02565 Re-exam Instructions
Version 1.0

Tue Herlau
tuhe@dtu.dk

August 22, 2024

1 General information

Evaluation form: The evaluation is graded on the 7-step scale. The grade is determined by an overall assessment of your project work and an oral exam. The project work will be weighted as approximately one fifth.

Project: Details about the project can be found on DTU Learn.

Questions: Please do not hesitate to contact me by email tuhe@dtu.dk or Discord if you want clarification of any questions you may have about the exam.

2 The oral exam

The oral-exam will be a **individual, without preparation, and without aids**. The duration is approximately 25 minutes. The oral exam is a stand-in for the written exam. The procedure is:

- You draw a random subject from the list of eight subjects described in section 2.2
- You give a brief, 5 minute presentation on the subject. It is recommended that you use the whiteboard for drawings, key definitions, etc. We may ask minor clarifying questions during the presentation.
- We will then proceed to ask questions about any other part of the syllabus
- You should **not** bring a computer, slides, notes, solved exercises, etc.

2.1 Reports

Existing reports are automatically transferred. Don't hand in reports you have not changed.

If you wish to update (improve) an existing report please use to the comment field on DTU learn to indicate which parts of the report was changed, and ensure you remain in compliance with the report description.

2.2 Oral exam subjects

The emphasis is on concrete, practical knowledge of basic definitions and how/why the methods work. It is better to be well prepared in the basics than have a high-level but non-operational/superficial understanding.

I have included single main example for each subject. The examples are chosen as the simplest/most natural illustration of key concepts in the subject. I recommend that you:

- Understand what problem the example solves,
- How it does so (i.e., how actions and other quantities are computed etc.),
- Be able to account for relevant theory behind the methods (general formulas use and other results relevant for the method)
- within reason, be able to account for variations in the example (changing conditions, parameters, etc.)
- within reason, be able to account for how you would approach implementing the method in python (i.e., consider if and how the topic was part of the earlier written exams).

The examples are intended to give us something concrete and practical to talk about, and for you to prepare at beforehand. In your presentation, you are very welcome to use the examples if you feel it is helpful.

Note the examples are not exhaustive, and the list of subject do not cover all relevant aspects of the course (such as Sarsa learning and other tabular methods).

1: The dynamical programming algorithm for finite-horizon control Hereunder how the principle of optimality relates to global optimality of the dynamical programming algorithm; knowledge of the proof of the principle of optimality is not required

- **Example: Inventory-control** This problem is described in the lecture notes and is used in first project. See also `ex02/inventory.py`.

2: PID control The PID control method. Hereunder what problem we are trying to solve, what the PID control rule is and why it looks the way it does.

- **Example: PID on the locomotive environment** The example is described in [Her24, sections 14.1 to 14.3] and also found in `ex04/pid.py`.

3: The discrete linear quadratic regulator linearization Hereunder the form of the linear-quadratic problem, the discrete LQR algorithm is derived, and how LQR can be applied to non-linear problems using linearization

- **Example: Linearization and balancing of the pendulum** The example described in [Her24, section 17.1] (linearization of the pendulum environment for upright balancing) and also found in `ex07/linearization_agent.py`

4: Direct methods for optimal control Hereunder the form of the continuous-time control problem and how it can be turned into a discrete optimization problem using using trapezoid collocation

- **Example: Pendulum swingup** The example described in [Her24, section 15.3.5] and also found in `ex05/direct_pendulum.py`

5: Bandit algorithms Explain terminology relating to bandits and bandit algorithms as seen in the course (Simple bandit and UCB1)

- **Example: Simple bandit agent** The method can be found in [SB18, Section 2.4] and we are concerned with the example described in [SB18, Figure 2.2], specifically as shown in `ex08/simple_agents.py`. Be mindful of what the plot shows and how it is produced.

6: Bellman's equations and their relationship to reinforcement learning Bellman's equations and their iterative form, the policy improvement theorem, and how they relate to the tabular learning methods taught in the course.

- **Example: Policy Iteration** We will consider policy iteration as applied to the small $4x4$ grid-world described in [SB18, Figure 4.1]. Concretely, we will consider the policy iteration example `ex09/policy_iteration.py`.

7: Eligibility traces Their definition and how they are used in reinforcement learning. The treatment in [SB18] is based on value-function approximations, however, I recommend sticking to the tabular case as in the lecture for simplicity.

- **Example: Sarsa(λ) and gridworld** Our example illustrate the updates to the Q-values sketched in [SB18, Example 12.1] using $\mathrm{Sarsa}(\lambda)$. Specifically, focus on the open gridworld demo we have used during the lectures, in particular `ex12/sarsa_lambda_open.py`. Don't concern yourself with the second part of the example with keyboard play; we are only interested in how the Q-values are updated using $\mathrm{Sarsa}(\lambda)$

8: Q-learning and Value-function approximations Extending a tabular method such as Q-learning using function approximators. What cost function do we attempt to update and how does it relate to standard (tabular) Q-learning?

- **Example: Q-learning with linear function approximators** We will consider the Mountain-Car example with linear function approximators (see [SB18, Figure Figure 10.2], however instead of semi-gradient Sarsa we will consider semi-gradient Q-learning[1]. The details of the linear tile-encoding used by [SB18] is irrelevant. You should only aim to understand this part at the level it is explained in [SB18, Section 9.5.4].

[1]Note that by the comments in the chapter they only differ in the estimation of the Q-values, and Q-learning is easier to implement than Sarsa

2.3 Optional gridworld demos

Most of our discussion of reinforcement learning has followed [SB18] in being centered around tabular methods in grid-world scenarios, and how the $Q(s, a)$-functions or $V(s)$-functions are defined and computed by the various methods. This makes for natural discussion subjects during the exam.

As a potential help, the file `exam_tabular_examples/tabular_examples.py` in the exercise folder summarize the methods we have seen, and allows you to re-produce the gridworld demos used in class. The demos are obviously optional and simply summarize methods familiar from the syllabus.

3 Syllabus

The course syllabus is the reading material from [Her24, SB18] as described in the reading plan as well as the lecture slides during the spring semester. Not all parts of a syllabus makes for useful discussion during an oral exam, and in particular I want to de-emphasize:

- The online λ-return algorithms such as online and true-online TD(λ). In other words, the methods which use the truncated $G^{\lambda}_{t:h}$-return (see [SB18, Section 12.3])

4 Hints:

An oral exam is supposed to be a discussion/conversation between the student and examiner. You have some control over this discussion by your selection of examples and which equations you choose to write when you give your presentation and answer questions.

It is therefore a good idea to have a firm grasp of the basics of each question – i.e., what is the basic problem formulation? What is the problem we are trying to solve? I recommend that you focus on a concrete understanding of what the things you choose to write/say, i.e. what is this expectation with respect to?, is the x you write here a vector, matrix or number?, etc.

Having a firm understandings of the basics of each question will very likely give you something to say regardless of what you are asked about. Conversely, memorizing an equation without knowing what it means is both very difficult and will likely result in reasonable follow-up questions that can be hard to answer.

Part of such a conversation is also to navigate the natural time-limitation. You should decide what the *'the most important thing'* is for each subject.

References

[Her24] Tue Herlau. Sequential decision making. (Freely available online), 2024.

[SB18] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. (Freely available online).