6.2 Importance Sampling

Importance Sampling (Rubinstein, 1981) is a standard technique for estimating the expected value of a random variable x with distribution d from samples, when the samples are drawn from another distribution d'. For instance, the target distribution d could be normal, while the sampling distribution d' is uniform (see Figure 6.1).

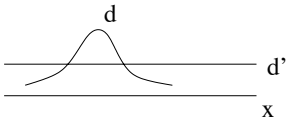

Figure 6.1. Different target and sampling distributions

In its classical form, importance sampling computes the expected value $E\{x \mid d\}$ based on a simple observation:

$$E_d\{x\} = \int_x x\, d(x)\, dx = \int_x x \frac{d(x)}{d'(x)} d'(x) dx = E_{d'}\left\{ x \frac{d(x)}{d'(x)} \right\},$$

which leads to the importance sampling estimator:

$$IS = \frac{1}{n} \sum_{i=1}^{n} x_i \frac{d(x_i)}{d'(x_i)}, \tag{6.2}$$

where x_i are samples selected according to d'. This estimator computes the average of the sample values, where each sample is weighted differently based on the ratio of its likelihood of occurring under the two distributions. This weighting gives more importance to samples that occur rarely under the sampling distribution d' but occur frequently under the target distribution d. If d and d' are the same, then all the samples have a weight of 1, and the estimator becomes the usual arithmetic average of the samples. The importance sampling estimator (6.2) is *consistent*, meaning that it converges with probability 1 to $E_d\{x\}$ as the number of samples goes to infinity, and *unbiased*, meaning its expected value after any number of samples is also $E_d\{x\}$ (Rubinstein, 1981).

A less known variant of this technique is *weighted importance sampling*, which performs a weighted average of the samples, with weights $\frac{d(x_i)}{d'(x_i)}$. The weighted importance sampling estimator is:

$$ISW = \frac{\sum_{i=1}^{n} x_i \frac{d(x_i)}{d'(x_i)}}{\sum_{i=1}^{n} \frac{d(x_i)}{d'(x_i)}} \tag{6.3}$$

The weighted importance sampling estimator (6.3) is a consistent but biased estimator of $E\{x \mid d\}$ (Rubinstein, 1981). Nevertheless, his estimator is often faster and more stable in practice than (6.2). Intuitively, this property is due to the fact that, if an unlikely event occurs, its weight will be very large, and will cause a large variation in the classical estimator. In the weighted estimator, the large weight appears in the denominator as well, and therefore smoothes the variation.

6.3 Applying Importance Sampling to MDPs

In the case of MDPs, the samples come in the form of episodes, which are complete sequences of states, actions and rewards, ending in a terminal state:

$$s_0 a_0 r_1 s_1 a_1 r_2 \ldots s_{T_m-1} a_{T_m-1} r_{T_m} s_{T_m}.$$

The goal is to estimate the state-action value function $Q^\pi(s,a)$ for a given state s and action a. Let M be the number of episodes containing state-action pair (s,a) and t_m be the first time t when $(s_t, a_t) = (s,a)$ in the mth of these episodes. Then we define the first-visit importance sampling estimate for $Q^\pi(s,a)$ as

$$Q^{IS}(s,a) \stackrel{\text{def}}{=} \frac{1}{M} \sum_{m=1}^{M} R_m w_m, \tag{6.4}$$

where R_m is the return following (s,a) in episode e,

$$R_m \stackrel{\text{def}}{=} r_{t_m+1} + \gamma r_{t_m+2} + \ldots + \gamma^{T_m - t_m - 1} r_{T_m}$$

and w_m is the importance sampling weight assigned to episode m:

$$w_m \stackrel{\text{def}}{=} \frac{\pi_{t_m+1}}{b_{t_m+1}} \frac{\pi_{t_m+2}}{b_{t_m+2}} \cdots \frac{\pi_{T_m-1}}{b_{T_m-1}}.$$

Here, and in the following sections, we denote by $\pi_t = \pi(s_t, a_t)$ and similarly $b_t = b(s_t, a_t)$. Similar estimators can be computed for every-visit Monte Carlo as well. First-visit estimators have the advantage of being unbiased (Singh & Sutton, 1996), and therefore we will use such estimators in this dissertation.

Similarly, we define the *weighted importance sampling estimator* (Sutton & Barto, 1998) as

$$Q^{ISW}(s,a) \stackrel{\text{def}}{=} \frac{\sum_{m=1}^{M} R_m w_m}{\sum_{m=1}^{M} w_m}. \tag{6.5}$$

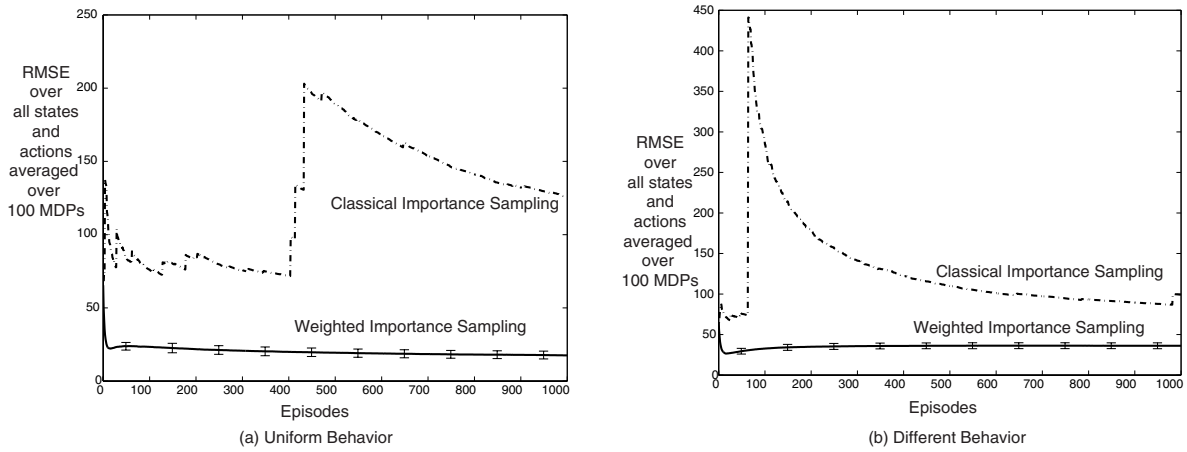

Figure 6.2. Comparison of classical and weighted importance sampling on 100 randomly generated MDPs. On the left, the behavior policy chose 50-50 from the two actions. On the right, the behavior policy chose with 20-80 probabilities, exactly opposite to the target policy. In both cases, the weighted algorithm is faster and more stable.

Figure 6.3 presents an empirical comparison of the classical and weighted importance sampling estimators. The comparison was performed using 100 different randomly generated MDPs. Each MDP has 100 states, one of which is terminal. Two actions were available in each nonterminal state, and each action branched to four next states, with random probabilities (the partition of unity was selected by

picking three random split points uniformly randomly from $[0, 1]$). The immediate rewards for each state-action pair were chosen uniformly randomly, between 0 and 1. The target policy was to select the first action with 80% probability and the second action with 20% probability. We used two different behavior policies. In the *uniform behavior* case (left panel) both actions were equally likely, whereas in the *different behavior* case, the first action was selected with 20% probability and the second action with 80% probability, resulting in a policy very different from the target policy. The initial state of each episode was chosen uniformly randomly from the nonterminal states. All the MDPs terminated with probability 1, so we used $\gamma = 1$.

Figure 6.3 shows, for each estimator, the root of the total mean squared error between the estimator and the true action values for the 200 state-action pairs, averaged over the 100 MDPs. this measure is computed at the beginning of learning, and after each of the first 1000 episodes. For the weighted importance sampling algorithm, the graph also includes error bars equal to one standard deviation. For the classical importance sampling, the maximum standard deviation is on the order of 3000, therefore we omitted the error bars. This result confirms the fact that the classical importance sampling algorithm has very high variance, which recommends against its use in practice. Also, as shown in the figure, the weighted version of the algorithm is faster and more stable than the classical version. This result was consistent across all MDPs we experimented with.

6.4 Per-Decision Importance Sampling

Both importance sampling algorithms presented so far require known Markov behavior policies. They are also inherently Monte Carlo algorithms, because they put a weight on the total return R_m obtained during an episode. There is no easy way of implementing either algorithm in an incremental fashion, for instance by performing TD-like updates after every step of the execution. In order to be able to perform such

updates, an algorithm should perform a weighting of each reward r_t obtained along the trajectory followed during the episode.

In this section we present a new algorithm that performs importance sampling weightings for each decision step along the way. Such a weighting can be computed if, instead of treating each return as one indivisible sample, we take into account the fact that the returns come from an underlying MDP. We will focus here on the Monte Carlo version of the estimator. In the next chapter we present a natural TD implementation.

In order to justify the estimator, let us examine the term $R_m w_m$ from equations (6.4) and (6.5): The terms of the sum can be naturally separated into two parts, one containing the $\frac{\pi}{b}$ ratios from t_{m+1} to $i-1$, and one containing the ratios from i to T_{m-1}. Intuitively, the weight on reward r_i should not depend on the future after time i, only on the history to that point. This is the idea behind the *per-decision importance sampling estimator*:

$$Q^{PD}(s,a) \stackrel{\text{def}}{=} \frac{1}{M} \sum_{m=1}^{M} \sum_{k=1}^{T_m - t_m} \gamma^{k-1} r_{t_m+k} \prod_{i=t_m+1}^{t_m+k-1} \frac{\pi_i}{b_i}. \tag{6.6}$$

The estimator weights each reward along a trajectory according to the likelihood of the trajectory up to that point, under the target and the behavior policy. If the target and the behavior policy are the same, the estimator is simply the average of the total returns from each episode. We now show that this estimate is indeed correct:

Theorem 5 *The per-decision importance sampling estimator Q^{PD} given by (6.6) is a consistent unbiased estimator of $Q^\pi(s,a)$.*

Proof: We know that the classical importance sampling estimator Q^{IS} is consistent and unbiased:

$$E\left\{ \left(\sum_{k=1}^{T-t} \gamma^{k-1} r_{t+k} \right) \prod_{i=t+1}^{T-1} \frac{\pi_i}{b_i} \mid s_t = s, a_t = a, b \right\} = Q^\pi(s,a).$$

We will show that the per-decision importance sampling estimator Q^{PD} has the same expected value as Q^{IS}. Let us move the importance sampling correction inside the sum, and examine the expectation for the k-th term:

$$E\left\{\gamma^{k-1} r_{t+k} \prod_{i=t+1}^{T-1} \frac{\pi_i}{b_i} \mid s_t = s, a_t = a, b\right\} =$$
$$= E\left\{\gamma^{k-1} r_{t+k} \frac{\pi_{t+1}}{b_{t+1}} \cdots \frac{\pi_{t+k-1}}{b_{t+k-1}} \mid s_t, a_t, \ldots s_{t+k-1}, a_{t+k-1}\right\}$$
$$\cdot E\left\{\frac{\pi_{t+k}}{b_{t+k}} \cdots \frac{\pi_{T-1}}{b_{T-1}} \mid s_t, a_t, \ldots s_{t+k}, a_{t+k}, b\right\}.$$

Since the underlying environment is an MDP, the second factor can be re-written as:

$$E\left\{\frac{\pi_{t+k}}{b_{t+k}} \cdots \frac{\pi_{T-1}}{b_{T-1}} \mid s_{t+k}, a_{t+k}, b\right\}.$$

The expected value of this term is 1. Therefore,

$$E\left\{\left(\sum_{k=1}^{T-t} \gamma^{k-1} r_{t+k}\right) \prod_{i=t+1}^{T-1} \frac{\pi_i}{b_i} \mid s_t = s, a_t = a, b\right\} =$$
$$E\left\{\sum_{k=1}^{T-t} \gamma^{k-1} r_{t+k} \prod_{i=t+1}^{t+k-1} \frac{\pi_i}{b_i} \mid s_t = s, a_t = a, b\right\},$$

which concludes the proof. $\diamond$

We can also devise a weighted version of the per-decision importance sampling algorithm. The reason for such a version is to smooth out large variations in the updates, if unlikely events happen. The idea is simply to divide the estimator by the sum of the weights during each episode:

$$Q^{PDW}(s,a) \stackrel{\text{def}}{=} \frac{\sum_{m=1}^{M} \sum_{k=1}^{T_m - t_m} \gamma^{k-1} r_{t_m+k} \prod_{i=t_m+1}^{t_m+k-1} \frac{\pi_i}{b_i}}{\sum_{m=1}^{M} \sum_{k=1}^{T_m - t_m} \gamma^{k-1} \prod_{i=t_m+1}^{t_m+k-1} \frac{\pi_i}{b_i}}. \tag{6.7}$$

This *weighted per-decision importance sampling estimator* is consistent but biased, just like the weighted importance sampling estimator Q^{ISW}.

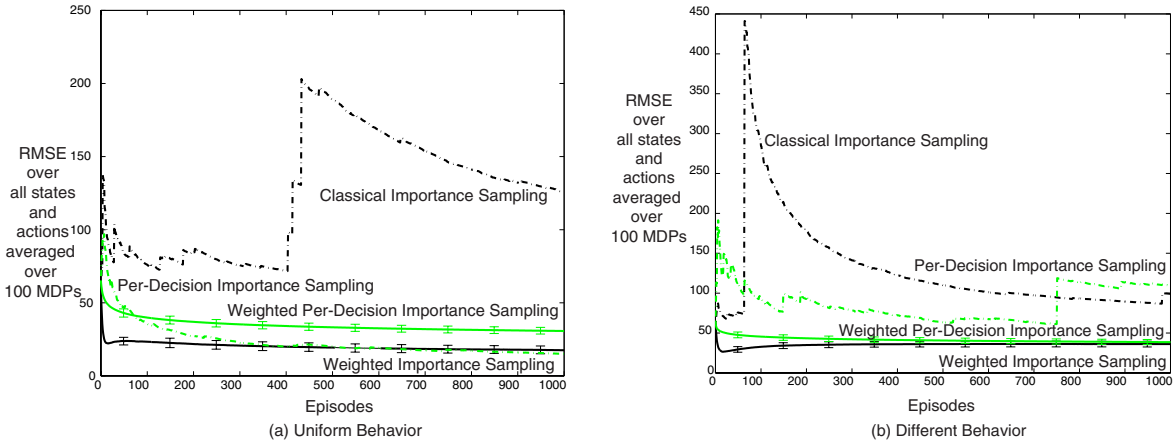

Figure 6.3. Comparison of Classical (Per-Return) and Per-Decision Monte Carlo Importance Sampling Algorithms

Figure 6.4 presents an empirical comparison of the per-decision algorithms with the classical (per-return) version, on the same testbed of 100 randomly generated MDPs (described in detail in section 6.3). The error measure is again the root of the total mean squared error for all the state-action pairs, averaged over the 100 MDPs. For the weighted per-decision algorithm, we also show error bars equal to one standard deviation. The standard deviation for the unweighted per-decision was on the order of 100 in the uniform behavior case (left panel) and on the order of 500 in the different behavior case (right panel). Since the weighted per-decision estimator has significantly smaller variance and more stable behavior, we recommend its use instead of the unweighted version, even though it is not consistently faster (as seen in the left panel).

6.5 Conclusions

In this chapter we presented Monte Carlo algorithms for policy evaluation, based on importance sampling corrections. One of the algorithms is a straightforward application of importance sampling. The other algorithm, per-decision importance sampling, is a new method, which takes into account the fact that the reward samples come from an MDP. We have shown that per-decision importance sampling algorithm