CS 747, Autumn 2020: Week 12, Lecture 1

Shivaram Kalyanakrishnan

Department of Computer Science and Engineering
Indian Institute of Technology Bombay

Autumn 2020

Reinforcement Learning

1. Policy gradient methods
2. Variance reduction in policy gradient methods
3. Batch reinforcement learning

Reinforcement Learning

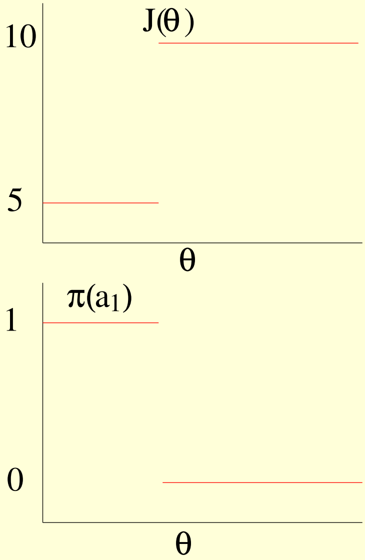
1. Policy gradient methods
2. Variance reduction in policy gradient methods
3. Batch reinforcement learning

Stochastic Policies

- Single state; actions a_1, a_2.
- $R(a_1) = 5$; $R(a_2) = 10$.
- Policy π; parameter θ.

$$\pi(a_1) = \begin{cases} 1 & \text{if } \theta < 0.6, \\ 0 & \text{otherwise.} \end{cases}$$

$$J(\theta) = \pi(a_1) \cdot 5 + \pi(a_2) \cdot 10.$$



Stochastic Policies

- Single state; actions a_1, a_2.
- $R(a_1) = 5$; $R(a_2) = 10$.
- Policy π; parameter θ.

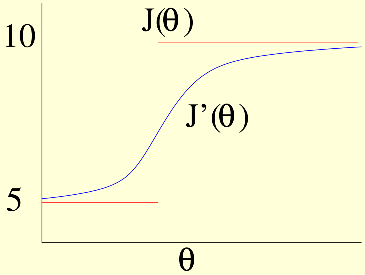
$$\pi(a_1) = \begin{cases} 1 & \text{if } \theta < 0.6, \\ 0 & \text{otherwise.} \end{cases}$$

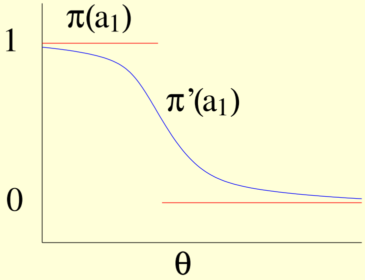
$$J(\theta) = \pi(a_1) \cdot 5 + \pi(a_2) \cdot 10.$$

- Policy π'; parameter θ.

$$\pi'(a_1) = \frac{1}{1 + e^{\theta - 0.6}}.$$

$$J'(\theta) = \pi'(a_1) \cdot 5 + \pi'(a_2) \cdot 10.$$





Idea

- If π is differentiable w.r.t. θ, so is (scalar) "policy value" J.

Idea

- If π is differentiable w.r.t. θ, so is (scalar) "policy value" J.
- We can "search" for "good" θ by iterating:

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} + \alpha \nabla_{\theta} J(\theta_{\text{old}}).$$

Idea

- If π is differentiable w.r.t. θ, so is (scalar) "policy value" J.
- We can "search" for "good" θ by iterating:

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} + \alpha \nabla_\theta J(\theta_{\text{old}}).$$

- **Example.** If we have features $x(s, a) \in \mathbb{R}^d$ for $s \in S, a \in A$, a common template for π is:

$$\pi(s, a) = \frac{e^{\theta \cdot x(s,a)}}{\sum_{b \in A} e^{\theta \cdot x(s,b)}},$$

where $\theta \in \mathbb{R}^d$ is the vector of policy parameters.

Idea

- If π is differentiable w.r.t. θ, so is (scalar) "policy value" J.
- We can "search" for "good" θ by iterating:

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} + \alpha \nabla_\theta J(\theta_{\text{old}}).$$

- **Example.** If we have features $x(s, a) \in \mathbb{R}^d$ for $s \in S, a \in A$, a common template for π is:

$$\pi(s, a) = \frac{e^{\theta \cdot x(s,a)}}{\sum_{b \in A} e^{\theta \cdot x(s,b)}},$$

where $\theta \in \mathbb{R}^d$ is the vector of policy parameters.
In this case, work out that

$$\nabla_\theta \pi(s, a) = \left(x(s, a) - \sum_{b \in B} \pi(s, b) x(s, b) \right) \pi(s, a).$$

Idea

- If π is differentiable w.r.t. θ, so is (scalar) "policy value" J.
- We can "search" for "good" θ by iterating:

$$\theta_{\text{new}} \leftarrow \theta_{\text{old}} + \alpha \nabla_\theta J(\theta_{\text{old}}).$$

- **Example.** If we have features $x(s, a) \in \mathbb{R}^d$ for $s \in S, a \in A$, a common template for π is:

$$\pi(s, a) = \frac{e^{\theta \cdot x(s,a)}}{\sum_{b \in A} e^{\theta \cdot x(s,b)}},$$

  where $\theta \in \mathbb{R}^d$ is the vector of policy parameters.
  In this case, work out that

$$\nabla_\theta \pi(s, a) = \left(x(s, a) - \sum_{b \in B} \pi(s, b) x(s, b) \right) \pi(s, a).$$

- But what's the connection between $\nabla_\theta J$ and $\nabla_\theta \pi(\cdot, \cdot)$?

Policy Gradient Theorem

- For simplicity assume episodic task with $\gamma = 1$.
- Assume there is a fixed start state s^0.
- We leave it implicit that π is fixed by parameter vector θ.
- $J(\theta) = V^\pi(s^0)$.
- We shall derive the connection between $\nabla_\theta J$ and $\nabla_\theta \pi(\cdot, \cdot)$.

Policy Gradient Theorem

For $s \in S, \nabla_\theta V^\pi(s) = \nabla_\theta \sum_{a \in A} \pi(s, a) Q^\pi(s, a)$

Policy Gradient Theorem

For $s \in S, \nabla_\theta V^\pi(s) = \nabla_\theta \sum_{a \in A} \pi(s, a) Q^\pi(s, a)$

$$= \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a)$$

$$+ \sum_{a \in A} \pi(s, a) \nabla_\theta \sum_{s' \in S} T(s, a, s')(R(s, a, s') + V^\pi(s'))$$

Policy Gradient Theorem

For $s \in S$, $\nabla_\theta V^\pi(s) = \nabla_\theta \sum_{a \in A} \pi(s,a) Q^\pi(s,a)$

$$= \sum_{a \in A} \nabla_\theta \pi(s,a) Q^\pi(s,a)$$

$$+ \sum_{a \in A} \pi(s,a) \nabla_\theta \sum_{s' \in S} T(s,a,s')(R(s,a,s') + V^\pi(s'))$$

$$= \sum_{a \in A} \left[\nabla_\theta \pi(s,a) Q^\pi(s,a) + \pi(s,a) \sum_{s' \in S} T(s,a,s') \nabla_\theta V^\pi(s') \right]$$

Policy Gradient Theorem

For $s \in S, \nabla_\theta V^\pi(s) = \nabla_\theta \sum_{a \in A} \pi(s,a) Q^\pi(s,a)$

$$= \sum_{a \in A} \nabla_\theta \pi(s,a) Q^\pi(s,a)$$

$$+ \sum_{a \in A} \pi(s,a) \nabla_\theta \sum_{s' \in S} T(s,a,s')(R(s,a,s') + V^\pi(s'))$$

$$= \sum_{a \in A} \left[\nabla_\theta \pi(s,a) Q^\pi(s,a) + \pi(s,a) \sum_{s' \in S} T(s,a,s') \nabla_\theta V^\pi(s') \right]$$

$$= \cdots = \sum_{x \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s \to x, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(x,a) Q^\pi(x,a),$$

Policy Gradient Theorem

For $s \in S, \nabla_\theta V^\pi(s) = \nabla_\theta \sum_{a \in A} \pi(s,a) Q^\pi(s,a)$

$$= \sum_{a \in A} \nabla_\theta \pi(s,a) Q^\pi(s,a)$$

$$+ \sum_{a \in A} \pi(s,a) \nabla_\theta \sum_{s' \in S} T(s,a,s')(R(s,a,s') + V^\pi(s'))$$

$$= \sum_{a \in A} \left[\nabla_\theta \pi(s,a) Q^\pi(s,a) + \pi(s,a) \sum_{s' \in S} T(s,a,s') \nabla_\theta V^\pi(s') \right]$$

$$= \cdots = \sum_{x \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s \to x, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(x,a) Q^\pi(x,a),$$

where $\mathbb{P}\{s \to x, k, \pi\}$ is the probability of reaching x from s in k steps if following π.

Policy Gradient Theorem

- Recall that $J(\theta) = V^{\pi}(s^0)$.

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^{\pi}(s, a).$$

Policy Gradient Theorem

- Recall that $J(\theta) = V^{\pi}(s^0)$.

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^{\pi}(s, a).$$

- But how to do gradient ascent? We do not know $\mathbb{P}\{s^0 \to s, k, \pi\}$ and $Q^{\pi}(s, a)$!

Policy Gradient Theorem

- Recall that $J(\theta) = V^{\pi}(s^0)$.

$$\nabla_{\theta} J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_{\theta} \pi(s, a) Q^{\pi}(s, a).$$

- But how to do gradient ascent? We do not know $\mathbb{P}\{s^0 \to s, k, \pi\}$ and $Q^{\pi}(s, a)$!
- We perform stochastic gradient ascent.
- We use the following fact. For any discrete, real-valued random variable X with pmf $p : X \to [0, 1]$,

$$\sum_{x \in X} p(x) x = \mathbb{E}[X].$$

Towards Gradient Ascent

- Generate episode $s^0, a^0, r^0, s^1, a^1, r^1, s^2, \dots, s^T = s^\top$ by acting according to π, parameterised by θ.

Towards Gradient Ascent

- Generate episode $s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^T = s^\top$ by acting according to π, parameterised by θ. Now observe:

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a)$$

Towards Gradient Ascent

- Generate episode $s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^T = s^\top$ by acting according to π, parameterised by θ. Now observe:

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a)$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \sum_{a \in A} \nabla_\theta \pi(s^t, a) Q^\pi(s^t, a) \right]$$

Towards Gradient Ascent

- Generate episode $s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^T = s^\top$ by acting according to π, parameterised by θ. Now observe:

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a)$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \sum_{a \in A} \nabla_\theta \pi(s^t, a) Q^\pi(s^t, a) \right]$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \sum_{a \in A} \pi(s^t, a) \frac{\nabla_\theta \pi(s^t, a)}{\pi(s^t, a)} Q^\pi(s^t, a) \right]$$

Towards Gradient Ascent

- Generate episode $s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^T = s^\top$ by acting according to π, parameterised by θ. Now observe:

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a)$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \sum_{a \in A} \nabla_\theta \pi(s^t, a) Q^\pi(s^t, a) \right]$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \sum_{a \in A} \pi(s^t, a) \frac{\nabla_\theta \pi(s^t, a)}{\pi(s^t, a)} Q^\pi(s^t, a) \right]$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \frac{\nabla_\theta \pi(s^t, a^t)}{\pi(s^t, a^t)} Q^\pi(s^t, a^t) \right]$$

Towards Gradient Ascent

- Generate episode $s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^T = s^\top$ by acting according to π, parameterised by θ. Now observe:

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a)$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \sum_{a \in A} \nabla_\theta \pi(s^t, a) Q^\pi(s^t, a) \right]$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \sum_{a \in A} \pi(s^t, a) \frac{\nabla_\theta \pi(s^t, a)}{\pi(s^t, a)} Q^\pi(s^t, a) \right]$$

$$= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \frac{\nabla_\theta \pi(s^t, a^t)}{\pi(s^t, a^t)} Q^\pi(s^t, a^t) \right] = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \frac{\nabla_\theta \pi(s^t, a^t)}{\pi(s^t, a^t)} G_{t:T} \right]$$

Towards Gradient Ascent

- Generate episode $s^0, a^0, r^0, s^1, a^1, r^1, s^2, \ldots, s^T = s^\top$ by acting according to π, parameterised by θ. Now observe:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a) \\
&= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \sum_{a \in A} \nabla_\theta \pi(s^t, a) Q^\pi(s^t, a) \right] \\
&= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \sum_{a \in A} \pi(s^t, a) \frac{\nabla_\theta \pi(s^t, a)}{\pi(s^t, a)} Q^\pi(s^t, a) \right] \\
&= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \frac{\nabla_\theta \pi(s^t, a^t)}{\pi(s^t, a^t)} Q^\pi(s^t, a^t) \right] = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \frac{\nabla_\theta \pi(s^t, a^t)}{\pi(s^t, a^t)} G_{t:T} \right] \\
&= \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \left(\nabla_\theta \ln \pi(s^t, a^t) \right) G_{t:T} \right].
\end{aligned}
$$

REINFORCE Algorithm

- Reference: Williams (1992).
- For clarity we show explicit dependence of π on parameter vector $\theta \in \mathbb{R}^d$.
- Assume θ is initialised arbitrarily.

Repeat for ever:

$\quad \theta_{\text{new}} \leftarrow \theta$.

$\quad$ Generate episode $s^0, a^0, r^0, s^1, \ldots, s^T = s^\top$, following π_θ.

$\quad$ **For** $t = 0, 1, \ldots, T-1$:

$\quad\quad G \leftarrow \sum_{k=t}^{T-1} r^k$. //This is $G_{t:T}$.

$\quad\quad \theta_{\text{new}} \leftarrow \theta_{\text{new}} + \alpha G \nabla_\theta \ln \pi_\theta(s^t, a^t)$.

$\quad \theta \leftarrow \theta_{\text{new}}$.

Reinforcement Learning

1. Policy gradient methods
2. Variance reduction in policy gradient methods
3. Batch reinforcement learning

Baseline Subtraction

- Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a).$$

Baseline Subtraction

- Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a).$$

- Let $B : S \to \mathbb{R}$ be an arbitrary function of state.

Baseline Subtraction

- Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a).$$

- Let $B : S \to \mathbb{R}$ be an arbitrary function of state. We claim

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) (Q^\pi(s, a) - B(s)).$$

- How come?

Baseline Subtraction

- Policy Gradient Theorem

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) Q^\pi(s, a).$$

- Let $B : S \to \mathbb{R}$ be an arbitrary function of state. We claim

$$\nabla_\theta J(\theta) = \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a)(Q^\pi(s, a) - B(s)).$$

- How come? Observe that

$$\sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} \sum_{a \in A} \nabla_\theta \pi(s, a) B(s)$$

$$= \sum_{s \in S} \sum_{k=0}^{\infty} \mathbb{P}\{s^0 \to s, k, \pi\} B(s) \nabla_\theta \sum_{a \in A} \pi(s, a) = 0.$$

Baseline Subtraction

- The policy gradient estimate can have high variance.

s	$Q^\pi(s, a_1)$	$Q^\pi(s, a_2)$	$Q^\pi(s, a_3)$	$V^\pi(s)$
s_1	105	79	100	90
s_2	10	6	13	12
s_3	-50	-60	-50	-55

Baseline Subtraction

- The policy gradient estimate can have high variance.

s	$Q^{\pi}(s, a_1)$	$Q^{\pi}(s, a_2)$	$Q^{\pi}(s, a_3)$	$V^{\pi}(s)$
s_1	105	79	100	90
s_2	10	6	13	12
s_3	-50	-60	-50	-55

- Common practice to subtract out $V^{\pi}(s)$—approximated independently as $\hat{V}(s)$.

- REINFORCE with baseline:

$$\theta_{\text{new}} \leftarrow \theta_{\text{new}} + \alpha \sum_{t=0}^{T-1} (G_{t:T} - \hat{V}(s^t)) \nabla_{\theta} \ln \pi_{\theta}(s^t, a^t).$$

Actor-critic Methods

- Even for fixed (s^t, a^t), can have high variance in $G_{t:T}$.

Actor-critic Methods

- Even for fixed (s^t, a^t), can have high variance in $G_{t:T}$.
- One approach is to do gradient ascent after averaging the gradient from a few episodes.

Actor-critic Methods

- Even for fixed (s^t, a^t), can have high variance in $G_{t:T}$.
- One approach is to do gradient ascent after averaging the gradient from a few episodes.
- Another approach is to bootstrap: to use $r^t + \hat{V}(s^{t+1})$ in place of $G_{t:T}$, where $\hat{V}(s^{t+1})$ is estimated independently.

Actor-critic Methods

- Even for fixed (s^t, a^t), can have high variance in $G_{t:T}$.
- One approach is to do gradient ascent after averaging the gradient from a few episodes.
- Another approach is to bootstrap: to use $r^t + \hat{V}(s^{t+1})$ in place of $G_{t:T}$, where $\hat{V}(s^{t+1})$ is estimated independently.
- Called the Actor-Critic architecture.

- Actor updates θ and hence π_θ.
- Critic evaluates π_θ (say using TD(0)) and provides input for the gradient ascent update.

$$\theta_{\text{new}} \leftarrow \theta_{\text{new}} + \alpha \sum_{t=0}^{T-1} (r^t + \hat{V}(s^{t+1}) - \hat{V}(s^t)) \nabla_\theta \ln \pi_\theta(s^t, a^t).$$

Actor-critic Methods

- Even for fixed (s^t, a^t), can have high variance in $G_{t:T}$.
- One approach is to do gradient ascent after averaging the gradient from a few episodes.
- Another approach is to bootstrap: to use $r^t + \hat{V}(s^{t+1})$ in place of $G_{t:T}$, where $\hat{V}(s^{t+1})$ is estimated independently.
- Called the Actor-Critic architecture.

\- Actor updates θ and hence π_θ.

\- Critic evaluates π_θ (say using TD(0)) and provides input for the gradient ascent update.

$$\theta_{\text{new}} \leftarrow \theta_{\text{new}} + \alpha \sum_{t=0}^{T-1} (r^t + \hat{V}(s^{t+1}) - \hat{V}(s^t)) \nabla_\theta \ln \pi_\theta(s^t, a^t).$$

- Not always provably convergent, but widely used in practice.

Reinforcement Learning

1. Policy gradient methods
2. Variance reduction in policy gradient methods
3. Batch reinforcement learning

Batch Updates to $\hat{Q}$

- We are back to value function-based learning.

Batch Updates to $\hat{Q}$

- We are back to value function-based learning.
- On-line methods such as TD(0) “extract” very little information from each transition; are computationally lightweight.

Batch Updates to $\hat{Q}$

- We are back to value function-based learning.
- On-line methods such as TD(0) “extract” very little information from each transition; are computationally lightweight.
- In many applications, samples are more expensive than computation; need to get more out of samples.

Batch Updates to $\hat{Q}$

- We are back to value function-based learning.
- On-line methods such as TD(0) “extract” very little information from each transition; are computationally lightweight.
- In many applications, samples are more expensive than computation; need to get more out of samples.
- Batch RL keeps transitions in memory, performs computationally heavier updates.

Batch RL outer loop

$\hat{Q} \leftarrow 0$, $D \rightarrow \emptyset$.

Repeat for ever: //Each iteration is a batch.

$\pi \leftarrow \epsilon$-greedy($\hat{Q}$).

Follow π for N episodes; gather data $D' = (s_i, a_i, r_i, s_{i+1})_{i=1}^{L}$.

$D \leftarrow D \cup D'$.

$\hat{Q} \leftarrow$ BatchUpdate(D, $\hat{Q}$).//$\hat{Q}$ optional in RHS.

Experience Replay

- Reference: Lin (1992).

BatchUpdateExperienceReplay$(D, \hat{Q})$

Repeat M times:
  Pick (s, a, r, s') uniformly at random from D.
  $\hat{Q}(s, a) \leftarrow \hat{Q}(s, a) + \alpha\{r + \gamma \max_{a' \in A} \hat{Q}(s', a') - \hat{Q}(s, a)\}$.
Return $\hat{Q}$.

- Sometimes $\hat{Q}$ reset/forgotten before the batch update.
- M usually large; hence multiple updates using each sample.

Fitted Q Iteration

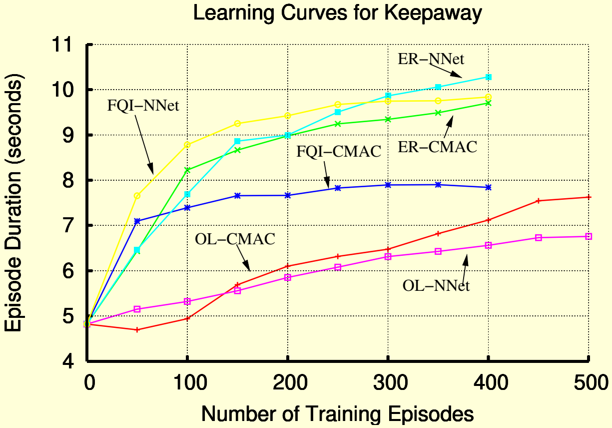
- Reference: Ernst, Geurts, Wehenkel (2005).
- Idea: obtain $\hat{Q}$ using supervised learning. Wait—labels?

BatchUpdateFittedQIteration(D)

$\hat{Q}_0 \leftarrow \mathbf{0}$.
For $i = 0, 1, \ldots, H - 1$:
  For $j \in \{1, 2, \ldots, L\}$: //Create a labeled data set.
    $x_j \leftarrow \text{FeatureVector}(s_j, a_j)$.
    $y_j \leftarrow r_j + \gamma \max_{a \in A} \hat{Q}_i(s_{j+1}, a)$.
  $\hat{Q}_{i+1} \leftarrow \text{SupervisedLearning}((x_j, y_j)_{j=1}^{L})$.
Return $\hat{Q}_H$.

- Will not diverge if the supervised learning model is an averager (nearest neighbour methods, decision trees, etc.).

Illustrative Graph



Batch Reinforcement Learning in a Complex Domain. Shivaram Kalyanakrishnan and Peter Stone, In Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007), pp.650–657, IFAAMAS, 2007.

Batch RL: Summary

- High computational complexity, low sample complexity.
- Can also be interpreted as a model-based approach (the data set D implicitly represents the model).
- Forms the basis for many modern neural network-based algorithms, such as DQN.
- Many variations possible
  - Gathering multiple batches of data in parallel.
  - Picking experience replay samples more intelligently.

Why So Many RL Methods?

- We have seen
  - Model-based RL,
  - On-line TD methods (Q-learning, Sarsa, Expected Sarsa),
  - Policy search (black box optimisation),
  - Policy gradient and actor-critic methods,
  - Batch RL methods.

Why So Many RL Methods?

- We have seen
  - Model-based RL,
  - On-line TD methods (Q-learning, Sarsa, Expected Sarsa),
  - Policy search (black box optimisation),
  - Policy gradient and actor-critic methods,
  - Batch RL methods.
- There is no single winner among these!

Why So Many RL Methods?

- We have seen
  - Model-based RL,
  - On-line TD methods (Q-learning, Sarsa, Expected Sarsa),
  - Policy search (black box optimisation),
  - Policy gradient and actor-critic methods,
  - Batch RL methods.
- There is no single winner among these!

  Effectiveness on a particular task depends on many factors: quality of features, type of representation, task horizon, state aliasing, constraints on computation and memory, etc.

Why So Many RL Methods?

- We have seen
  - Model-based RL,
  - On-line TD methods (Q-learning, Sarsa, Expected Sarsa),
  - Policy search (black box optimisation),
  - Policy gradient and actor-critic methods,
  - Batch RL methods.
- There is no single winner among these!

  Effectiveness on a particular task depends on many factors: quality of features, type of representation, task horizon, state aliasing, constraints on computation and memory, etc.
- Other topics we will cover:
  - Monte Carlo Tree search,
  - Multiagent RL,
  - Case studies: Atari games, AlphaGo.