Frequentist Learning

Frequentist learning = counting!

Frequentist Learning

Frequentist learning $=$ counting!

Suppose we have a state-action (s, a) pair with m successor states, and want to learn the probabilities

$$P(s, a, s_1), \ldots, P(s, a, s_m).$$

Frequentist Learning

Frequentist learning = counting!

Suppose we have a state-action (s, a) pair with m successor states, and want to learn the probabilities

$$P(s, a, s_1), \ldots, P(s, a, s_m).$$

Definition (Frequentist learning)

1. Take N samples of (s, a),
2. Count how many times we see successor state s_i, call this $\#(s, a, s_i)$,
3. We estimate $\tilde{P}(s, a, s_i) = \frac{\#(s,a,s_i)}{N}$.

Frequentist learning - why does this work?

An MDP is Markovian: transition probability $P(s, a, s_i)$ is independent of all other transition probabilities.

Frequentist learning - why does this work?

An MDP is Markovian: transition probability $P(s, a, s_i)$ is independent of all other transition probabilities.

$\tilde{P}(s, a, \cdot)$ forms a valid probability distribution:

$$N = \sum_j \#(s, a, s_j) \implies \sum_i \tilde{P}(s, a, s_i) = \sum_i \frac{\#(s, a, s_i)}{\sum_j \#(s, a, s_j)} = 1.$$

Key problem in frequentist learning

Frequentist learning is sensitive to observations.

Key problem in frequentist learning

Frequentist learning is sensitive to observations.

If we do not observe a transition, we have $\#(s, a, s_i) = 0$, and then we learn $\tilde{P}(s, a, s_i) = 0$.

Key problem in frequentist learning

Frequentist learning is sensitive to observations.

If we do not observe a transition, we have $\#(s, a, s_i) = 0$,
and then we learn $\tilde{P}(s, a, s_i) = 0$.

What to do if we know that this transition exists, i.e., $P(s, a, s_i) > 0$?

Bayesian Learning

Bayesian learning allows us to incorporate prior knowledge.

Bayesian Learning

Bayesian learning allows us to incorporate prior knowledge.

General idea:

$$Posterior \propto Prior \cdot Likelihood.$$

Bayesian Learning

Bayesian learning allows us to incorporate prior knowledge.

General idea:

$$Posterior \propto Prior \cdot Likelihood.$$

Conjugate prior: for certain families of priors and likelihoods, the posterior distribution is already known.

Bayesian Learning

Bayesian learning allows us to incorporate prior knowledge.

General idea:

$$Posterior \propto Prior \cdot Likelihood.$$

Conjugate prior: for certain families of priors and likelihoods, the posterior distribution is already known.

The Dirichlet distribution is conjugate to the multinomial likelihood:

$$Dirichlet \propto Dirichlet \cdot Multinomial.$$

Bayesian learning starts again with counting in a data set.

Bayesian Learning

Bayesian learning starts again with counting in a data set.

Suppose we have a state-action (s, a) pair with m successor states, and want to learn the probabilities

$$P(s, a, s_1), \ldots, P(s, a, s_m).$$

Again we take $N = \#(s, a)$ samples and count how many times we see s_i: $k_i = \#(s, a, s_i)$.

Updating distributions

These counts have a multinomial likelihood

$$Mn(k_1, \ldots, k_m \mid P(s,a,\cdot)) \propto \prod_{i=1}^{m} P(s,a,s_i)^{k_i}.$$

Updating distributions

These counts have a multinomial likelihood

$$Mn(k_1, \ldots, k_m \mid P(s, a, \cdot)) \propto \prod_{i=1}^{m} P(s, a, s_i)^{k_i}.$$

The Dirichlet distribution is a conjugate prior to the multinomial likelihood:

$$Dir(P(s, a, \cdot) \mid \alpha_1, \ldots, \alpha_m) \propto \prod_{i=1}^{m} P(s, a, s_i)^{\alpha_i - 1}.$$

Updating distributions

These counts have a multinomial likelihood

$$Mn(k_1, \ldots, k_m \mid P(s, a, \cdot)) \propto \prod_{i=1}^{m} P(s, a, s_i)^{k_i}.$$

The Dirichlet distribution is a conjugate prior to the multinomial likelihood:

$$Dir(P(s, a, \cdot) \mid \alpha_1, \ldots, \alpha_m) \propto \prod_{i=1}^{m} P(s, a, s_i)^{\alpha_i - 1}.$$

Given a prior Dirichlet distribution and a multinomial likelihood, we can update the prior to a posterior Dirichlet distribution with

$$Dir(P(s, a, \cdot) \mid \alpha_1 + k_1, \ldots, \alpha_m + k_m).$$

MAP estimation

After computing the posterior distribution $Dir(P(s, a, \cdot) \mid \alpha_1, \ldots, \alpha_m)$, we derive point estimates via the mode:

$$\tilde{P}(s, a, s_i) = \frac{\alpha_i - 1}{\left(\sum_{j=1}^{m} \alpha_j\right) - m}.$$

Key problem in Bayesian learning

Bayesian learning (MAP estimation) can be heavily biased to the prior.

Key problem in Bayesian learning

Bayesian learning (MAP estimation) can be heavily biased to the prior.

Hence, a challenge is choosing a good prior as starting point.

Key problem in Bayesian learning

Bayesian learning (MAP estimation) can be heavily biased to the prior.

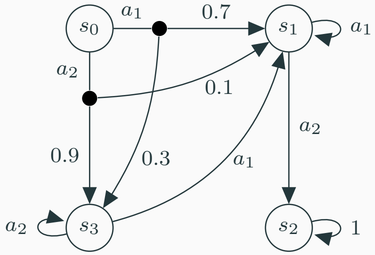
Hence, a challenge is choosing a good prior as starting point.

A Dirichlet distribution with $\alpha_i = \alpha_j$ for all i, j yields a uniform distribution.
The higher the values for α_i, the more data you need to shift away from the prior.

Depending on the specific situation, better choices may exist!

Example

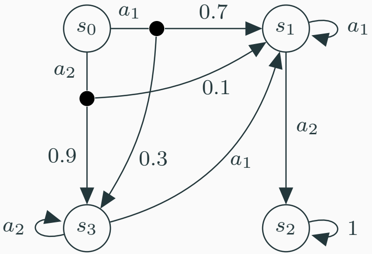
Suppose we want to learn (s_0, a_1) in the MDP:



Suppose $N = 20$, $\#(s_0, a_1, s_1) = 13$, $\#(s_0, a_1, s_3) = 7$.

Example

Suppose we want to learn (s_0, a_1) in the MDP:

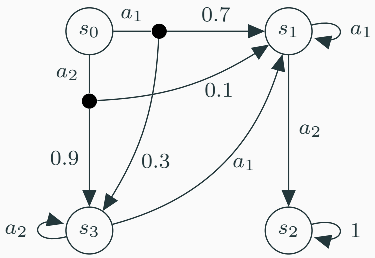


Suppose $N = 20$, $\#(s_0, a_1, s_1) = 13$, $\#(s_0, a_1, s_3) = 7$.

- Frequentist: $\tilde{P}(s_0, a_1, s_1) = \frac{13}{20} = 0.65$, $\tilde{P}(s_0, a_1, s_3) = \frac{7}{20} = 0.35$.

Example

Suppose we want to learn (s_0, a_1) in the MDP:



Suppose $N = 20$, $\#(s_0, a_1, s_1) = 13$, $\#(s_0, a_1, s_3) = 7$.

- Frequentist: $\tilde{P}(s_0, a_1, s_1) = \frac{13}{20} = 0.65$, $\tilde{P}(s_0, a_1, s_3) = \frac{7}{20} = 0.35$.
- Bayesian: Assume prior Dirichlet distribution with $\alpha_1 = \alpha_3 = 10$.

  Posterior: $\alpha_1 = 10 + 13$, $\alpha_3 = 10 + 7$.

  MAP-estimation:

  $\tilde{P}(s_0, a_1, s_1) = \frac{22}{38} = 0.579$,
  $\tilde{P}(s_0, a_1, s_3) = \frac{16}{38} = 0.421$.

PAC Learning

Probably approximately correct (PAC) learning: formal guarantee on the result.

We construct an IMDP with the following intervals:

PAC Learning

Probably approximately correct (PAC) learning: formal guarantee on the result.

We construct an IMDP with the following intervals:

1. Compute point estimates via frequentist or Bayesian learning for every transition (s, a, s'),

PAC Learning

Probably approximately correct (PAC) learning: formal guarantee on the result.

We construct an IMDP with the following intervals:

1. Compute point estimates via frequentist or Bayesian learning for every transition (s, a, s'),
2. Choose an error rate $\epsilon \in (0, 1)$, and compute the error rate for the whole model: $\epsilon_M = \epsilon / \sum_{s,a} |Post(s,a)_{>1}|$, where $|Post_{>1}(s, a)|$ is the number of successor states of (s, a) with probabilities in $(0, 1)$. Then use ϵ_M to compute $\delta_M = \sqrt{\log(2/\epsilon_M)/2N}$.

PAC Learning

Probably approximately correct (PAC) learning: formal guarantee on the result.

We construct an IMDP with the following intervals:

1. Compute point estimates via frequentist or Bayesian learning for every transition (s, a, s'),
2. Choose an error rate $\epsilon \in (0, 1)$, and compute the error rate for the whole model: $\epsilon_M = \epsilon / \sum_{s,a} |Post(s,a)_{>1}|$, where $|Post_{>1}(s, a)|$ is the number of successor states of (s, a) with probabilities in $(0, 1)$. Then use ϵ_M to compute $\delta_M = \sqrt{\log(2/\epsilon_M)/2N}$.
3. For each transition, construct the interval $\tilde{P}(s, a, s') \pm \delta_M$: $\underline{P}(s, a, s') = P(s, a, s') - \delta_M$, $\overline{P}(s, a, s') = P(s, a, s') + \delta_M$.

PAC Learning

Probably approximately correct (PAC) learning: formal guarantee on the result.

We construct an IMDP with the following intervals:

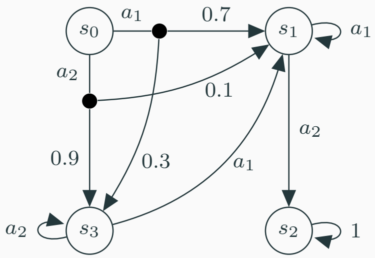
1. Compute point estimates via frequentist or Bayesian learning for every transition (s, a, s'),
2. Choose an error rate $\epsilon \in (0, 1)$, and compute the error rate for the whole model: $\epsilon_M = {}^{\epsilon}\!/\!\sum_{s,a} |Post(s,a)_{>1}|$, where $|Post_{>1}(s, a)|$ is the number of successor states of (s, a) with probabilities in $(0, 1)$. Then use ϵ_M to compute $\delta_M = \sqrt{\log({}^2\!/\epsilon_M)/2N}$.
3. For each transition, construct the interval $\tilde{P}(s, a, s') \pm \delta_M$: $\underline{P}(s, a, s') = P(s, a, s') - \delta_M$, $\overline{P}(s, a, s') = P(s, a, s') + \delta_M$.

Then with probability of at least $1 - \epsilon$ the true MDP M is contained in the IMDP $\mathcal{M}$:

$$\Pr(M \in \mathcal{M}) \geq 1 - \epsilon.$$

Example

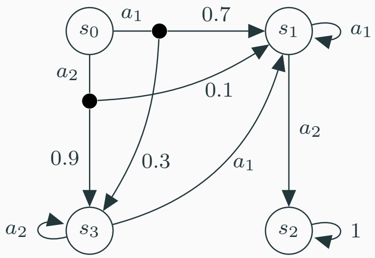
Suppose we want to learn (s_0, a_1) in the MDP:



Suppose we have $N = 20$, $\tilde{P}(s_0, a_1, s_1) = 0.65$, $\tilde{P}(s_0, a_1, s_3) = 0.35$, and set $\epsilon = 0.01$.

Example

Suppose we want to learn (s_0, a_1) in the MDP:

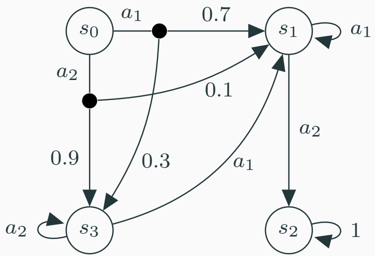


Suppose we have $N = 20$, $\tilde{P}(s_0, a_1, s_1) = 0.65$, $\tilde{P}(s_0, a_1, s_3) = 0.35$, and set $\epsilon = 0.01$.

- $\sum_{s,a} |Post_{>1}(s, a)| = 2 + 2 = 4$,

Example

Suppose we want to learn (s_0, a_1) in the MDP:

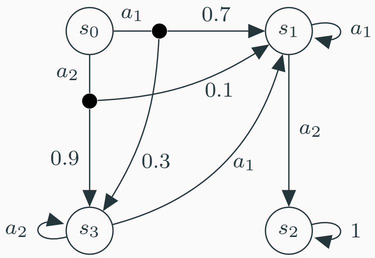


Suppose we have $N = 20$, $\tilde{P}(s_0, a_1, s_1) = 0.65$, $\tilde{P}(s_0, a_1, s_3) = 0.35$, and set $\epsilon = 0.01$.

- $\sum_{s,a} |Post_{>1}(s, a)| = 2 + 2 = 4$,
- $\epsilon_M = 0.0025$, $\delta_M = \sqrt{\frac{\log(2/\epsilon_M)}{2N}} = 0.409$,

Example

Suppose we want to learn (s_0, a_1) in the MDP:

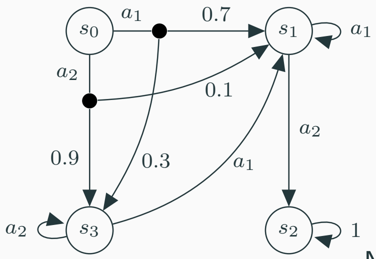


Suppose we have $N = 20$, $\tilde{P}(s_0, a_1, s_1) = 0.65$, $\tilde{P}(s_0, a_1, s_3) = 0.35$, and set $\epsilon = 0.01$.

- $\sum_{s,a} |Post_{>1}(s,a)| = 2 + 2 = 4$,
- $\epsilon_M = 0.0025$, $\delta_M = \sqrt{\frac{\log(2/\epsilon_M)}{2N}} = 0.409$,
- $\underline{P}(s_0, a_1, s_1) = 0.65 - 0.409 = 0.241$,
- $\overline{P}(s_0, a_1, s_1) = 0.65 + 0.409 = 1.059 \equiv 1.0$,
- $\underline{P}(s_0, a_1, s_3) = 0.35 - 0.409 = -0.059 \equiv 0.0$,
- $\overline{P}(s_0, a_1, s_3) = 0.35 + 0.409 = 0.759$.

Example

Suppose we want to learn (s_0, a_1) in the MDP:



Suppose we have $N = 20$, $\tilde{P}(s_0, a_1, s_1) = 0.65$, $\tilde{P}(s_0, a_1, s_3) = 0.35$, and set $\epsilon = 0.01$.

- $\sum_{s,a} |Post_{>1}(s, a)| = 2 + 2 = 4$,
- $\epsilon_M = 0.0025$, $\delta_M = \sqrt{\frac{\log(2/\epsilon_M)}{2N}} = 0.409$,
- $\underline{P}(s_0, a_1, s_1) = 0.65 - 0.409 = 0.241$,
- $\overline{P}(s_0, a_1, s_1) = 0.65 + 0.409 = 1.059 \equiv 1.0$,
- $\underline{P}(s_0, a_1, s_3) = 0.35 - 0.409 = -0.059 \equiv 0.0$,
- $\overline{P}(s_0, a_1, s_3) = 0.35 + 0.409 = 0.759$.

Note that values are forced into the $[0, 1]$ interval.

Key problems in PAC learning

1. The amount of data required for useful guarantees is enormous,
2. PAC learning assumes the underlying distribution(s) are fixed.

Linearly Updating Intervals

Linearly updating intervals (LUI): no formal guarantees, but fast and flexible when underlying distributions change.

Linearly Updating Intervals

Linearly updating intervals (LUI): no formal guarantees, but fast and flexible when underlying distributions change.

We assume two intervals for each transition:

1. An interval of prior transition probabilities $[\underline{P}(s,a,s'), \overline{P}(s,a,s')]$,
2. A strength interval $[\underline{n}(s,a,s'), \overline{n}(s,a,s')]$.

(1) Serves as prior that will be updated,
(2) Controls how much data we need.

LUI Computation

Assume we want to update transitions $(s, a, s_1), \ldots, (s, a, s_m)$.

LUI Computation

Assume we want to update transitions $(s, a, s_1), \ldots, (s, a, s_m)$.

1. Collect data, and let $N = \#(s, a)$ and $k_i = \#(s, a, s_i)$

LUI Computation

Assume we want to update transitions $(s, a, s_1), \ldots, (s, a, s_m)$.

1. Collect data, and let $N = \#(s, a)$ and $k_i = \#(s, a, s_i)$
2. Update lower bound:

$$\underline{P}(s, a, s_i)' = \begin{cases} \frac{\overline{n}(s,a,s_i)\underline{P}(s,a,s_i)+k_i}{\overline{n}(s,a,s_i)+N} & \text{if } \forall j. \frac{k_j}{N} \geq \underline{P}(s, a, s_j) \text{ (prior-data agreement),} \\ \frac{\underline{n}(s,a,s_i)\underline{P}(s,a,s_i)+k_i}{\underline{n}(s,a,s_i)+N} & \text{if } \exists j. \frac{k_j}{N} < \underline{P}(s, a, s_j) \text{ (prior-data conflict).} \end{cases}$$

LUI Computation

Assume we want to update transitions $(s, a, s_1), \ldots, (s, a, s_m)$.

1. Collect data, and let $N = \#(s, a)$ and $k_i = \#(s, a, s_i)$
2. Update lower bound:

$$\underline{P}(s, a, s_i)' = \begin{cases} \frac{\overline{n}(s,a,s_i)\underline{P}(s,a,s_i)+k_i}{\overline{n}(s,a,s_i)+N} & \text{if } \forall j. \frac{k_j}{N} \geq \underline{P}(s, a, s_j) \text{ (prior-data agreement),} \\ \frac{\underline{n}(s,a,s_i)\underline{P}(s,a,s_i)+k_i}{\underline{n}(s,a,s_i)+N} & \text{if } \exists j. \frac{k_j}{N} < \underline{P}(s, a, s_j) \text{ (prior-data conflict).} \end{cases}$$

3. Update upper bound:

$$\overline{P}(s, a, s_i)' = \begin{cases} \frac{\overline{n}(s,a,s_i)\overline{P}(s,a,s_i)+k_i}{\overline{n}(s,a,s_i)+N} & \text{if } \forall j. \frac{k_j}{N} \leq \overline{P}(s, a, s_j) \text{ (prior-data agreement),} \\ \frac{\underline{n}(s,a,s_i)\overline{P}(s,a,s_i)+k_i}{\underline{n}(s,a,s_i)+N} & \text{if } \exists j. \frac{k_j}{N} > \overline{P}(s, a, s_j) \text{ (prior-data conflict).} \end{cases}$$

LUI Computation

Assume we want to update transitions $(s, a, s_1), \ldots, (s, a, s_m)$.

1. Collect data, and let $N = \#(s, a)$ and $k_i = \#(s, a, s_i)$
2. Update lower bound:

$$\underline{P}(s, a, s_i)' = \begin{cases} \frac{\overline{n}(s,a,s_i)\underline{P}(s,a,s_i)+k_i}{\overline{n}(s,a,s_i)+N} & \text{if } \forall j. \frac{k_j}{N} \geq \underline{P}(s, a, s_j) \text{ (prior-data agreement),} \\ \frac{\underline{n}(s,a,s_i)\underline{P}(s,a,s_i)+k_i}{\underline{n}(s,a,s_i)+N} & \text{if } \exists j. \frac{k_j}{N} < \underline{P}(s, a, s_j) \text{ (prior-data conflict).} \end{cases}$$

3. Update upper bound:

$$\overline{P}(s, a, s_i)' = \begin{cases} \frac{\overline{n}(s,a,s_i)\overline{P}(s,a,s_i)+k_i}{\overline{n}(s,a,s_i)+N} & \text{if } \forall j. \frac{k_j}{N} \leq \overline{P}(s, a, s_j) \text{ (prior-data agreement),} \\ \frac{\underline{n}(s,a,s_i)\overline{P}(s,a,s_i)+k_i}{\underline{n}(s,a,s_i)+N} & \text{if } \exists j. \frac{k_j}{N} > \overline{P}(s, a, s_j) \text{ (prior-data conflict).} \end{cases}$$

4. Return updated transitions $[\underline{P}(s, a, \cdot)', \overline{P}(s, a, \cdot)']$
   and strengths $[\underline{n}(s, a, \cdot) + N, \overline{n}(s, a, \cdot) + N]$.

Example (single interval)

Prior	strength	estimate	posterior	strength
$[0.0, 1.0]$	$[0, 10]$	$\frac{1}{2}$	$[0.083, 0.917]$	$[2, 12]$
$[0.0, 1.0]$	$[0, 10]$	$\frac{50}{100}$	$[0.45, 0.55]$	$[100, 110]$
$[0.0, 1.0]$	$[0, 1000]$	$\frac{50}{100}$	$[0.045, 0.95]$	$[100, 1100]$
$[0.4, 0.6]$	$[0, 10]$	$\frac{1}{1}$	$[0.45, 1.0]$	$[1, 11]$
$[0.4, 0.6]$	$[10, 100]$	$\frac{1}{1}$	$[0.406, 0.636]$	$[11, 101]$

UCRL2 - the general idea

For simplicty, we only consider UCRL2 that learns the transition function.

UCRL2 - the general idea

For simplicty, we only consider UCRL2 that learns the transition function.

Initialize: set confidence parameter $\delta \in (0,1)$ and time counter $t = 1$.

1. Build L_1 MDP with

$$\tilde{P}(s,a,s') = \frac{\#(s,a,s')}{\max\{1,\#(s,a)\}}, \quad d(s,a) = \sqrt{\frac{14|S|\log(|A|t_k/\delta)}{\max\{1,\#(s,a)\}}},$$

UCRL2 - the general idea

For simplicty, we only consider UCRL2 that learns the transition function.
Initialize: set confidence parameter $\delta \in (0, 1)$ and time counter $t = 1$.

1. Build L_1 MDP with

$$\tilde{P}(s, a, s') = \frac{\#(s, a, s')}{\max\{1, \#(s, a)\}}, \quad d(s, a) = \sqrt{\frac{14|S| \log(|A|t_k/\delta)}{\max\{1, \#(s, a)\}}},$$

2. Compute optimistic policy π (next slide),

UCRL2 - the general idea

For simplicty, we only consider UCRL2 that learns the transition function.
Initialize: set confidence parameter $\delta \in (0,1)$ and time counter $t = 1$.

1. Build L_1 MDP with

$$\tilde{P}(s,a,s') = \frac{\#(s,a,s')}{\max\{1, \#(s,a)\}}, \quad d(s,a) = \sqrt{\frac{14|S|\log(|A|t_k/\delta)}{\max\{1, \#(s,a)\}}},$$

2. Compute optimistic policy π (next slide),
3. Sample data using π,

UCRL2 - the general idea

For simplicty, we only consider UCRL2 that learns the transition function.

Initialize: set confidence parameter $\delta \in (0, 1)$ and time counter $t = 1$.

1. Build L_1 MDP with

$$\tilde{P}(s, a, s') = \frac{\#(s, a, s')}{\max\{1, \#(s, a)\}}, \quad d(s, a) = \sqrt{\frac{14|S| \log(|A| t_k / \delta)}{\max\{1, \#(s, a)\}}},$$

2. Compute optimistic policy π (next slide),
3. Sample data using π,
4. Repeat.

Solving the optimistic inner problem efficiently (L_1 MDPs)

For UCRL2 we need to compute the optimistic value and policy:

$$V_{n+1}(s) = \max_{a \in A} \left\{ R(s,a) + \gamma \sup_{P(s,a) \in \mathcal{P}(s,a)} \left\{ \sum_{s' \in S} P(s,a)(s') V_n(s') \right\} \right\}$$

Solving the optimistic inner problem efficiently (L_1 MDPs)

For UCRL2 we need to compute the optimistic value and policy:

$$V_{n+1}(s) = \max_{a \in A} \left\{ R(s,a) + \gamma \sup_{P(s,a) \in \mathcal{P}(s,a)} \left\{ \sum_{s' \in S} P(s,a)(s') V_n(s') \right\} \right\}$$

To do so, we have a similar algorithm as for IMDPs:

1. Order $s_1, \ldots, s_m$ such that $V_n(s_1) \geq \cdots \geq V_n(s_m)$,

Solving the optimistic inner problem efficiently (L_1 MDPs)

For UCRL2 we need to compute the optimistic value and policy:

$$V_{n+1}(s) = \max_{a \in A} \left\{ R(s,a) + \gamma \sup_{P(s,a) \in \mathcal{P}(s,a)} \left\{ \sum_{s' \in S} P(s,a)(s') V_n(s') \right\} \right\}$$

To do so, we have a similar algorithm as for IMDPs:

1. Order $s_1, \ldots, s_m$ such that $V_n(s_1) \geq \cdots \geq V_n(s_m)$,
2. Set $P(s_1) = \min\{1, \tilde{P}(s_1) + d/2\}$ and for $j > 1$: $P(s_j) = \tilde{P}(s_j)$,
3. $l = m$,

Solving the optimistic inner problem efficiently (L_1 MDPs)

For UCRL2 we need to compute the optimistic value and policy:

$$V_{n+1}(s) = \max_{a \in A} \left\{ R(s,a) + \gamma \sup_{P(s,a) \in \mathcal{P}(s,a)} \left\{ \sum_{s' \in S} P(s,a)(s') V_n(s') \right\} \right\}$$

To do so, we have a similar algorithm as for IMDPs:

1. Order $s_1, \ldots, s_m$ such that $V_n(s_1) \geq \cdots \geq V_n(s_m)$,
2. Set $P(s_1) = \min\{1, \tilde{P}(s_1) + d/2\}$ and for $j > 1$: $P(s_j) = \tilde{P}(s_j)$,
3. $l = m$,
4. While $\sum_j P(s_j) > 1$:
    - $P(s_l) = \max\{0, 1 - \sum_{j \neq l} P(s_j)\}$,
    - $l = l - 1$,

Solving the optimistic inner problem efficiently (L_1 MDPs)

For UCRL2 we need to compute the optimistic value and policy:

$$V_{n+1}(s) = \max_{a \in A} \left\{ R(s,a) + \gamma \sup_{P(s,a) \in \mathcal{P}(s,a)} \left\{ \sum_{s' \in S} P(s,a)(s') V_n(s') \right\} \right\}$$

To do so, we have a similar algorithm as for IMDPs:

1. Order $s_1, \ldots, s_m$ such that $V_n(s_1) \geq \cdots \geq V_n(s_m)$,
2. Set $P(s_1) = \min\{1, \tilde{P}(s_1) + d/2\}$ and for $j > 1$: $P(s_j) = \tilde{P}(s_j)$,
3. $l = m$,
4. While $\sum_j P(s_j) > 1$:
   - $P(s_l) = \max\{0, 1 - \sum_{j \neq l} P(s_j)\}$,
   - $l = l - 1$,
5. Return P.

UCRL2 - full algorithm

Set $\delta \in (0,1)$, $t = 1$, $\#(s,a) = 0$, $\#(s,a,s') = 0$,

For episode $k = 1, 2, \ldots,$ do:

UCRL2 - full algorithm

Set $\delta \in (0,1)$, $t = 1$, $\#(s,a) = 0$, $\#(s,a,s') = 0$,

For episode $k = 1, 2, \ldots,$ do:

1. Build L_1 MDP at episode k:

UCRL2 - full algorithm

Set $\delta \in (0,1)$, $t = 1$, $\#(s,a) = 0$, $\#(s,a,s') = 0$,

For episode $k = 1, 2, \ldots,$ do:

1. Build L_1 MDP at episode k:
   1.1 $t_k = t$,
   1.2 $\tilde{P}(s,a,s') = \frac{\#(s,a,s')}{\max\{1,\#(s,a)\}}$, $d(s,a) = \sqrt{\frac{14|S|\log(|A|t_k/\delta)}{\max\{1,\#(s,a)\}}}$

UCRL2 - full algorithm

Set $\delta \in (0,1)$, $t = 1$, $\#(s,a) = 0$, $\#(s,a,s') = 0$,

For episode $k = 1, 2, \ldots,$ do:

1. Build L_1 MDP at episode k:
   1.1 $t_k = t$,
   1.2 $\tilde{P}(s,a,s') = \frac{\#(s,a,s')}{\max\{1,\#(s,a)\}}$, $d(s,a) = \sqrt{\frac{14|S|\log(|A|t_k/\delta)}{\max\{1,\#(s,a)\}}}$
   1.3 Compute optimistic policy π_k in L_1 MDP $(S, A, \tilde{P}, d, R, \gamma)$,

UCRL2 - full algorithm

Set $\delta \in (0,1)$, $t = 1$, $\#(s,a) = 0$, $\#(s,a,s') = 0$,

For episode $k = 1, 2, \ldots,$ do:

1. Build L_1 MDP at episode k:
   1.1 $t_k = t$,
   1.2 $\tilde{P}(s,a,s') = \frac{\#(s,a,s')}{\max\{1,\#(s,a)\}}$, $d(s,a) = \sqrt{\frac{14|S|\log(|A|t_k/\delta)}{\max\{1,\#(s,a)\}}}$
   1.3 Compute optimistic policy π_k in L_1 MDP $(S, A, \tilde{P}, d, R, \gamma)$,
2. Sampling:
   2.1 Set local counters $\forall(s,a,s') : v_k(s,a) = 0, v_k(s,a,s') = 0$,

UCRL2 - full algorithm

Set $\delta \in (0,1)$, $t = 1$, $\#(s,a) = 0$, $\#(s,a,s') = 0$,

For episode $k = 1, 2, \ldots,$ do:

1. Build L_1 MDP at episode k:
   1.1 $t_k = t$,
   1.2 $\tilde{P}(s,a,s') = \frac{\#(s,a,s')}{\max\{1,\#(s,a)\}}$, $d(s,a) = \sqrt{\frac{14|S|\log(|A|t_k/\delta)}{\max\{1,\#(s,a)\}}}$
   1.3 Compute optimistic policy π_k in L_1 MDP $(S, A, \tilde{P}, d, R, \gamma)$,
2. Sampling:
   2.1 Set local counters $\forall(s,a,s') : v_k(s,a) = 0, v_k(s,a,s') = 0$,
   2.2 While $v_k(s,\pi_k(s)) < \max\{1, \#(s,\pi_k(s))\}$:
      - Execute action $a = \pi_k(s)$, update counter $v_k(s,a) = v_k(s,a) + 1$

UCRL2 - full algorithm

Set $\delta \in (0,1)$, $t = 1$, $\#(s,a) = 0$, $\#(s,a,s') = 0$,

For episode $k = 1, 2, \ldots,$ do:

1. Build L_1 MDP at episode k:
   1.1 $t_k = t$,
   1.2 $\tilde{P}(s,a,s') = \frac{\#(s,a,s')}{\max\{1,\#(s,a)\}}$, $d(s,a) = \sqrt{\frac{14|S|\log(|A|t_k/\delta)}{\max\{1,\#(s,a)\}}}$
   1.3 Compute optimistic policy π_k in L_1 MDP $(S, A, \tilde{P}, d, R, \gamma)$,
2. Sampling:
   2.1 Set local counters $\forall(s,a,s') : v_k(s,a) = 0, v_k(s,a,s') = 0$,
   2.2 While $v_k(s,\pi_k(s)) < \max\{1, \#(s,\pi_k(s))\}$:
      - Execute action $a = \pi_k(s)$, update counter $v_k(s,a) = v_k(s,a) + 1$
      - Observe successor state s', update counter $v_k(s,a,s') = v_k(s,a,s') + 1$,

UCRL2 - full algorithm

Set $\delta \in (0,1)$, $t = 1$, $\#(s,a) = 0$, $\#(s,a,s') = 0$,

For episode $k = 1, 2, \dots,$ do:

1. Build L_1 MDP at episode k:
   1.1 $t_k = t$,
   1.2 $\tilde{P}(s,a,s') = \frac{\#(s,a,s')}{\max\{1,\#(s,a)\}}$, $d(s,a) = \sqrt{\frac{14|S|\log(|A|t_k/\delta)}{\max\{1,\#(s,a)\}}}$
   1.3 Compute optimistic policy π_k in L_1 MDP $(S, A, \tilde{P}, d, R, \gamma)$,
2. Sampling:
   2.1 Set local counters $\forall(s,a,s') : v_k(s,a) = 0, v_k(s,a,s') = 0$,
   2.2 While $v_k(s, \pi_k(s)) < \max\{1, \#(s, \pi_k(s))\}$:
   - Execute action $a = \pi_k(s)$, update counter $v_k(s,a) = v_k(s,a) + 1$
   - Observe successor state s', update counter $v_k(s,a,s') = v_k(s,a,s') + 1$,
   - Set s' as the current state: $s = s'$, update $t = t + 1$,

UCRL2 - full algorithm

Set $\delta \in (0,1)$, $t = 1$, $\#(s,a) = 0$, $\#(s,a,s') = 0$,

For episode $k = 1, 2, \ldots,$ do:

1. Build L_1 MDP at episode k:
   1.1 $t_k = t$,
   1.2 $\tilde{P}(s,a,s') = \frac{\#(s,a,s')}{\max\{1,\#(s,a)\}}$, $d(s,a) = \sqrt{\frac{14|S|\log(|A|t_k/\delta)}{\max\{1,\#(s,a)\}}}$
   1.3 Compute optimistic policy π_k in L_1 MDP $(S, A, \tilde{P}, d, R, \gamma)$,
2. Sampling:
   2.1 Set local counters $\forall(s,a,s') : v_k(s,a) = 0, v_k(s,a,s') = 0$,
   2.2 While $v_k(s, \pi_k(s)) < \max\{1, \#(s, \pi_k(s))\}$:
   - Execute action $a = \pi_k(s)$, update counter $v_k(s,a) = v_k(s,a) + 1$
   - Observe successor state s', update counter $v_k(s,a,s') = v_k(s,a,s') + 1$,
   - Set s' as the current state: $s = s'$, update $t = t + 1$,

   2.3 End episode k, update global counters $\#(s,a) \mathrel{+}= v_k(s,a)$, $\#(s,a,s') \mathrel{+}= v_k(s,a,s')$