NOVA IMS Information Management School

Reinforcement Learning

Lecture 4

Model Free Methods

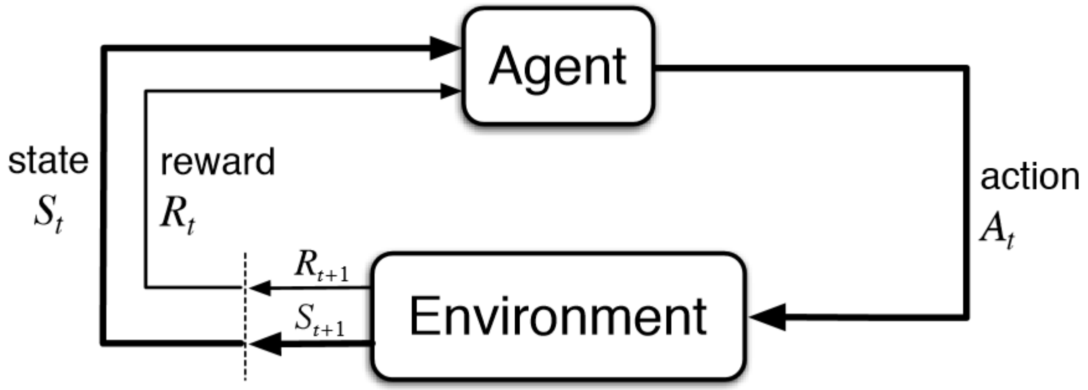
May 2025

Nuno Alpalhão
nalpalhao@novaims.unl.pt

Instituto Superior de Estatística e Gestão de Informação
Universidade Nova de Lisboa

Last Class RL Agent’s Components

- All components are functions
  - **Policies** $\pi : S \rightarrow A$ (or to probabilities over A)
  - **Value functions** $v : S \rightarrow R$
    - **Q functions** $q : S \times A \rightarrow R$
  - **Models** $m : S \rightarrow S$ and/or $r : S \rightarrow R$
  - **State update** $u : S \times O \rightarrow S$



Last Class

Model Free - Monte Carlo

Monte Carlo (MC) is one of the most popular and most commonly used algorithms in various fields ranging from physics and mechanics to computer science. The Monte Carlo algorithm is used in reinforcement learning (RL) when the model of the environment is not known.

Using **Dynamic Programming** (DP) to **find an optimal policy** where **we know the model dynamics**, which is **transition and reward probabilities**. But how can we determine the optimal policy when we don't know the model dynamics? In that case, we use the Monte Carlo algorithm; it is extremely powerful for finding optimal policies when we don’t have knowledge of the environment.

Last Class

In the **Monte Carlo Prediction** setup (in the prediction vs control realm) we are solely trying to achieve a **value function**.

We are **assuming an a priori policy** (which we normally do not have).

Disadvantages of Monte Carlo Learning

Monte Carlo algorithms can be used to learn value predictions:

- When episodes are long, **learning can be slow.**
- **Wait until an episode ends** before we can learn.
- Return can have **high variance.**

Last Class

Prediction vs Control

Estimating v_π or Q^π is called policy evaluation or, simply, **prediction.**

- Given a policy, what is my expected return under that behaviour?
- Given this treatment protocol/trading strategy, what is my expected return?

Estimating v_* or Q^* is sometimes called **control**, because these can be used for policy optimisation.

- What is the optimal way of behaving? What is the optimal value function?
- What is the optimal treatment? What is the optimal control policy to minimise time, fuel consumption, etc?

Incremental Monte Carlo Updates

Update $\boldsymbol{v(s)}$ **incrementally** after each episode:

For each state S_t with return G_t:

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

In non-stationary problems, it can be useful to track a running mean, i.e. forget old episodes:

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

Last Class

Incremental Monte Carlo Updates Overview

Learn v_π online from experience under policy π.

Incremental every-visit Monte Carlo:

- Update value $v(S_t)$ toward actual return G_t:

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

α can take many values:

$\alpha = \frac{1}{N(S_t)}$ or even $\alpha = 1$

Monte Carlo Key Limitations

Generally high variance estimator:

- Reducing variance can require a lot of data.
- In cases where data is very hard or expensive to acquire, or the stakes are high, Monte Carlo may be impractical.

Requires episodic settings:

- Episode must end before data can be used to update the value function.

Model Free Approaches Temporal-Difference Learning

“If one had to identify one idea as central and novel to reinforcement learning, it would undoubtedly be temporal-difference (TD) learning.” – Sutton and Barto 2017

NOVA IMS Information Management School

Model Free Approaches Temporal-Difference Learning

Temporal Difference methods **learn immediately from each step in an episode**.

Temporal Difference **is model-free**: no knowledge of MDP transitions / rewards.

Temporal Difference learns from incomplete episodes, by **bootstrapping**.

Temporal Difference can be used in episodic or **infinite-horizon** non-episodic settings.

Temporal-Difference Learning – Mathematical Intuition

Bellman's equations:

$$v_\pi(s) = E\left[R_{t+1} + \gamma\, v_\pi(S_{t+1}) \mid S_t = s, A_t \sim \pi(S_t)\right]$$

Approximate by iterating:

$$v'(s) = E\left[R_{t+1} + \gamma\, v(S_{t+1}) \mid S_t = s, A_t \sim \pi(S_t)\right]$$

We can sample this:

$$v'(S_t) = R_{t+1} + \gamma\, v(S_{t+1})$$

Model Free Approaches Temporal-Difference Learning

We can sample this:

$$v'(S_t) = \boldsymbol{R_{t+1}} + \gamma\, v(S_{t+1})$$

This is likely quite noisy — better to take a small step (with parameter α):

$$v'(S_t) = v(S_t) + \alpha[\boldsymbol{R_{t+1}} + \gamma\, v(S_{t+1}) - v(S_t)]$$

Remember in **Incremental Monte Carlo Updates**:

$$v'(S_t) \leftarrow v\,(S_t) + \alpha\big(\boldsymbol{G_t} - v\,(S_t)\big)$$

Model Free Approaches Temporal-Difference Learning

This is likely quite noisy — better to take a small step (with parameter α):

$$v'(S_t) = v(S_t) + \alpha[R_{t+1} + \gamma\, v(S_{t+1}) - v(S_t)]$$

$R_{t+1} + \gamma\, v(S_{t+1})$ is called the **Temporal Difference Target**

$R_{t+1} + \gamma\, v(S_{t+1}) - v(S_t)$ is called the **Temporal Difference Error**

α, associated with the learning rate, can take several values

Model Free Approaches Temporal-Difference Learning

$R_{t+1} + \gamma\, v(S_{t+1})$ is called the **Temporal Difference Target**

Looking at the target, we notice that we are only taking into consideration the next immediate reward.

This is one specific form of Temporal Differences leaning, namely TD(0).

For example, in TD(3):

$R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \gamma^3\, v(S_{t+1})$ is the **Temporal Difference Target**

Note that TD(∞), would mean that the target $R_{t+1} + \gamma R_{t+2} + \cdots = \boldsymbol{G_t}$

Model Free Approaches Bootstrapping and Sampling

Bootstrapping: update involves an estimate:

- Monte Carlo does not bootstrap.
- Dynamic Programming bootstraps.
- Temporal Differences learning bootstraps.

Sampling: update samples an expectation:

- Monte Carlo samples.
- Dynamic Programming does not sample.
- Temporal Differences learning samples.

Model Free Approaches — Temporal difference learning

We can apply the same idea to the Q function (action values)

Temporal-difference learning for action values:

- Update value $\boldsymbol{q(S_t, A_t)}$
- Towards estimated return $\boldsymbol{R_{t+1} + \gamma . q(S_{t+1}, A_{t+1})}$

$$q'(S_t, A_t) \leftarrow q(S_t, A_t) + \alpha[R_{t+1} + \gamma . q(S_{t+1}, A_{t+1}) - q(S_t, A_t)]$$

This algorithm is known as **SARSA.**

NOVA IMS Information Management School

Model Free Approaches Temporal difference learning

Temporal Difference is model-free (no knowledge of MDP) and learn directly from experience.

Temporal Difference can learn from incomplete episodes, by bootstrapping.

Temporal Difference can learn during each episode.

Comparing Monte Carlo and Temporal Differences

Temporal Difference can learn before knowing the final outcome:

- Temporal Difference can learn online after every step.
- Monte Carlo must wait until end of episode before return is known.

Comparing Monte Carlo and Temporal Differences

Temporal Difference can learn without the final outcome:

- Temporal Difference can learn from incomplete sequences.
- Monte Carlo can only learn from complete sequences.
- Temporal Difference works in continuing (non-terminating) environments.
- Monte Carlo only works for episodic (terminating) environments.

Comparing Monte Carlo and Temporal Differences

Temporal Difference is independent of the temporal span of the prediction:

- Temporal Difference can learn from single transitions.
- Monte Carlo must store all predictions (or states) to update at the end of an episode.

Temporal Difference needs reasonable value estimates.

Model Free Approaches — Bias/Variance Trade-Off

Monte Carlo return $G_t = R_{t+1} + \gamma . R_{t+2} + \ldots$ is an unbiased estimate of $v_\pi(S_t)$.

Temporal Difference target $R_{t+1} + \gamma\, v(S_{t+1})$ is a biased estimate of $v_\pi(S_t)$.

unless $v(S_{t+1}) = v_\pi(S_{t+1})$ where π is an optimal policy.

Model Free Approaches Bias/Variance Trade-Off

But the Temporal Difference target has lower variance:

- Return depends on many random actions, transitions, rewards.
- Temporal Difference target depends on one random action, transition, reward.

Model Free Approaches Bias/Variance Trade-Off

In some cases, Temporal Differences can have an irreducible bias.

The world may be partially observable:

- Monte Carlo would implicitly account for all the latent variables.

The function to approximate the values may fit poorly.

In the tabular case, both Monte Carlo and Temporal Differences will converge.

Advantages and Disadvantages of MC vs. TD

Temporal Differences **exploits Markov** property:

- Can help in **fully-observable** environments.

Monte Carlo **does not exploit Markov** property:

- Can help in **partially-observable** environments.

With finite data, or with function approximation, **the solutions may differ.**

Advantages and Disadvantages of MC vs. TD

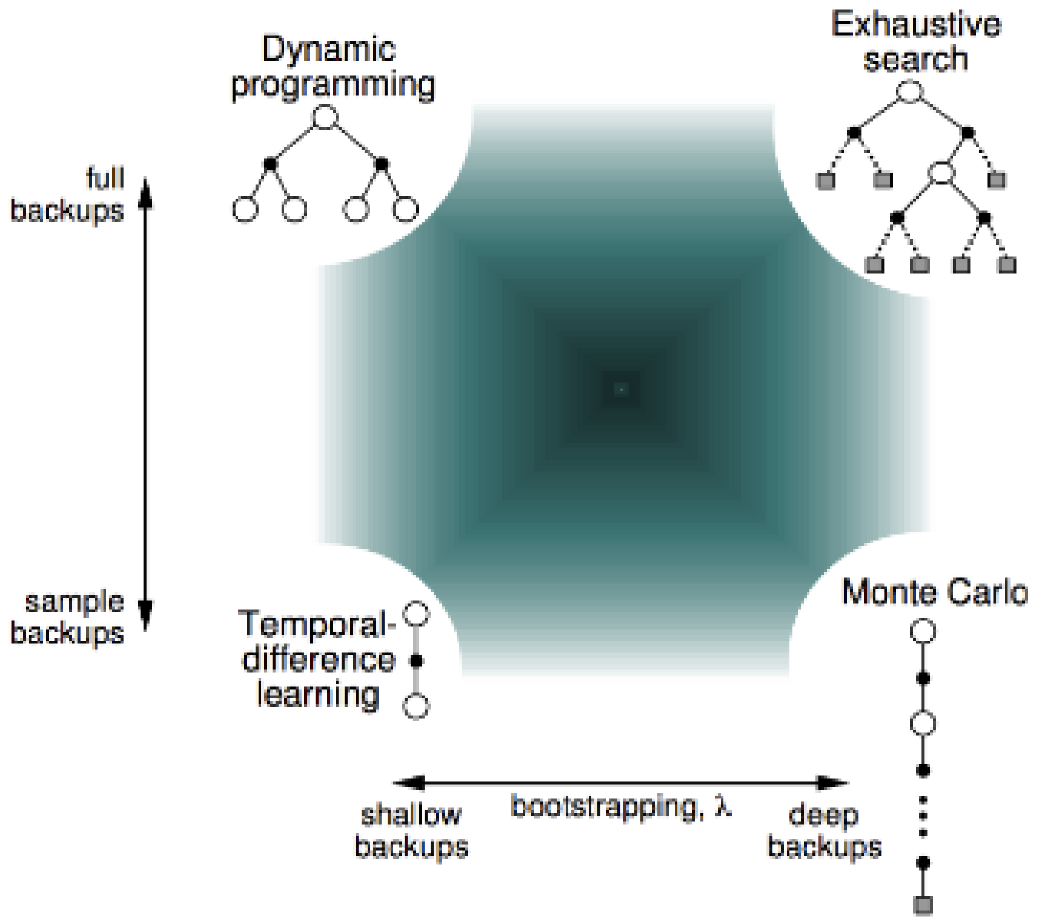
Monte Carlo has **high variance**, **zero bias**:

- Good convergence properties.
- Not very sensitive to initial value.
- Very simple to understand and use.

Temporal Differences **has low variance**, **some bias**:

- Usually more efficient than Monte Carlo.
- Temporal Differences usually converges to $v_\pi(s)$.
- More sensitive to initial value.

Unified View of Reinforcement Learning



Model Free Approaches

Multi-Step Temporal Differences

Temporal Differences uses value estimates which might be inaccurate:

- In addition, information can propagate back quite slowly.
- In Monte Carlo information propagates faster, but the updates are noisier.
- We can go in between Temporal Differences and Monte Carlo.

Model Free Approaches

Multi-Step Temporal Differences

$R_{t+1} + \gamma\, v(S_{t+1})$ is called the **Temporal Difference Target**

Looking at the target we notice that we are only taking into consideration the next immediate reward.

This is one particular form of Temporal Differences leaning, namely TD(0).

For example in TD(3):

$R_{t+1} + \gamma^1 R_{t+2} + \gamma^2 R_{t+3} + \gamma^3\, v(S_{t+3})$ is the **Temporal Difference Target**

Note that TD(∞), would mean that the target $R_{t+1} + \gamma R_{t+2} + \cdots = \boldsymbol{G_t}$

Multi-Step Temporal Differences

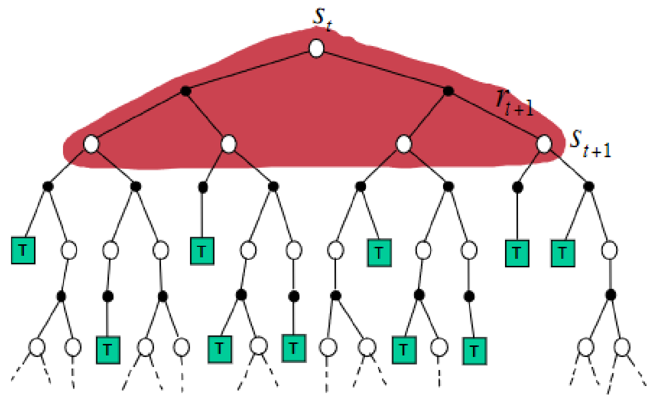
Multi-step returns have benefits from both Temporal Differences and Monte Carlo:

- Bootstrapping can have issues with bias.
- Monte Carlo can have issues with variance.

Model Free Approaches — Prediction Overview

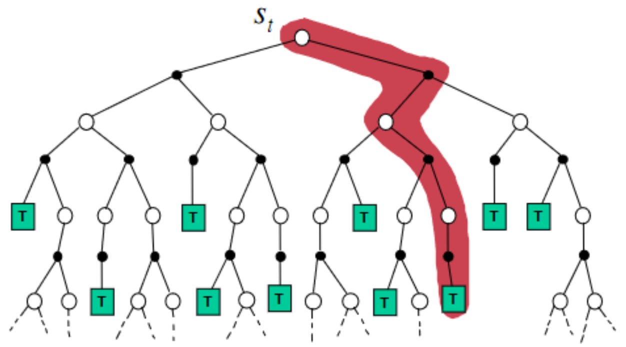
Visual Interpretation

$$v(S_t) \leftarrow \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \mid A_t \sim \pi(S_t)\right]$$



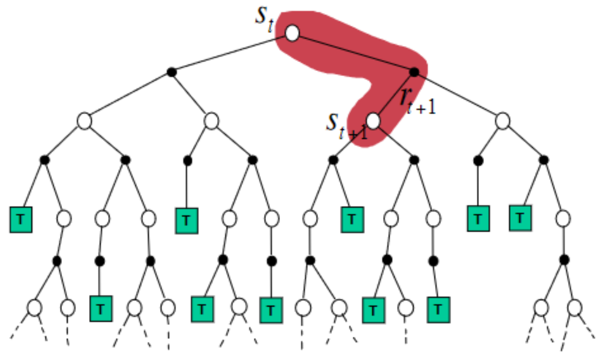
Dynamic Programming

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



Monte Carlo

$$v(S_t) \leftarrow v(S_t) + \alpha\left(R_{t+1} + \gamma v(S_{t+1}) - v(S_t)\right)$$



Temporal Differences TD(0)

*** Images from Open AI with UCL course**

Model Free Control

Model Free Control

Let’s discuss **control** methods!

Model Free Control Exploration vs Exploitation

Remember two of the fundamental problems in reinforcement learning

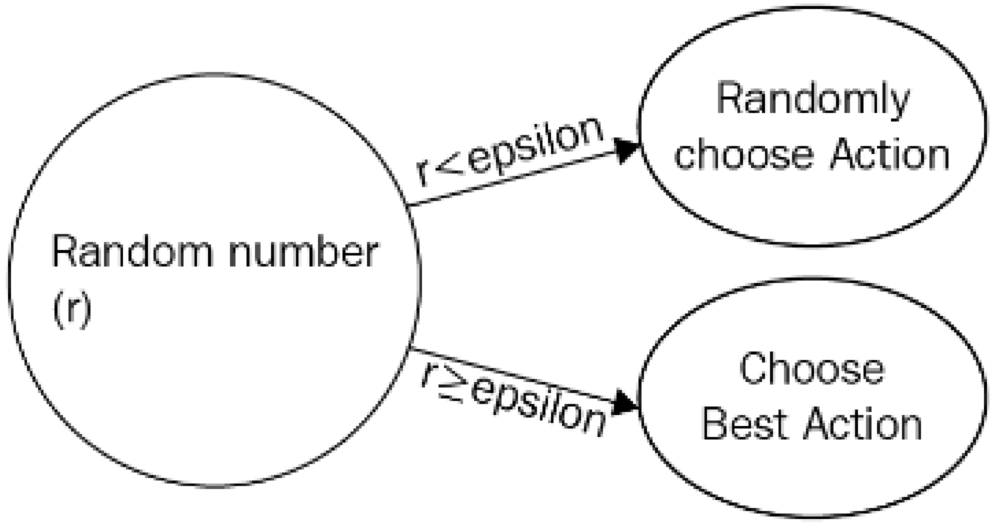
- **Exploration** finds more information about the environment.
- **Exploitation** exploits known information to maximise reward.
- It is usually important to explore as well as exploit.

Model Free Control Greedy Exploration

Simplest idea for ensuring continual exploration.

All actions are tried with non-zero probability.

With a given probability choose the greedy action.

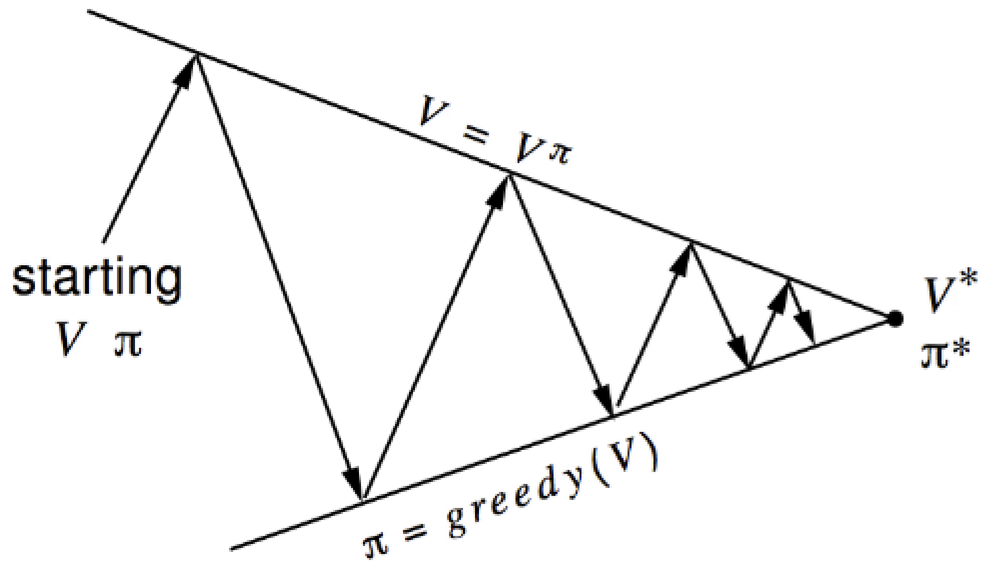


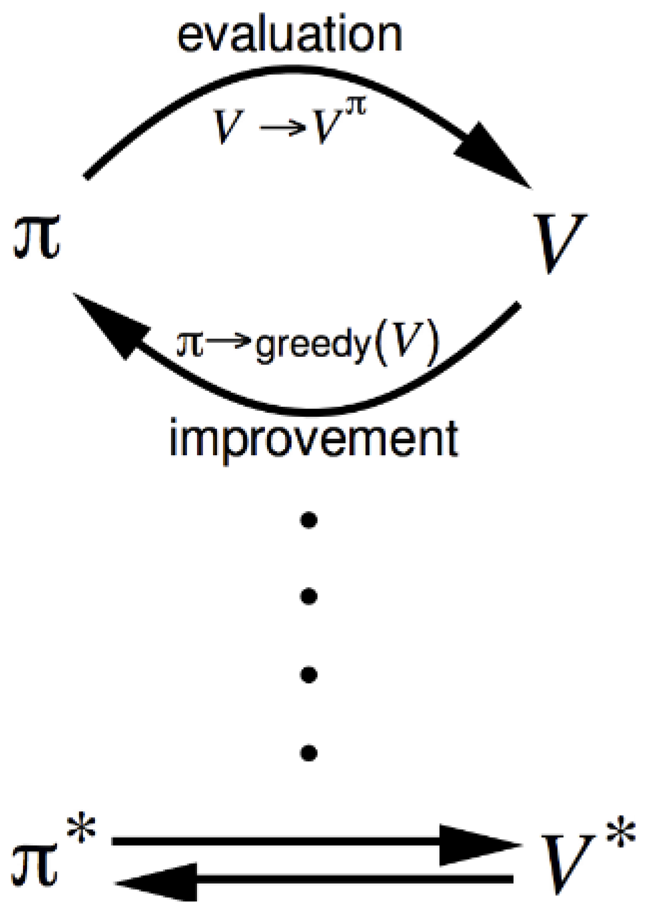
Model Free Control

Model Free Policy Iteration

Policy Iteration is seen as a **greedy** approach.

The idea is to sequentially improve a value function (**policy evaluation**) and then a policy (**policy improvement**).





Model Free Control Model Free Policy Iteration

Greedy policy improvement over $v(s)$ requires model of MDP:

$$\pi'(s) = argmax_a E\,[\boldsymbol{R_{t+1}} + \boldsymbol{\gamma}.\boldsymbol{v}(\boldsymbol{S_{t+1}}) \mid S_t = s, A_t = a]$$

Greedy policy improvement over $q(s,a)$ is **model free**:

$$\pi'(s) = argmax_a\, q(s,a)$$

This makes action values convenient.

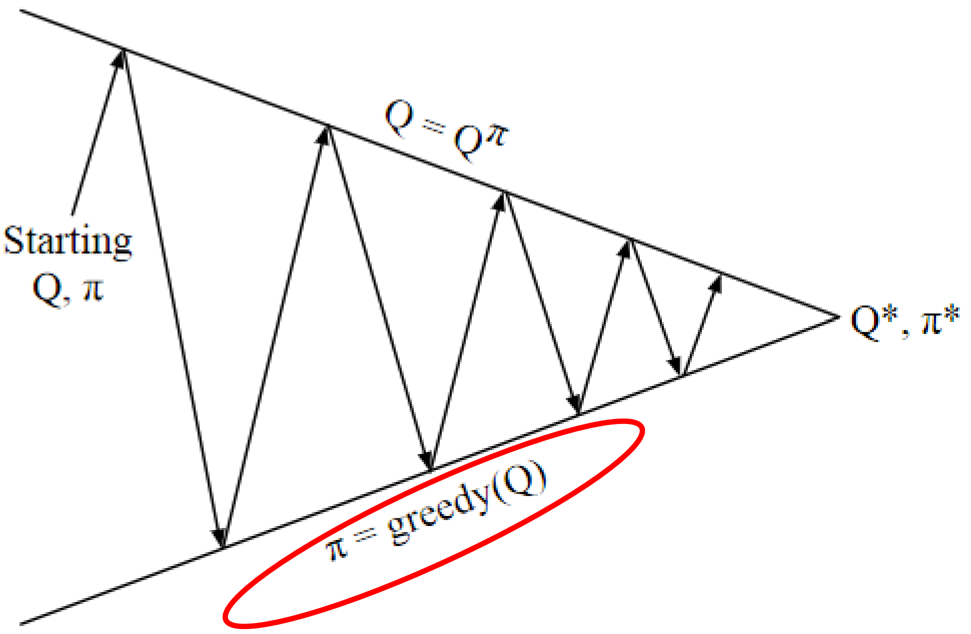
Model Free Control

Generalised Policy Iteration with Action-Value Function

Monte-Carlo policy evaluation, $q \approx q_\pi$.

Greedy policy improvement? No exploration!

(Can’t sample all states and actions when learning by interacting)



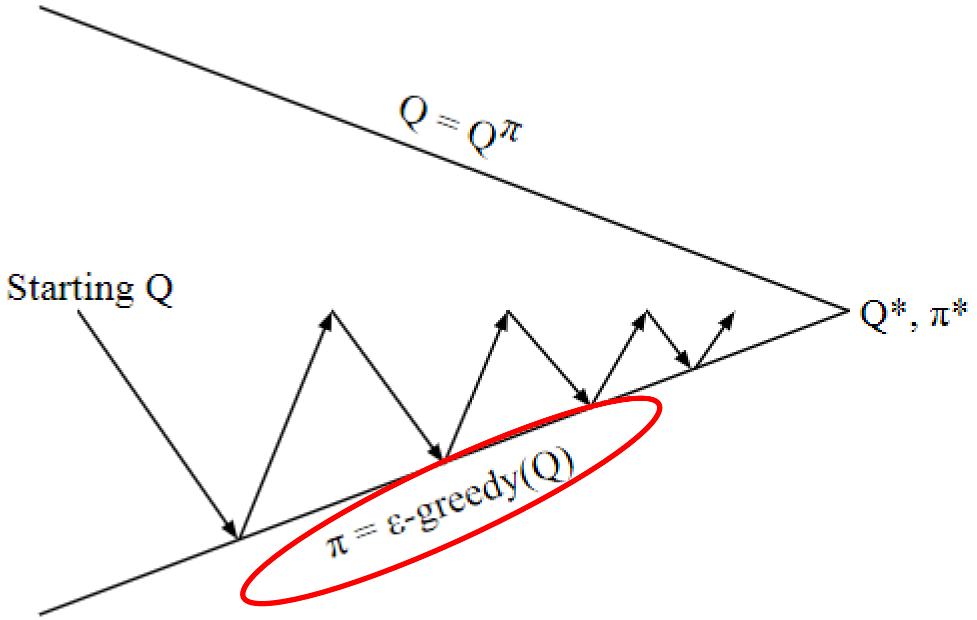
Model Free Control

Monte-Carlo Generalized Policy Iteration

Every episode:

Policy evaluation Monte-Carlo policy evaluation, $q \approx q_\pi$.

Policy improvement greedy policy improvement.



Model Free Control

GLIE

Definition

Greedy in the Limit with Infinite Exploration (GLIE)

- All state-action pairs are explored infinitely many times:

$$\forall s, a \qquad limit\; t \rightarrow \infty \qquad N\,(s, a) = \infty$$

- The policy converges to a greedy policy:

$$limit\;\; t \rightarrow \infty \qquad \pi_t\,(a|s) = I[a = argmax_{a'}\, q(s, a')]$$

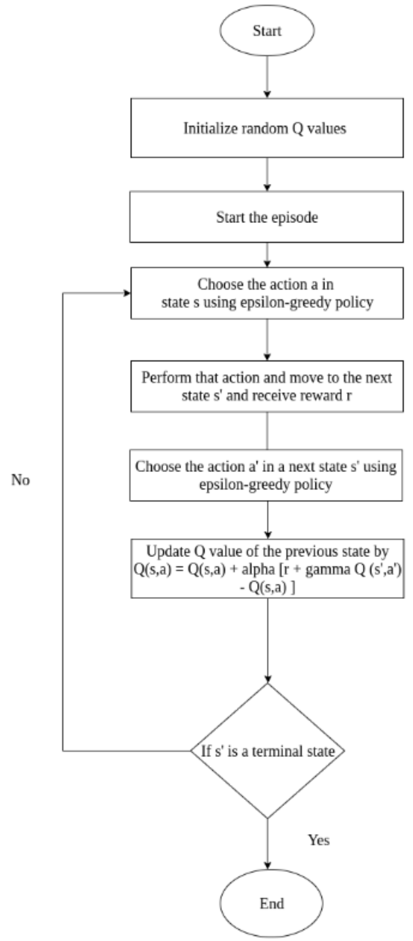
Model Free Control

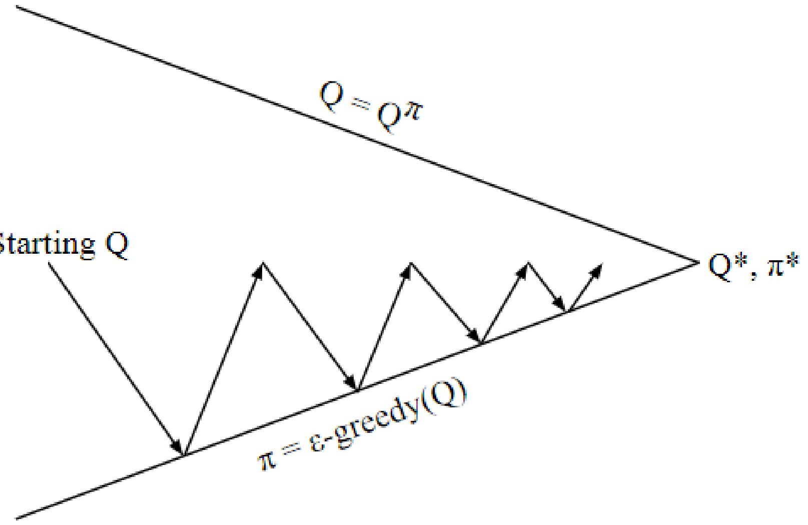
Theorem

GLIE Model-free control **converges to the optimal action-value function**, $q \rightarrow q_*$

Model Free Control

Updating Action-Value Functions with SARSA


Start
Initialize random Q values
Start the episode
Choose the action a in state s using epsilon-greedy policy
Perform that action and move to the next state s' and receive reward r
Choose the action a' in a next state s' using epsilon-greedy policy
Update Q value of the previous state by Q(s,a) = Q(s,a) + alpha [r + gamma Q (s',a') - Q(s,a)]
If s' is a terminal state
No
Yes
End



<u>Every time-step:</u>

Policy evaluation SARSA, $q \approx q_\pi$

Greedy policy improvement

$$q'(S_t, A_t) \leftarrow q(S_t, A_t) + \alpha[R_{t+1} + \gamma . q(S_{t+1}, A_{t+1}) - q(S_t, A_t)]$$

NOVA IMS Information Management School

Model Free Control

GLIE

Theorem

Tabular SARSA **converges to the optimal action-value function**, $q \to q_*$, if the policy is GLIE.

Model Free Control Several Algorithms Seen

Dynamic programming

- Policy Iteration
- Value Iteration

Monte Carlo

Temporal Differences

- Temporal Differences (SARSA)
- Q-Learning

Next Class

On and Off-Policy Learning

On-policy learning

Learn about behaviour policy π from experience sampled from π.

Off-policy learning

Learn about target policy π from experience sampled from μ.

Evaluate target policy $\pi(a|s)$ to compute $v_\pi(s)$ or $q_\pi(s,a)$.

While using behaviour policy $\mu(a|s)$ to generate actions.

Next Class

On and Off-Policy Learning

On-policy learning

- “Learn on the job”.

Off-policy learning

- “Look over someone’s shoulder”.

Off-policy TD and Q-learning.

Thank You!

Morada: Campus de Campolide, 1070-312 Lisboa, Portugal

Tel: +351 213 828 610 | Fax: +351 213 828 611