Value Prediction with Function Approximation

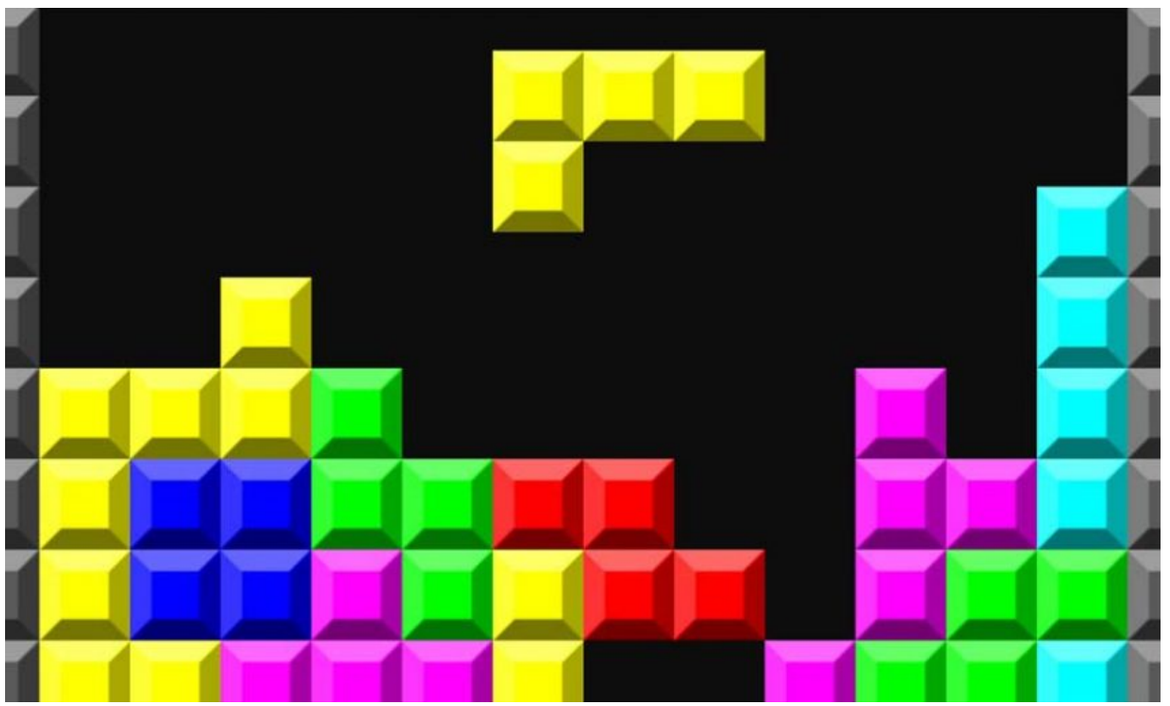
Reading: Algs for RL (Szepesvári), Sec 3.2

Generalization for value prediction

- Major limitation of tabular RL: does not scale to large state space
  - most methods require that we run into the same state multiple times
  - when the state space is large, you might not see the same state even twice!
  - In other words: sample complexity scales with |S|
  - need generalization
- For value prediction problem, generalization requires that we have some prior knowledge about the form the value function
  - linear function approximation: design features $\phi(s) \in \mathbb{R}^d$ ("featurizing states"), and approximate $V^\pi(s) \approx \theta^\top \phi(s)$
  - only unknown: θ. d unknowns vs |S| unknowns!

Example: linear function approximation for tetris

- An example featurization:
  - let the height of the pile in i-th column be the i-th feature
  - dimensionality of feature = #columns
  - (probably doesn't work; just an example)
- Feature engineering requires a lot of prior knowledge, domain insights, and trial and error, just as in supervised learning!



Monte-Carlo Value Prediction

- Draw a starting state s_i from the exploratory initial distribution, roll out a trajectory using π from s_i, and let G_i be the (random) discounted return
- Collect $\{(s_i, G_i)\}$ pairs
- Least square regression: $\mathbf{min}_\theta \frac{1}{n} \sum_{i=1}^{n} (\theta^\top \phi(s_i) - G_i)^2$
- Why this works?
  - Assume $\{(s_i, G_i)\}$ are i.i.d., let (s, G) be variables equal in distribution
  - The expected version of the objective: $\mathbf{min}_\theta \mathbb{E}_{s,G}[(\theta^\top \phi(s) - G)^2]$
  - If we do not restrict ourselves to linear functions, the function that minimizes this objective is $s \mapsto \mathbb{E}[G \mid s]$ $(= V^\pi(s))$!
  - If true $V^\pi(s)$ happens to take linear form, the regression will find it in the limit (of infinite data)
  - Finite sample regime: bias & variance trade-off

Monte-Carlo Value Prediction

- The same idea applies to non-linear value function approximation
- More generally & abstractly, think of function approximation as searching over a restricted **function space**, which is a set whose members are functions that map states to real values.
- Function space of linear value function approximation:
  $\mathcal{F} = \{V_\theta : \theta \in \mathbb{R}^d\}$, where $V_\theta(s) = \theta^\top \phi(s)$
  - typically only a small subset of all possible functions
  - Using "all possible functions" = tabular!
  - Equivalently, tabular MC value prediction can be recovered by choosing ϕ as the identity features $\phi(s) = \{\mathbb{I}[s = s']\}_{s' \in S}$
- $\min_{V_\theta \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} (V_\theta(s_i) - G_i)^2$
- Plug in any function approximator of your choice
- SGD: uniformly sample i and $\theta \leftarrow \theta - \alpha \cdot (V_\theta(s_i) - G_i) \cdot \nabla V_\theta(s_i)$

TD(0) with function approximation

- tabular: $V(s_t) \leftarrow V(s_t) + \alpha(r_t + \gamma V(s_{t+1}) - V(s_t))$
- When we update $V(s_t)$, the target is $r_t + \gamma V(s_{t+1})$
- Batch version of the algorithm: one Bellman update can be approximated (using all data) as

$$V_{k+1} \leftarrow \arg\min_{V_\theta \in \mathcal{F}} \frac{1}{T} \sum_{t=1}^{T} (V_\theta(s_t) - r - \gamma V_k(s_{t+1}))^2$$

- SGD + "no-wait": $\theta \leftarrow \theta - \alpha \cdot (V_\theta(s_t) - r - \gamma V_\theta(s_{t+1})) \cdot \nabla V_\theta(s_t)$
- When using linear function approximation $V_\theta(s) = \phi(s)^\top \theta$, we have $\theta \leftarrow \theta - \alpha \cdot (\phi(s_t)^\top \theta - r - \gamma \phi(s_{t+1})^\top \theta) \cdot \phi(s_t)$
- When using chain rule, we only take gradient on $V_\theta(s_t)$ and ignore $V_\theta(s_{t+1})$; the latter is treated as a constant (it plays the role of V_k)

What if…? (not required)

- What happens if we also differentiate $V_\theta(s_{t+1})$?
- This corresponds to $\arg\min_{V_\theta \in \mathcal{F}} \sum_{(s,r,s')} (V_\theta(s) - r - \gamma V_\theta(s'))^2$
  - no iteration anymore; a clean optimization objective
  - (most RL algorithms with bootstrapped target do not have a fixed optimization objective; objective changes over time)
- Assume for simplicity that, each data point is generated by (1) sampling s i.i.d. from some exploratory distribution, and (2) generating r and s' conditioned on $(s, \pi(s))$
- Replacing empirical objective with the population version, the objective becomes $\mathbb{E}_{s,r,s'}[(V_\theta(s) - r - \gamma V_\theta(s'))^2]$

What if…? (not required)

- $\mathbb{E}_{s,r,s'}[(V_\theta(s) - r - \gamma V_\theta(s'))^2]$ can be decomposed into two terms
  - First term: $\mathbb{E}_s[(V_\theta(s) - (\mathcal{T}^\pi V_\theta)(s))^2]$
    - This is good! measures how much V_θ violates Bellman eq
    - A version of Bellman error $\|V - \mathcal{T}^\pi V\|$
  - Second term: $\gamma^2 \mathbb{E}_s[\mathrm{Var}_{s'|s,\pi(s)}[V_\theta(s')]]$
    - (assumes deterministic rewards)
    - This is bad! An additional term that penalizes functions that has large variance w.r.t. random state transitions
    - Special case: 0 when environment is deterministic
- So it's actually a sensible algorithm for deterministic environments, but doesn't work when stochasticity is significant

Resolutions (not required)

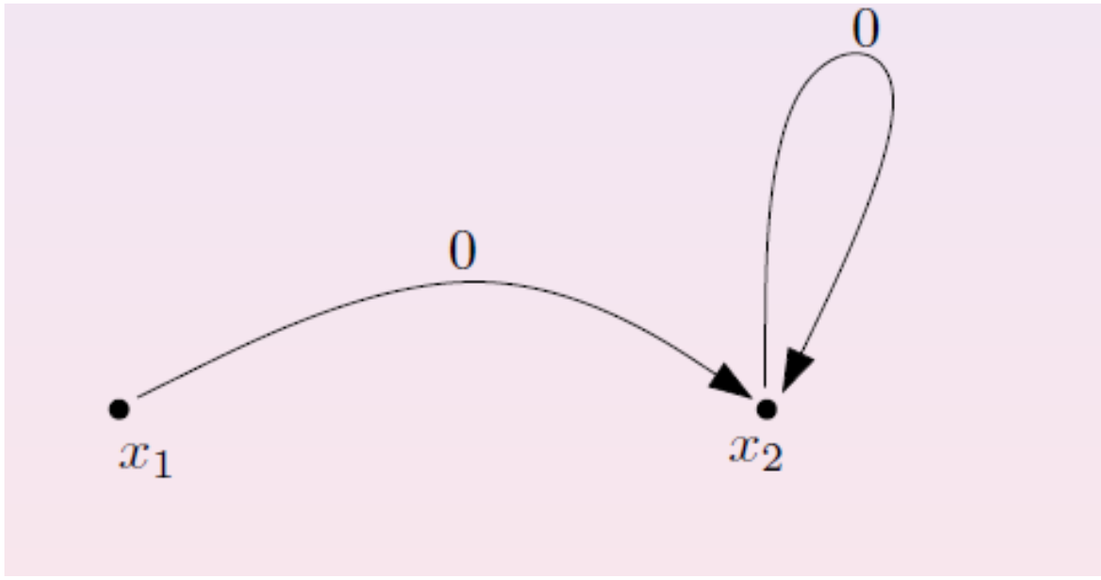
- If we have a simulator…
  - For each s in data, draw another independent state transition
  - Minimize objective $\mathbb{E}[(V_\theta(s) - r - \gamma V_\theta(s'_A))(V_\theta(s) - r - \gamma V_\theta(s'_B)]$
  - "Double sampling" and Baird's residual algorithm (Bellman residual minimization)
  - Exercise: do you need to double sample the reward if reward is stochastic?
  - The conditional variance term is eliminated by double sampling
- If we can only draw 1 next-state (as with any natural data generation process)…
  - Estimate the conditional variance term and subtract from the objective
  - A minimax formulation (not covered in this course)
  - For further readings, see 598 slides on FQI.

Convergence?

- TD with function approximation can diverge in general
- Is it because of…
  - Randomness in SGD?
    - Nope. Even the batch version doesn't converge.
  - Sophisticated, non-linear func approx?
    - Nope. Even linear doesn't converge.
    - That our function class does not capture V^π?
      - Nope. Even if V^π can be exactly represented in the function class ("realizable"), it still does not converge.
  - 

2.1 Counter-example for least-square regression [Tsitsiklis and van Roy, 1996]

An MDP with two states x_1, x_2, 1-d features for the two states: $f_{x_1} = 1, f_{x_2} = 2$. Linear Function approximation with $\tilde{V}_\theta(x) = \theta f_x$.



$$
\begin{aligned}
\theta_k &:= \arg\min_\theta \frac{1}{2}(\theta - \text{target}_1)^2 + (2\theta - \text{target}_2)^2 \\
&= \arg\min_\theta \frac{1}{2}(\theta - \gamma\theta^{k-1} f_{x_2})^2 + (2\theta - \gamma\theta^{k-1} f_{x_2})^2 \\
&= \arg\min_\theta \frac{1}{2}(\theta - \gamma 2\theta^{k-1})^2 + (2\theta - \gamma 2\theta^{k-1})^2
\end{aligned}
$$

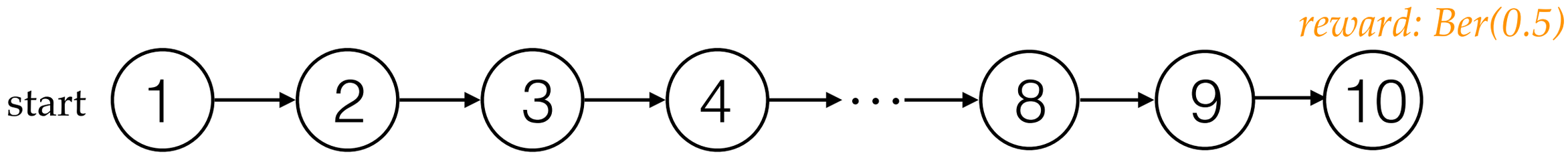
$$
(\theta - \gamma 2\theta^{k-1}) + 2(2\theta - \gamma 2\theta^{k-1}) = 0 \Rightarrow 5\theta = 6\gamma\theta^{k-1}
$$

$$
\theta_k = \frac{6}{5}\gamma\theta_{k-1}
$$

This diverges if $\gamma \geq 5/6$.

credit: course notes from Shipra Agrawal

A simple example (finite horizon, γ=1)



Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

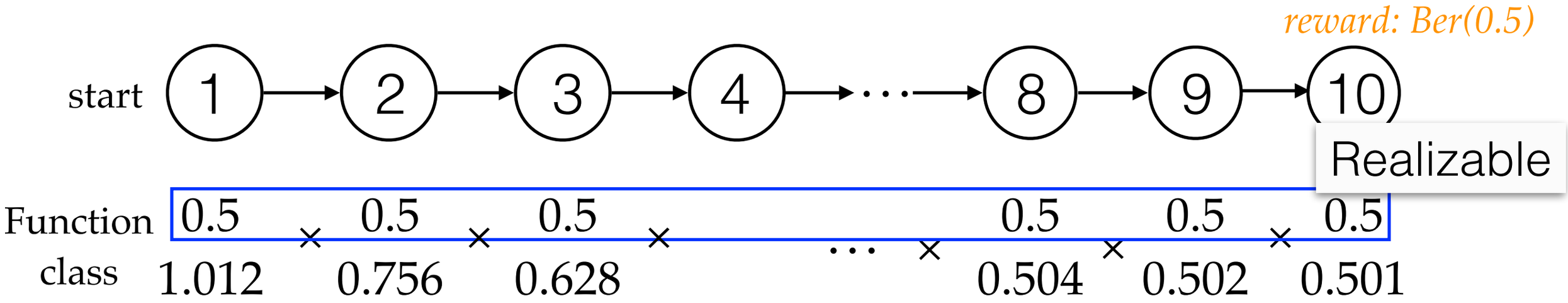
Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.501 0.501

…

Iter #10: 0.501 0.501 0.501 0.501 … 0.501 0.501 0.501

- Dataset $D = \{(s, r, s')\}$ looks like:
  {(①, 0, ②), (②, 0,③), …, (⑩, 1, *end*), …, (⑩, 0, *end*)}

How things go wrong (w/ restricted class)



Iter #1: Data: (⑩, 1, *end*), …, (⑩, 0, *end*) ⇨ 0.501

Iter #2: Data: (⑨, 0, ⑩) ⇨ (⑨, 0+0.501) ⇨ 0.502 0.501

…

Iter #10: !!! 1.012 0.756 0.628 … 0.502 0.501

Example given in Dann et al'18

Non-convergence

- Why things go wrong?
- Bellman update is a contraction, but here we have an additional projection step: $V_{k+1} \leftarrow \Pi_{\mathcal{F}} \mathcal{T}^{\pi} V_k$, projected Bellman update may NOT be a contraction (even with linear function approximation)
  - it is still a contraction in some special cases; will see
- In other words: in each iter, we solve a regression problem where the target function is $T^{\pi}V$, where V can be arbitrary function in $\mathcal{F}$
- The fact that $V^{\pi} \in \mathcal{F}$ does not imply that $T^{\pi}V$ is in $\mathcal{F}$! We may do quite poorly in the regression problem, and the iteration does not mimic a Bellman update
- Why tabular is fine? $\mathcal{F}$ is fully expressive so $T^{\pi}V$ is always in $\mathcal{F}$.
  - Similarly for func approx, if we assume that $\mathcal{F}$ is closed under T^{π}, can prove some good properties of TD
- All alg based on bootstrapped targets suffer from this issue
  - Compare to the behavior of Monte-Carlo