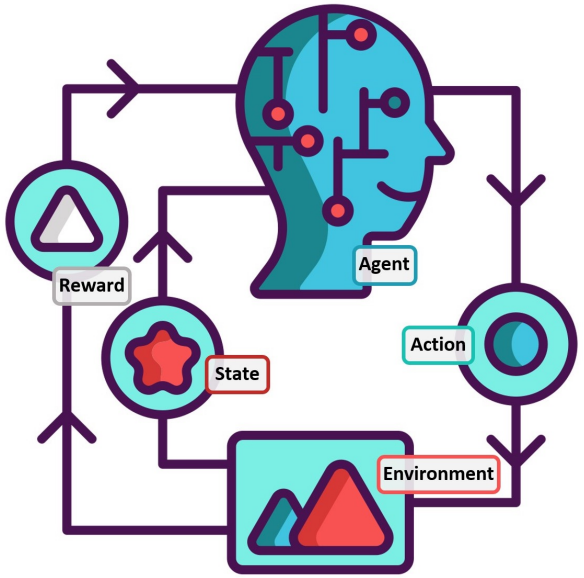
AMCO
ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND CONTROL RESEARCH GROUP

Reinforcement Learning
2025/2026

RL2
Reinforcement Learning Research Lab @ UniPD

Lecture #08
Off-Policy Monte Carlo & Temporal Difference Learning



Gian Antonio Susto

Announcement before starting: this week's lab

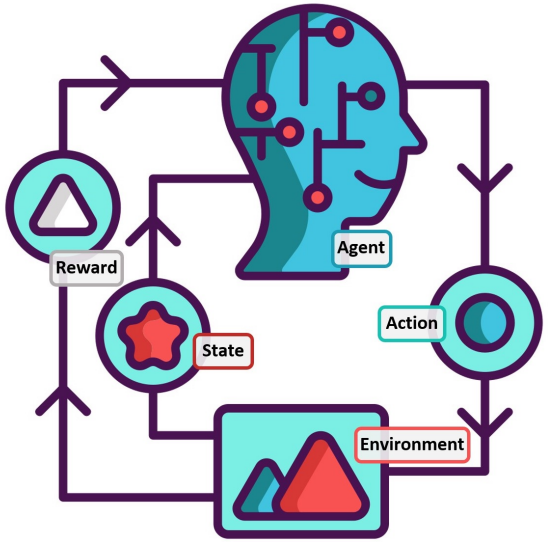
- Friday 14:30 -> Room Me
- Lab on Monte Carlo (MC) methods: windy grid-world environment

Announcement before starting: this week's lab

- Friday 14:30 -> Room Me
- Lab on Monte Carlo (MC) methods: windy grid-world environment
- Blackjack lab from previous year available on-line
- Send 'kudos' to Niccolò Turcato

niccolo.turcato@phd.unipd.it
https://www.linkedin.com/in/niccol%C3%B2-t-3411b1118/

Recap: Monte Carlo Methods



- Model-free approaches for the ‘full’ RL problem: MC methods exploits the fact that expectations can be approximated with averages
- On policy approaches

Prediction Lecture #05

Control Lecture #06 (GPI on state-action functions, Exploring Starts, ε-soft policies)

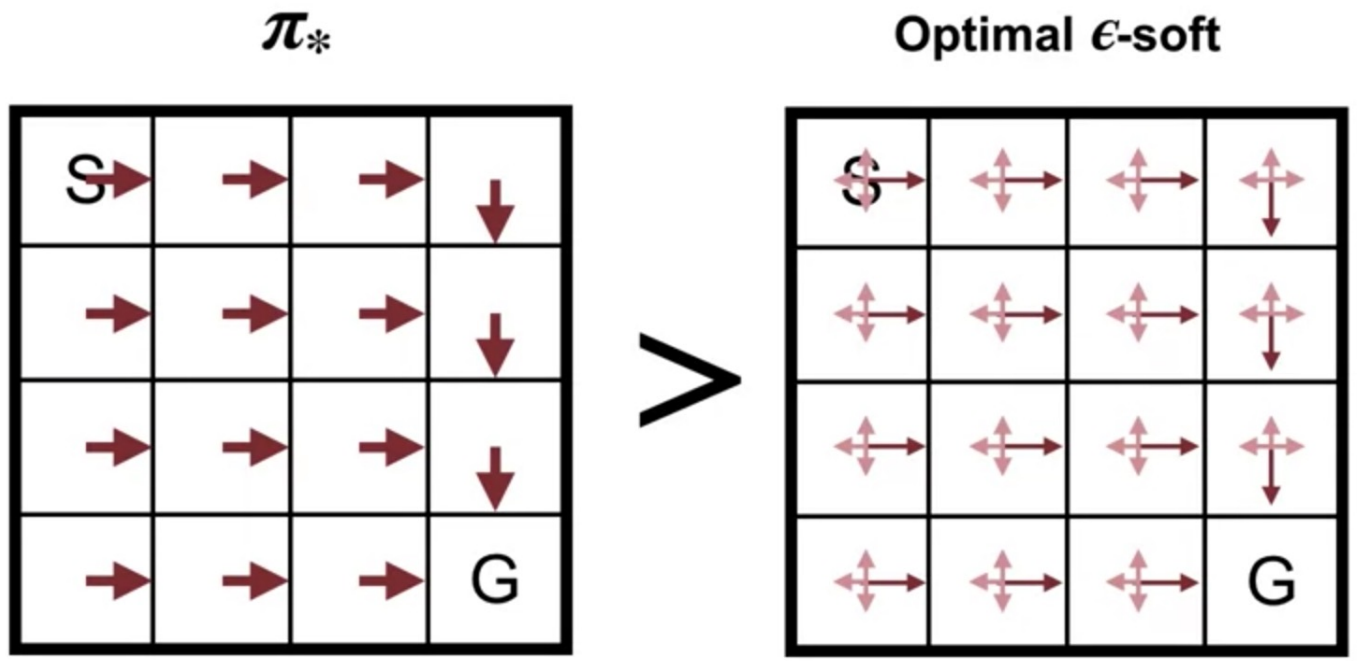
- Off-policy approaches

Prediction Lecture #07-08

Control Optional material on the book (Section 5.7)

Recap: in Control ε-soft policies are not optimal!

- It can be shown (policy improvement theorem) that the previous algorithm allow improvement (optional, see the book)
- ε-soft policies are not optimal!
- However, they may work quite well and may represent a feasible option when ES cannot be applied (which is true in many cases!)



Can we do better?

A. White, M. White 'Sample-based Learning Methods'

Recap: on-policy vs. off-policy

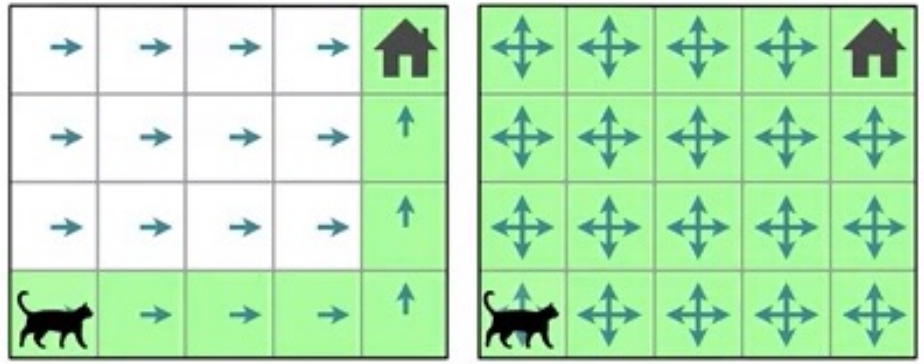
- What we have seen so far are on-policy learning
- On-policy ('learning on the job'): learn about policy π from experience sampled from policy π
- Off-policy ('look over someone's shoulder'): learn about policy π (target policy) from experience sampled from policy b (behaviour policy)
- In off-policy we use the behaviour policy to collect data from the environment in order to evalute an optimal policy

Why using a behaviour policy?

- We can leave the exploration to the behaviour policy and reach optimal policies!
- We may want to gain experience from observing other agents (or humans)
- Re-use experienc generated from old policies $\pi_1, \pi_2, \pi_3, \ldots, \pi_{t-1}$
- We can learn multiple policies while following one policy

Target policy

Behaviour policy

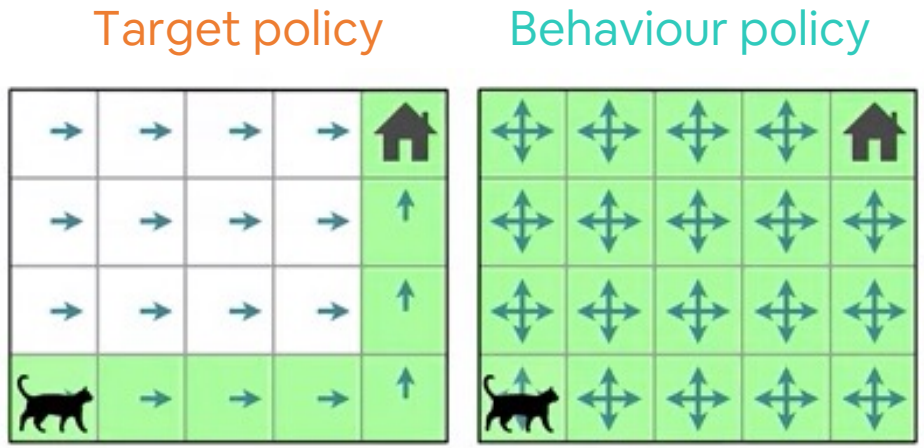


A. White, M. White 'Sample-based Learning Methods'

Recap: on-policy vs. off-policy

Things I have to consider to make off-policy work

- [Hypothesis #01] Knowledge of behaviour policy (and of target policy, obvious)
- [Hypothesis #02] Coverage
- Importance Sampling



Recap: Off-policy learning - Coverage

- If we resort to off-policy learning, we must ensure coverage:

$$\text{if } \pi(a|s) > 0 \Longrightarrow b(a|s) > 0$$

- Which means that we must ensure that the behaviour policy take explore the (state, action) pair that have non-zero probabilty in the target policy
- If this do not happen, the agent cannot learn the correct action value for that state because it never observed samples of what would happen.

Recap: Off-policy learning – Importance Sampling

DEFINITION. The *expectation* of a discrete random variable X taking the values $a_1, a_2, \ldots$ and with probability mass function p is the number

$$\mathrm{E}[X] = \sum_i a_i \mathrm{P}(X = a_i) = \sum_i a_i p(a_i).$$

We can consider this as a new random variable

$$\mathbb{E}_\pi[X] = \sum_{x \in X} x\pi(x) = \sum_{x \in X} x\rho(x)b(x) = \mathbb{E}_b[X\rho(x)]$$

- We have a way to 'correct' the expectation if we draw samples from a different policy
- To actually use this with data, we exploit MC (a weighted sampled average)

$$\mathbb{E}_\pi[X] \sim \frac{1}{n} \sum_{i=1,\ldots,n} x_i \rho(x_i)$$

Importance Sampling Ratio:

$$\rho(x) = \frac{\pi(x)}{b(x)}$$

Prediction: The Bandits Example

- 2-arm bandits: I already have a policy π that chooses a_1 with 75% probability and a_2 with 25%
- I'm looking over the shoulder of another agent that has 50% probabilty for each action (policy b)
- Initial estimations $q_\pi(a_i) = q_b(a_i) = 0$

Prediction: The Bandits Example

- 2-arm bandits: I already have a policy π that chooses a_1 with 75% probability and a_2 with 25%
- I'm looking over the shoulder of another agent that has 50% probabilty for each action (policy b)
- Initial estimations $q_\pi(a_i) = q_b(a_i) = 0$
- Episode #01: a_2 -> +1

$q_b(a_2) = 1$, $q_\pi(a_2) = 1 * 25/50 = 0.5$

Importance Sampling Ratio:

$$\rho(x) = \frac{\pi(x)}{b(x)}$$

Prediction: The Bandits Example

- 2-arm bandits: I already have a policy π that chooses a_1 with 75% probability and a_2 with 25%
- I'm looking over the shoulder of another agent that has 50% probabilty for each action (policy b)
- Initial estimations $q_\pi(a_i) = q_b(a_i) = 0$
- Episode #01: a_2 -> +1

$q_b(a_2) = 1$, $q_\pi(a_2) = 1 * 25/50 = 0.5$

- Episode #02: a_1 -> -1

$q_b(a_1) = -1$, $q_\pi(a_1) = -1 * 75/50 = -1.5$

Importance Sampling Ratio:

$$\rho(x) = \frac{\pi(x)}{b(x)}$$

Prediction: The Bandits Example

- 2-arm bandits: I already have a policy π that chooses a_1 with 75% probability and a_2 with 25%
- I'm looking over the shoulder of another agent that has 50% probabilty for each action (policy b)
- Initial estimations $q_\pi(a_i) = q_b(a_i) = 0$
- Episode #01: a_2 -> +1

$q_b(a_2) = 1$, $q_\pi(a_2) = 1 * 25/50 = 0.5$

- Episode #02: a_1 -> -1

$q_b(a_1) = -1$, $q_\pi(a_1) = -1 * 75/50 = -1.5$

- Episode #03: a_1 -> +2

$q_b(a_1) = 0.5(-1+2) = 0.5$, $q_\pi(a_1) = 0.5*(-1+2) * 75/50 = 0.75$

Importance Sampling Ratio:

$$\rho(x) = \frac{\pi(x)}{b(x)}$$

Prediction: Off-policy learning

- In RL we have to deal with sequences!
- We use returns generated from b to evaluate π, we cannot directly compute v_π as the average of the returns seen!
- We have to correct each return thanks to importance sampling
- Given a starting state S_t the probability of the subsequent state-action trajectory under policy π is

$$\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}$$

Prediction: Off-policy learning

- In RL we have to deal with sequences!
- We use returns generated from b to evaluate π, we cannot directly compute v_π as the average of the returns seen!
- We have to correct each return thanks to importance sampling
- Given a starting state S_t the probability of the subsequent state-action trajectory under policy π is

$$\begin{aligned}
&\Pr\{A_t, S_{t+1}, A_{t+1}, \ldots, S_T \mid S_t, A_{t:T-1} \sim \pi\} \\
&\quad = \pi(A_t|S_t)p(S_{t+1}|S_t, A_t)\pi(A_{t+1}|S_{t+1}) \cdots p(S_T|S_{T-1}, A_{T-1}) \\
&\quad = \prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k),
\end{aligned}$$

Probability of a trajectory in an environment under policy π

Prediction: Off-policy learning

- The relative probability of the trajectory under the target and behaviour policy (the importance-sampling ratio) is

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k,A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k,A_k)} =$$

Probability of a trajectory in an environment under policy π

Probability of a trajectory in an environment under policy b

- We can now exploit returns obtain with b to estimate v_π with MC

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$

- We can consider importance-sampling ratios as ‘weights’

Prediction: Off-policy learning

- The relative probability of the trajectory under the target and behaviour policy (the importance-sampling ratio) is:

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k, A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k, A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

Probability of a trajectory in an environment under policy π

Probability of a trajectory in an environment under policy b

- We can now exploit returns obtain with b to estimate v_π with MC

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$

- We can consider importance-sampling ratios as ‘weights’

Prediction: Off-policy learning

- The relative probability of the trajectory under the target and behaviour policy (the importance-sampling ratio) is:

$$\rho_{t:T-1} \doteq \frac{\prod_{k=t}^{T-1} \pi(A_k|S_k)p(S_{k+1}|S_k,A_k)}{\prod_{k=t}^{T-1} b(A_k|S_k)p(S_{k+1}|S_k,A_k)} = \prod_{k=t}^{T-1} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}$$

- We can now exploit returns obtain with b to estimate v_π with MC

$$\mathbb{E}[\rho_{t:T-1} G_t \mid S_t = s] = v_\pi(s)$$

- We can consider importance-sampling ratios as 'weights'

Monte Carlo off-policy Prediction

Input: a policy π to be evaluated

Initialize

$V(s) \in$

Retur

Loop fo

Gene

$G \leftarrow$

Loop

G

A

V

$W \leftarrow W \dfrac{}{b(A_t \mid S_t)}$

$T-1, A_{T-1}, R_T$

0

Algorithm 1 Off-policy Monte Carlo prediction algorithm for estimating $V \approx v_\pi$

Initialize:
  $V(s) \in \mathbb{R}$ (arbitrarily), for all $s \in \mathcal{S}$
  $C(s) = 0$ (accumulating weights), for all $s \in \mathcal{S}$
  π as the target policy and b as the behavior policy (where $b(a|s) > 0$ if $\pi(a|s) > 0$)
Loop forever (for each episode):
  Generate an episode using b: $S_0, A_0, R_1, S_1, A_1, R_2, \ldots, S_{T-1}, A_{T-1}, R_T$
  $G \leftarrow 0$
  $W \leftarrow 1$
  For $t = T-1$ down to 0:
    $W \leftarrow W \cdot \frac{\pi(A_t|S_t)}{b(A_t|S_t)}$
    If $W = 0$ then:
      Break
    $G \leftarrow \gamma G + R_{t+1}$
    $C(S_t) \leftarrow C(S_t) + W$
    $V(S_t) \leftarrow V(S_t) + \frac{W}{C(S_t)}(G - V(S_t))$

Monte Carlo off-policy Prediction

Differences w.r.t. on-policy

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, **arbitrarily, for all** $s \in S$

$Returns(s) \leftarrow$ **an empty list, for all** $s \in S$

Loop forever (for each episode):

Generate an episode following b: $S_0, A_0, R_1, S_1 \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$ $W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$

$G \leftarrow \gamma W G + R_{t+1}$

Append G **to** $Returns(S_t)$

$V(S_t) \leftarrow$ **average**$(Returns(S_t))$

$$W \leftarrow W \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$$

Monte Carlo off-policy Prediction

Differences w.r.t. on-policy

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, **arbitrarily, for all** $s \in S$

$Returns(s) \leftarrow$ **an empty list, for all** $s \in S$

Initialization

Loop forever (for each episode):

Generate an episode following b: $S_0, A_0, R_1, S_1 \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0 \quad W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$

$G \leftarrow \gamma W G + R_{t+1}$

Append G **to** $Returns(S_t)$

$V(S_t) \leftarrow$ **average**$(Returns(S_t))$

$$W \leftarrow W \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$$

Monte Carlo off-policy Prediction

Differences w.r.t. on-policy

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, **arbitrarily, for all** $s \in S$

$Returns(s) \leftarrow$ **an empty list, for all** $s \in S$

We are using a different policy for data generation

Loop forever (for each episode):

Generate an episode following b: $S_0, A_0, R_1, S_1 \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0 \quad W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$

$G \leftarrow \gamma W G + R_{t+1}$

Append G **to** $Returns(S_t)$

$V(S_t) \leftarrow$ **average**$(Returns(S_t))$

$$W \leftarrow W \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$$

Monte Carlo off-policy Prediction

Differences w.r.t. on-policy

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, **arbitrarily, for all** $s \in S$

$Returns(s) \leftarrow$ **an empty list, for all** $s \in S$

We are correcting for the weights given by the Importance Sampling

Loop forever (for each episode):

Generate an episode following b: $S_0, A_0, R_1, S_1 \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$ $W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$

$G \leftarrow \gamma W G + R_{t+1}$

Append G **to** $Returns(S_t)$

$V(S_t) \leftarrow$ **average**$(Returns(S_t))$

$$W \leftarrow W \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$$

Monte Carlo off-policy Prediction

Differences w.r.t. on-policy

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, **arbitrarily, for all** $s \in S$

$Returns(s) \leftarrow$ **an empty list, for all** $s \in S$

Loop forever (for each episode):

Generate an episode following b: $S_0, A_0, R_1, S_1 \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$ $W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$

$G \leftarrow \gamma W G + R_{t+1}$

Append G **to** $Returns(S_t)$

$V(S_t) \leftarrow$ **average**$(Returns(S_t))$

$$W \leftarrow W \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$$

We are correcting for the weights given by the Importance Sampling

We are computing the 'weights' incrementally, backwards for efficiency!

$$\rho_{t:T-1} \doteq \prod_{k=t}^{T-1} \frac{\pi(A_k \mid S_k)}{b(A_k \mid S_k)}$$

$$= \rho_t \rho_{t+1} \rho_{t+2} \cdots \rho_{T-2} \rho_{T-1}$$

$\longleftarrow$

$W_1 \leftarrow \rho_{T-1}$

$W_2 \leftarrow \rho_{T-1}\rho_{T-2}$

$W_3 \leftarrow \rho_{T-1}\rho_{T-2}\rho_{T-3}$

Monte Carlo off-policy Prediction

Differences w.r.t. on-policy

Input: a policy π to be evaluated

Initialize:

$V(s) \in \mathbb{R}$, **arbitrarily, for all** $s \in S$

$Returns(s) \leftarrow$ **an empty list, for all** $s \in S$

Loop forever (for each episode):

Generate an episode following b: $S_0, A_0, R_1, S_1 \ldots, S_{T-1}, A_{T-1}, R_T$

$G \leftarrow 0$ $W \leftarrow 1$

Loop for each step of episode, $t = T-1, T-2, \ldots, 0$; **while** $W \neq 0$

$G \leftarrow \gamma W G + R_{t+1}$

Append G **to** $Returns(S_t)$

$V(S_t) \leftarrow$ **average**$(Returns(S_t))$

$$W \leftarrow W \frac{\pi(A_t \mid S_t)}{b(A_t \mid S_t)}$$

Additional check to improve efficiency

Off-policy Control Monte Carlo

- To apply off-policy MC control, keep in mind that coverage may change over time:

$$\pi(a|s) > 0 \Longrightarrow b(a|s) > 0$$

$$\pi'(a|s) > 0 \Longrightarrow b(a|s) > 0 \text{ (?)}$$

Off-policy Control Monte Carlo

- To apply off-policy MC control, keep in mind that coverage may change over time:

$$\pi(a|s) > 0 \Longrightarrow b(a|s) > 0$$
$$\pi'(a|s) > 0 \Longrightarrow b(a|s) > 0 \text{ (?)}$$

- In practice off-policy MC control is not effective! Over many steps, off-policy estimations are really high-variance and slow to converge!

Off-policy Control Monte Carlo

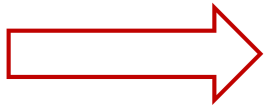
- To apply off-policy MC control, keep in mind that coverage may change over time:

$$\pi(a|s) > 0 \Longrightarrow b(a|s) > 0$$
$$\pi'(a|s) > 0 \Longrightarrow b(a|s) > 0 \text{ (?)}$$

- In practice off-policy MC control is not effective! Over many steps, off-policy estimations are really high-variance and slow to converge!
- However, with Temporal Difference learning, we'll make off-policy work also on practice!

Monte Carlo methods: Exam

$\mathcal{P}, \mathcal{R}$
known



Data,
experience

True Expectations

Sample Averages

- All the content of Chapter 5 are Exam material, beside:

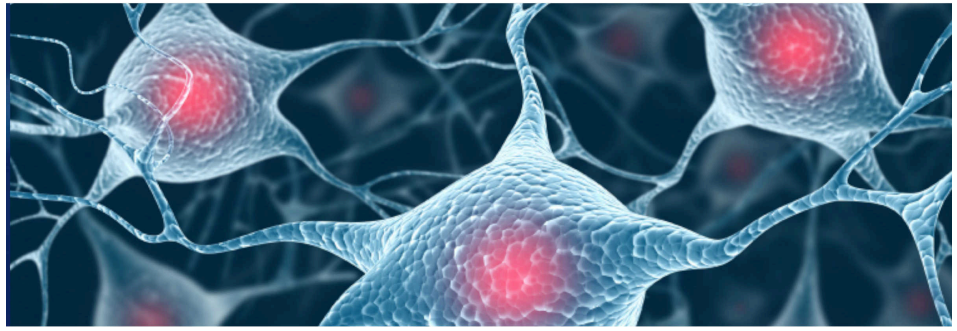
i. it can be considered optional the policy improvement theorem for ε-soft policies (in section 5.4)

ii. Sections 5.5 & 5.6 are related to Importance sampling: mandatory topic, but consider the simplified version presented in (this) class

iii. Sections 5.7, 5.8 and 5.9 can be skipped/optional (Section 5.7 talks about off-policy MC control)

- Pay particular attention to the algorithms!
- (Slides are always exam material – unless differently stated)

Temporal-difference Learning (Chapter 6)



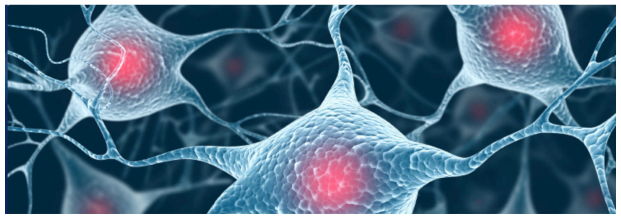
Intro to Temporal-difference (TD) Learning

From Chapter 6: ‘

If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning’

Intro to Temporal-difference (TD) Learning

- TD learning is another approach to model-free RL (for both prediction and control)
- TD learning applies bootstrapping, it works for incomplete episode (ie. It works also with continuous episodes)
- An historical result in RL is TD-gammon, a computer program developed by Gerald Tesauro (IBM) in 1992, that exploited TD-learning to play backgammon
- TD learning has still wide applications in RL and in other fields (in Neuroscience for example the firing of dopamine follow TD learning principles)
- We will see how TD learning is a general framework that also include Monte-Carlo Methods: we will start considering TD(0) and then the general framework



Limits of Monte Carlo approaches

1. MC only works on episodic tasks

Limits of Monte Carlo approaches

1. MC only works on episodic tasks

(However, with low γ values for example, approximations of the returns can be achieved)

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots$$

Limits of Monte Carlo approaches

1. MC only works on episodic tasks

(However, with low γ values for example, approximations of the returns can be achieved)

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \gamma^3 R_{t+4} + \cdots$$

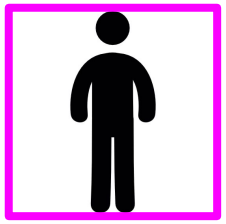
2. MC does not exploit available information as soon as they are collected: we are not using the rewards – that are informative! – as soon as they are available!

Limits of Monte Carlo approaches

‘Coming home in 20 mins’

Total Elapsed time:
0 min

Limits of Monte Carlo approaches



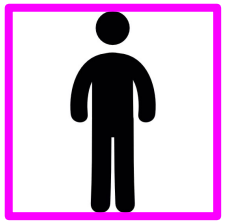
‘Coming home in 20 mins’

Sudden 20 mins call

Total Elapsed time:
20 min

Monte Carlo

Limits of Monte Carlo approaches



‘Coming home in 20 mins’

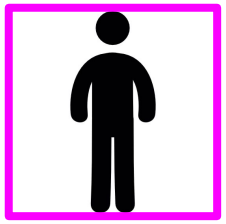
Sudden 20 mins call

Total Elapsed time:
30 min

Traffic: 10 mins more

Monte Carlo

Limits of Monte Carlo approaches



‘Coming home in 20 mins’

Sudden 20 mins call

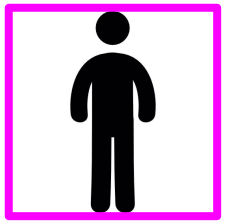
Total Elapsed time:
35 min

Traffic: 10 mins more

Monte Carlo

Gasoline: 5 mins more

Limits of Monte Carlo approaches



‘Coming home in 20 mins’

Sudden 20 mins call

Traffic: 10 mins more

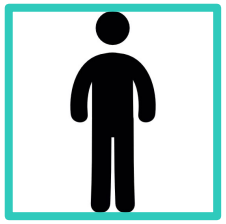
Gasoline: 5 mins more

Driving: 20 mins

Total Elapsed time:
55 min

Monte Carlo

Limits of Monte Carlo approaches



‘Coming home in 20 mins’

Sudden 20 mins call

‘Coming home in 20 mins’

Total Elapsed time:
55 min

Traffic: 10 mins more

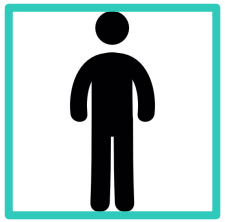
‘Coming home in 20 mins’

Gasoline: 5 mins more

‘Coming home in 20 mins’

Driving: 20 mins

Limits of Monte Carlo approaches



‘Coming home in 20 mins’

Sudden 20 mins call

‘Coming home in 20 mins’

Total Elapsed time:
55 min

Traffic: 10 mins more

‘Coming home in 30 mins’

Temporal Difference

Gasoline: 5 mins more

‘Coming home in 25 mins’

Driving: 20 mins

Recall from Lecture #02: Incremental Estimations are recurrent in RL

$$NewEstimate \leftarrow OldEstimate + StepSize \left[Target - OldEstimate \right]$$

We can apply this to MC methods:

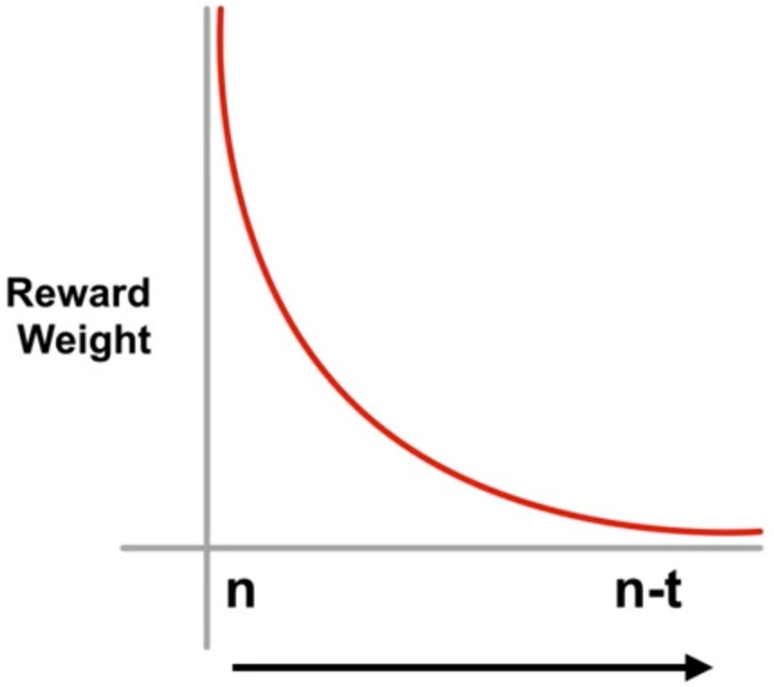
- Version ‘exact mean’

$$N(S_t) \leftarrow N(S_t) + 1$$

$$V(S_t) \leftarrow V(S_t) + \frac{1}{N(S_t)} \left(G_t - V(S_t) \right)$$

- Version ‘step size’ (better for non-stationary problems)

$$V(S_t) \leftarrow V(S_t) + \alpha \left(G_t - V(S_t) \right)$$



Prediction: Monte Carlo vs. TD-learning

The main difference between MC and TD is

- MC methods wait for the actual return G_t (that is available at the end of the episode) to update the estimation of $v_\pi(s)$

$$V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$$

Prediction: Monte Carlo vs. TD-learning

The main difference between MC and TD is

- MC methods wait for the actual return G_t (that is available at the end of the episode) to update the estimation of $v_\pi(s)$

$$V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$$

- TD(0) uses the immediate reward (that is available after taking one action) to update the estimation of the so-called TD target

$$V(S_t) \leftarrow V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

Prediction: Monte Carlo vs. TD-learning

The main difference between MC and TD is

- MC methods wait for the actual return G_t (that is available at the end of the episode) to update the estimation of $v_\pi(s)$

$$V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$$

- TD(0) uses the immediate reward (that is available after taking one action) to update the estimation of the so-called TD target

$$V(S_t) \leftarrow V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

The TD target can be seen as the 'best' estimation of G_t at time t+1:

$$G_t = R_{t+1} + \gamma G(S_{t+1}) \sim R_{t+1} + \gamma V(S_{t+1})$$

Prediction: Monte Carlo vs. TD-learning

The main difference between MC and TD is

- MC methods wait for the actual return G_t (that is available at the end of the episode) to update the estimation of $v_\pi(s)$

$$V(S_t) \leftarrow V(S_t) + \alpha\,(G_t - V(S_t))$$

- [illegible] available after taking one action) to update the estimation of the so-called TD target

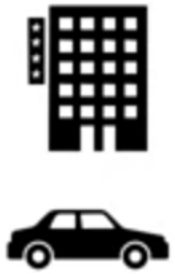
This quantity is called the *TD error*

$$V(S_t) \leftarrow V(S_t) + \alpha\,(R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$

The TD target can be seen as the 'best' estimation of G_t at time t+1:

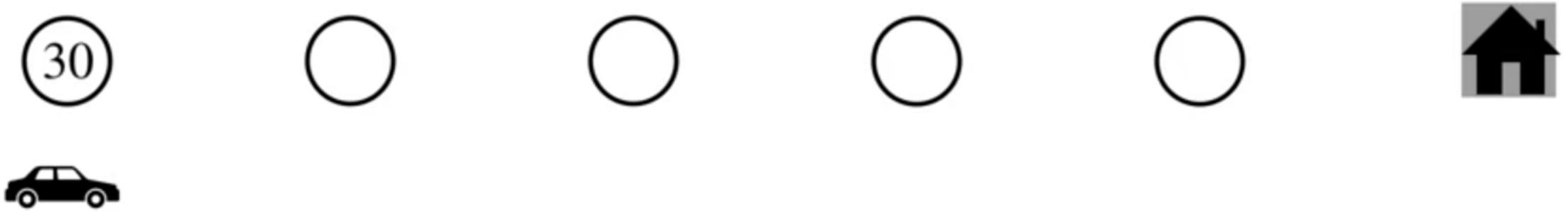
$$G_t = R_{t+1} + \gamma G(S_{t+1}) \sim R_{t+1} + \gamma V(S_{t+1})$$

Prediction: TD(0) - Driving example



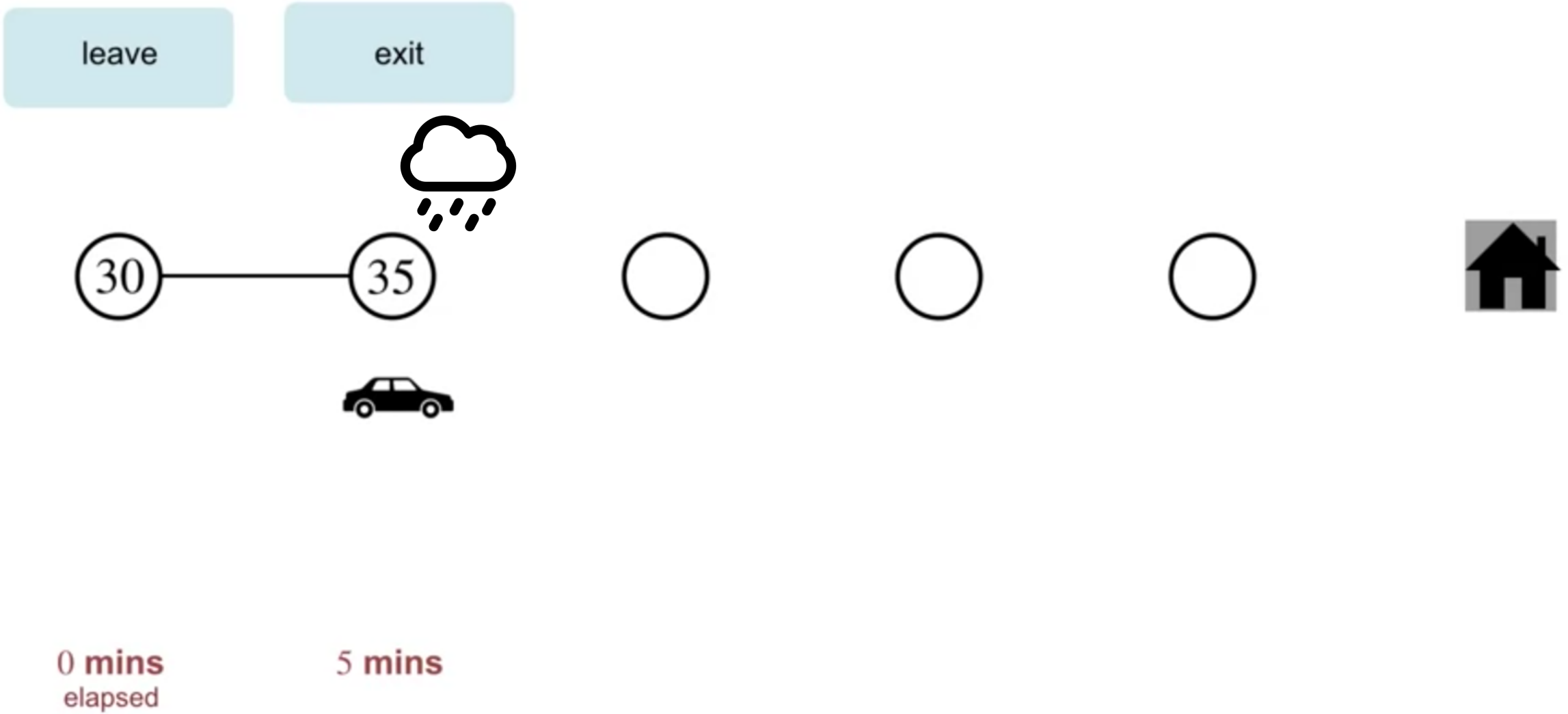
A. White, M. White 'Sample-based Learning Methods'

leave

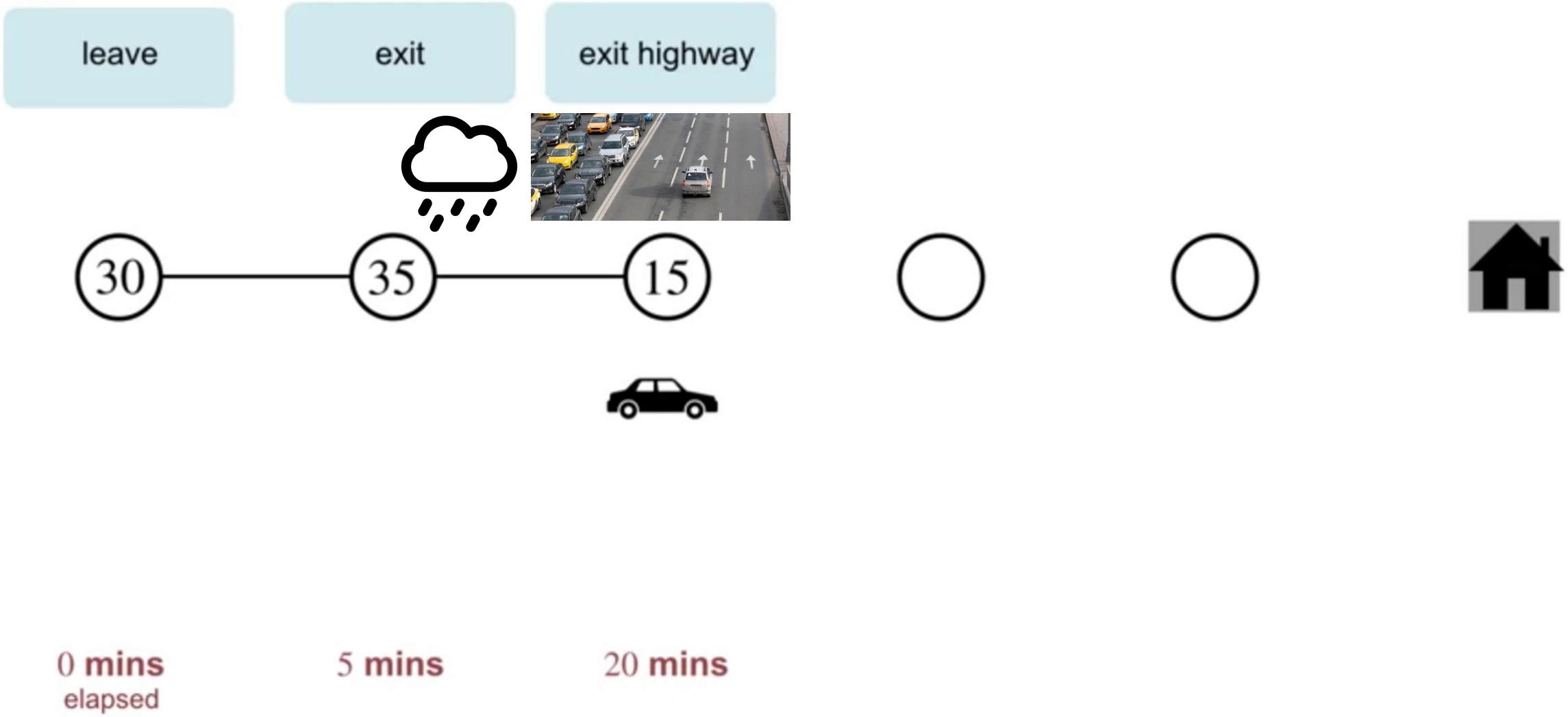


0 **mins**
elapsed

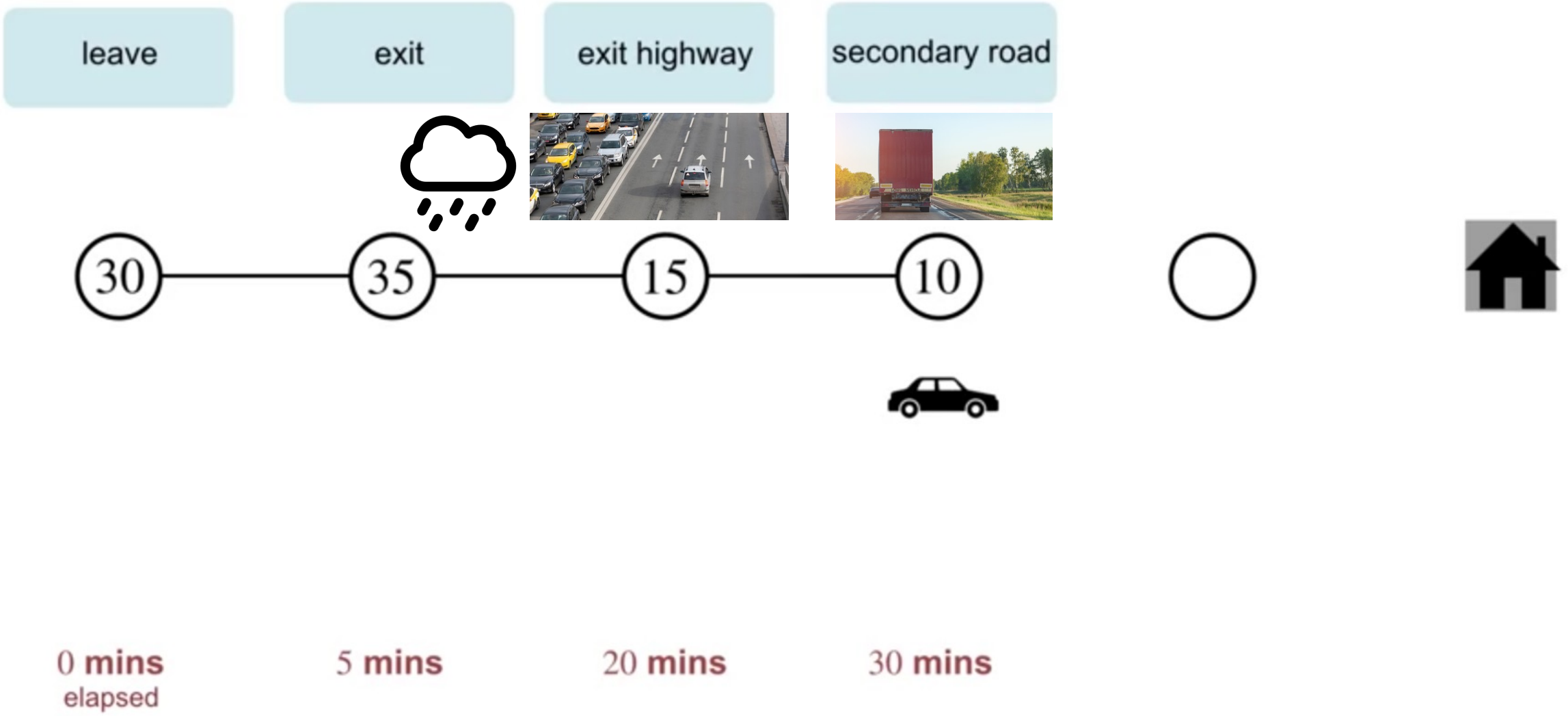
A. White, M. White 'Sample-based Learning Methods'



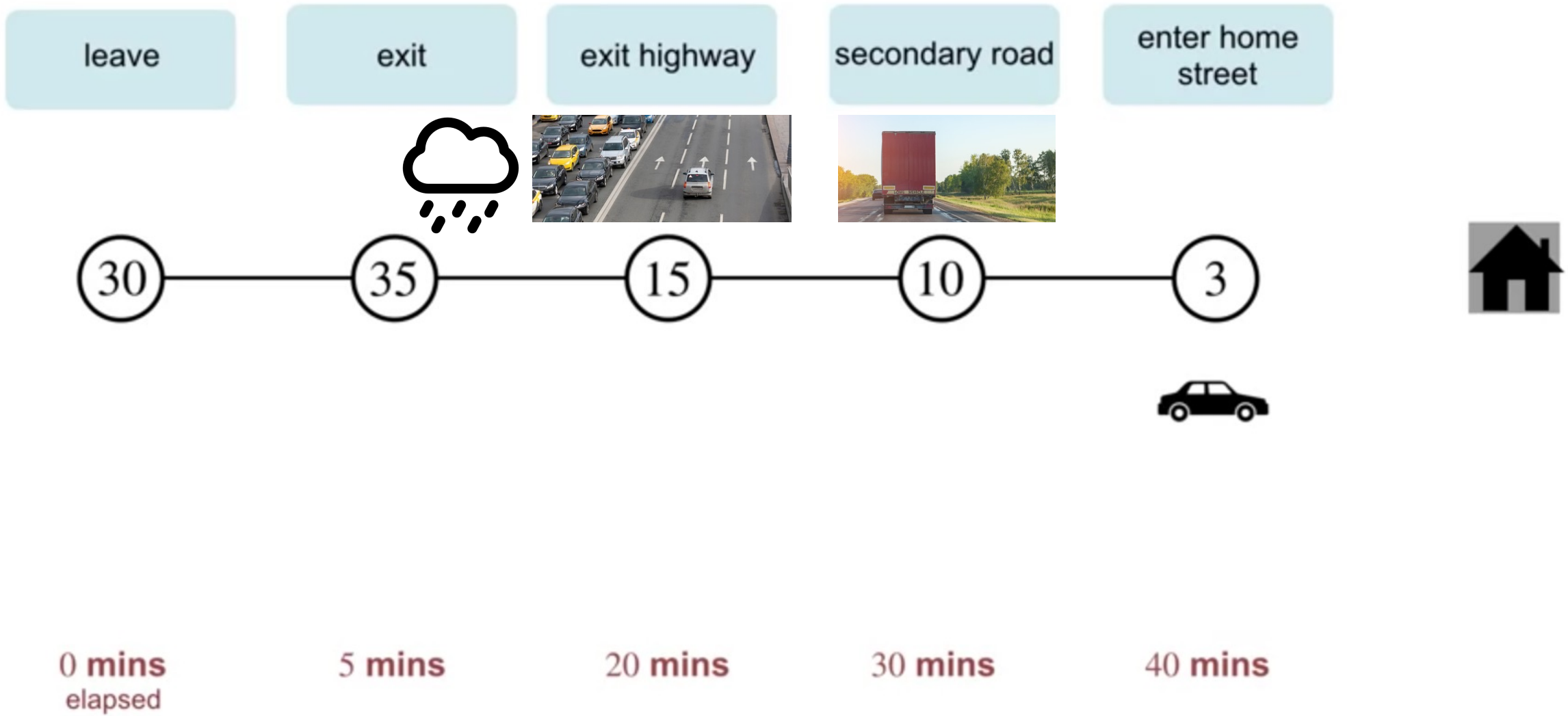
A. White, M. White 'Sample-based Learning Methods'



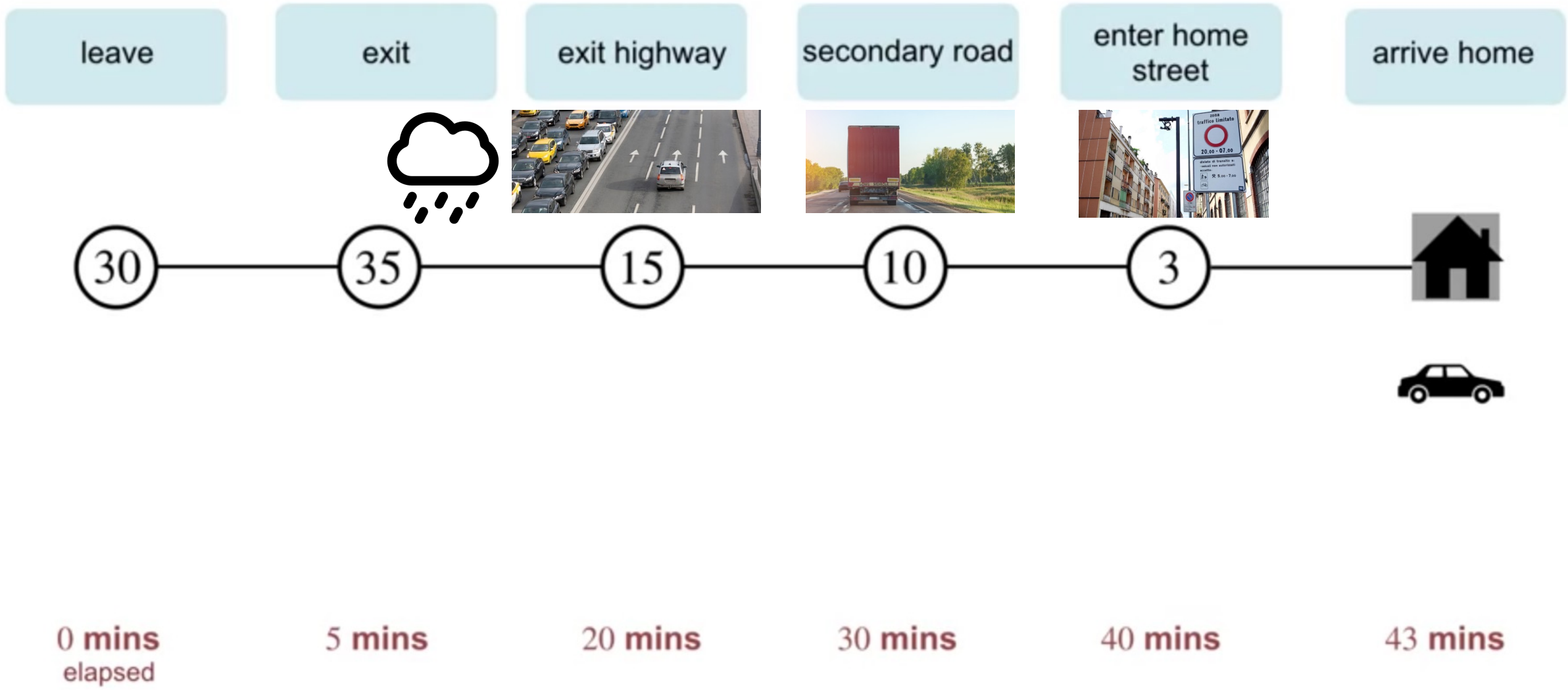
A. White, M. White 'Sample-based Learning Methods'



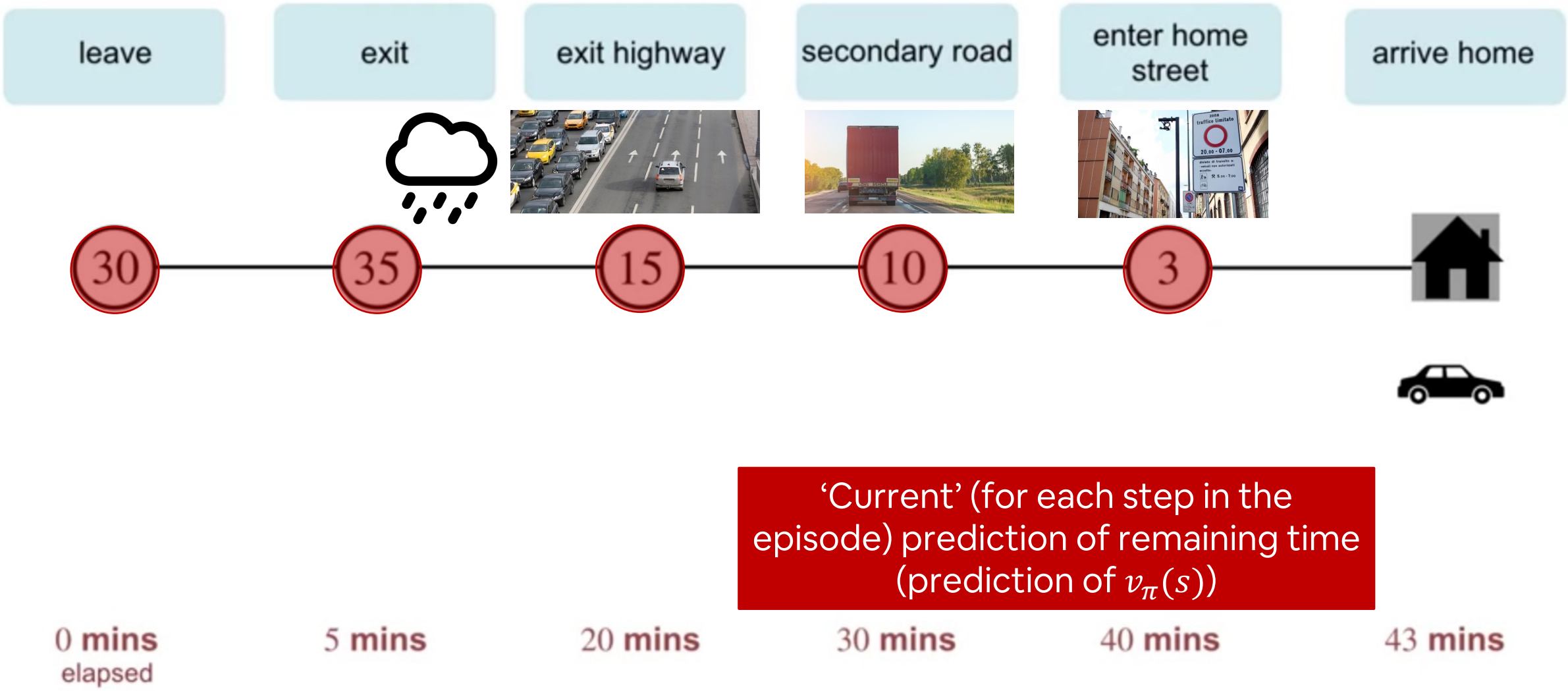
A. White, M. White 'Sample-based Learning Methods'



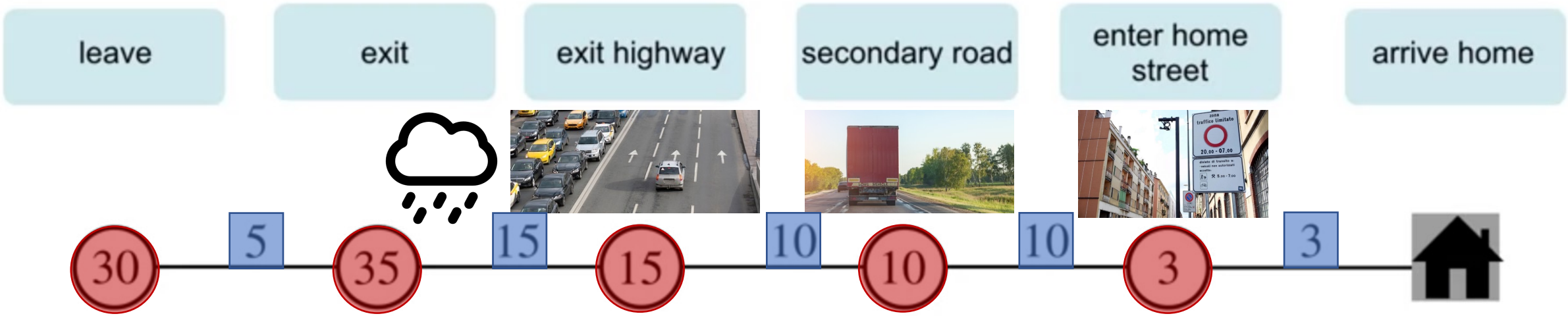
A. White, M. White 'Sample-based Learning Methods'



A. White, M. White 'Sample-based Learning Methods'



A. White, M. White 'Sample-based Learning Methods'

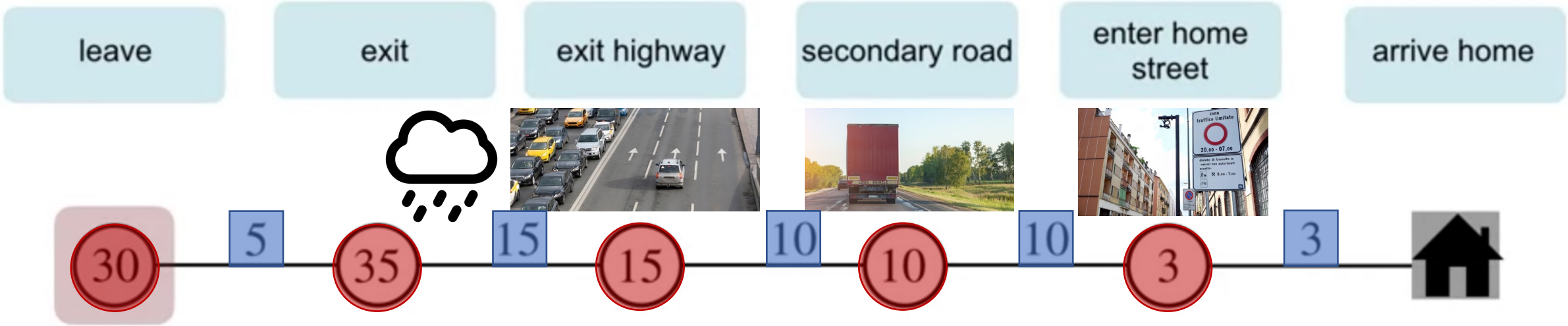


Rewards: the elapsed time for each steps

'Current' (for each step in the episode) prediction of remaining time (prediction of $v_\pi(s)$)

0 mins elapsed	5 mins	20 mins	30 mins	40 mins	43 mins

A. White, M. White 'Sample-based Learning Methods'



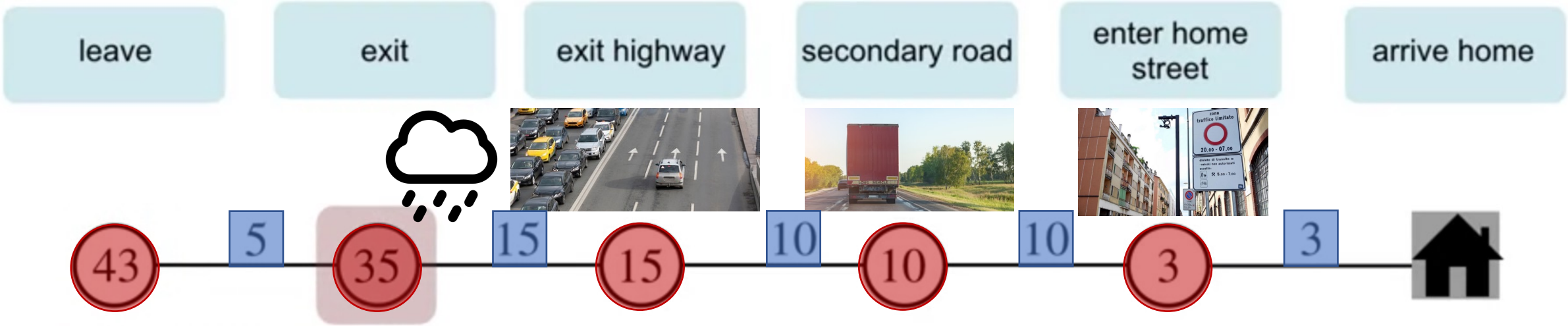
$$G_0 = 5 + 15 + 10 + 10 + 3 = 43$$

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \qquad \alpha = 1 \quad \gamma = 1$$

$$V(\textbf{leave}) \leftarrow V(\textbf{leave}) + \alpha[G_0 - V(\textbf{leave})]$$
$$30 \qquad\qquad 43 \qquad\qquad 30$$

Monte Carlo

A. White, M. White 'Sample-based Learning Methods'



$$G_1 = 15 + 10 + 10 + 3 = 38$$

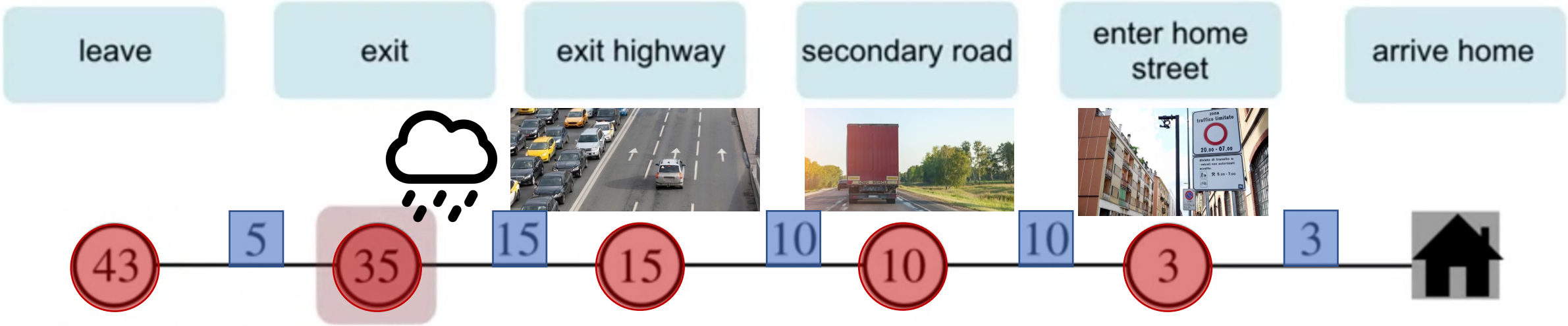
$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

$$\alpha = 1 \quad \gamma = 1$$

$$V(\textbf{exit}) \leftarrow V(\textbf{exit}) + \alpha[G_1 - V(\textbf{exit})]$$

Monte Carlo

A. White, M. White 'Sample-based Learning Methods'



$$G_1 = 15 + 10 + 10 + 3 = 38$$

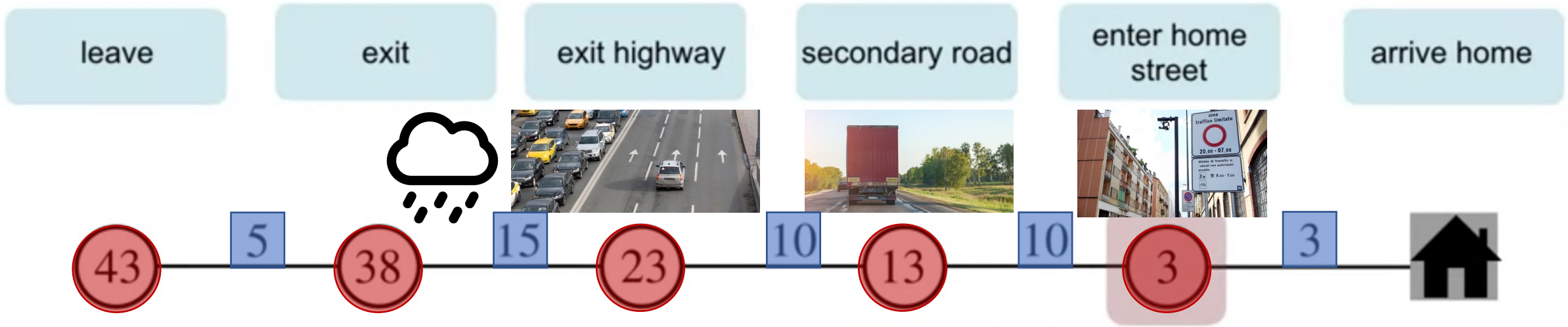
$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

$$\alpha = 1 \quad \gamma = 1$$

$$V(\textbf{exit}) \leftarrow V(\textbf{exit}) + \alpha[G_1 - V(\textbf{exit})]$$

Monte Carlo

A. White, M. White 'Sample-based Learning Methods'



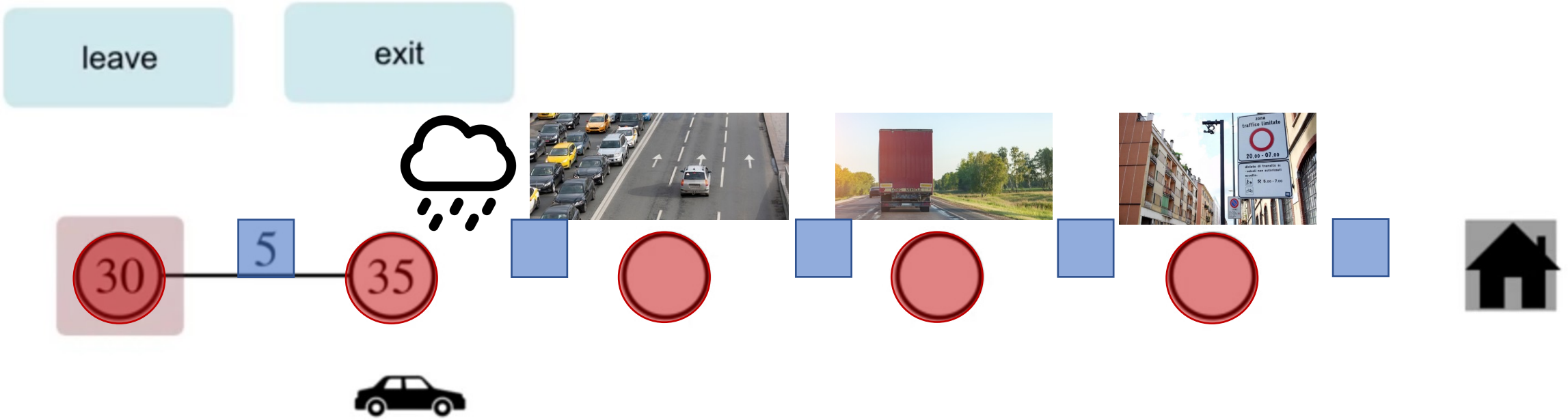
$$G_4 = 3$$

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)] \qquad \alpha = 1 \quad \gamma = 1$$

$$V(\textbf{enter home st.}) \leftarrow V(\textbf{enter home st.}) + \alpha[G_4 - V(\textbf{enter home st.})]$$
$$3 \qquad\qquad\qquad 3 \qquad\qquad 3$$

Monte Carlo

A. White, M. White 'Sample-based Learning Methods'



$$V(S_t) \leftarrow V(S_t) + \alpha[\, R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

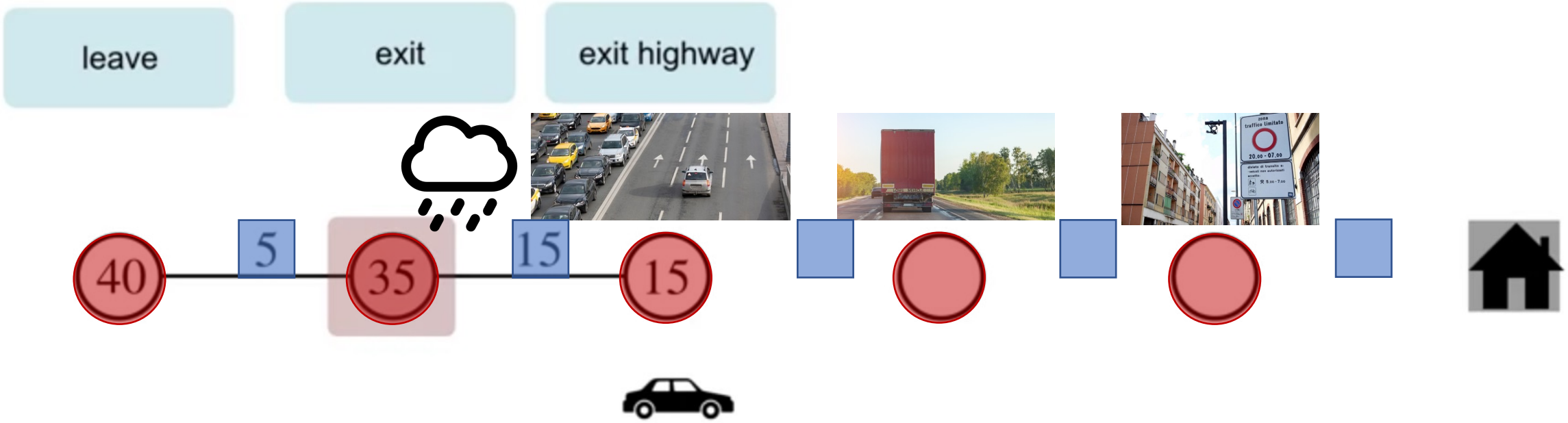
$$\alpha = 1 \quad \gamma = 1$$

$$V(\textbf{leave}) \leftarrow V(\textbf{leave}) + \alpha\,[\, R_1 + \gamma\, V(\textbf{exit}) - V(\textbf{leave})\,]$$

$$30 \qquad 5 \qquad 35 \qquad 30$$

TD(0)

A. White, M. White 'Sample-based Learning Methods'



$$V(S_t) \leftarrow V(S_t) + \alpha[\, R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

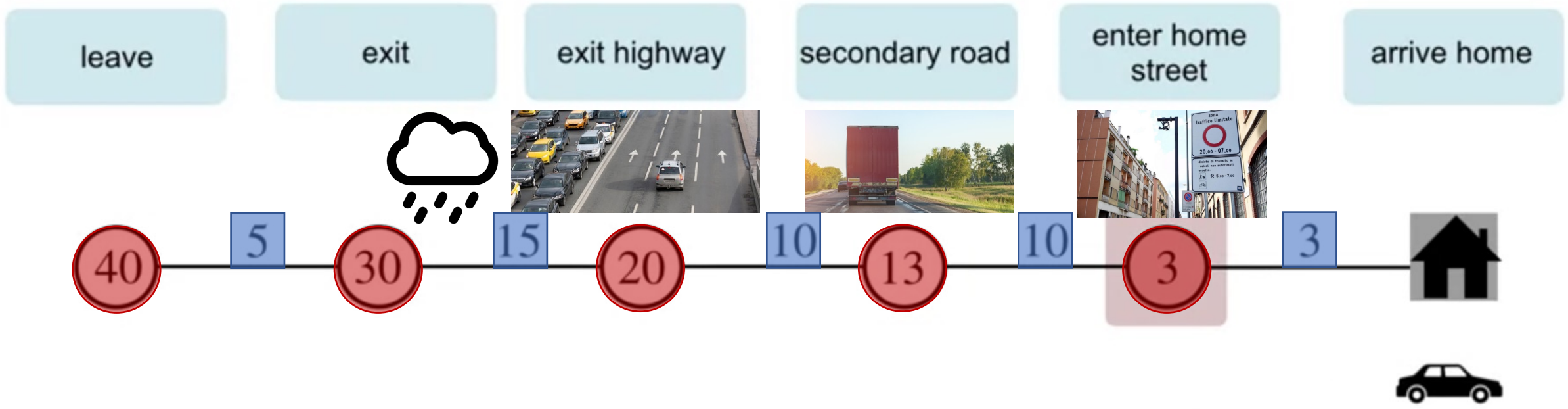
$\alpha = 1 \quad \gamma = 1$

$$V(\textbf{exit}) \leftarrow V(\textbf{exit}) + \alpha\,[\, R_2 + \gamma\, V(\textbf{exit highway}) - V(\textbf{exit})\,]$$

$$35 \qquad 15 \qquad 15 \qquad 35$$

TD(0)

A. White, M. White 'Sample-based Learning Methods'



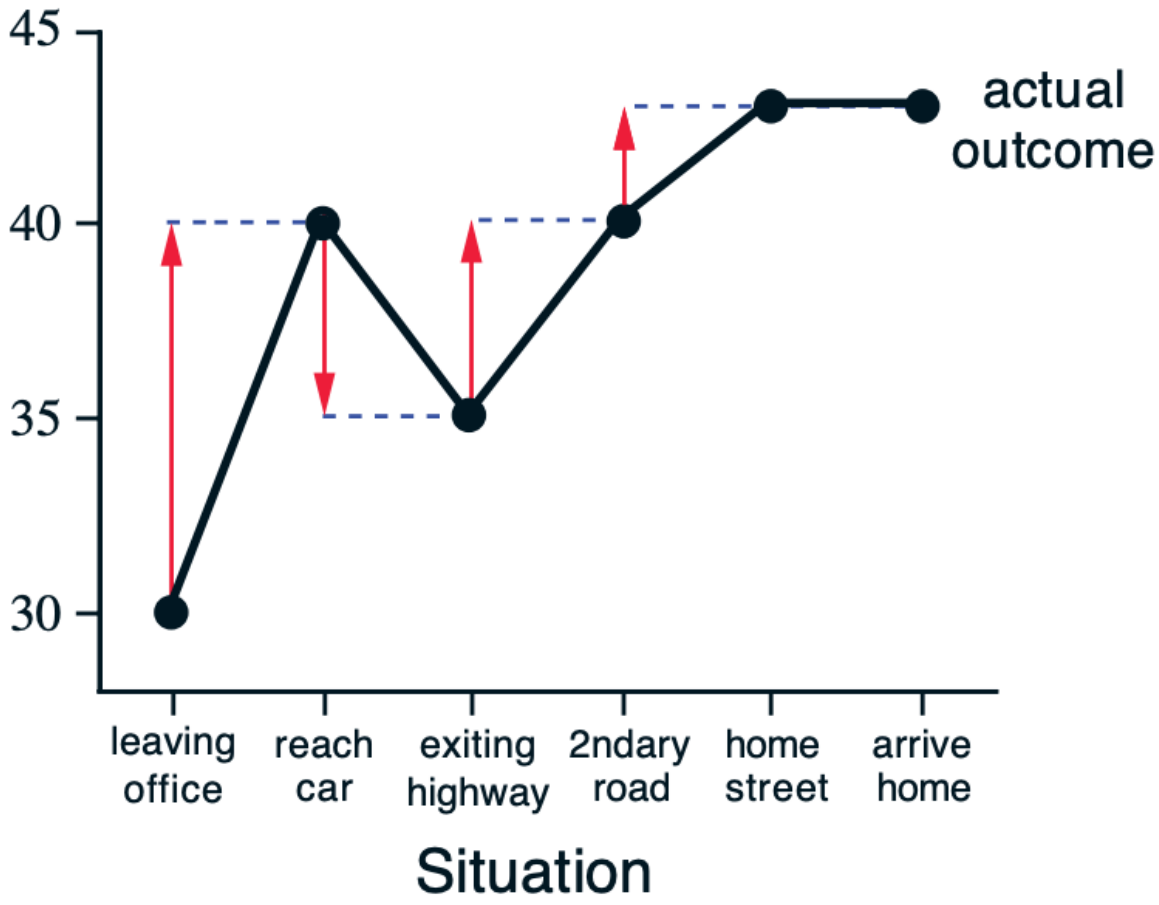
$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \qquad \alpha = 1 \quad \gamma = 1$$

$$V(\textbf{enter home st.}) \leftarrow V(\textbf{enter home st.}) + \alpha[R_5 + \gamma V(\textbf{home}) - V(\textbf{arrive home st.})]$$
$$3 \qquad\qquad 3 \qquad 0 \qquad 3$$

TD(0)

A. White, M. White 'Sample-based Learning Methods'

Prediction: TD(0) - Driving example



Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated

Algorithm parameter: step size $\alpha \in (0, 1]$

Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(terminal) = 0$

Loop for each episode:

  Initialize S

  Loop for each step of episode:

    $A \leftarrow$ action given by π for S

    Take action A, observe R, S'

    $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$

    $S \leftarrow S'$

  until S is terminal

We will see how the choice of α can have a major impact!

Tabular TD(0) for estimating v_π

Input: the policy π to be evaluated
Algorithm parameter: step size $\alpha \in (0, 1]$
Initialize $V(s)$, for all $s \in \mathcal{S}^+$, arbitrarily except that $V(\mathit{terminal}) = 0$

Loop for each episode:
    Initialize S
    Loop for each step of episode:
        $A \leftarrow$ action given by π for S
        Take action A, observe R, S'
        $V(S) \leftarrow V(S) + \alpha\big[R + \gamma V(S') - V(S)\big]$
        $S \leftarrow S'$
    until S is terminal

TD-Learning

- TD methods update their estimates based on other estimates, ie. they bootstrap
- TD can learn before knowing the final outcome:

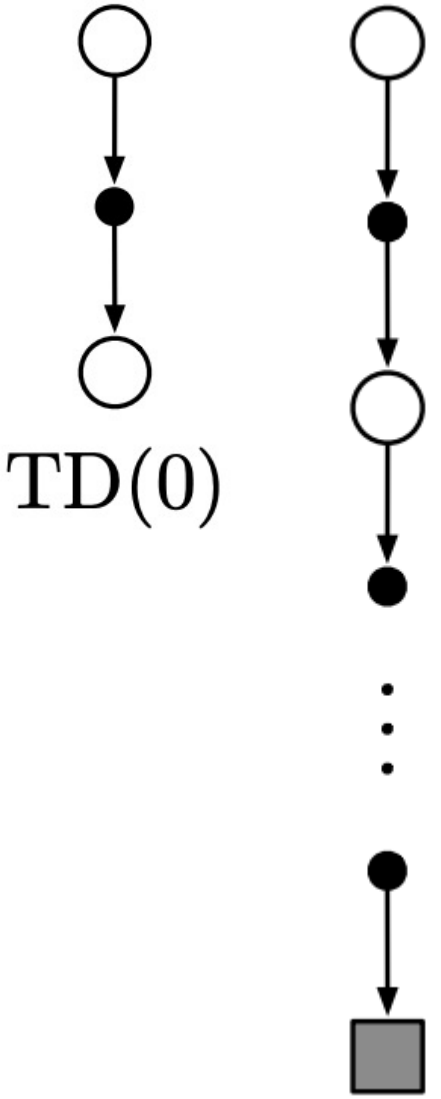
TD can learn online after every step / MC must wait until end of episode before return is known

- TD can learn without the final outcome

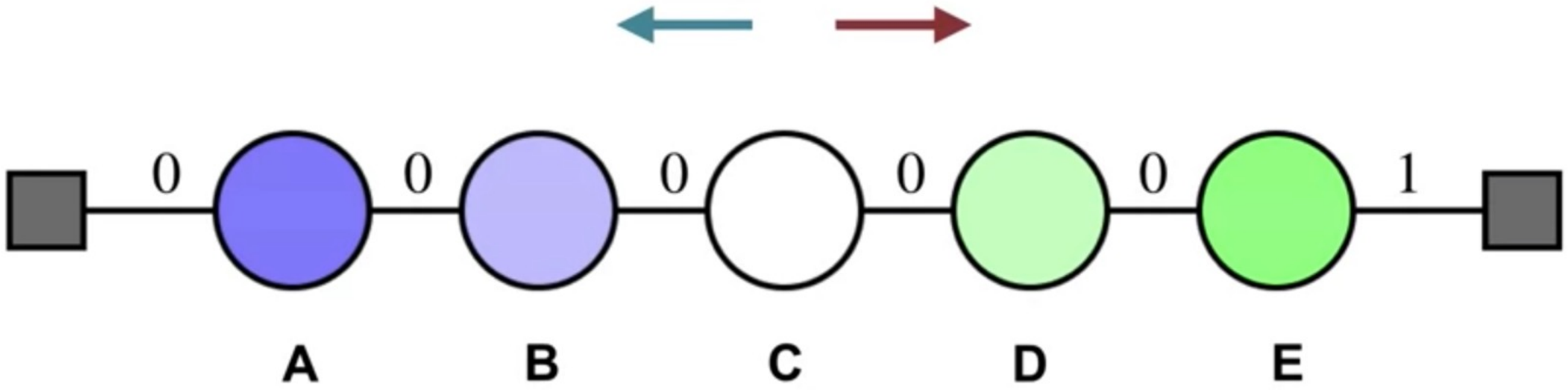
i. TD can learn from incomplete sequences / MC can only learn from complete sequences

ii. TD works in continuing (non-terminating) environments / MC only works for episodic (terminating) environments

- TD methods still converge to the right estimation of v_π
- Typically, TD approaches are faster to converge than MC!



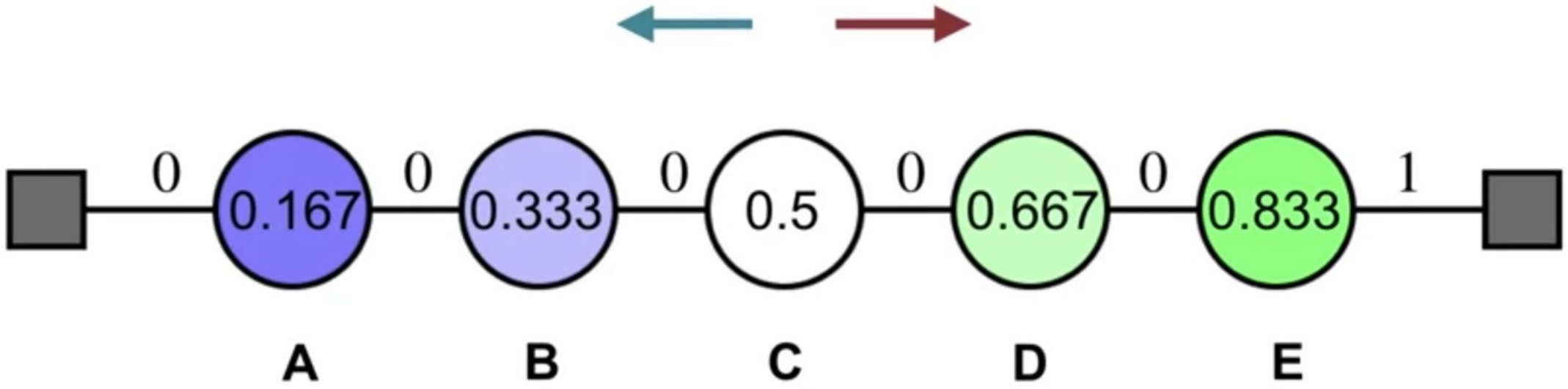
Prediction: TD(0) – Random Walk



$$\pi(\,.\,|\,s) = 1/2 \quad \forall s \in \mathcal{S} \qquad\qquad \gamma = 1$$

A. White, M. White 'Sample-based Learning Methods'

Prediction: TD(0) – Random Walk

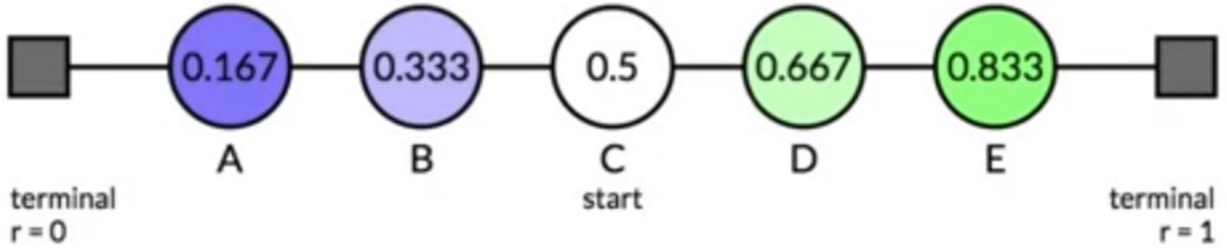


$$\pi(\,.\,|\,s) = 1/2 \quad \forall s \in \mathcal{S} \qquad\qquad \gamma = 1$$

A. White, M. White 'Sample-based Learning Methods'

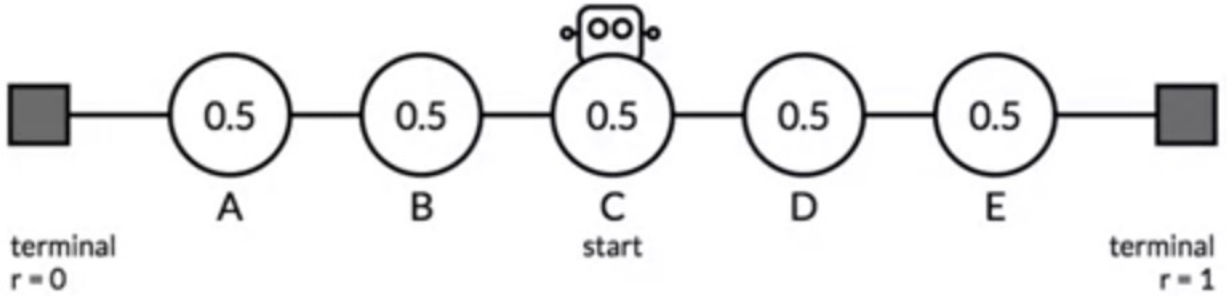
Episode #1

Target / Exact Values



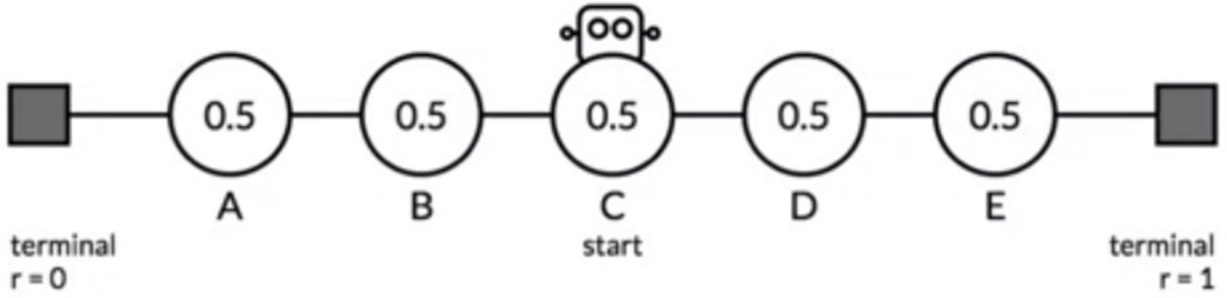
Updates using TD Learning

$$V(S_t) \leftarrow V(S_t) + \alpha[\, R_{t+1} + \gamma V(S_{t+1}) \; - V(S_t)]$$



In the following:
$\alpha = 0.5$

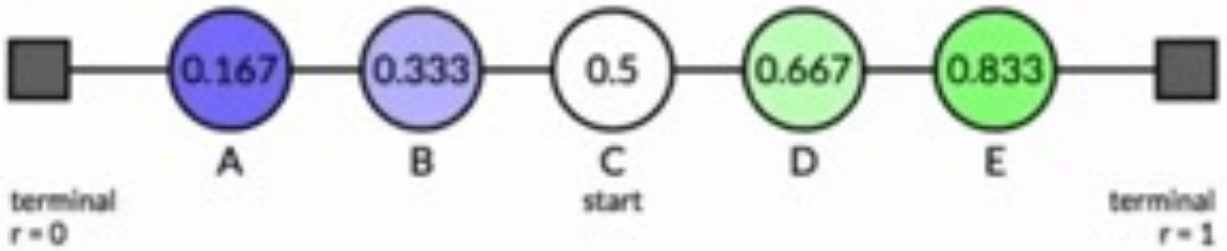
Updates using Monte Carlo



A. White, M. White 'Sample-based Learning Methods'

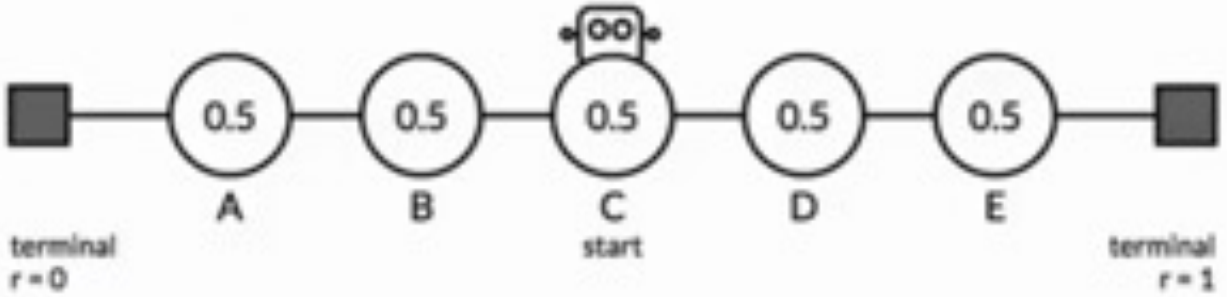
Episode #1

Target / Exact Values

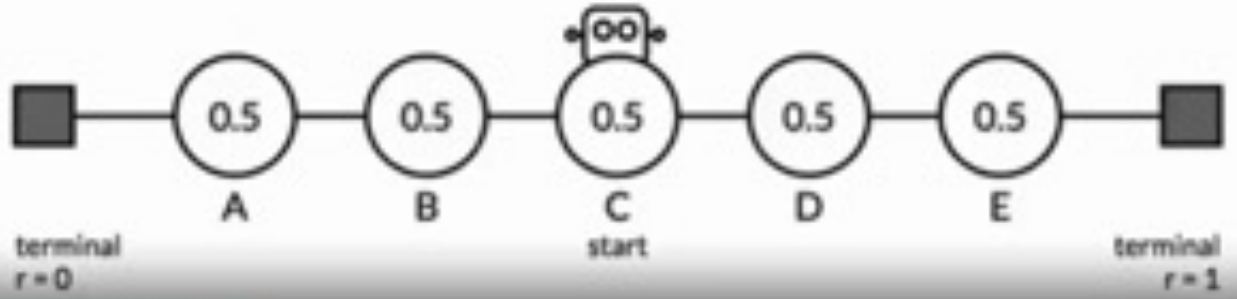


Updates using TD Learning

$$V(S_t) \leftarrow V(S_t) + \alpha[\, R_{t+1} + \gamma V(S_{t+1}) \, - V(S_t)]$$



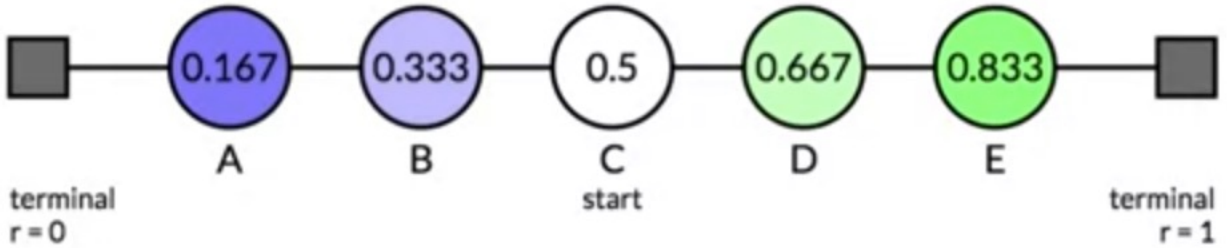
Updates using Monte Carlo



A. White, M. White 'Sample-based Learning Methods'

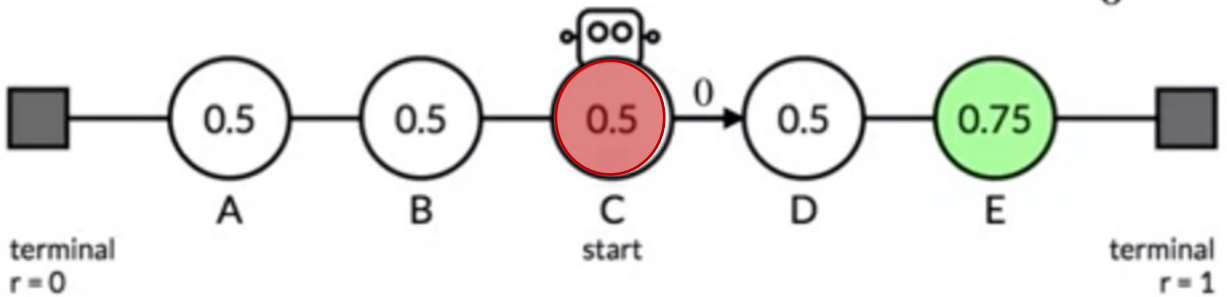
Episode #1

Target / Exact Values

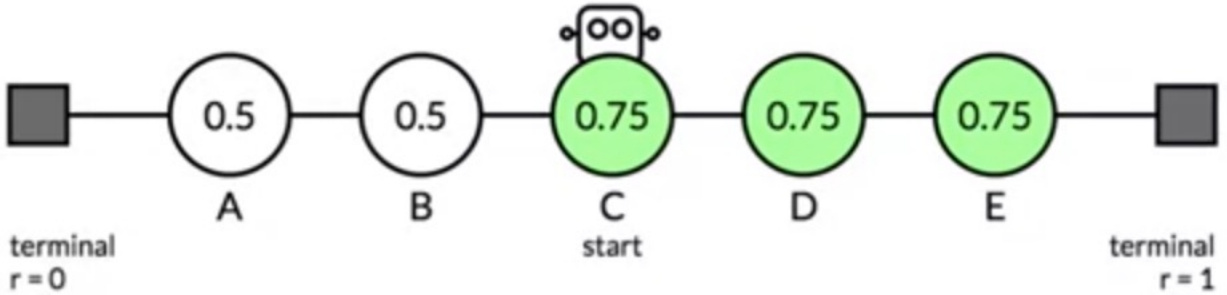


Updates using TD Learning

$$V(S_t) \leftarrow V(S_t) + \alpha[\underset{0}{R_{t+1}} + \gamma \underset{0.5}{V(S_{t+1})} - \underset{0.5}{V(S_t)}]$$



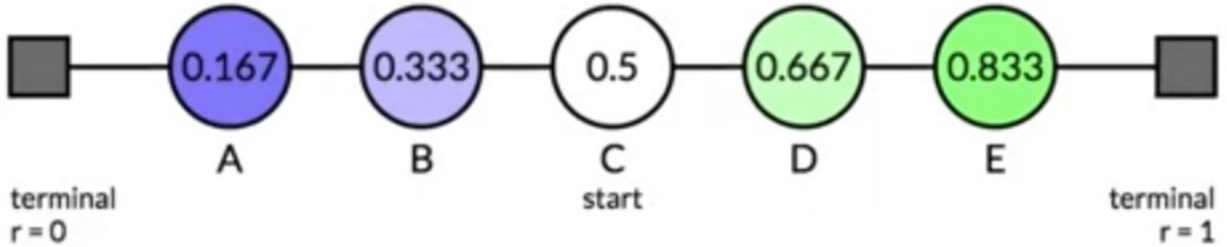
Updates using Monte Carlo



A. White, M. White 'Sample-based Learning Methods'

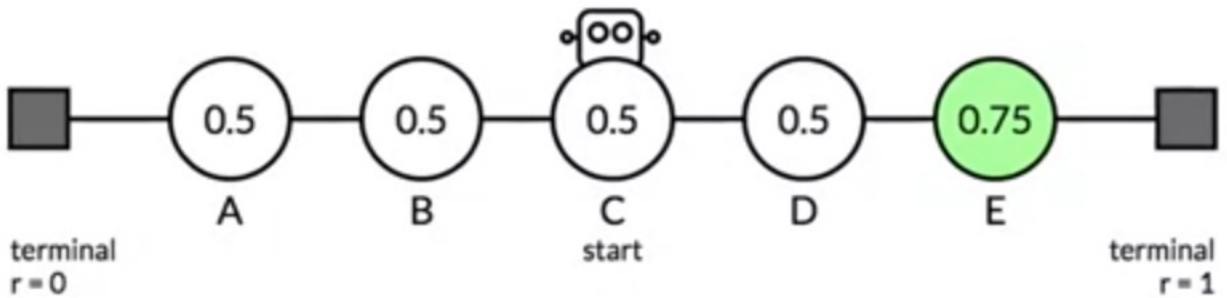
Episode #1

Target / Exact Values



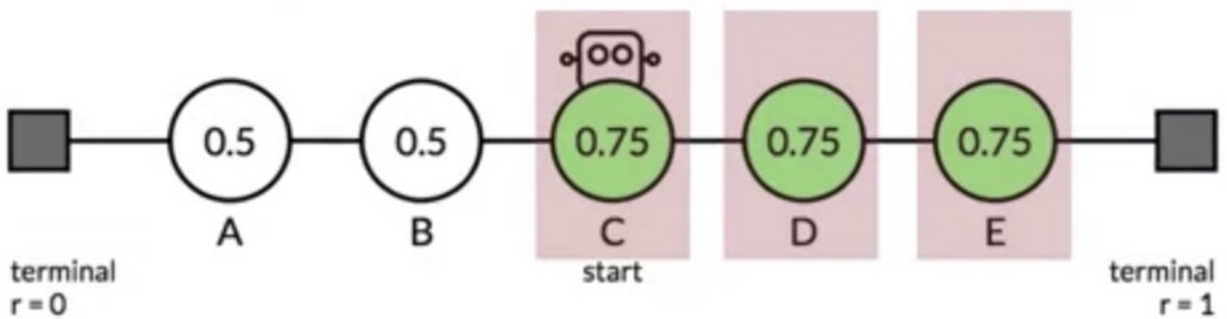
Updates using TD Learning

$$V(S_t) \leftarrow V(S_t) + \alpha[\, R_{t+1} + \gamma V(S_{t+1}) \, - V(S_t)]$$



Updates using Monte Carlo

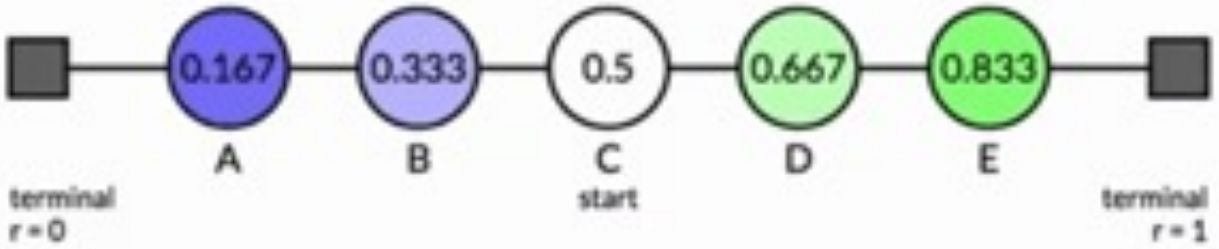
$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$



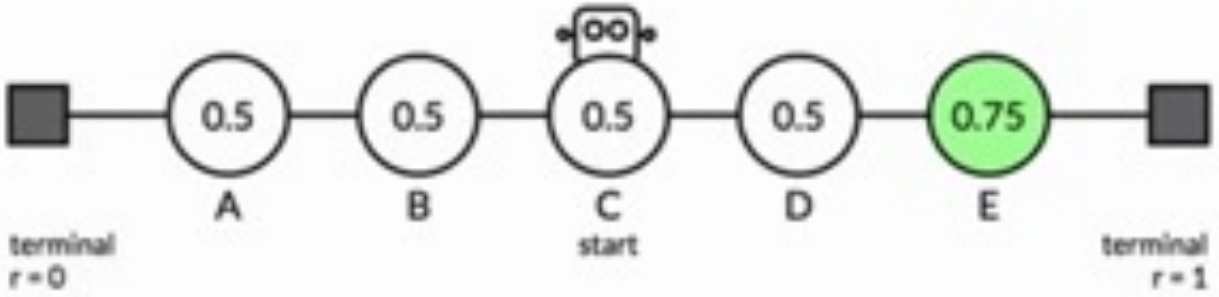
A. White, M. White 'Sample-based Learning Methods'

Episode #2

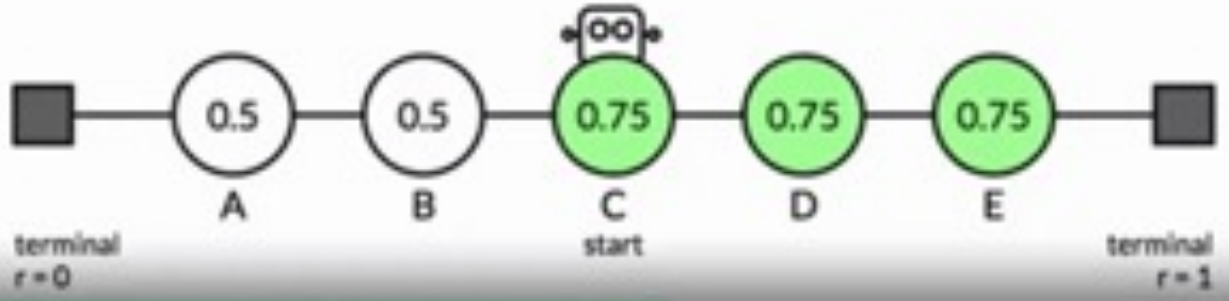
Target / Exact Values



Updates using TD Learning

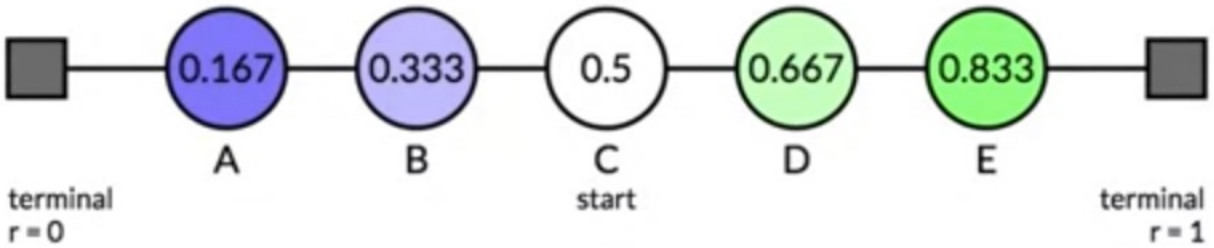


Updates using Monte Carlo

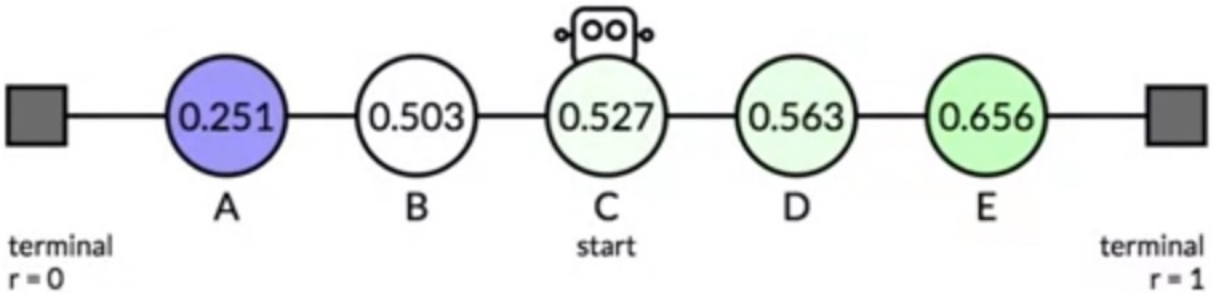


A. White, M. White 'Sample-based Learning Methods'

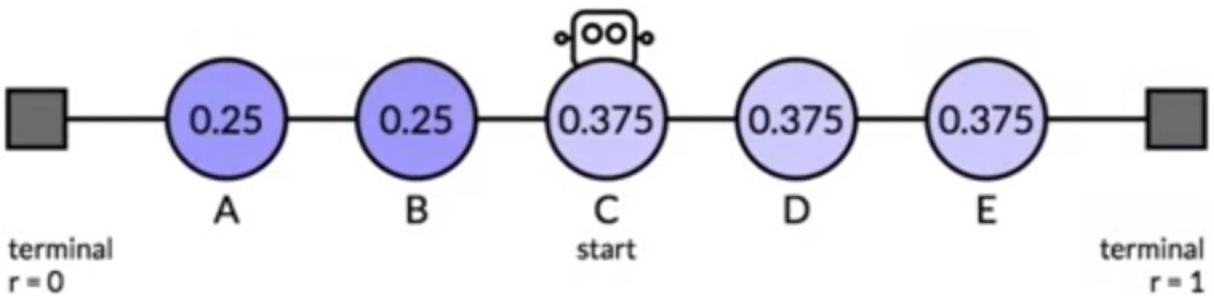
Target / Exact Values



Updates using TD Learning



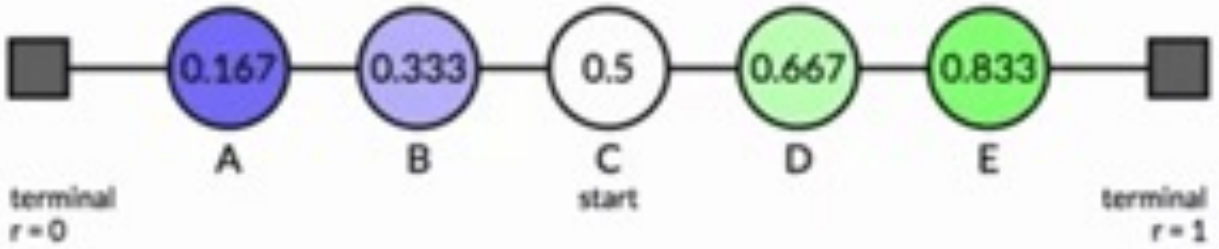
Updates using Monte Carlo



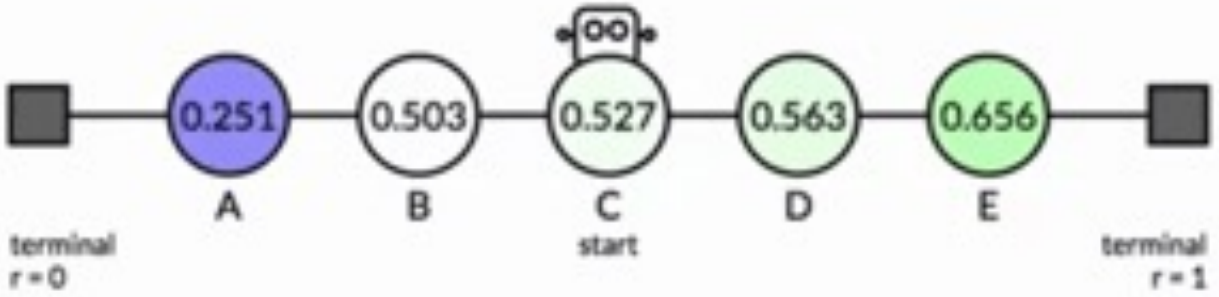
A. White, M. White 'Sample-based Learning Methods'

Additional Episodes

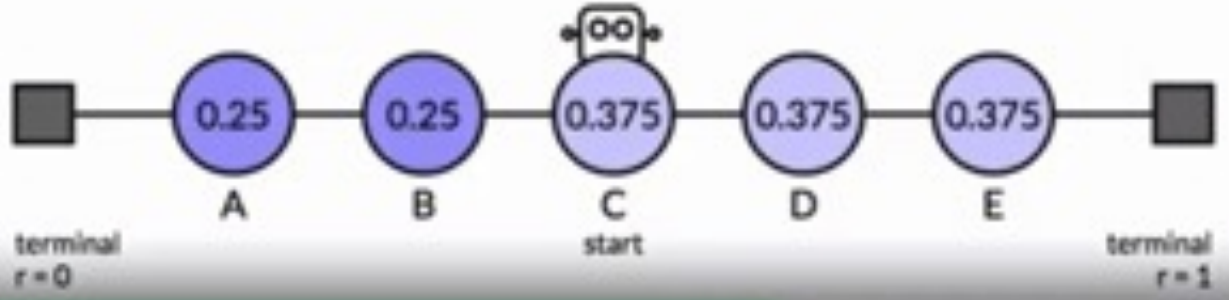
Target / Exact Values



Updates using TD Learning



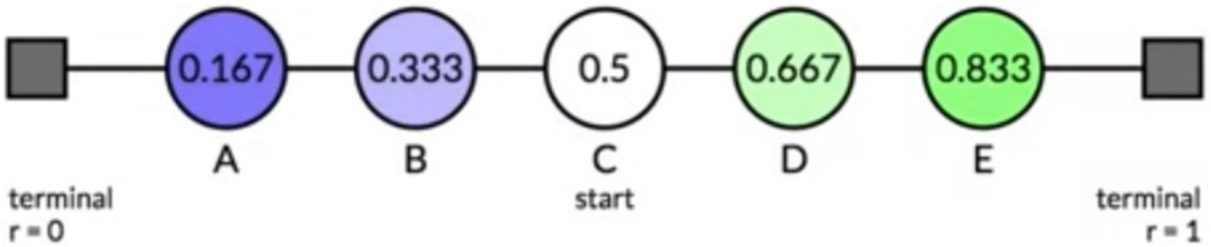
Updates using Monte Carlo



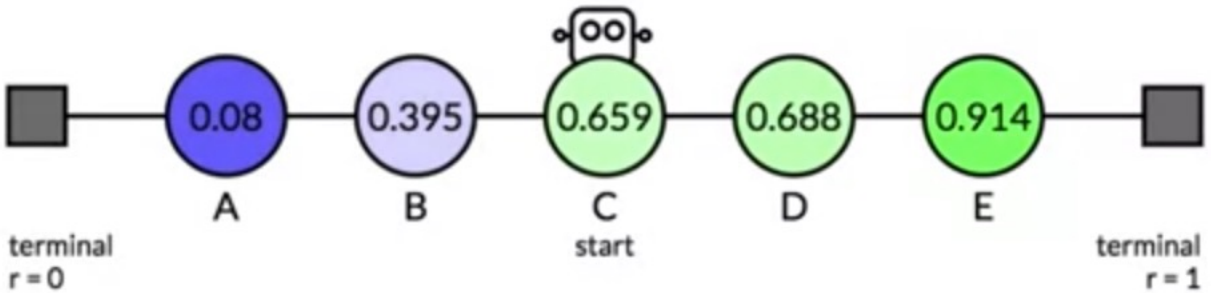
A. White, M. White 'Sample-based Learning Methods'

Additional Episodes

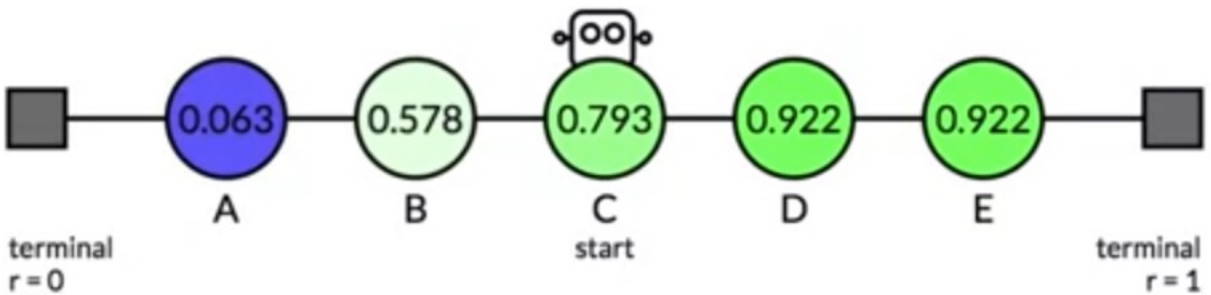
Target / Exact Values



Updates using TD Learning

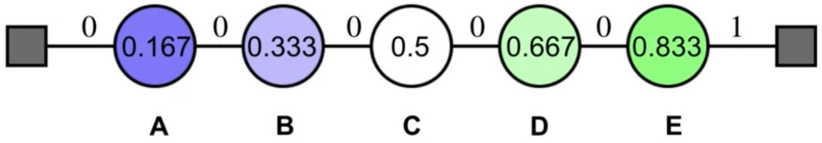


Updates using Monte Carlo

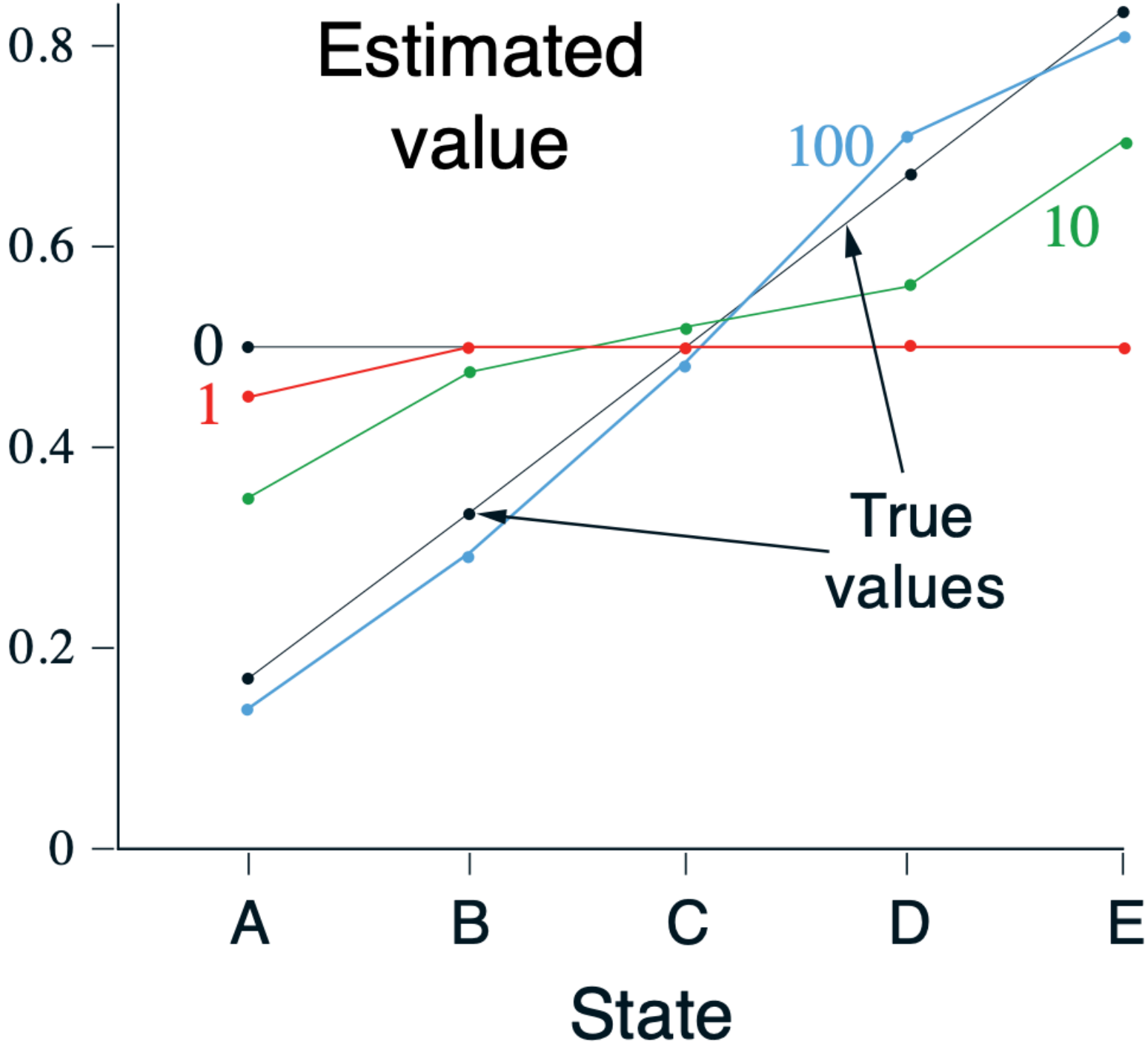


A. White, M. White 'Sample-based Learning Methods'

Prediction: TD(0) – Random Walk

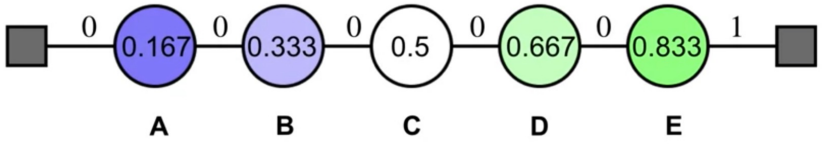


A. White, M. White 'Sample-based Learning Methods'

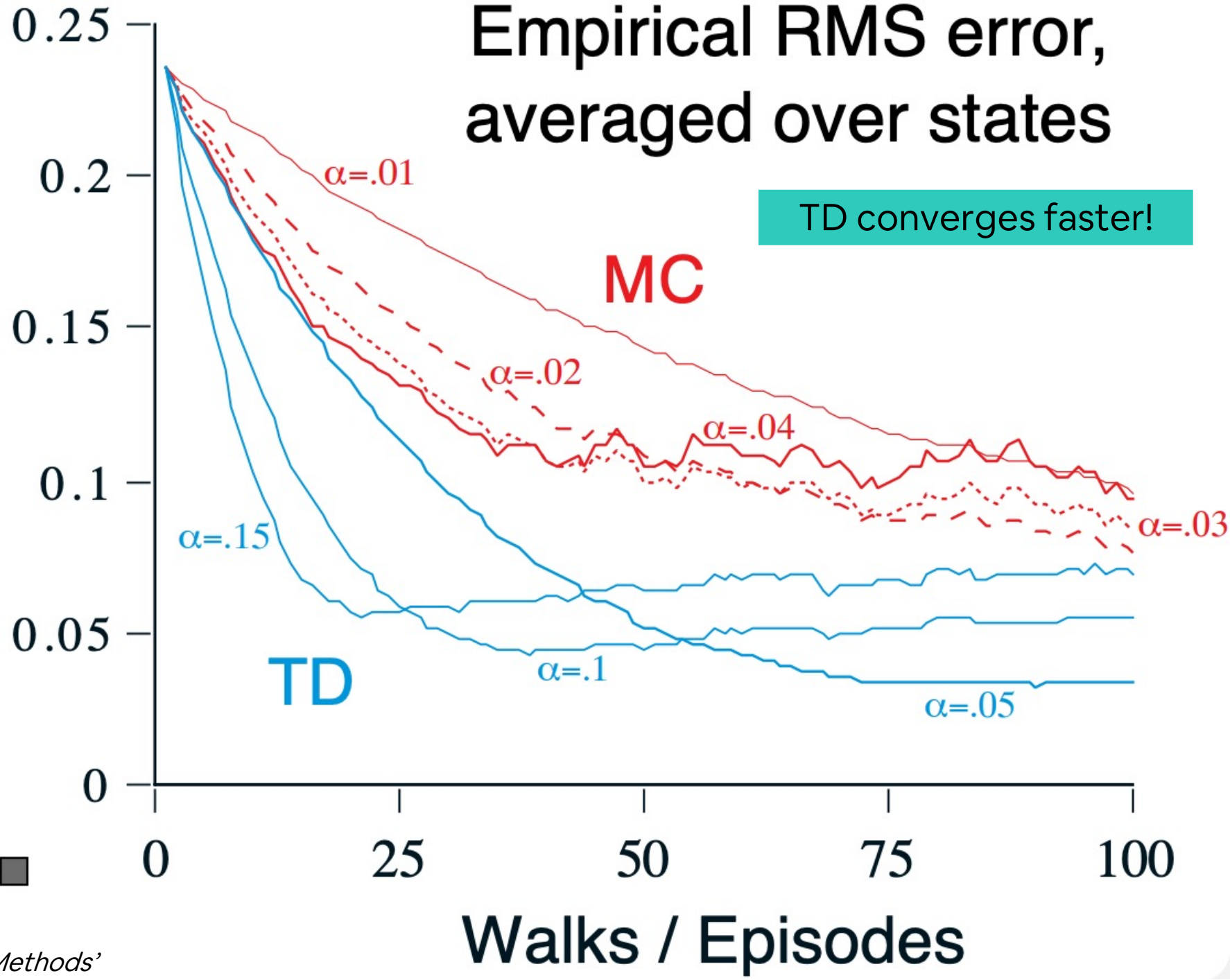


Prediction: TD(0) – Random Walk

Estimation error - Root Mean Square (RMS) – averaged over the 5 states, the averaged over 100 runs

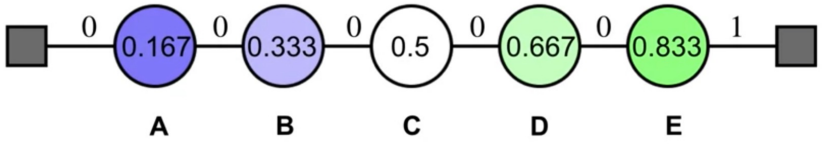


A. White, M. White 'Sample-based Learning Methods'

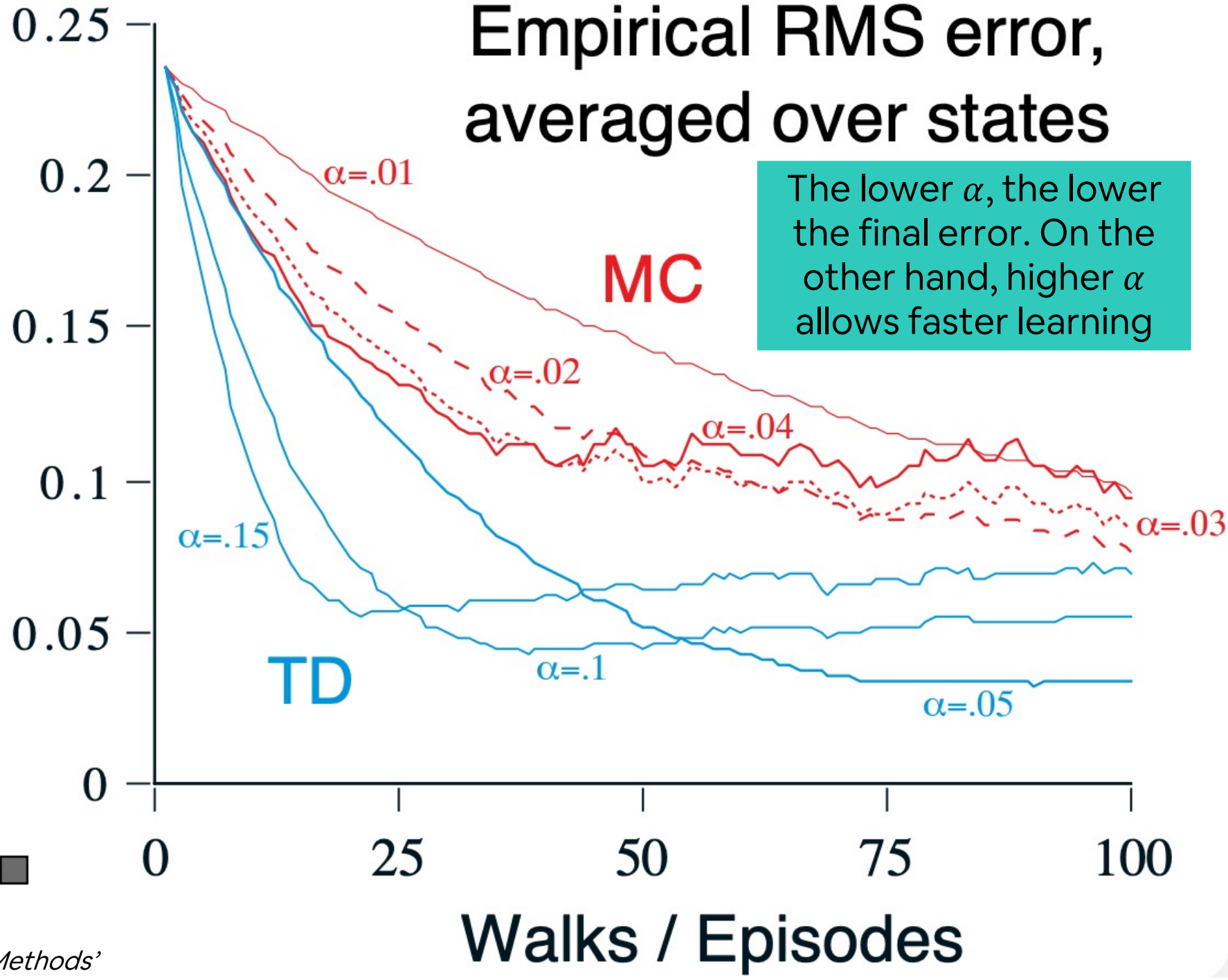


Prediction: TD(0) – Random Walk

Estimation error - Root Mean Square (RMS) – averaged over the 5 states, the averaged over 100 runs



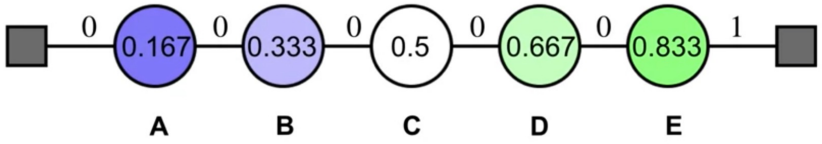
A. White, M. White 'Sample-based Learning Methods'



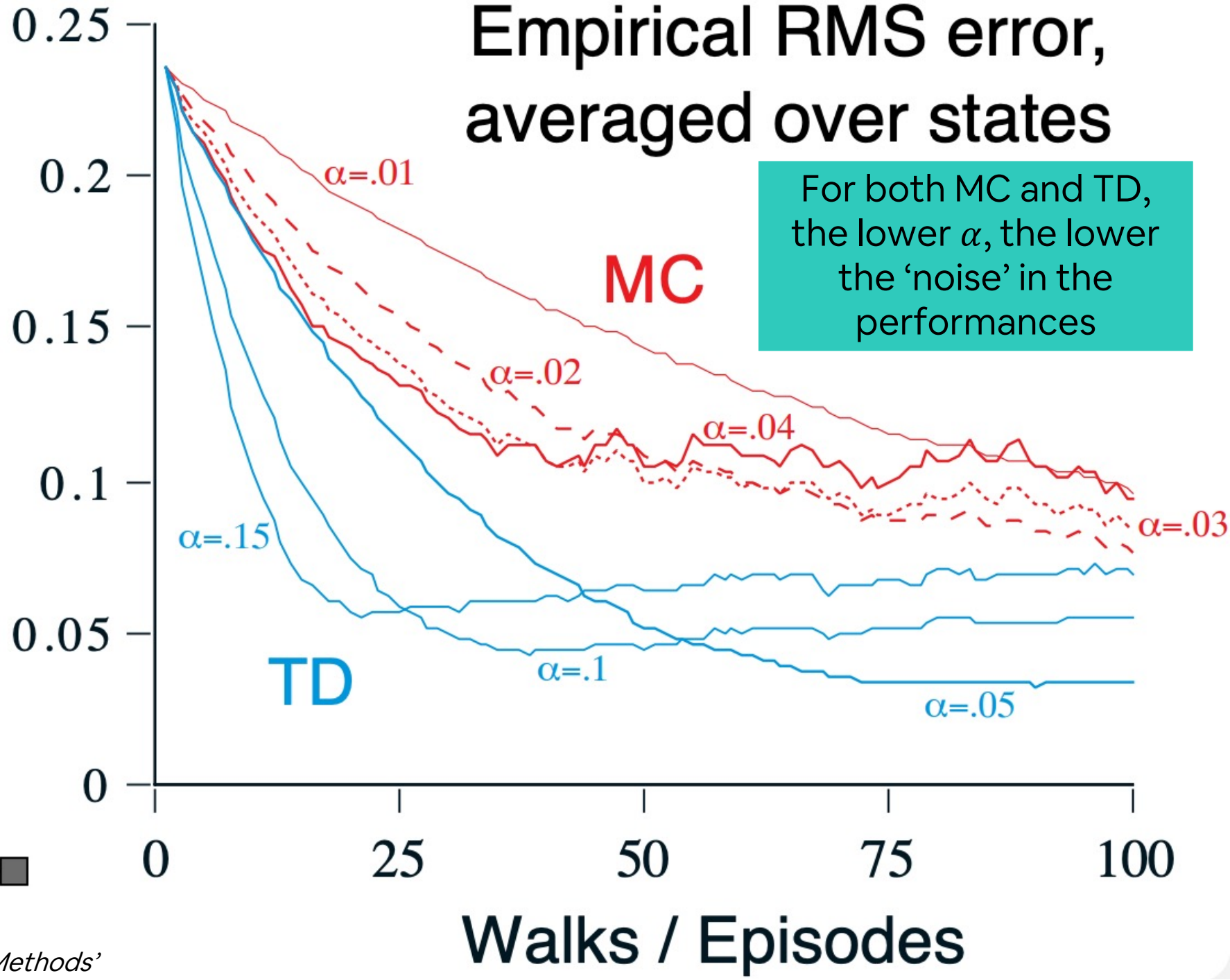
The lower α, the lower the final error. On the other hand, higher α allows faster learning

Prediction: TD(0) – Random Walk

Estimation error - Root Mean Square (RMS) – averaged over the 5 states, the averaged over 100 runs



A. White, M. White 'Sample-based Learning Methods'



Bias/Variance Trade-off

- The return $G_t = R_{t+1} + \gamma R_{t+2} + \ ... + \gamma^{T-1} R_T$ is an unbiased estimate of $v_\pi(S_t)$
- True TD target $R_{t+1} + \gamma v_\pi(S_{t+1})$ (we don't know the true value function!) is an unbiased estimate of $v_\pi(S_t)$
- TD target $R_{t+1} + \gamma V(S_{t+1})$ (the target that we move to, it can be quite a wrong estimate!) is a biased estimate of $v_\pi(S_t)$
- TD target is much lower variance than the return:

1. Return depends on many random actions, transitions, rewards
2. TD target depends on one random action, transition, reward

Monte Carlo vs. TD(0)

- MC has high variance, zero bias

1. Good convergence properties (even with function approximation – Chapter 9, we will see this later in the course)
2. Not very sensitive to initial value
3. Very simple to understand and use

- TD has low variance, some bias

1. Usually more efficient than MC
2. TD(0) converges to $v_\pi(s)$ (but not always with function approximation)
3. More sensitive to initial values

Credits

- Image of the course is taken from C. Mahoney ‘Reinforcement Learning’ https://towardsdatascience.com/reinforcement-learning-fda8ff535bb6
- Many concepts and material for examples have been taken from A. White, M. White ‘Sample-based Learning Methods’
- The bias/variance trade-off was taken from D. Silver ‘Reinforcement Learning’ course @ UCL

AMCO
ARTIFICIAL INTELLIGENCE, MACHINE LEARNING AND CONTROL RESEARCH GROUP

Reinforcement Learning 2025/2026

RL2
Reinforcement Learning Research Lab @ UniPD

Thank you!
Questions?

Lecture #08
Off-Policy Monte Carlo & Temporal Difference Learning

Gian Antonio Susto