UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Faculty of Engineering, Built Environment and Information Technology

EAI 320

Intelligent Systems

Practical 4 Guide

Updated on 27 May 2020

Concept: Prof. Warren P. du Plessis
Guide written by: Llewellyn Strydom
Guide revised by: Michael Teles

I. General Instructions

This section contains general instructions which are applicable to all assignments.

- The prescribed LaTeX format should be used, and the generated portable document format (PDF) file should be submitted.
- The commented Python source code should be submitted.
- Code must be submitted via the EAI 320 ClickUP page. Do not email code to the lecturer or Assistant Lecturer (AL) as emailed code will be considered not to have been submitted.
- The names of submitted files must have the format provided below.
  `eai320_prac_{practical nr.}_{student nr.}_task_{X}.{extension}`
- No late assignments will be accepted. No excuses for late submission will be accepted.
- Each student must do their own work. Academic dishonesty is unacceptable and cases will be reported to the university Legal Office for suspension.
- The code should include the standard declaration of originality for individual assignments provided in the General Study Guide of the Department of Electrical, Electronic and Computer Engineering. All information from other sources must be clearly identified and referenced.
- The code provided to students should also not be included in submissions.
- Any attempt to interfere with the operation of the framework used (e.g. to modify the scores of agents) will be regarded as academic dishonestly.

A. Submission

Code should only be submitted via the

- `Practicals` → `Practical 4` → `Practical 4 Task 1`
- `Practicals` → `Practical 4` → `Practical 4 Task 2`

links on the EAI 320 ClickUP page.

Only assignments submitted in via ClickUP will be accepted. Do not email code to the lecturer or AL as emailed code will be ignored.

Late assignments will not be accepted! Accepting late assignments is extremely unfair on those students who submit their work timeously because their tardy colleagues are effectively given additional time to complete the same work. Students are advised to submit the day before the deadline to avoid inevitable problems with ClickUP, internet connections, unsynchronised clocks, load shedding, hard-drive failure, computer theft, etc.. Students who choose to submit close to the deadline accept the risk associated with their actions, and no excuses for late submissions will be accepted.

Students will be allowed to submit updated copies of their assignments until the deadline, so there will be no excuse for submitting late. Rather be marked on an incomplete early version of your assignment than fail to submit anything.

B. Academic Dishonesty

Academic dishonesty is completely unacceptable. Students should thus familiarise themselves with the University of Pretoria's rules on academic dishonesty summarised in the study guide and the university's rules. Students found guilty of academic dishonesty will be reported to the Legal Office of the University of Pretoria for suspension.

Students are required to include the standard originality declaration for individual assignments provided in the General Study Guide of the Department of Electrical, Electronic and

Computer Engineering as part of their code. This standard originality declaration of originality includes a statement that the student submitting the report is aware of the fact that academic dishonesty is unacceptable and a statement that the submitted work is the work of that student. Failure to include this declaration of originality will mean that the report will be considered not to have been submitted.

While students are encouraged to work together to better understand the work, each student is required to independently write their own code. No part of any student's work may be the same as any part of another student's work.

Students should clearly indicate material from other sources and provide complete references to those sources. Examples of commonly-used sources include the textbook [1] and this document [2]. Note that this does not mean that students may reuse code and/or information found in books, on the internet, or in other sources as students are required to complete the tasks themselves.[1]

II. Scenario

The goal of this assignment is to implement a naïve Bayes classifier. The naïve Bayes classifier is an algorithm that solves classification problems by making a simple assumption: that all features in the input vector $\mathbf{f}$ are conditionally independent given the class. Although this assumption does not always hold, naïve Bayes classifiers have been widely used and have shown to perform well on a variety of tasks [1].

The naïve Bayes classifier is based on Bayes' law, which can be written as

$$p\left(C_k|\mathbf{f}\right) = \frac{p(\mathbf{f}|C_k)p\left(C_k\right)}{p(\mathbf{f})}, \tag{1}$$

where $p\left(C_k|\mathbf{f}\right)$ is the probability that an object belongs to a certain class C_k given a feature vector $\mathbf{f}$. Since the denominator of Bayes' law acts simply as a normalisation term, only the numerator need be considered when implementing a naïve Bayes classifier. The numerator term can be rewritten as

$$\begin{aligned} p(\mathbf{f}|C_k)p\left(C_k\right) &= p\left(f_1, \ldots, f_n, C_k\right) && (2)\\ &= p\left(f_1|f_2, \ldots, f_n, C_k\right)p\left(f_2, \ldots, f_n, C_k\right) && (3)\\ &= p\left(f_1|f_2, \ldots, f_n, C_k\right)p\left(f_2|f_3, \ldots, f_n, C_k\right)p\left(f_3, \ldots, f_n, C_k\right) && (4)\\ &= p\left(f_1|f_2, \ldots, f_n, C_k\right)p\left(f_2|f_3, \ldots, f_n, C_k\right)\ldots p\left(f_{n-1}|f_n, C_k\right)p\left(f_n|C_k\right)p\left(C_k\right) && (5)\end{aligned}$$

using the product rule of probability [3].

If the elements from the feature vector are considered to be conditionally independent then Eq. 1 simplifies to

$$p\left(C_k|\mathbf{f}\right) \propto p\left(C_k\right)\prod_{i=1}^{F} p\left(f_i|C_k\right), \tag{6}$$

where F is the number of features.

[1]The objective of all academic assignments is fundamentally that students learn by completing the assignments. Merely reusing code and/or information found elsewhere defeats this objective because a key part of the learning process is performing the tasks oneself.

The naïve Bayes classifier implements the following simple decision rule to select the class with the highest posterior probability $\hat{C}$ for classification:

$$\hat{C} = \underset{k\in\{1,\ldots,K\}}{\operatorname{argmax}}\, p\left(C_k\right) \prod_{i=1}^{F} p\left(f_i | C_k\right). \tag{7}$$

For more information on naïve Bayesian classifiers please refer to the prescribed textbook [1].

III. Instructions

A. Task 1

The task is to train a naïve Bayes classifier on historical game data to play a game of rock-paper-scissors (RPS), and use it to implement an agent. The agent has to be trained prior to playing the game, and the relevant prior and marginal probabilities have to be contained in the code that the student submits.

The historical game data is provided in a comma-separated values (CSV) file, `data.csv` and is the same file that was used for the previous practical. If agent 1 is the agent that is implemented using the naïve Bayes classifier and agent 2 is the opposing agent, the objects that agent 1 and agent 2 will play in the current round are represented by x_t and y_t respectively. The history contained in the first column of `data.csv` is then in the form $[x_{t-2}, y_{t-2}, x_{t-1}, y_{t-1}]$, and the second column represents the next object that the opponent played given the two game history. The optimal move for a given history is thus the move that will beat the opponent's move in the second column. Thus, the naïve Bayes classifier should aim to predict the optimal choice of x_t given a history of two moves.

B. Task 2

The task is to implement a RPS agent that implements a version of the naïve Bayes classifier that is able to learn in an online fashion. The agent will thus be expected to learn the strategy of an arbitrary opposing agent by recording the history of moves played during a game and calculating the relevant probabilities that are required to implement a naïve Bayes classifier.

In order to ensure that efficient code is written and to allow for easy testing, this agent must play 100 matches of 1 000 rounds each against a simple bot (like `only_rock.py` or `only_scissors.py`) in less than 1 minute when running on a single central processing unit (CPU) core.

IV. Submission Requirements

Students are not required to submit a technical report for this assignment.
Since no report is required for this assignment, there is a large emphasis on the fact that submitted code must be well-commented as well as easy to understand and follow.

A. Code Instructions

Each student is required to submit code conforming to the requirements listed below. A mark of zero will be awarded if the submitted code file does not run, produces errors, or does not follow the instructions.

- The code should be written in such a way that it is easy to follow.
- All code must be commented to a point where the implementation of the underlying algorithm can be determined.
- The submission must be a single file.
- Code submitted code for Task 1 will be evaluated using the command
```
rpsrunner.py -m 100 -r 1000 student.py only_rock.py,\
    only_paper.py,only_scissors.py,beat_common.py
```
- Submitted code, for Task 2, will be evaluated using the command
```
rpsrunner.py -m 100 -r 1000 -t 1 student.py agent.py,
```
  , where *agent.py* is a suitable opposing agent.
- All print statements, plots, file writes, and any other functions that were used for unit testing etc. must be removed before submission.
- All submissions must be written using Python 3 as a result of the fact that `rpsrunner.py` is written in Python 3.
- The code must include the author's name and the revision number/date of the revision of the program as a comment at the top of the program.
- The code submitted must be the final implementation of the assignment.
- TurnItIn will not accept code with the file extension `.py`, so the file extensions should be changed to `.txt` before submission.

References

[1] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*, 3rd ed. Prentice Hall, 2010.
[2] L. Strydom, *EAI 320 – Practical 6 Guide*, University of Pretoria, 25 Apr. 2019.
[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

Appendix A
Abbreviations