Introduction to Deep Reinforcement Learning

Coding 1 – Search Algorithms, Reinforcement Learning , Deep Neural Networks

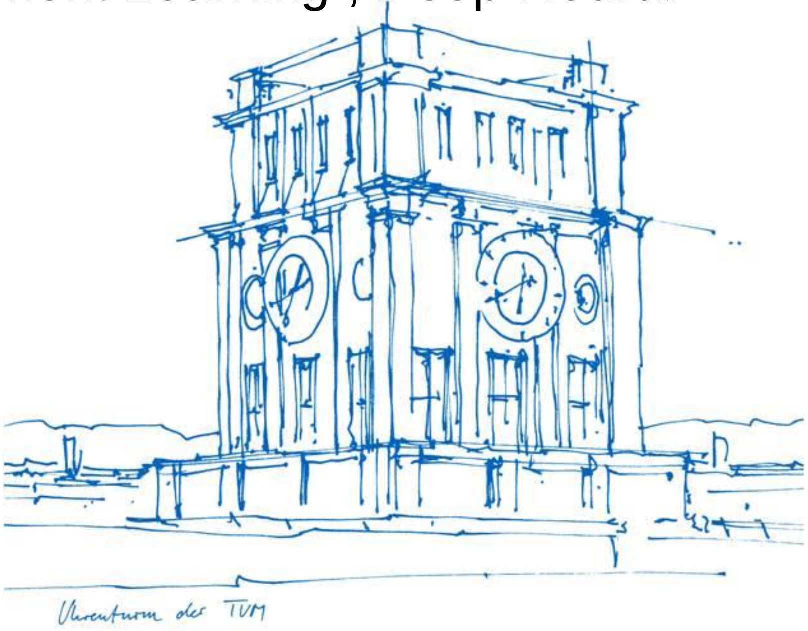
Prof. Dr. Maximilian Schiffer

Professorship for Business Analytics and Intelligent Systems

TUM School of Management

Technical University of Munich

Winter Term 2024/2025



Exercise Types

Repetition of topics + Discussion of worksheets

- We upload the respective worksheets on Moodle, usually on the day of the corresponding lecture
- In the exercise we repeat essential parts of the lecture and discuss the results of the worksheet
- Worksheets are not mandatory but will help you to pass the exam successfully!

Coding exercises

- *We hand out three coding exercises*
  1. *Search Algorithms + Neural Networks + Reinforcement Learning*
  2. *DQN*
  3. *Policy Gradients*
- *For every coding exercise we present you the exercise, and discuss with you the results*
- *We use Python + TensorFlow 2 for the practical exercises*
- *Python knowledge recommended*
- *Coding exercises are not mandatory but will help you to pass the exam successfully!*

Installation: Download code

We uploaded the code in a *.zip folder to Moodle

1. Download the *.zip folder
2. Extract the *.zip folder

Installation: Code structure

```
../path/to/your/directory
|       reinforcement_learning
        |       config.py
        |       environment.py
        |       run_Q_learning.py
        |       util.py
|       requirements.txt
|       deep_neural_network
        |       data/
        |       config.py
        |       preprocessing.py
        |       train_deep_neural_network.py
|       search
        |       config.py
        |       environment.py
        |       run_search.py
        |       util.py
|       README.md
```

README.md → Here you can also find the complete exercise description

Installation: Python

- We use python3 for the Coding Exercise
- Check if python is installed on your computer with
  `python --version` or
  `python3 --version` command in the terminal
- If python is not installed: install python: https://www.python.org/

PyCharm

We have to distinguish between a programming language and an Integrated Development Environment (IDE). An IDE is an application that consolidates different aspects of writing code, like editing source code, debugging, etc…

Programming language: python

IDE: PyCharm (feel free to use any other IDE if you want)

- Install PyCharm: https://www.jetbrains.com/help/pycharm/installation-guide.html
- You can make a student license to use PyCharm Professional

Installation: Python

Virtual Environment

- A virtual environment is an isolated Python environment where the dependencies of a project are installed – so the dependencies do not have to be installed system wide
- This gives the possibility to have different versions of dependencies in your projects
- Install a virtual environment with the terminal
  - https://packaging.python.org/guides/installing-using-pip-and-virtual-environments/
- Install a virtual environment with PyCharm
  - https://www.jetbrains.com/help/pycharm/creating-virtual-environment.html

Packages

- A Package is a collection of modules with prebuilt functions that you can use within your code
- You only need the packages named in the requirements.txt file to solve the exercise
- Install the packages from the requirements.txt file in your virtual environment
  - Activate virtual environment: `source env/bin/activate` (Unix) `.\env\Scripts\activate` (Windows)
  - Install packages from requirements.txt: *`pip install -r requirements.txt`*

Presentation of Coding Exercises: Search Algorithms

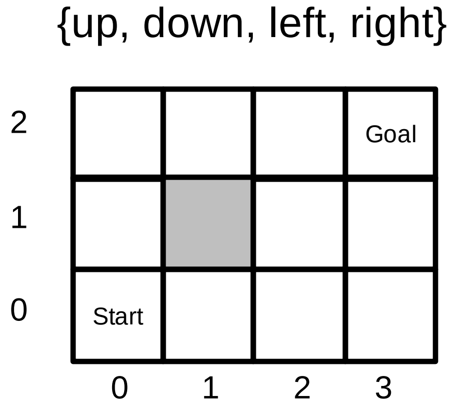
The objective of this task is to implement Graph Search algorithms to find a path from a start state START to a goal state GOAL in a deterministic Gridworld environment.

Action space: {up, down, left, right}

When an action would lead the agent to run against a wall this transition (s,a,s') does not exist.

Implement the three algorithms:

- Depth-First Search
- Breadth-First Search
- Uniform-Cost Search



run_search.py: Here you have to implement the code (#implementHere) (Code)

util.py: Defines the agent class and auxiliary functions

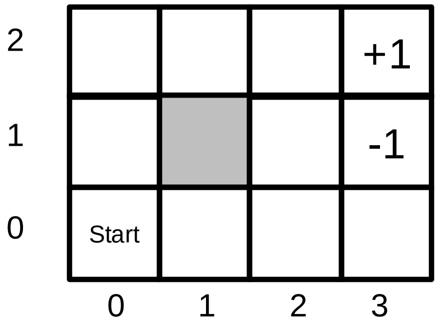
environment.py: Defines all functions to act with the environment

Presentation of Coding Exercises: Reinforcement Learning

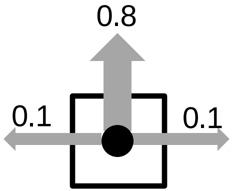
Write a code to train an agent that maximizes its reward in a non-deterministic Gridworld environment by using Q-Learning.
When choosing an action, with probability 0.8 the agent executes this action, whereas with a probability of 0.1 (0.1) the agent moves in a right angle to the left (right) side of the intended direction.

Action space: {up, down, left, right}

{up, down, left, right}



run_Q_learning.py: Here you have to implement the code (#implementHere) (Code)
util.py: Defines the agent class and auxiliary functions
environment.py: Defines all functions to act with the environment
config.py: defines hyperparameters – feel free to change hyperparameters in order to experience how results depend on hyperparameter settings



Presentation of Coding Exercises: Deep Neural Networks

Write code to train a Deep Neural Network that predicts the target variable of the following data sets

1. Iris
2. Bank
3. Wine

Watch the preprocessing.py file for further information. Or:
https://archive.ics.uci.edu/ml/datasets/iris
https://archive.ics.uci.edu/ml/datasets/bank+marketing
https://archive.ics.uci.edu/ml/datasets/Wine+Quality

Iris

- **Predict the species of a flower** {Iris Setosa, Iris Versicolour, Iris Virginica}
- Classification
- Features
  - sepal length in cm
  - sepal width in cm
  - petal length in cm
  - petal width in cm

Bank

- **Predict if a client will subscribe for a bank term deposit or not** {yes, no}
- Classification
- Features
  - Numercial features and categorical features preprocessed with one-hot encoding

Wine

- **Predict the quality of a wine**
- Regression
- Features
  - Numerical features

Presentation of Coding Exercises: Deep Neural Networks

config.py: defines hyperparameters – feel free to change hyperparameters in order to experience how results depend on hyperparameter settings

preprocessing.py: You do not have to change anything in this file but you can have a look how the data sets have been preprocessed

train_deep_neural_network.py: Here you have to implement the code

1. Define the deep neural network models (#defineModel)
   1. With subclass of Model (#defineModel_model)
   2. With sequential API (#defineModel_sequential)
   3. With functional API (#defineModel_functional)
2. Read in the data sets (#readInData)

Option1 – args.implementation=‘detail’

3. Define metrics to track the learning progress (#defineMetrics)
4. Define the loss function (#defineLoss)
5. Define gradient (#defineGradient)
6. Define the optimization algorithm (#defineOptimization)
7. Train the model (#trainModel)
8. Evaluate the model (#evaluateModel)
9. Use the model to make predictions (#predictTarget)

Option2 – args.implementation=‘no_detail’

10. Define metrics to track the learning progress (#defineMetrics2)
11. Compile the model (#compileModel2)
12. Fit the model (#fitModel2)
13. Evaluate model (#evaluateModel2)

The objective of this exercise is to learn how **to implement Deep Neural Networks** in order to prepare you for *Coding Exercise2: Deep Reinforcement Learning*. You will see the results for predicting target variables are not good. In research, when using Deep Neural Networks to predict target variables we would implement more preprocessing steps and use more sophisticated model architectures to make better predictions.

Solution

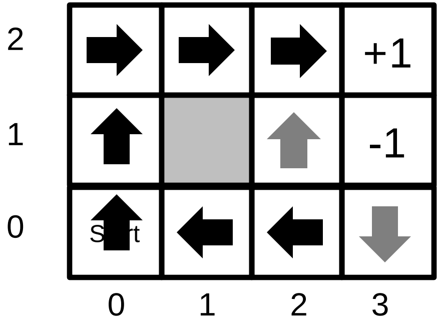
Search

BFS: [(0,0) → up, (0,1) → up, (0,2) → right, (1,2) → right, (2,2) → right, (3,2)]

DFS: [(0,0) → right, (1,0) → right, (2,0) → right, (3,0) → up, (3,1) → up, (3,2)]

UCS: [(0,0) → up, (0,1) → up, (0,2) → right, (1,2) → right, (2,2) → right, (3,2)]

Reinforcement Learning



Deep Neural Network

The loss value should decay smoothly. You see the development of the loss value when running your code with args.implementation=„detail“