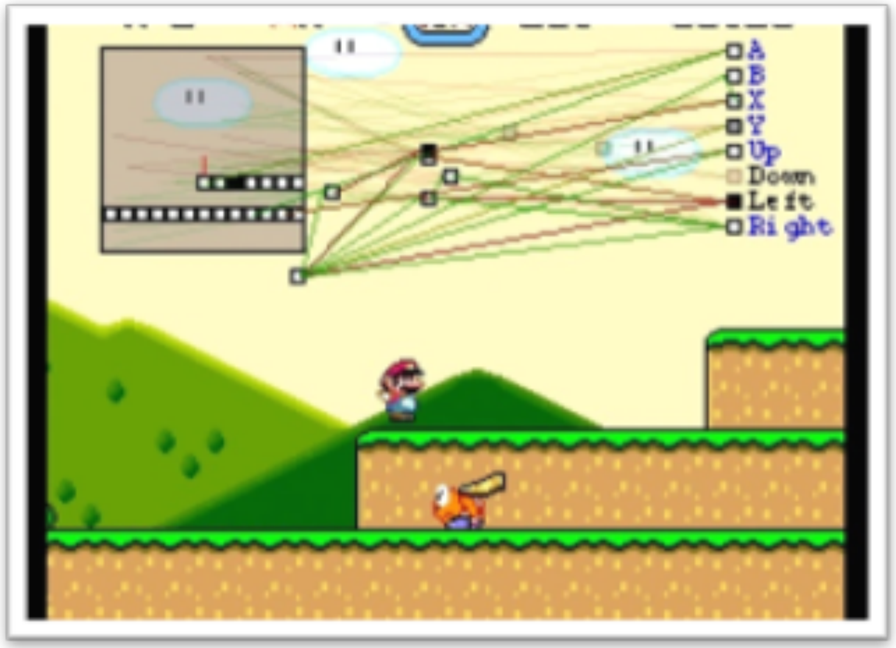
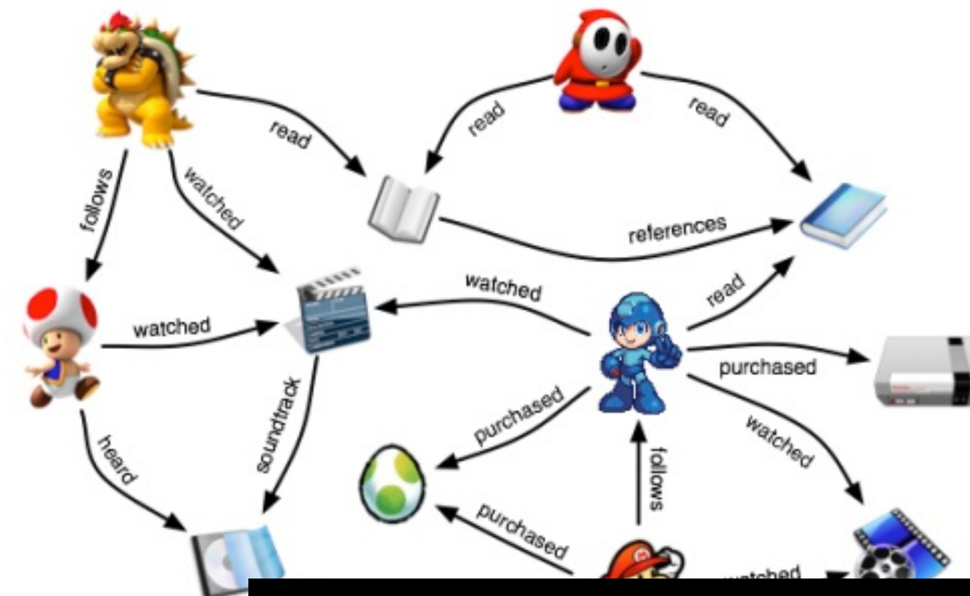
# Reinforcement Learning In Feature Space From Small Data

Mengdi Wang

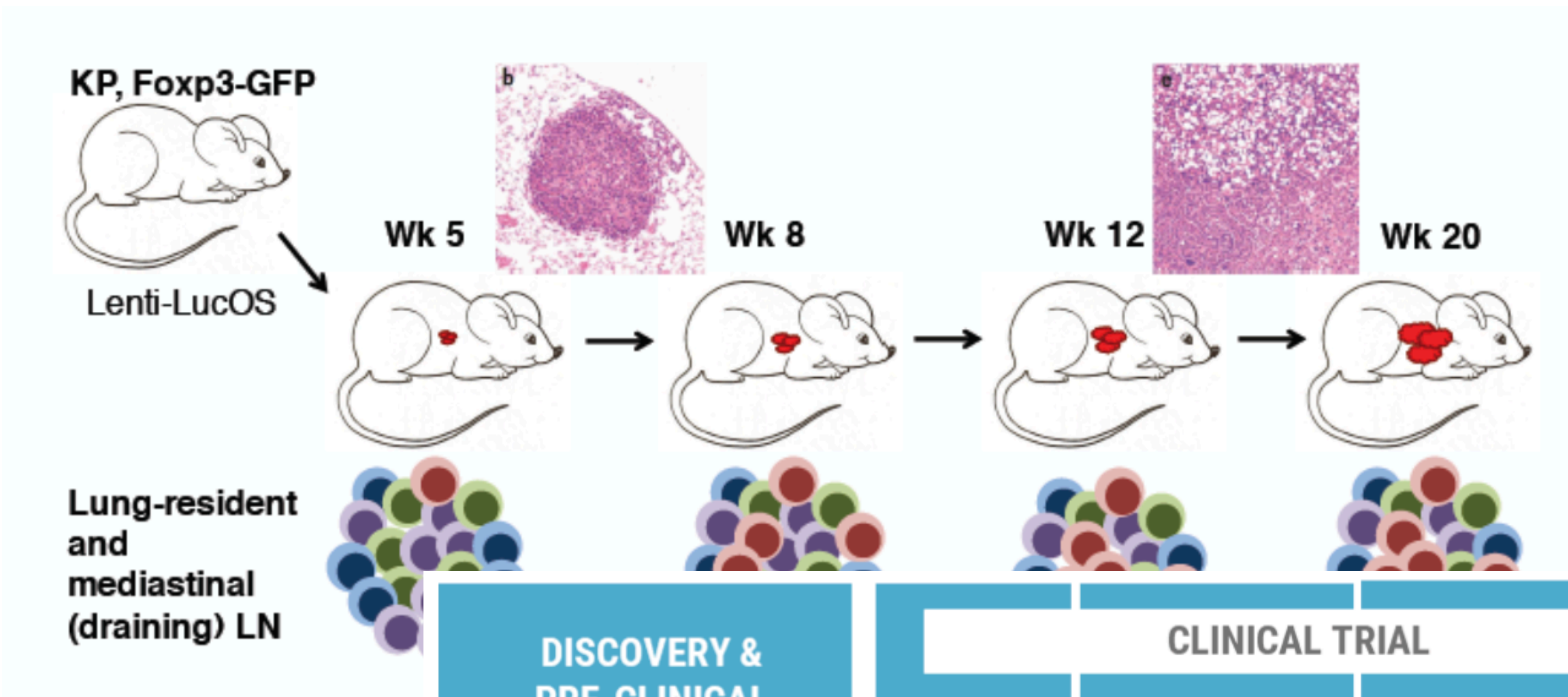
PRINCETON UNIVERSITY

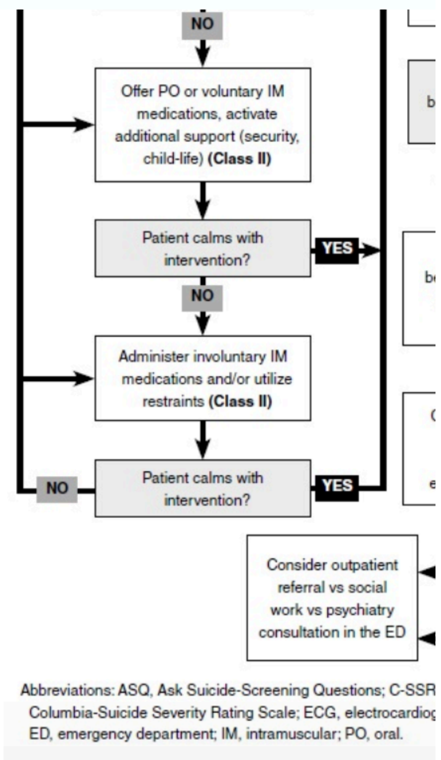


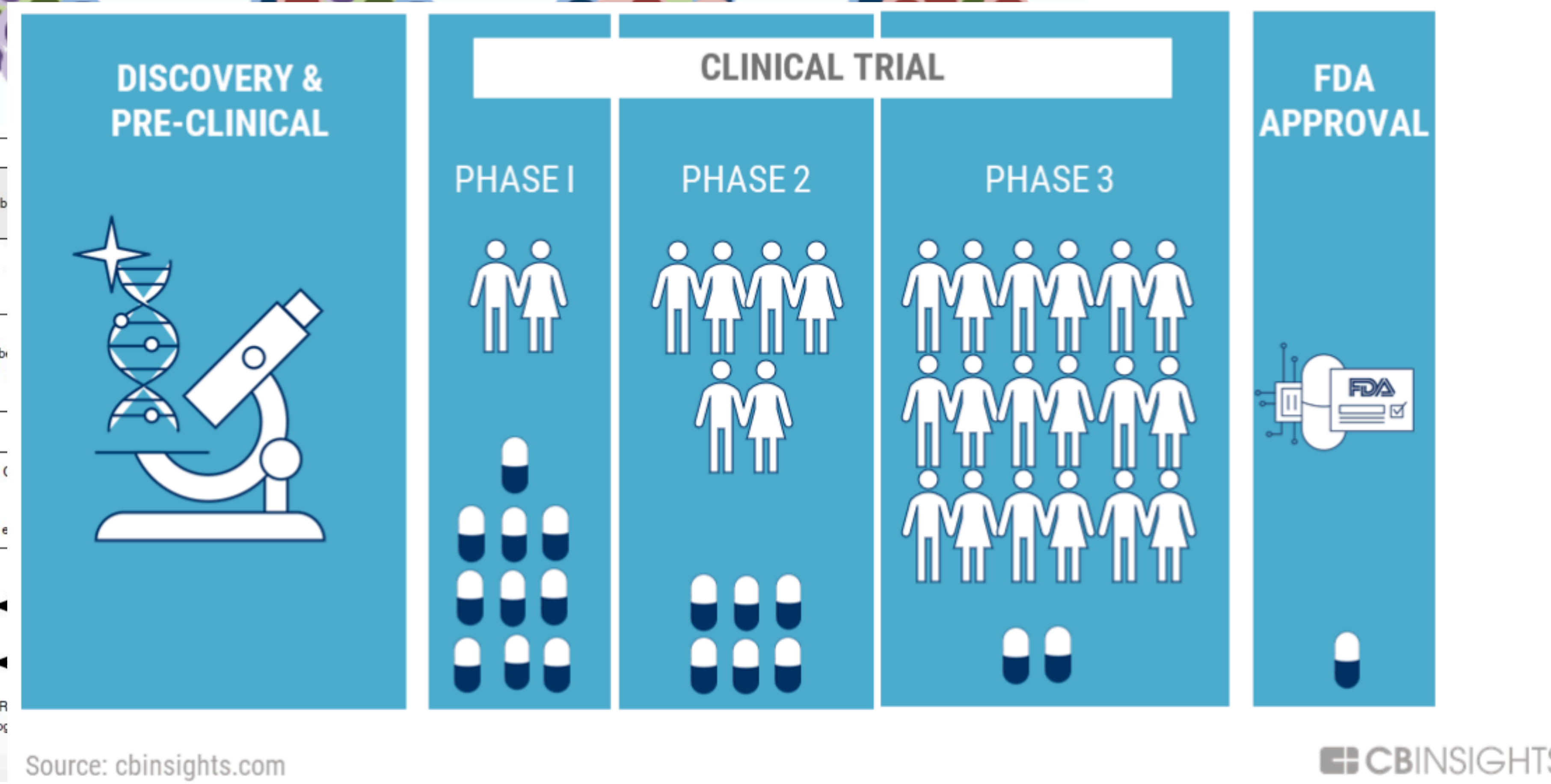





Reinforcement learning achieves phenomenal empirical successes




What if the data/trial is limited and costly

*How many samples are needed to learn an 90%-optimal policy?*

*How much regret to pay when learning to control on-the-fly?*

# Markov decision process

- A finite set of states $S$
- A finite set of actions $A$
- Reward is given at each state-action pair $(s,a)$:

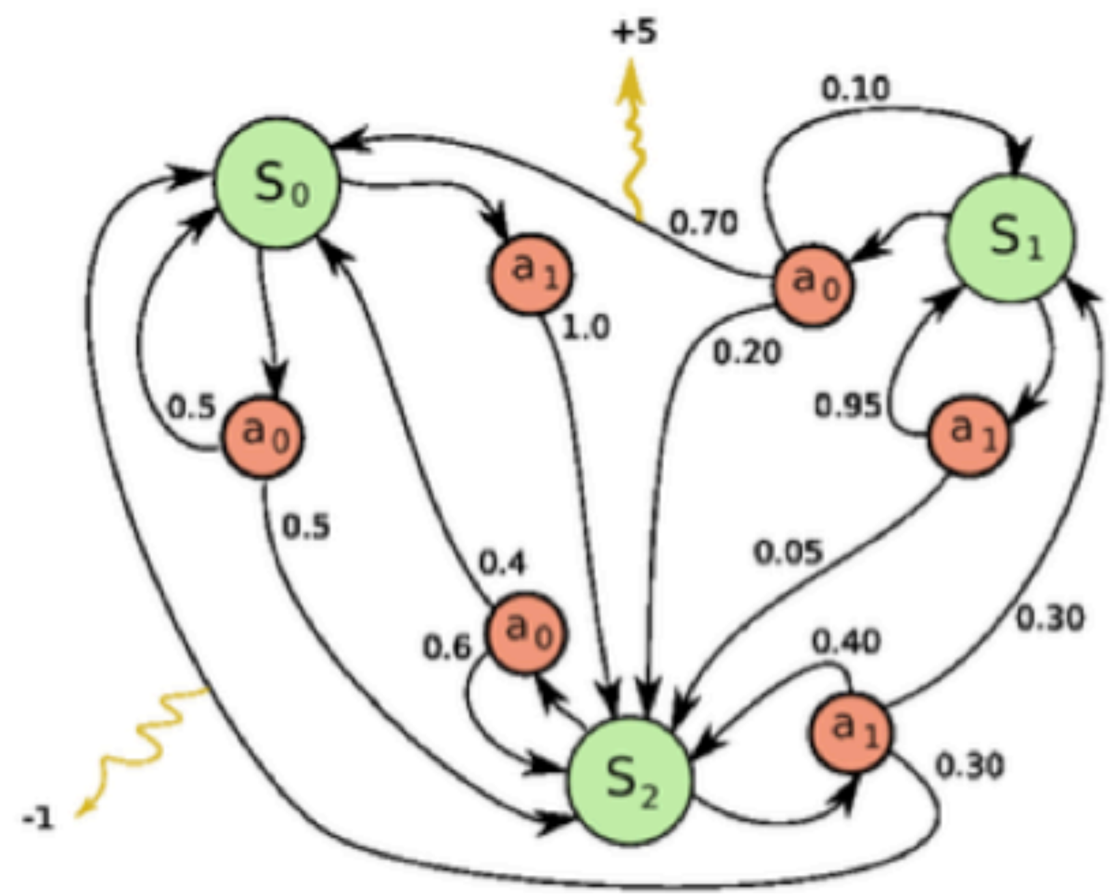
  $r(s,a) \in [0,1]$
- State transits to $s'$ with prob.

  $P(s' | s,a)$

- Find a best policy $\pi : S \to A$ such that

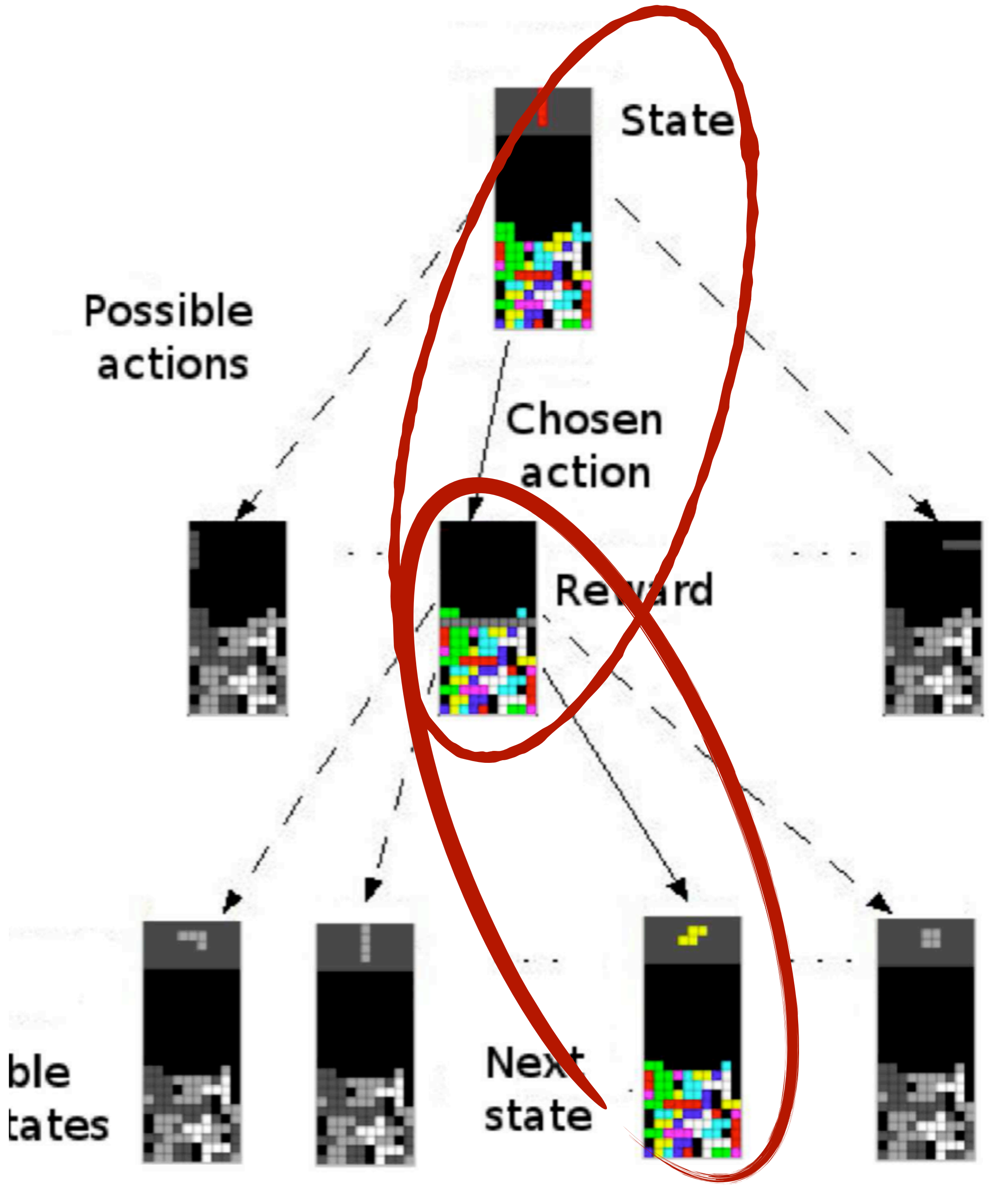
$$\max_{\pi} v^{\pi} = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$$

- *$\gamma \in (0,1)$ is a discount factor*




*We call if "tabular MDP" if there is no structural knowledge at all*

# What does a sample mean?




Samples are state-transition triplets (s,a,s')

# Use empirical risk minimization for RL?

**Data:** Sample state-transition triplets $\{(s, a, s')\}$

**Step 1:** Estimate the transition model and compute empirical transition density

$$\hat{P}(s' \mid s, s) = \frac{\#\ times\ (s, a, s')\ appeared}{\#\ times\ (s, a)\ appeared}$$

**Step 2:** Solve the empirical MDP problem by dynamic programming

$$\hat{\pi} = \mathbf{argmax}_{\pi} \mathbb{E}^{\pi}_{\hat{P}} \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t) \right]$$

- *Hard to analyze:* tons of dependencies and nonlinearity [AMK13, AKY19]
- *Hard to implement:* it is a model-based approach (large memory overhead + computation bottleneck)
- *Which are model-free:* Q-learning, actor-critic, policy gradients

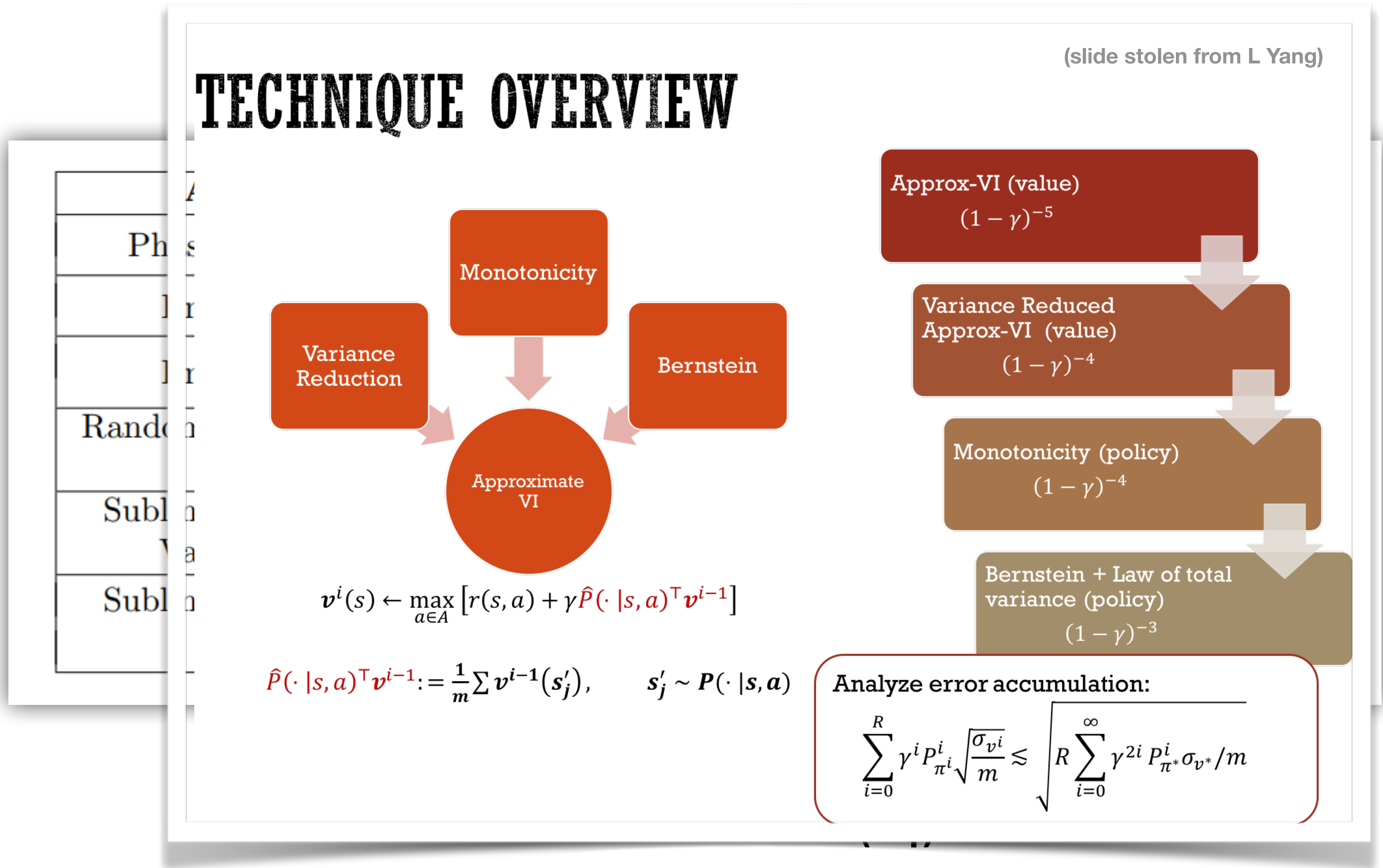
# Prior efforts: algorithms and sample complexity results

| Algorithm | Sample Complexity | References |
|---|---|---|
| Phased Q-Learning | $\tilde{O}(C\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^7\epsilon^2})$ | [KS99] |
| Empirical QVI | $\tilde{O}(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^5\epsilon^2})$ [2] | [AMK13] |
| Empirical QVI | $\tilde{O}(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^3\epsilon^2})$ if $\epsilon = \tilde{O}(\frac{1}{\sqrt{(1-\gamma)\lvert\mathcal{S}\rvert}})$ | [AMK13] |
| Randomized Primal-Dual Method | $\tilde{O}(C\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^4\epsilon^2})$ | [Wan17] |
| Sublinear Randomized Value Iteration | $\tilde{O}\left(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^4\epsilon^2}\right)$ | [SWWY18] |
| Sublinear Randomized QVI | $\tilde{O}\left(\frac{\lvert\mathcal{S}\rvert\lvert\mathcal{A}\rvert}{(1-\gamma)^3\epsilon^2}\right)$ | This Paper |

**$1/(1-\gamma)=1+\gamma+\gamma^2+\ldots$ is the effective horizon**

**Lots of efforts about $1/(1-\gamma)$**

# Prior efforts: algorithms and sample



# Complexity and Regret for Tabular MDP

- **Information-theoretical limit** (Azar et al. 2013)**:** Any method finding an ε-optimal policy with probability 2/3 needs at least sample size

$$\Omega\left(\frac{|SA|}{(1-\gamma)^3\epsilon^2}\right)$$

- **The optimal sampling-based algorithm** (with Sidford, Yang, Ye, 2018, Agarwal et al, 2019)**:** With a generative model, finding ε-optimal policy with probability 1-δ using sample size

$$O\left(\frac{|SA|}{(1-\gamma)^3\epsilon^2}\log\frac{1}{\delta}\right)$$

*Statistical complexity of RL (in this basic setting) is finally well understood*

# S

**is way too big**

Suppose states are vectors of dimension $d$

Vanilla discretization of state space gives $|S| = 2^d$

Size of policy space = $|A|^{|S|}$

Log of policy space size = $|S| \log(|A|) > 2^d$

# Rethinking Bellman equation

**Bellman equation is the optimality principal for MDP** (in the average-reward case, where γ=1)

$$\bar{v}^* + v^*(s) = \max_a \left\{ \sum_{s' \in \mathcal{S}} P_a(s, s') v^*(s') + r_a(s) \right\}, \qquad \forall s \in \mathcal{S}$$

- The max operation applies to every state-action pair -> nonlinearity + high dim

**Bellman equation is equivalent to a bilinear saddle point problem** (Wang 2017)

$$\min_{v} \max_{\mu \in \Delta} \left\{ L(v, \mu) = \sum_a \left( \mu_a^T ((I - P_a) v + r_a) \right) \right\}$$

value function

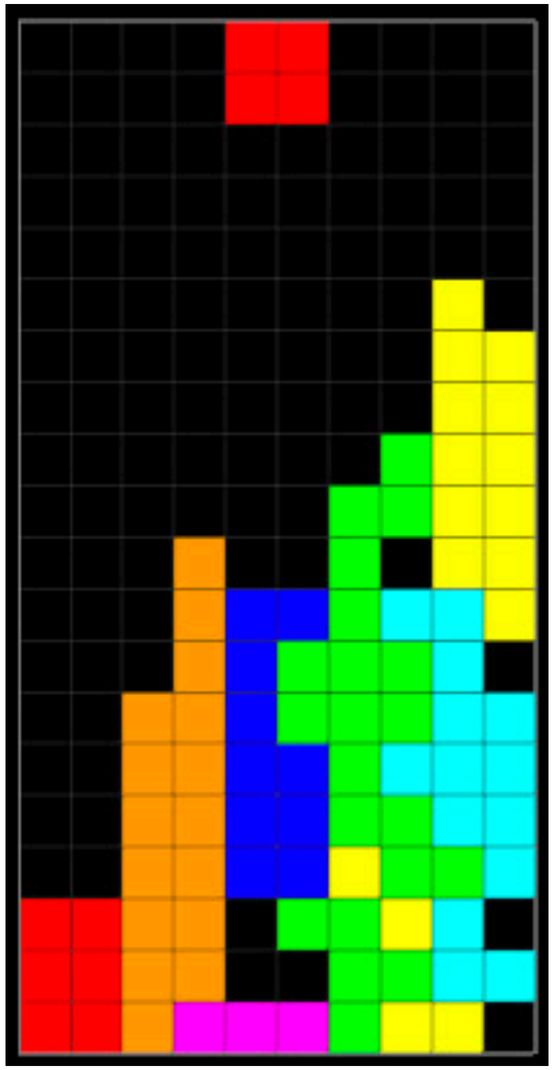
stationary state-action distribution

- Strong duality between value function and invariant measure
- SA x S linear program

# State Feature Map

- Suppose we are given a state feature map

$$state \mapsto [\phi_1(state), \ldots, \phi_N(state)] \in \mathbb{R}^N$$



- Can we do better?
- Tetris can be solved well using 22 features and linear models
    - Feature 1: Height of wall
    - Feature 2: Number of holes

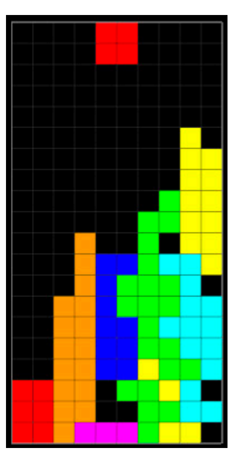
# Representing value function using linear combination of features

- The value function of a policy is the expected cumulative reward as the initial state varies:

$$V^\pi : \mathcal{S} \to \mathbb{R}, \qquad V^\pi(s) = \mathbb{E}^\pi \left[ \sum_{t=0}^{H} r(s_t, a_t) \mid s_0 = s \right]$$

- Suppose that the high-dimensional value vector admits a linear model:

$$V^\pi(s) \approx w_1 \phi_1(s) + \ldots + w_N \phi_N(s)$$

- Value of  = $w_1$ x Height of Wall + $w_2$ x # Holes + …

- Linear model for value function approximation has lots of limitations (later)

# Reducing Bellman equation using features

$$\bar{v}^* + v^*(s) = \max_a \left\{ \sum_{s' \in \mathcal{S}} P_a(s, s') v^*(s') + r_a(s) \right\}, \forall s$$

**Bellman eq:**

$\left\{ \begin{array}{l} \text{High-dim} \\ \text{Nonlinear} \end{array} \right.$

$$\min_{v} \max_{\mu \in \Delta} \left\{ L(v, \mu) = \sum_a \left( \mu_a^T ((I - P_a)v + r_a) \right) \right\}$$

**Bellman saddle point:**

$\{$ High-dim

$$\min_{v \in \textbf{Span}(\Phi)} \max_{\mu \in \textbf{Span}(\Phi\Psi^T)} L(v, \mu)$$

$$\left\{ \begin{array}{l} v(\cdot) \approx \sum_{i=1}^{r_S} w_i \phi_i(\cdot) \\ \mu(s, a) \approx \sum_{i=1}^{r_S} \sum_{j=1}^{r_A} u_{ij} \phi_i(s) \psi_j(a) \end{array} \right.$$

$$\min_{w \in \Re^{r_S}} \max_{u \in \Re^{r_A}} \sum_a \left( \Psi_{a*} u_a^T \Phi^T (I - P_a) \Phi \tilde{v} + r_a \right)$$

$\left\{ \begin{array}{l} \text{Low-dim} \\ \text{Convex-concave} \\ \text{Strong duality} \\ \text{Parametric} \end{array} \right.$

# Sample complexity of RL with features

**Suppose that good state and action features are known**

- For average-reward RL, a primal-dual policy learning method finds the optimal policy using sample size (with YC, LL, 2018)

$$\Theta\left(C \cdot \frac{|N_S N_A|}{\epsilon^2}\right)$$

where C is polynomial in mixing and ergodicity parameters

- Sample-Optimal Parametric Q-Learning for discounted RL （with LY, 2019)

$$\Theta\left(\frac{|N_S N_A|}{\epsilon^2(1-\gamma)^3}\right)$$

- Matching the information-theoretic minimax lower bound.

- *Reduced* ***S*** *to* ***$N_S$ $N_A$ (# state-action features)***

# Learning to Control On-The-Fly

- Prior sample complexity analysis assumes a **generative model:**
  - One can draw transitions from any pre-specified state-action pair (enough exploration guaranteed)
  - Sample-optimal algorithms draw the same number of samples per state or per representative state (w. Sidford,Yang,Ye18, w. Yang Jia 19, Agarwal metal 19)
- In practice, we have to **learn on-the-fly:**
  - H-horizon stochastic control problem, starting at a fixed state $s_0$
  - A learning algorithm learns to control by repeatedly acting in the real world
  - It would act in realtime, observe state transitions, and adapt its control policy every episode
  - Impossible to visit all states frequently

# Episodic Reinforcement Learning

- **Regret of a learning algorithm** $\mathscr{K}$

$$\textbf{Regret}_{\mathscr{K}}(T) = \mathbb{E}_{\mathscr{K}}\left[\sum_{n=1}^{N}\left(V^*(s_0) - \sum_{h=1}^{H} r\left(s_{n,h}, a_{n,h}\right)\right)\right],$$

where T= NH, and the sample state-action path $\{s_{n,h}, a_{n,h}\}$ is generated on-the-fly by the learning algorithm $\mathscr{K}$

- **Challenges:**
  - Long-term effect of a single wrong decision
  - Data dependency: Almost all the transition samples are dependent
  - Exploration-exploitation tradeoff
  - More complicated than multi-arm bandit (naive reduction yields A^S arms)

# Hilbert space embedding of transition kernel

- **Suppose we are given state-action feature maps**

$$state, action \mapsto [\phi_1(state, action), \ldots, \phi_d(state, action)] \in \mathbb{R}^N$$

$$state \mapsto [\psi_1(state), \ldots, \psi_{d'}(state)] \in \mathbb{R}^{d'}$$

- Assume that the unknown transition kernel can be fully embedded in the feature space, i.e., there exists a transition core M* such that

$$P(s' \mid s, a) = \phi(s, a)^\top M^* \psi(s') \,.$$

- The decomposition structure is equivalent to using linear model for value function approximation with 0 Bellman error (w LY 2019)

- Low-dim assumption on V is closely related to low-dim assumption on P

# The MatrixRL Algorithm

- At the beginning of the (n+1)th episode, suppose the samples collected so far are

$$\{(s_{n,h}, a_{n,h}), s_{n,h+1}\} \to \{\phi_{n,h}, \psi_{n,h}\} := \{\phi(s_{n,h}, a_{n,h}), \psi(s_{n,h+1})\}$$

- We will use their corresponding feature vectors.

- Estimate the transition core via matrix ridge regression

$$M_n = \arg\min_{M} \sum_{n'<n,h\le H} \left\| \psi_{n',h}^{\top} K_{\psi}^{-1} - \phi_{n',h}^{\top} M \right\|_2^2 + \|M\|_F^2.$$

Where $K_\psi$ is a precomputed matrix

- However, using empirical estimate greedily would lead to poor exploration

- Borrow ideas from linear bandit (Dani et al 08, Chu et al 11, …)

# The MatrixRL Algorithm

- **Construct a matrix confidence ball** around the estimated transition core

$$B_n = \left\{ M \in \mathbb{R}^{d \times d'} : \quad \|(A_n)^{1/2}(M - M_n)\|_F \leq \sqrt{\beta_n} \right\}$$

- **Find optimistic Q-function estimate**

$$Q_{n,h}(s,a) = r(s,a) + \max_{M \in B_n} \phi(s,a)^\top M \Psi^\top V_{n,h+1}, \quad Q_{n,H} = 0$$

where the value estimate is given by

$$V_{n,h}(s) = \Pi_{[0,H]}\left[ \max_a Q_{n,h}(s,a) \right]$$

- **In the new episode, choose actions greedily by** $\max_a Q_{n,h}(s,a)$

- The optimistic Q encourage exploration: (s,a) with higher uncertainty gets tried more often

(RL in Feature Space: Matrix Bandit, Kernels, and Regret Bounds, Preprint, 2019)

# Regret Analysis

- **Theorem** Under the embedding assumption and regularity assumptions, the T-time-step regret of MatrixRL satisfies with high probability thats

$$\textbf{Regret}(T) \leq C \cdot dH^2 \cdot \sqrt{T},$$

- First polynomial regret bound for RL in feature space.
- *Independent of S*
- Minimax optimal?
- *It is optimal in d and T, close to optimal in H*

(RL in Feature Space: Matrix Bandit, Kernels, and Regret Bounds, Preprint, 2019)

# The special case where Ψ= I

- A nonparametric model where P cannot be encoded using a small # of parameters

$$P(s' \mid s, a) = \phi(s, a)^\top M^* \psi(s'), \qquad \textbf{where} \quad \psi = I\,.$$

- It only needs features to describe left principal space of P

- **In this case, MatrixRL has closed-form updates:**

$$Q_{n,h}(s,a) = r(s,a) + \phi(s,a)^\top M_n V_{n,h+1} + C\sqrt{\beta_n}\sqrt{\phi_{n,h}^T A_n^{-1} \phi_{n,h}}, \quad Q_{n,H} = 0$$

- **Theorem** Under the embedding assumption and if ψ= I, the T-time-step regret of MatrixBandit is

$$\textbf{Regret}(T) \le C \cdot d^{3/2} H^2 \cdot \sqrt{T},$$

-

# From feature to kernel

Suppose that we are given a kernel function over the state-action space instead of explicit feature maps

$$K((s, a), (s', a'))$$

- RL in kernel space? (Ormoneit & San 02, Ormoneit & Glynn 02, …)
- Kernel presents a very flexible framework for extrapolating information from seen states to unseen states
- We consider the generic assumption that the transition kernel belongs to the product Hilbert spaces spanned by these features:

$$P \in \mathscr{H}_\phi \times \mathscr{H}_\psi$$

# MatrixRL has a equivalent kernelization

**Algorithm 2** KernelMatrixRL: Reinforcement Learning with Kernels

**Input:** An episodic MDP environment $M = (\mathcal{S}, \mathcal{A}, P, s_0, r, H)$, kernel functions $k_\phi, k_\psi$;
Total number of episodes $N$;
**Initialize:** empty reply buffer $\mathcal{B} = \{\}$;
**for** episode $n = 1, 2, \ldots, N$ **do**
For $(s, a) \in \mathcal{S} \times \mathcal{A}$, let

$$w_n(s,a) := \sqrt{k_\phi[(s,a),(s,a)] - \mathbf{k}_{\Phi_{n-1},s,a}^\top (I + \mathbf{K}_{\Phi_{n-1}})^{-1} \mathbf{k}_{\Phi_{n-1},s,a}};$$

$$x_n(s,a) := \mathbf{k}_{\Phi_{n-1},s,a}^\top (I + \mathbf{K}_{\Phi_{n-1}})^{-1} \mathbf{K}_{\Psi_{n-1}} (\overline{\mathbf{K}}_{\Psi_{n-1}} \overline{\mathbf{K}}_{\Psi_{n-1}}^\top)^{-1} \overline{\mathbf{K}}_{\Psi_n};$$

Let $\{Q_{n,h}\}$ be defined as follows:

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}: \quad Q_{n,H+1}(s,a) := 0 \quad \text{and}$$
$$\forall h \in [H]: \quad Q_{n,h}(s,a) := r(s,a) + x_n(s,a)^\top V_{n,h+1} + \eta_n w_n(s,a), \tag{9}$$

where

$$V_{n,h}(s) = \Pi_{[0,H]}\big[\max_a Q_{n,h}(s,a)\big] \qquad \forall s, a, n, h;$$

and $\eta_n$ is a parameter to be determined;
**for** stage $h = 1, 2, \ldots, H$ **do**
Let the current state be $s_{n,h}$;
Play action $a_{n,h} = \arg\max_{a \in \mathcal{A}} Q_{n,h}(s_{n,h}, a)$;
Record the next state $s_{n,h+1}$: $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_{n,h}, a_{n,h}, s_{n,h+1})\}$;
**end for**
**end for**

**Theorem** $\quad$ **Regret**$(T) \le O\left(\|P\|_{\mathcal{H}_\phi \times \mathcal{H}_\psi} \cdot \log(T) \cdot \widetilde{d} \cdot H^2 \cdot \sqrt{T}\right)$

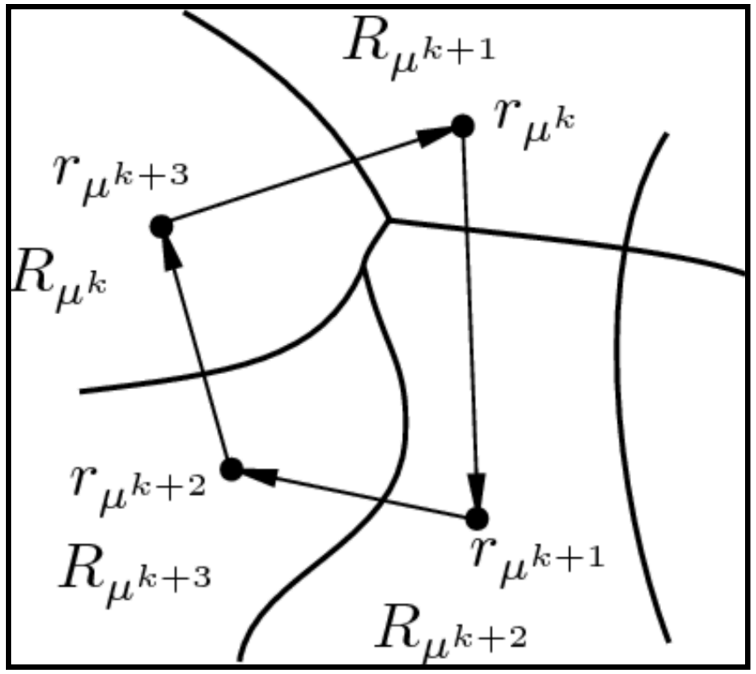
RL regret in kernel space depends on Hilbert space norm of the transition kernel and effective dimension of the kernel space

(RL in Feature Space: Matrix Bandit, Kernels, and Regret Bounds, w. Lin Yang, 2019)

# Pros and cons for using features for RL

- Deep connection to regression. Theoretical guarantee
- Easy to implement. Not many parameters to tune.

- Rely on good known features
- Pathological policy oscillation and chattering
- Not as rich as nonlinear models




(Bertsekas 07)

- *Not very surprising that good features can reduce the dimensionality of RL … Can we do well without known features?*
- *Many works in this domain, eg state representation learning (Lesort et al 08), latent state encoding (Du et al 19)*

# What could be good state features?

- Given a stationary Markov chain with transition operator *P* and one-step reward function *r*, the average-reward difference-of-value function is given by

$$v = \lim_{T\to\infty} \left( r + Pr + P^2 r + \cdots + P^T r - (T\bar{r}) \cdot \mathbf{1} \right).$$

- Suppose that *P* admits the decomposition

$$P = \Phi \tilde{P} \Psi^T$$

- Both the value *v* and the invariant measure ξ lie in low-dim spaces:

$$v \in \textbf{Span}(\Phi) \qquad \xi \in \textbf{Span}(\Psi)$$

**Good value features ϕ shall span the column space of P**

# Learning features automatically from time series data

- Consider a state-transition trajectory

$$X_1, X_2, \cdots, X_t, \cdots$$

- Spectral decomposition of the transition operator

$$\mathbb{P}(X_{t+1} \mid X_t) \approx \sum_i^r u_i(X_t) v_i(X_{t+1})$$

**Markov features**

- $u_i(\cdot)$ 's $v_i(\cdot)$'s are natural features for RL
- Reward-independent

**Estimate $x \to \Psi(x)$ from data to "preserve dynamics" (approximate leading singular functions of P)**

$$\max_{\Psi: X \mapsto \mathbb{R}^r, \Psi_j \in H} \mathbf{Tr} \left( \int \Psi(x) p(x, y) \Psi(y)^T dx dy \right)$$

- Statistical error bounds and information-theoretic limits proved (w AZ 2018, w YD, KZ, 2018, w YS, YD, GH, 2019)

# Kernelized state embedding from random features

**Data:** A high-dimensional time series and a kernel space with K

$$X_1, X_2, \cdots, X_t, \cdots, \qquad \textbf{where} \quad X_t \in \mathbb{R}^d$$

**Solution:**

1. Open up the kernel space and approximate with random features

$$K(x, y) \approx \phi(x)^\top \phi(y) \qquad \phi(\,\cdot\,) = [\phi_1(\,\cdot\,), \ldots, \phi_N(\,\cdot\,)]^\top$$

2. Estimate a projection matrix of the transition kernel onto the K space

$$\hat{Q} = \frac{1}{T} \sum_{t=1}^{T} \phi(X_t) \phi(X_{t+1})^\top$$

3. Find the best rank-r approximation $\quad \hat{Q} = \hat{U} \Lambda \hat{V}^T, \qquad \hat{Q}_r = \hat{U}_r \Lambda_r \hat{V}_r^T$

**Output:** Low-dim state embedding (a kernelized diffusion map)

$$X \mapsto P(\,\cdot\,|X) \mapsto \hat{\Psi}(X) := \phi(X)^\top \hat{U}_r \in \mathbb{R}^r$$

- Minimax-optimal error bounds for recovering P proved in (w Sun, Duan, Gong 2019)

# Some theory

- The diffusion distance between two states is $dist(x, y) = \|p(\cdot \mid x) - p(\cdot \mid y)\|$

- Kernelized state embedding preserves the diffusion distance up to error

$$| dist(x, y) - \|\hat{\Psi}(x) - \hat{\Psi}(x)\| | \leq O\left(\sqrt{\frac{r\kappa t_{mix}}{n}}\right), \quad \forall x, y$$

where r is rank, k is MC's condition number, n is the length of trajectory.

# Finding Metastable State Clusters

- We want to find a partition of the state space such that that *states within the same set shares similar future paths*

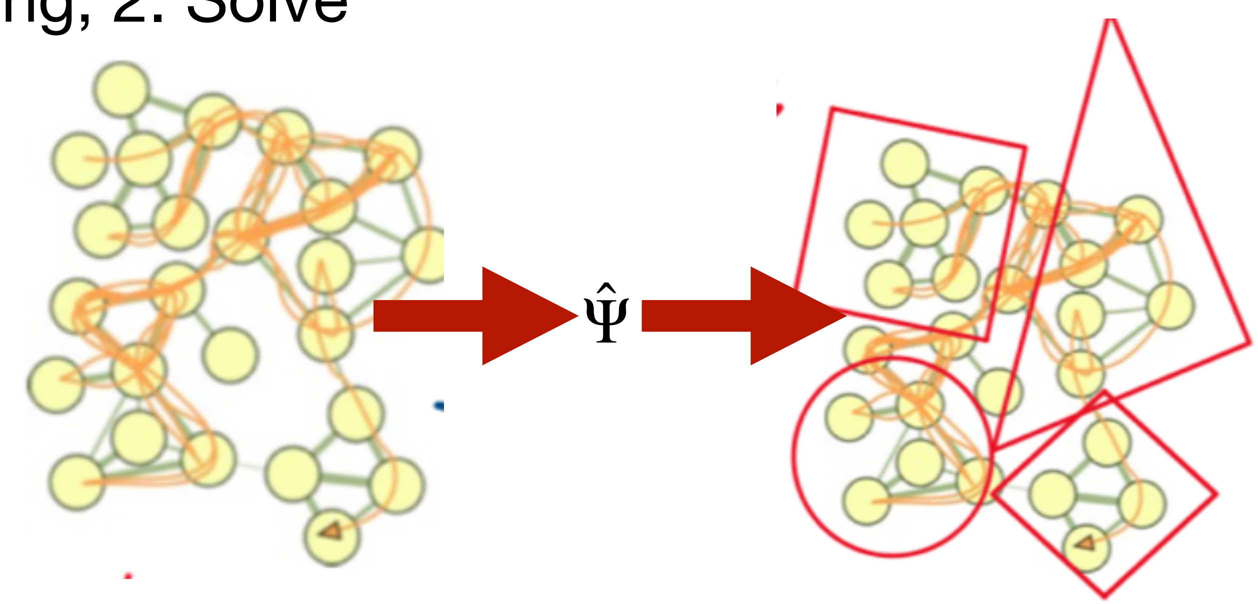
$$\min_{\Omega_1,\cdots,\Omega_m} \min_{q_1,\cdots,q_m} \sum_{i=1}^{m} \int_{\Omega_i} \pi(x) \| p(\cdot \mid x) - q_i(\cdot) \|_{L^2}^2 dx,$$

- If the MC is reversible, the problem finds the optimal metastable partition [E 2008]

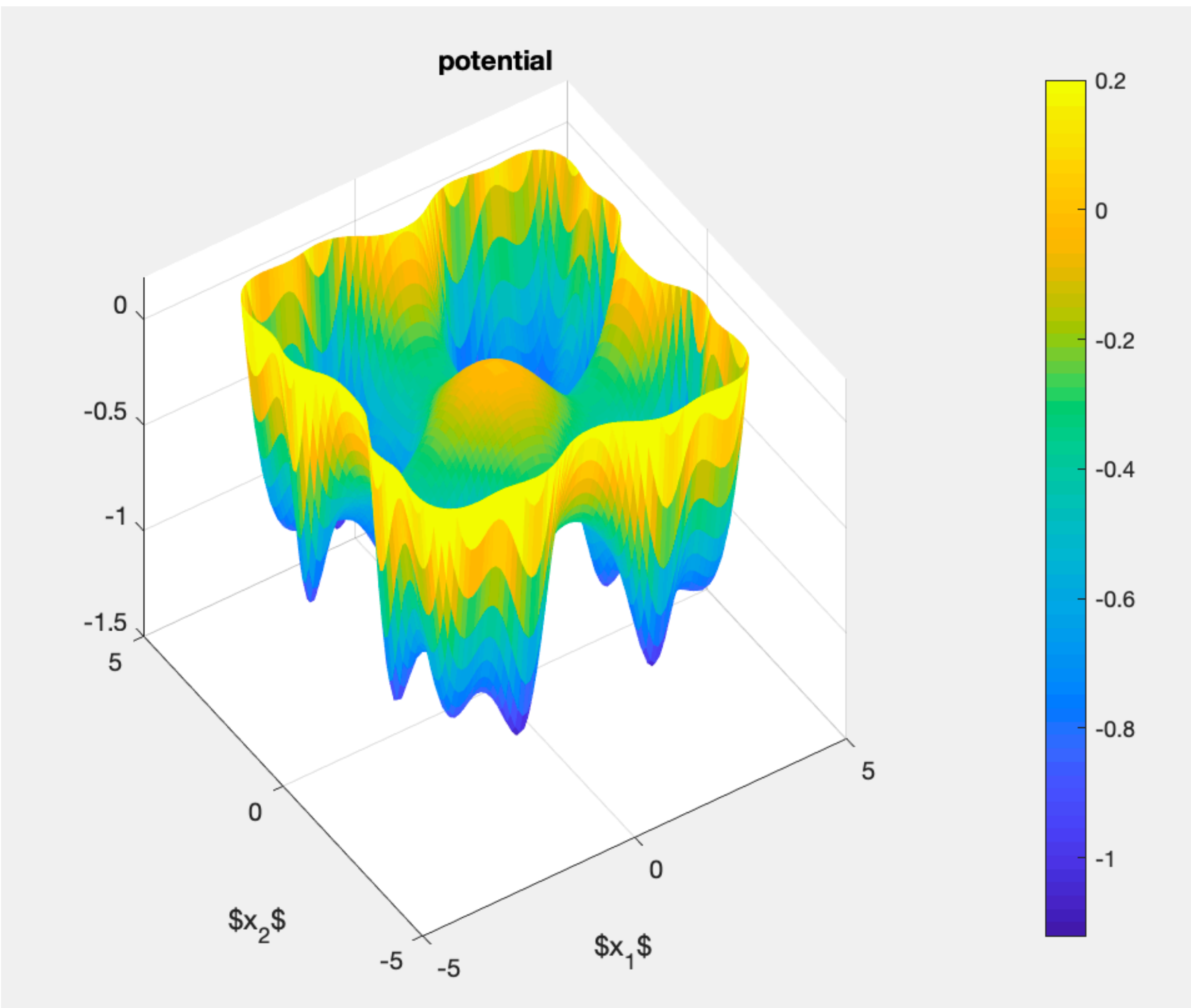
$$(A_1^*, \cdots, A_m^*) = \textbf{argmax}_{A_1,\cdots,A_m} \sum_{k=1}^{m} p(A_k \mid A_k)$$

- **Solution:** 1. Estimate state embedding; 2. Solve

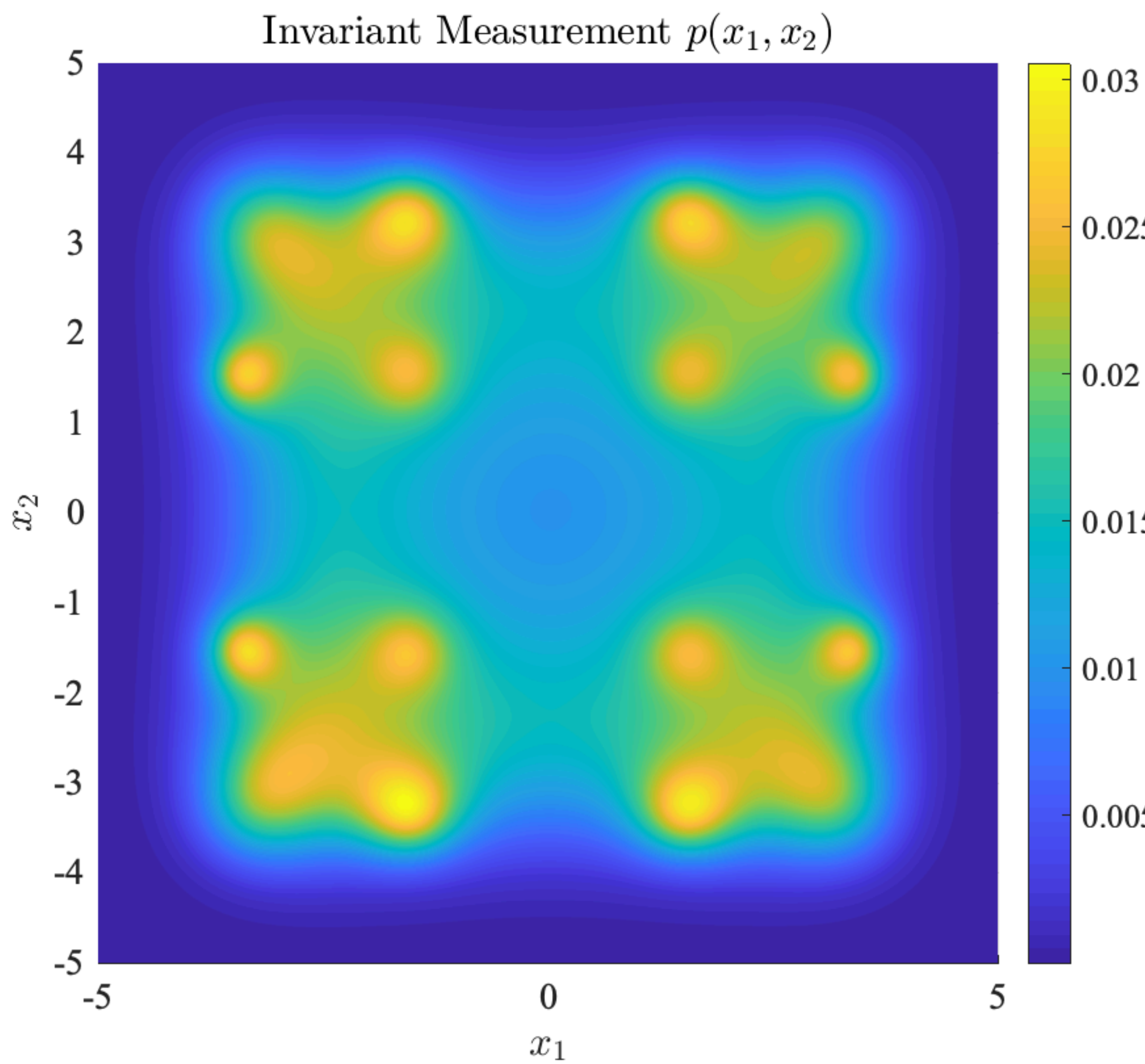
$$\min_{(\Omega_1,\cdots,\Omega_m)} \min_{s_1,\cdots,s_k \in \mathbb{R}^r} \sum_{i=1}^{m} \sum_{i \in [N]} \| \hat{\Psi}(x_i) - s_i \|^2 dx$$



# Example: stochastic diffusion process

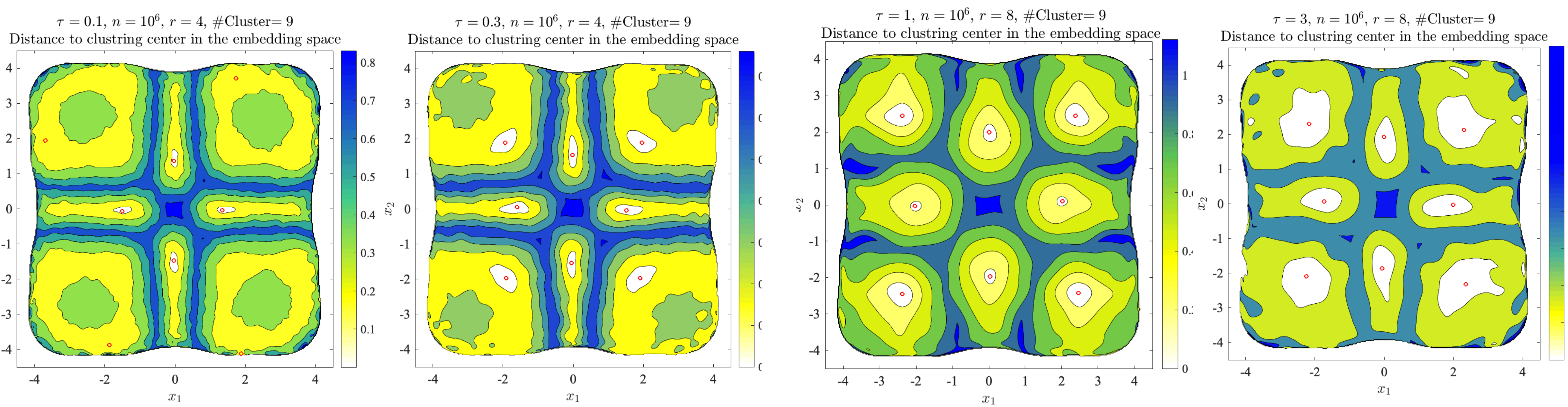



**Potential Function**




**True Invariant Measure**

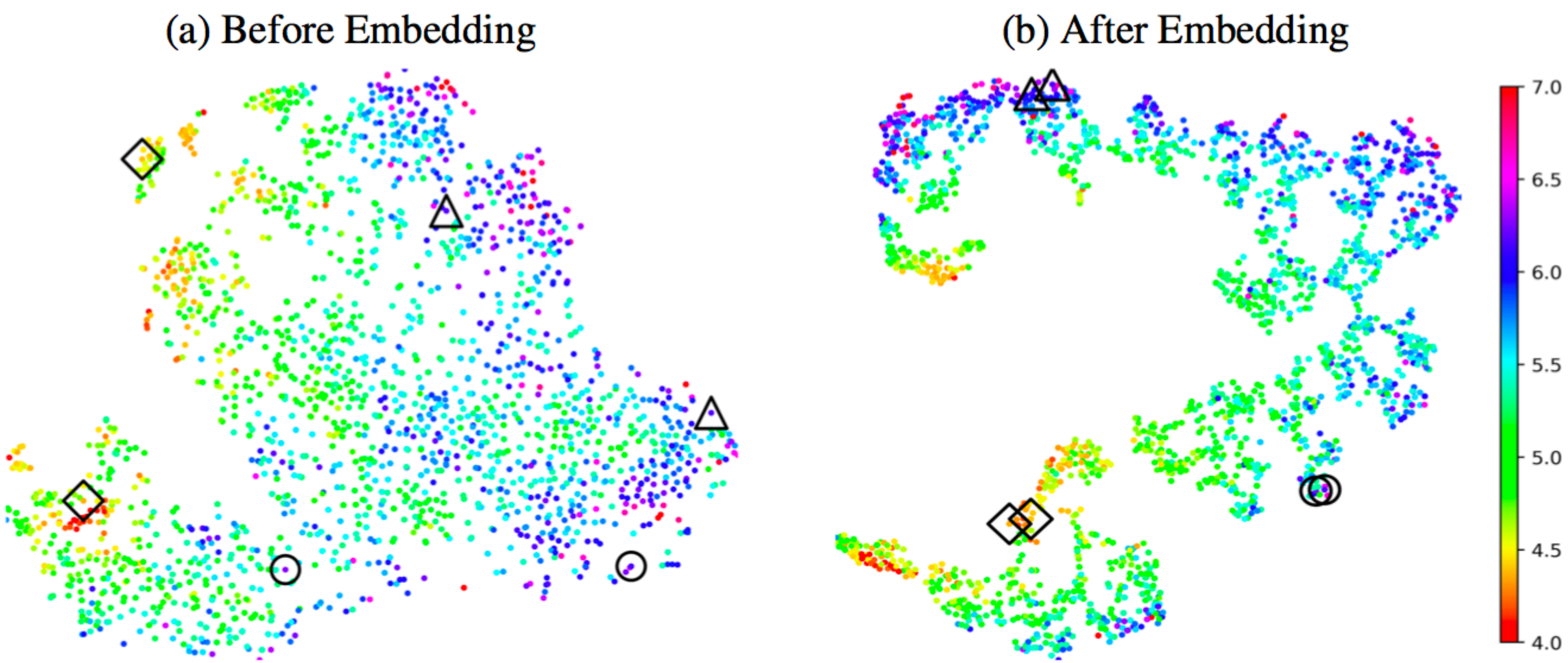
# Metastable clusters learned from $P^t$




**t=0.1** **t=0.3** **t=1** **t=3**

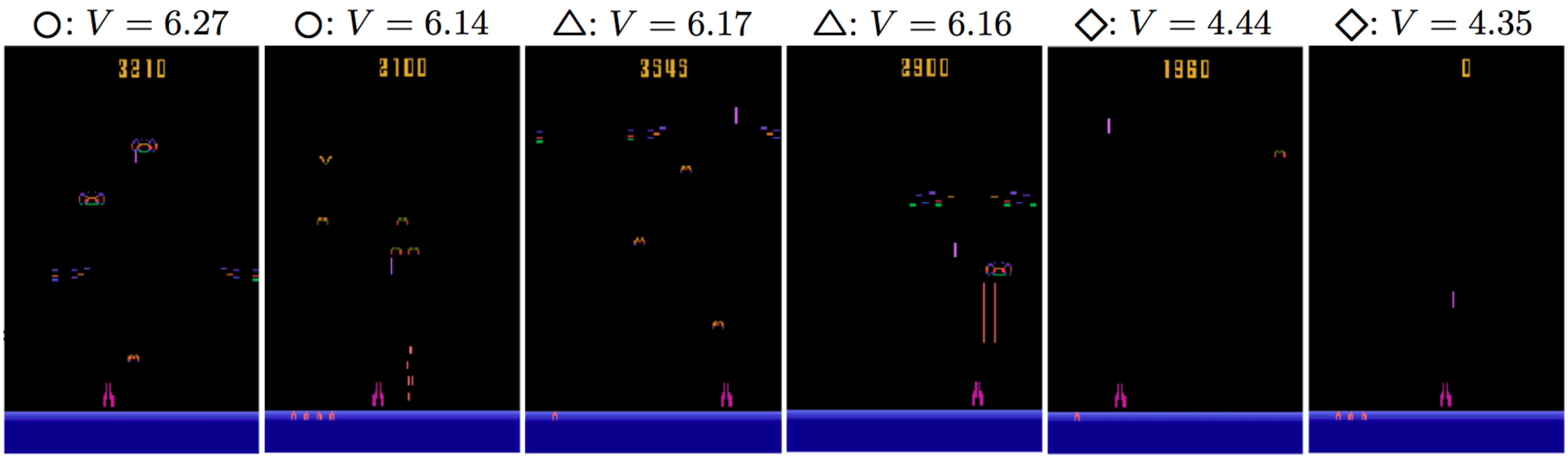
**Learning metastable sets from state trajectories**

# Example: State Trajectories of Demon Attack




**Visualization of game states before and after embedding in t-SNE plots.**

# Game states that are close after embedding



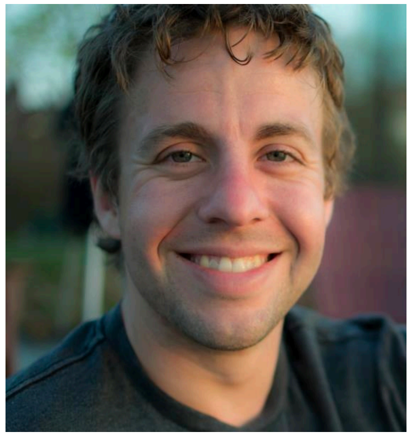

**About to score; both moving to the left**

**New demons appearing**

**Waiting for new targets; moving to center from opposite ends**

State embedding identifies states as similar in low-dim space if they share similar future paths

# Collaborators



Aaron Sidford

Yinyu Ye

Anru Zhang

Lin Yang

Tracy Ke

Yichen Chen

Yaqi Duan

Hao Gong

Yifan Sun

Zeyu Jia

**Thank you!**