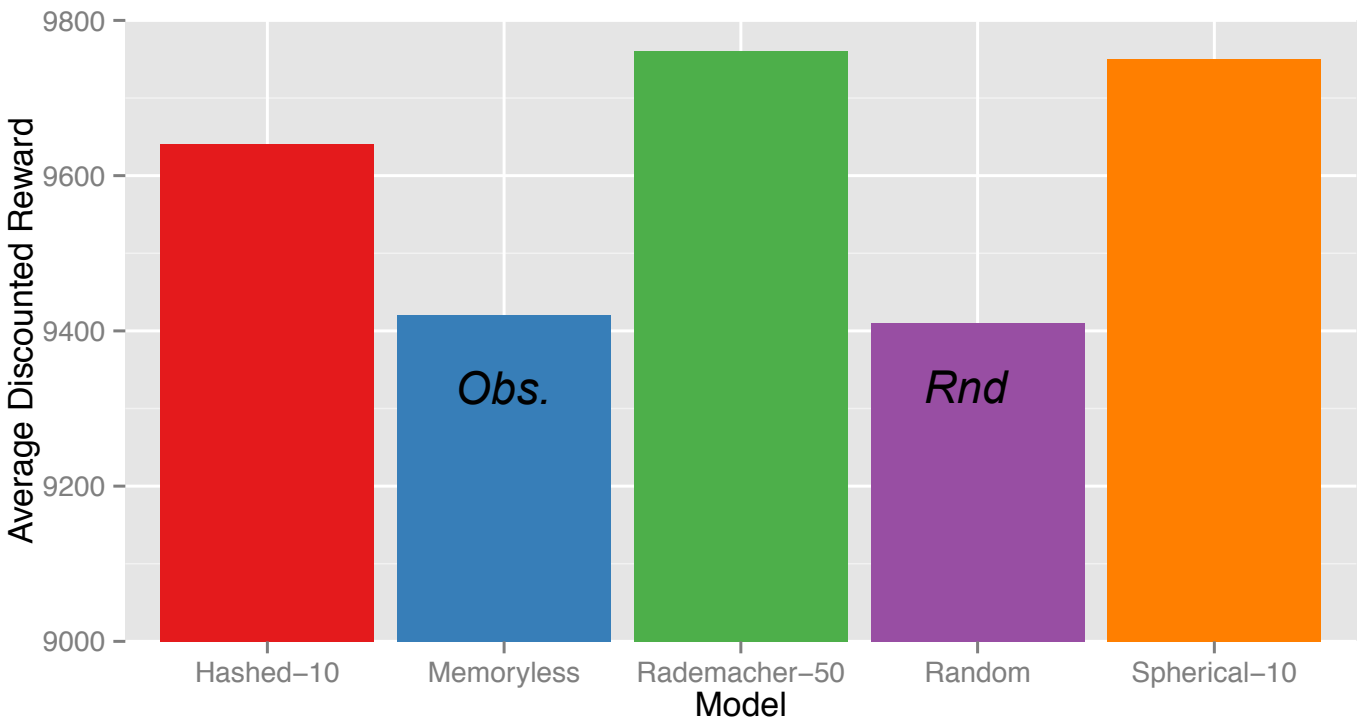


(b) *S-PocMan*

Adaptive migratory bird management

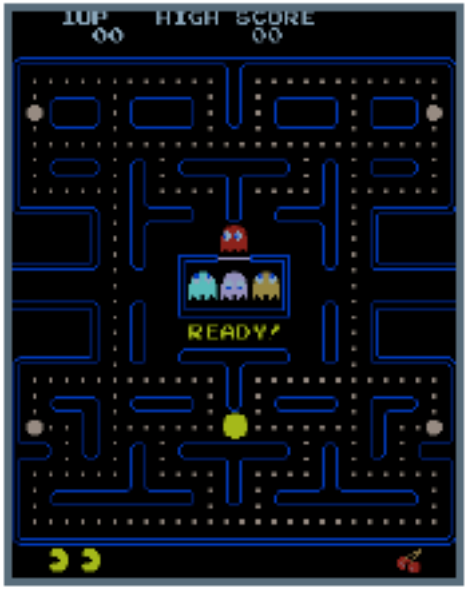
IJCAI 2013 Challenge problem (Nicol et al. 2013):

Observe bird population level at different breeding sites over time + protection level on intertidal sites on their migratory path.



Learning (more!) representations for RL

s

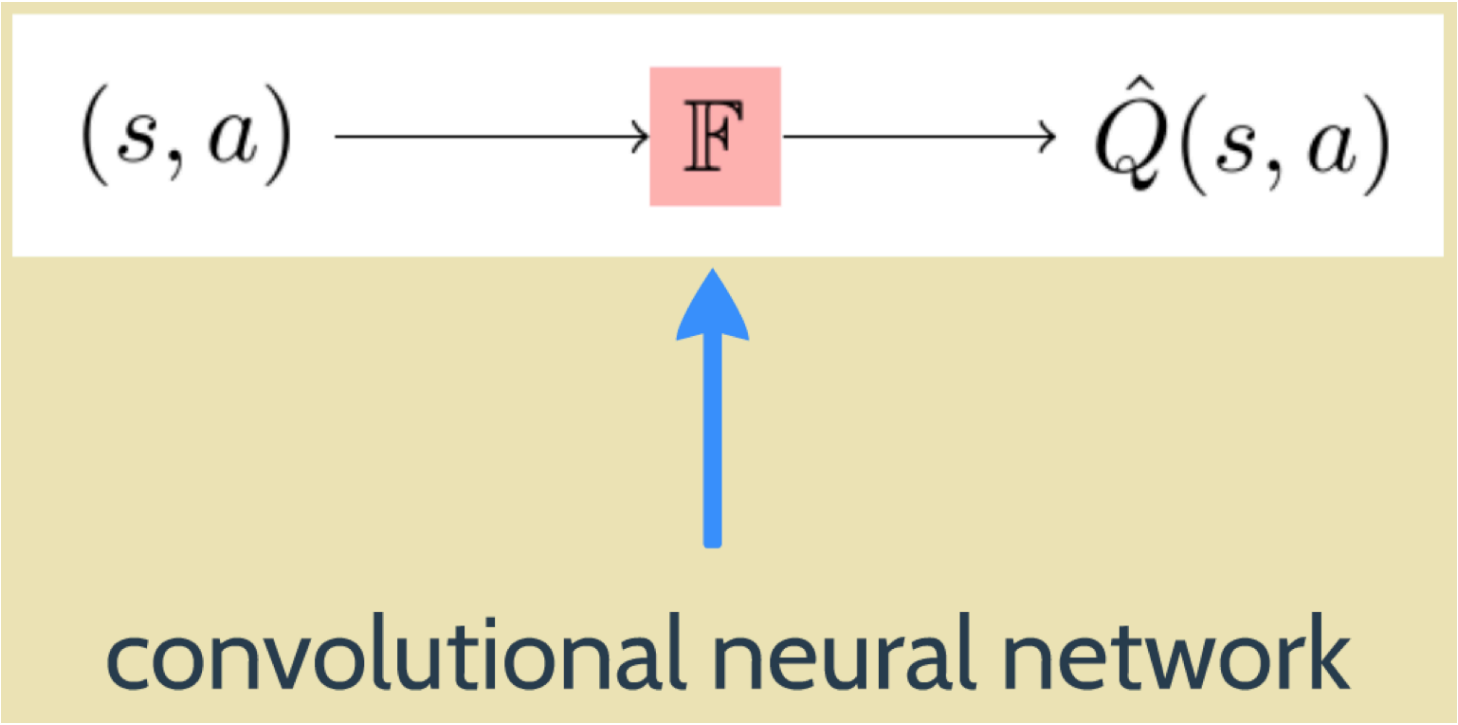


Original state

$\phi(s)?$

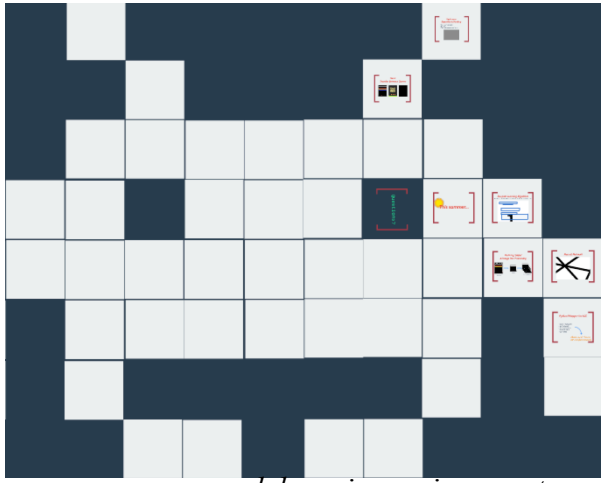
$Q(\phi(s),a)$

Deep Q-learning



Deep Q-learning

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



www.arcadelearningenvironment.org

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)