Reinforcement Learning KU (DAT.C307UF), WS24
Assignment 2
Iterative Policy Evaluation, Planning in a Grid World

Thomas Wedenig
thomas.wedenig@tugraz.at

Teaching Assistant:	Erwin Pfeiler
Points to achieve:	25 pts
Deadline:	20.12.2024 23:59
Hand-in procedure:	You can work in groups of **at most two people**. **Exactly one** team member uploads three files to TeachCenter: **The report (.pdf)**, `policy_iteration.py`, and `frozen_lake_utils.py`. The first page of the report must be the **cover letter**. Do not upload a folder. Do not zip the files.
Plagiarism:	If detected, 0 points for all parties involved. If this happens twice, we will grade the group with "Ungültig aufgrund von Täuschung"

General Remarks

Your submission will be graded based on:

- Clarity and correctness (Is your code doing what it should be doing?)
- Include intermediary steps, and textually explain your reasoning process.
- Quality of your plots (Is everything clearly legible/visible? Are axes labeled? …)
- If some results or plots are not described in the report, they will **not** count towards your points.
- Code in comments will not be executed or graded. Make sure that your code runs.
- For all pen and paper exercies, clearly report all intermediate steps – only presenting the final solution is insufficient.

As detailed in `P5_Intro_to_Gymansium_and_PyTorch.ipynb`, create a `conda` environment with Python 3.11.5 and install all dependencies using `pip install -r requirements.txt`. Do not import any other modules in the python files you are given. In the class `PolicyIteration`, only add your implementation in the corresponding sections marked with `TODO`. Do not change the function signatures. Failure to adhere to these rules may result in point deductions.

The file `frozen_lake_utils.py` provides plotting utilities that you can use. You do not need to edit this file (you can however, as this file is also included in your submission).

Math Recap

The following concepts have also been discussed in the lecture.

Definition 1 *Let V be a vector space. Then $f : V \mapsto \mathbb{R}_0^+$ is a norm on V provided the following hold:*

1. *$f(\mathbf{v}) = 0$ if and only if $\mathbf{v} = \mathbf{0}$*
2. *For any $\lambda \in \mathbb{R}, \mathbf{v} \in V, f(\lambda\mathbf{v}) = |\lambda| f(\mathbf{v})$*
3. *For any $\mathbf{v}, \mathbf{u} \in V, f(\mathbf{u} + \mathbf{v}) \leq f(\mathbf{v}) + f(\mathbf{u})$*

A vector space together with a norm is called a normed vector space.

According to Definition 1, a norm is a function that assigns a nonnegative number to each vector, which can be interpreted as some notion of "length". The norm of a vector $\mathbf{v}$ is often denoted by $||\mathbf{v}||$. In the n-dimensional Euclidean vector space $V = \{\mathbf{v} \,|\, \mathbf{v} = (x_1, \ldots, x_n)^\top,\ x_1, \ldots, x_n \in \mathbb{R}\}$, a common choice of norm is the max norm $||\mathbf{v}||_\infty = \max_i |v_i|$.

A norm $||\cdot||$ gives rise to a *distance measure* between two vectors $\mathbf{v}$ and $\mathbf{u}$ by taking the norm of their difference, i.e. $||\mathbf{v} - \mathbf{u}||$.

Definition 2 *Let $(\mathbf{v}^{(n)}; n \geq 0)$ be a sequence of vectors of a normed vector space $V = (V, ||\cdot||)$. Then $\mathbf{v}^{(n)}$ is called a Cauchy-sequence if $\lim_{n\mapsto\infty} \sup_{m\geq n} ||\mathbf{v}^{(n)} - \mathbf{v}^{(m)}|| = 0$, i.e., the elements of the sequence become arbitrarily close as the sequence continues.*

Definition 3 *A normed vector space V is called* complete *if every Cauchy sequence in V converges to some element in V.*

As an example, every Cauchy sequence in the real numbers $\mathbb{R}$ converges to some real number—hence, the real numbers form a complete vector space. On the other hand, we can construct Cauchy sequences in the rational numbers $\mathbb{Q}$ which converge to some irrational number (e.g. to the number $\pi = 3.141592\ldots$)—hence, the rational numbers are *not* a complete vector space (they are still a normed vector space though).

Definition 4 *A complete, normed vector space is called a* Banach space.

Definition 5 *Let $V = (V, ||\cdot||)$ be a normed vector space. A mapping $T : V \mapsto V$ is called L-*Lipschitz *if for any $\mathbf{u}, \mathbf{v} \in V$,*

$$||T(\mathbf{u}) - T(\mathbf{v})|| \leq L||\mathbf{u} - \mathbf{v}||. \tag{1}$$

T is called a contraction *if it is L-Lipschitz with $L < 1$. In this case, L is called the contraction factor of T, and T is called an L-contraction.*

Definition 6 *Let $T : V \mapsto V$ be some mapping defined on some vector space V. Any vector $\mathbf{v} \in V$ for which $T(\mathbf{v}) = \mathbf{v}$ is called a* fixed point *of T.*

The **Banach fixed-point theorem** says that a contraction T in a Banach space always has a *unique fixed point*, and iterating T will always converge to it:

Theorem 1 *Let V be a Banach space and $T : V \mapsto V$ be a contraction mapping. Then T has a unique fixed point $\mathbf{v}^*$. Furthermore, for any $\mathbf{v}^{(0)} \in V$, let $(\mathbf{v}^{(n)}; n \geq 0)$ be the sequence of vectors defined via $\mathbf{v}^{(n+1)} = T(\mathbf{v}^{(n)})$. For any $\mathbf{v}^{(0)}$, this sequence converges to $\mathbf{v}^*$.*

1 Iterative Policy Evaluation [10 points]

In the lecture, we learned about computing the value function v_π by solving the Bellman equation via closed-form matrix inversion. However, this approach does not scale well to large-scale MDPs. To this end, we considered an iterative approach based on the Bellman equation, i.e., interpreting the Bellman equation as an update rule:

$$V_{new}(s) \leftarrow r(s) + \gamma \sum_{s'} p(s'|s)\, V_{old}(s'), \tag{2}$$

where γ is the discounting factor, $r(s)$ is the expected reward function and $p(s'|s)$ is the state transition. Iterative Policy Evaluation is detailed in Algorithm 1.

Algorithm 1: Iterative Policy Evaluation, for estimating $V \approx v_\pi$

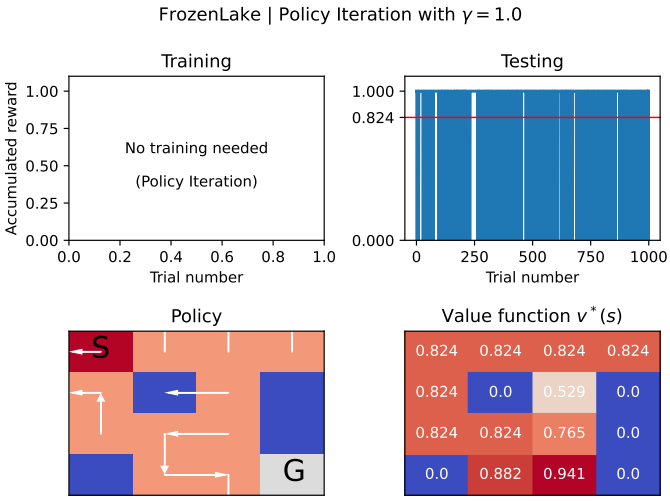
Input: π, the policy to be evaluated
Data: a small threshold $\varepsilon > 0$ determining accuracy of estimation
Output: $V \approx v_\pi$
initialize $V(s)$ arbitrarily for all $s \in \mathcal{S}$, and $V(\text{terminal})$ to 0;
repeat
  $\Delta \leftarrow 0$;
  $V_{old}(s) \leftarrow V(s)$ for all $s \in \mathcal{S}$;
  foreach $s \in \mathcal{S}$ **do**
    $V(s) \leftarrow r(s) + \gamma \sum_{s'} p(s'|s)\, V_{old}(s')$;
    $\Delta \leftarrow \max(\Delta, |V_{old}(s) - V(s)|)$;
  end
until $\Delta < \varepsilon$;

Task: Prove that for $0 \leq \gamma < 1$, Iterative Policy Evaluation (Algorithm 1) always converges to v_π, for any MDP, any policy π and any initialization of $V(s)$. Clearly show all steps of the proof. Hint: Use Banach's Fixed Point Theorem.

2 Planning in a Grid World [15 Points]

A grid world is a typical environment with finite action and state spaces. We will solve the FrozenLake[1] environment from the `gymnasium` package (a fork of OpenAI's Gym package). The agent controls the movement of a character in a grid world. Some tiles of the grid are walkable, and others lead to the agent falling into the water. Additionally, the movement direction of the agent is uncertain and only partially depends on the chosen direction (the lake is slippery). The agent receives a reward of 1 if it moves the goal state and 0 else. The episode ends as soon as the agent falls into the water or reaches the goal state. Further information can be found in the Gymnasium Documentation.

(a) FrozenLake environment. The starting state is on the top left while the goal state is on the bottom right.



(b) Value function and policy that were acquired using *Policy Iteration* ($\gamma = 1$). Testing the policy using 1000 episodes yields an average sum of rewards of 0.824.

a) Assume we know the dynamics of the underlying MDP (i.e., the state transition probabilities and the expected rewards). Fill in the TODOs in `policy_iteration.py`: You are supposed to implement the *Policy Iteration* algorithm, which consists of repeatedly *evaluating a policy* (using *Iterative Policy Evaluation*) and improving this policy by acting greedily w.r.t. the Q function of the old policy. When finished with the TODOs, execute the file: This will run policy iteration until convergence for both the undiscounted case ($\gamma = 1$) and a discounted case ($\gamma = 0.95$). In both scenarios, the code framework will produce a plot which includes (1) the policy you have found, (2) the corresponding value function, and (3) the reward that was accumulated in 1000 simulated episodes (and its average). Include both plots ($\gamma = 1$ and $\gamma = 0.95$) in your report. Explain any differences between the two policies, the two value functions, and between the average test success rate (i.e., the fraction of simulated episodes in which the agent reached the goal state). Briefly discuss why these differences appear.

b) Assume that s_{start} is the starting state. In the undiscounted case ($\gamma = 1$), what's the relationship between $v^*(s_{\text{start}})$ and the average reward you encounter when acting according to π^* for N episodes (i.e., `np.mean(policy_iteration.test_policy(num_episodes=N))`)? What happens as $N \to \infty$? Write down the definition of the v function in this particular case ($\gamma = 1$, binary reward per episode).

c) In general: Is the optimal policy π^* unique? Is the optimal value function v^* unique? Explain your reasoning.

[1] https://gymnasium.farama.org/environments/toy_text/frozen_lake/