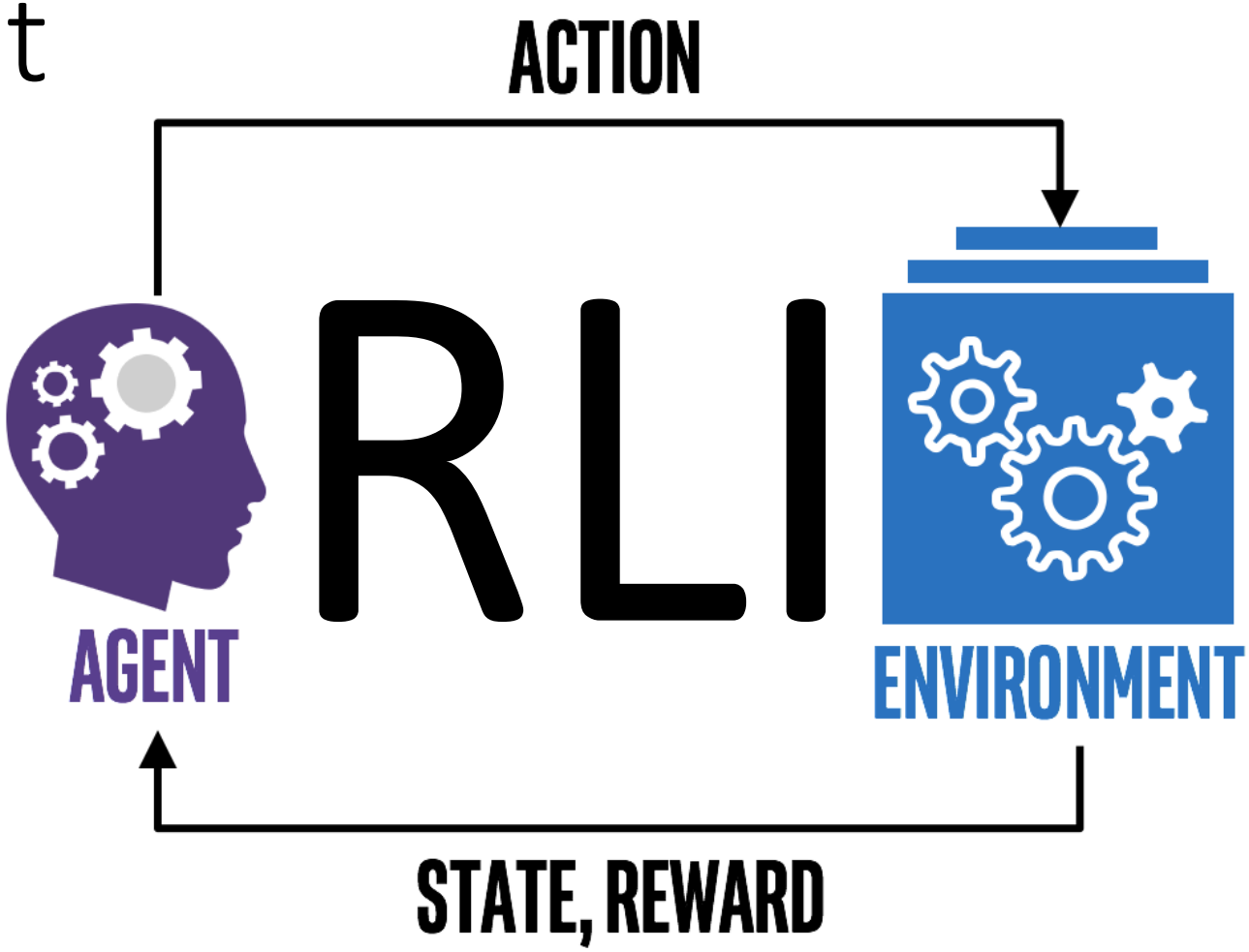
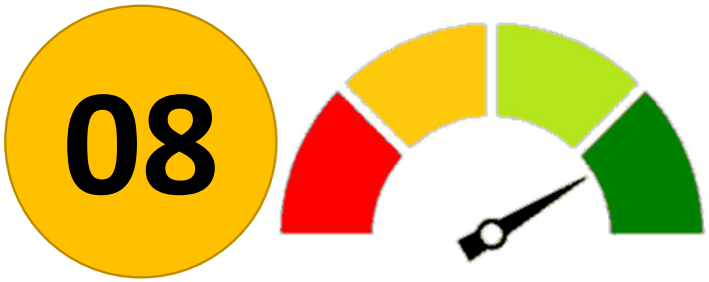
Reinforcement Learning Introduction





José Manuel Rey, IE

Reinforcement Learning Introduction Assignment 08.00

Date: Ene/29 2026

Due Date: Feb/25 2026 [SESSION 15]

Ongoing grading (maximum)

50 Individual Assignment [mini-groups]

Assignment Description Overview

This assignment consists of 3 parts/questions:

Assignment 08.01 → "Jack's Car Rental Problem – Version 1" [35%]

- jack_car_problem_v0.ipynb

Assignment 08.02 → "Blackjack Off-Policy" [35%][7 questions/5% per question]

- Off-Policy MC Control with Weighted Importance Sampling.ipynb

Assignment 08.03 → "Pass the Pigs (DP)" [30%]

- Pig Dice Game (numba) RL Dynamic Programming (Assignment).ipynb

Assignment Description Overview

For each part/question you should elaborate a self-contained ".ipynb" file (derived from the documents delivered with the assignment), providing (when required) a clear concise explanation of the conceptual approach, the diagrams or models representing the problem, comments for the code used and the explanation of the solution

(*) Note: the files blackjack_gym.py, plotting.py, are included in the assignment files as they are used as external libraries by the jupyter notebooks [no changes should be made in any of them]

SUBMIT YOUR WORK BY ZIPPING TOGETHER THE 3 NOTEBOOKS (COMPLETED WITH YOUR ANSWERS) IN A SINGLE SUBMISSION FILE NAMED:

"RLI_08_00 – {mini-group number}.zip" (see the attached .txt for the mini-groups)

Administrative and additional notes

- Find attached (in the zip file) the configuration of the mini-groups for the assignment:
  - →RLI_BCSAI_C3{A or B}_2026 - ASSIGNMENT 0 RLI_08_00 TEAMS.txt
  - →Note: you will find two versions of this file, either (A) or (B). You will be listed in the one corresponding to your IE class

  Only submit ONE zip file per mini-group. Choose ONE member of each group for that submission. In case you send more than one, ensure that all the submissions by the different members are identical or **very explicitly** indicate which version is the definitive one, as if not the system will choose just ONLY one of them (randomly) for the review and evaluation of the whole mini-group.
- As usual, the code should run without warnings or errors. All additional required files should be included in the ZIP file, and if you were using any "novel" additional external library for your work, clearly indicate a brief description of its purpose, the version used and the "**`!pip install <package>`**" required for its installation

Assignment 08.01 Description Overview Jack's Car Rental Problem – version 1

You must adapt the solutions obtained for the original "Jack's Car Rental Problem" [Example 4.2: Jack's Car Rental in Reinforcement Learning - 2nd ed Richard Sutton and Andrew Barto] (we will call them version 0) to the new conditions proposed in Sutton & Barto "Exercise 4.7 (programming)" (we will call them version 1)

Use the jupyter notebook created for solving the first problem (*jack_car_problem_v0.ipynb*) and adapt it conveniently for the second problem (name you new *document as jack_car_problem_v1.ipynb*)

The description of each problem is repeated below in the following slides

Jack's Car Rental (Version 0)

Here are key specific points about the problem:

- 2 locations A, B
- Each location can only hold 20 cars.
- Every time a car is rented, we earn $10 (Reward)
- Every time we move a car overnight to another location, it costs us $2 (Negative Reward).
- The maximum number of cars we can move overnight is 5 (Action).
- The number of cars requested and returned at each location (n) on any given day are Poisson random variables.
- The expected number (lambda) of **rental requests at the first and second location is 3 and 4 respectively.**
- The expected number of **rental returns at the first and second location is 3 and 2 respectively.**
- Our discount rate for future returns, (γ), is 0.9.
- The time step are days (thus, one step in an iteration can be considered a full day), the state is the number of cars at each location at the end of the day, and the actions are the net number of cars moved between the two locations overnight.

Jack's Car Rental (Version 1)

Let's see what happens if we add some non-linearities [*as they are costs or conditions not directly proportional to the number of cars moved, as they were the original ones stated for the problem in version 0*] like the following to the problem above:

- One of Jack's employees at the first location rides a bus home each night and lives near the second location. He is happy to shuttle one car to the second location for free.
- Each additional car still costs $2, as do all cars moved in the other direction.
- In addition, Jack has limited parking space at each location. If more than 10 cars are kept overnight at a location (after any moving of cars), then an additional cost of $4 must be incurred to use a second parking lot (independent of how many cars are kept there) → We will have one more reward of -$4 for the second parking lot if needed

Assignment 08.02 Description Overview Blackjack Off-Policy

Complete the code appropriately in the attached notebook (*Off-Policy MC Control with Weighted Importance Sampling.ipynb*) in order to create a working version of an implementation on an Off-Policy Monte Carlo Control method with Weighted Importance Sampling

ASSIGNMENT (Technical Note) blackjack_gym.py

- This exercise also introduces the use of OpenAI compliant environments (OpenAI Gymnasium) [blackjack_gym.py]
  - **Note:** No changes are required in this code for the purpose of this assignment
- For compatibility issues we will be using **OpenAI Gymnasium** version >0.26.0. The current version (as of Jan/2026) is 1.2.3.
- The suggested (minimalistic) installation is: `pip install gymnasium[box2d]` or just `pip install gymnasium`
- In the case that you were already using some other older versions of OpenAI Gym (<=0.21.00), and want to keep it active (for compatibility with some older code you want to run) create and use a virtual python environment to avoid conflicts

Assignment 08.03 Description Overview Pass the Pigs

Complete the code appropriately in the attached notebook (*Pig Dice Game (numba) RL Dynamic Programming (Assignment).ipynb*) in order to create a working version of an implementation on a RL Dynamic Programming (either Policy-Iteration-Improvement or Value-Iteration) for finding a optimal strategy for “Pass the Pigs” (or “Pig Dice Game”)

Objective

- Find and optimal policy for playing the game (in either of the possible variants: GAME_PIG_DICE or GAME_PASS_THE_PIGS) using a RL Dynamic Programming algorithm
- You can choose either a Policy Iteration & Improvement algorithm or a Value Iteration algorithm
- Review the general documentation included in the assignment description for obtaining a general intuition of the mechanics of the game

Notes

- For guiding and helping you in understanding the types of representations (states, actions, policies) used and the system dynamics of the environment you are provided with some code for examples of policies [Policy helpers (opponent's policies)] and for the transitioning rules of the game [Basic simplified environment → As numba functions]
- You will not need such functions but, however, you might just need to include (copy) pieces of this code into your final algorithm for implementing the sweeps of state/action/state transitions in your algorithm
- Additionally, you are provided with a basic example for the visualization of the resulting policies [Utility and plotting functions]. You may use those or either implement other specific tools/plots that you consider more convenient for the visualization, interpretation or explanation of the policies obtained
- The code is adapted for numba for faster execution, but in this case you can implement your algorithms practically with no change from standard python syntax. In the case, that you find difficulties or errors that make the debugging process more complicated, just comment out the @njit decorators

Instructions

- Review the notebook and fill in the missing code appropriately.
- Check the final executability and results of the resulting code.

Appendix

brief description of the game
“Pass the Pigs”

Pass the Pigs (Dice Game)
Is it worth being greedy?

- Pass the Pigs is a commercial version (*) of the dice game Pig, but using custom asymmetrical throwing dice, similar to shagai.
- Each turn involves one player throwing two model pigs, each of which has a dot on one side. The player gains or loses points, or is eliminated from the game based on the way the pigs land.
- Each turn lasts until the player throwing either rolls the pigs in a way that wipes out their current turn score, wipes out their total game score, or decides to stop their turn, add their turn score to their total score and pass the pigs to the next player.
- The winner is the first player to reach a predetermined total score, usually 100 points.

(*) It was created by David Moffatt and published by Recycled Paper Products as Pig Mania! in 1977. The publishing license was later sold to Milton Bradley and the game renamed Pass the Pigs. In 1992, publishing rights for North America were sold to Winning Moves Games USA, which acquired the game outright from David Moffat Enterprises in early 2017.

Dice Game Pig

The simple dice version

Players take turns rolling the dice, accumulating points with each **roll** or passing the turn to their opponent either when the dice roll is a one (in which case they also lose the points accumulated in the current turn) or when they voluntarily decide to **pass** (in which case they add the points accumulated in the turn to their score)

PIG OUT
Lose points won in the turn
Swap player

ROLL
Add points to the turn

PASS
Add points of the turn to the total score and Swap player

Rules of Game

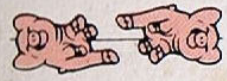
NUMBER OF PLAYERS: Two or more.

PLAYING TIME: Average time for 4 players is 20 minutes.

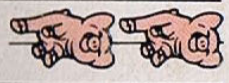
HOW PLAY BEGINS: Each player tosses the pigs once—player with the highest roll goes first. In the event of a tie those players toss again until a winner is decided. A scorekeeper or "Grunt" is appointed. Play begins with players taking turns as pigs are passed in a counter clockwise direction.

SCORING POINTS: The pigs are tossed by placing them both into the pigsty and tossing them simultaneously onto a smooth surface. A caked surface of mud is best but a table top can be used. Points are scored as follows:

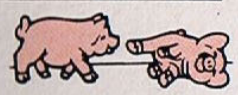
Pig Out: Pigs land on opposite sides, one pig on left side, the other on right side. 0 points — player must pass pigs to next player.



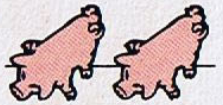
Siders: Pigs land on their same sides. 1 point and player tosses again.



Hoofer: One pig lands standing on all four hoofs. 5 points and player tosses again.



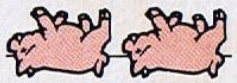
Double Leaning Jowler: Both pigs land in the leaning jowler position. 60 points and player tosses again.



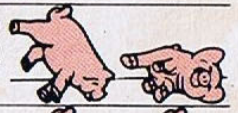
Double Hoofer: Both pigs land in the hoofer position. 20 points and player tosses again.

Razorback: One pig lands on its back. 5 points and player tosses again.

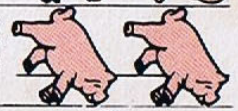
Double Razorback: Both pigs land in the razorback position. 20 points and player tosses again.



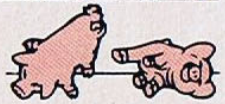
Snouter: One pig lands on its snout and two forehoofs. 10 points and player tosses again.



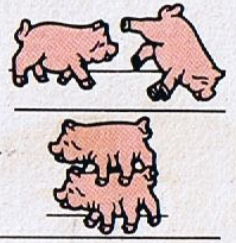
Double Snouter: Both pigs land in the snouter position. 40 points and player tosses again.



Leaning Jowler: One pig lands on its left jowl supported by left ear and left forehoof. 15 points and player tosses again.



Mixed Combo: Any combination of Hoofer, Razorback, Snouter, or Leaning Jowler, add total score: i.e., Hoofer (5 pts.) and Snouter (10 pts.) = 15 points and player tosses again.



Piggy Back: Since this is an unnatural position for pigs, player forfeits game.

Makin' Bacon: Both pigs touching in any position. Player loses all points earned for that turn of play, and must pass the pigs to the next player.

HOG CALLING—"SOOEE": Any player other than the pig tosser may hog call. He does this by shrieking "Sooee" prior to the tossing of the pigs. The first player to scream "Sooee" gets to hog call—there can be only one hog caller per toss. The hog caller predicts what positions the pigs will land on the upcoming toss.

If correct, the hog caller earns double the number of points scored on toss. The pig tosser must subtract the same number of points accumulated from his score and pass the pigs to the next player (a player can never go below 0 points and must have 1 or more points to hog call).

If incorrect, hog caller must subtract from his score double the points scored on toss. Pig tosser earns double the points scored on toss and continues play. If "pig out" or "makin' bacon" are thrown, regular rules apply with no points earned or lost.

The player tossing the pigs retains possession of the pigs until he throws "pig out" or "makin' bacon," or is "hog called" correctly by another player.

HOW TO WIN: When one player reaches 100 or more points after completing his turn (i.e., a player continues to accumulate points over 100 as long as there is no "pig out," "makin' bacon," or successful "Sooee"), the final round begins. Each of the remaining players has one turn to surpass the score of the leader. If the score is bettered, another round takes place in which all the other players (including the old leader) have one more opportunity to surpass the new leader. If no one scores higher than the leader in one complete turn, the game is over and the winner has won!

OPTIONAL ADULT RULES: FOR A MORE VOLATILE FINISH All play is the same except that, in the final round, the player with over 100 points *MUST*—repeat, *MUST*—HOG CALL each opponent in a last round; if, after this last-round is complete, the HOG CALLER still has over 100 points, he wins. If, however, an unsuccessful "SOOEE" or two causes the leader's score to fall below 100, the game continues as normal with no further obligation to HOG CALL—until someone again goes over 100 and *MUST* HOG CALL each opponent in that final round. The winner, as above, is the HOG CALLER who survives the FINAL ROUND of mandatory HOG CALLING all the other players by remaining over 100 points.

Scoring

Generally, a position's score is inversely proportional to its likelihood of occurring.

→ If one, and only one, of the pigs is lying on its side, that pig scores nothing, and the other pig scores as follows:

- **Razorback** - The pig is lying on its back - 5 points
- **Trotter (Hoofer)** - The pig is standing on all four trotters - 5 points
- **Snouter** - The pig is resting on its snout and front trotters - 10 points
- **Leaning Jowler** - The pig is resting on its trotter, **snout and ear** - 15 points

→ If neither pig is lying on its side, the score is the **sum** of the individual positions.

- Mixed Combo - Sum of individual scores

→ If both pigs land in the same position, the **sum is doubled**; equivalently, the score for the individual position is quadrupled:

- Double Razorback - (5 + 5) × 2 = 20 points
- Double Trotter - (5 + 5) × 2 = 20 points
- Double Snouter - (10 + 10) × 2 = 40 points
- Double Leaning Jowler - (15 + 15) × 2 = 60 points

→ If both pigs land on their sides, the score is as follows:

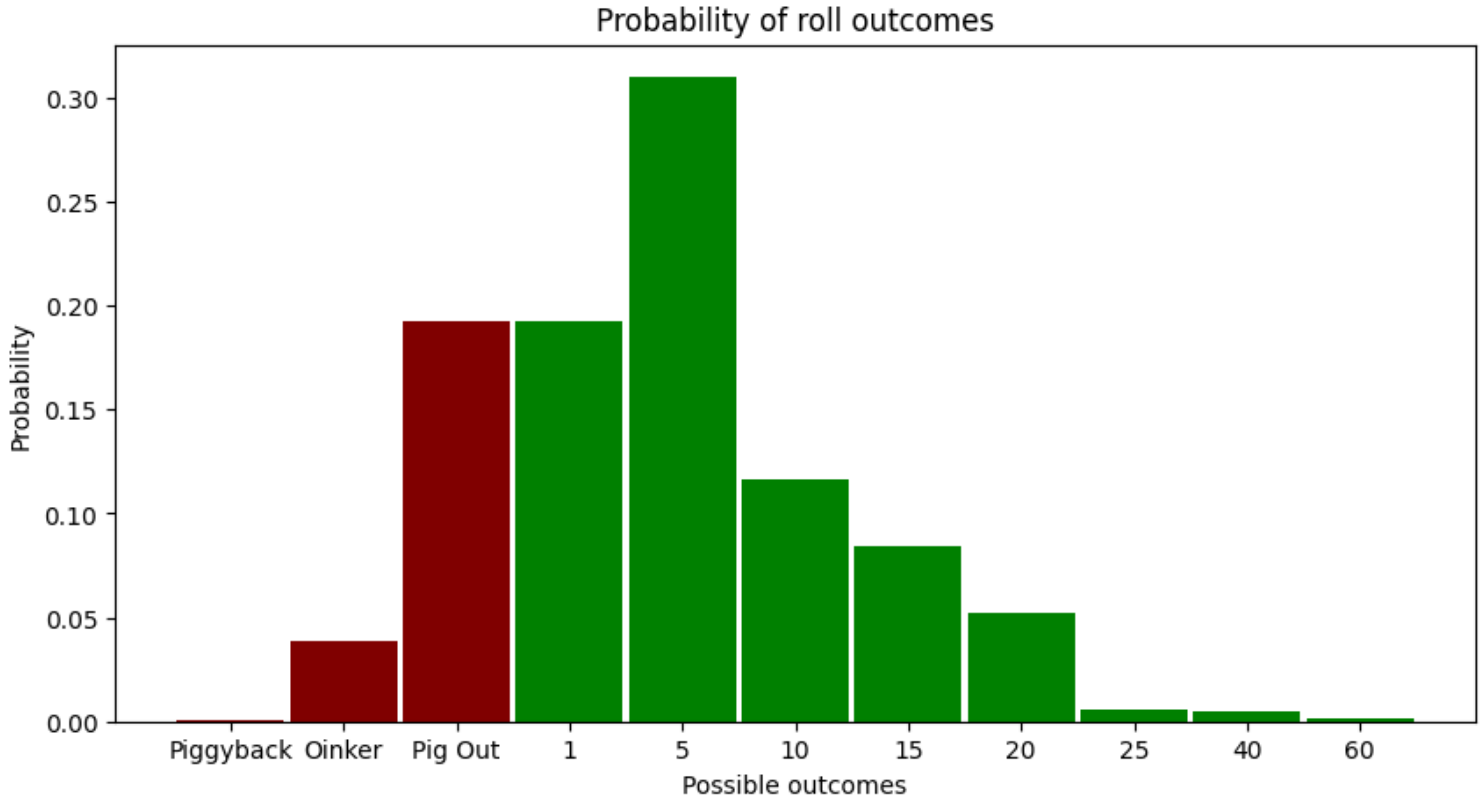
- **Sider** - Pigs lie on the same side, either spot up or spot down - 1 point
- **Pig Out** - Pigs lie on opposite sides - Player's score for the turn is wiped out; play passes to next player.

→ Finally, if the pigs come to rest touching each other:

- **Makin' Bacon (or Oinker)** - Both pigs are touching and both are resting on the table - Player's total score from the game is wiped out; play passes to next player.
- **Piggyback** - One pig rests on top of the other pig and not the table - Player is eliminated from the game; play passes to next player.

Scores and Rules (Summary)

Position	Percentage (1)	Percentage (2)	Percentage (3)	Percentage	Single	Double
Side (no dot) Pink	34.9%	34.1%	65% / 2	62% / 2	0	1
Side (dot) Dot	30.2%	32.9%	65% / 2	62% / 2	0	1
Razorback	22.4%	19.5%	20%	15%	5	20
Trotter	8.8%	9.2%	9%	10%	5	20
Snouter	3.0%	3.5%	4%	7%	10	40
Leaning Jowler	0.61%	0.8%	1%	4%	15	60
Pig Out					Pigs lie on opposite sides Player's score for the turn is wiped out	
Makin' Bacon/Oinker				2%	(Back to Zero) Player's total score from the game is wiped out	
Piggyback					Player is eliminated	


Probability of roll outcomes
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
Piggyback
Oinker
Pig Out
1
5
10
15
20
25
40
60
Possible outcomes