WASP Software Engineering Assignment

Jack Sandberg

Introduction of research field

In the multi-armed bandit (MAB) problem, we consider an agent that must select an arm to pull from a set of one-armed bandits. After selecting an arm, the agent is given a random reward based on which arm is selected. The average reward for each arm is unknown to the agent and the agent's goal is to maximize the total reward received over a sequence of rounds.

A multi-armed bandit researcher is then interested in designing policies or algorithms that ensure that the agent maximizes its total reward. Good policies must strike a balance between exploration (selecting all arms often enough) and exploitation (only selecting the arm believed to be best).

The simple formulation of the MAB problem allows us to express many problems as MABs, such as news recommendations, molecule design, or optimal sensor placement. In many real-world applications, we have more information about the structure of the problem which is not efficiently represented in the MAB. By considering extensions of the simple MAB problem, we can derive algorithms that can more efficiently solve these problems.

A specific extension of interest in my research is the *combinatorial* and *contextual* MAB (CCMAB) problem in which the agent must select multiple arms every round and is also given extra information about the arms. In my research so far, we have developed and applied CCMAB algorithms to the problem of finding energy-efficient routes for electric vehicles in traffic networks.

Principles from Robert's lectures

1. During the lectures, we discussed the risk-levels of AI systems depending on where in the process they are used based on (Feldt et al., 2018), see Figure 1. The risk level is the lowest for process based systems and highest for runtime systems. The paper mentions that the canonical example for runtime systems is autonomous and self-adaptive systems software systems. MAB research is a prime example of such a system since, by design, the agent should adapt its policy based on the observed rewards. In practice, this could mean that putting my research into products might be more challenging since it can be seen as more risky. Therefore, it will be important to develop good methods for monitoring and understanding the algorithms.
2. Building and deploying AI systems can be complicated and can go wrong in many different places, all the way from initial data collection to monitoring of the deployed system. As was highlighted in the lectures, turning AI development into a structured engineering discipline requires a broad skill set. Martínez-Fernández et al. (2022) highlighted that building multidisciplinary teams is necessary for effective AI development. For any future collaborative research, it will be worthwhile to look for collaborators with complementary skill sets. Doing so can ensure that the team as a whole has a

greater understanding of the challenges involved with the project. However, it should be noted that this point likely applies to most research.

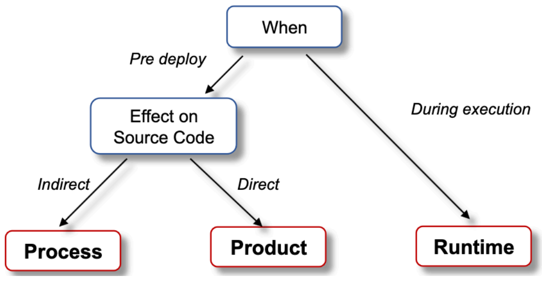

Figure 1: From (Feldt et al., 2018), overview of the different points of application (PA) in the AI-SEAL taxonomy.

Principles from guest lectures

1. During the SAAB guest lecture, the air traffic management industry was discussed and it was mentioned that the industry is very resistant to any changes. Changing working practices requires very careful and methodical strategies of winning people over. However, parts of the AI field on the other hand is very rapidly evolving and will happily throw out old models for the latest SOTA. When trying to apply AI to more traditional or slowly evolving industries, one should consider how the introduction can be made gradual to make the transition easier and gentler. My research project is more towards the academic side and might not relate to this particularly well, but I think a broader point of the lecture is that one must carefully adapt a "sales-pitch" of whatever you are selling to your audience/customer (regardless if its your research or a product). The SAAB lecture highlighted how
2. During the Volvo guest lecture, an example of using LLMs for generating parts of software tests was discussed. The automotive industry, and Volvo in particular, is very safety oriented and thus also tend to make changes in an incremental manner. In a sense, it is a bit surprising to see a safety oriented company use (what is often regarded as) an unreliable technology such as LLMs. However, I think this is a good demonstration of the risk-levels from Robert's lectures - applying a unreliable technology to an internal process is less risky than deploying it live in a product. The field of MABs is applied most often at a runtime level and may therefore be subject to more scrutiny in traditional industries. But by thinking about how it can be integrated and prove its value in internal processes, the field could see broader adoption in more risk-adverse industries.

A Case Study on AI Engineering Practices: Developing an Autonomous Stock Trading System

a) In (Grote & Bogner, 2023), the authors apply AI engineering practices in a case study on developing an autonomous stock trading system. After surveying the literature, the authors identify 10 practices that they will apply, some examples are: *standardize and automate data quality check procedures*, *use error validation and categorization*, and *continuously measure model quality, performance and drift*. Structured field notes are used to document the experiences. The authors give a thorough account of how the practices played out and whether they found it helpful. However, the case study only considers a single project with mostly one developer (the main author) and it can therefore be difficult to generalize the results. Moreover, the author describes his background as more software engineering oriented which makes some of the practices seem rather basic or at times even inappropriate. For example, practice 4 states that one should use cross-validation to avoid testing an ML model on data it has already seen. This is basic practice for any practitioner but the authors seem to have either omitted or missed that blindly applying cross-validation in an application with time-dependence, such as *stock trading*, gives the model access to future information which would invalidate any result obtained.

b) The paper describes the process of creating, developing and deploying an autonomous system that must adapt to the environment. In my research on MABs, we develop algorithms that adapt over time through online learning and should act autonomously. Thus, the experiences of Grote & Bogner can help me adopt better AI engineering practices in my research.

c) In a larger AI-intensive software project, the core idea of the paper (identifying and applying sound AI engineering practices) would help to ensure the final product is built using high quality data, is easier to monitor and conforms better to the outlined objective. As stated, MAB algorithms are designed to be used in online fashion and be adaptive. Following practice 5 of (Grote & Bogner, 2023) (*continuously measure model quality, performance and drift*) would help ensure that the model does not degrade after deployment. Similarly, Grote & Bogner described how talking to multidisciplinary stakeholders (practice 10) helped in understanding what the model needs to do and helped them make the code much more performant. In many MAB applications, the goal you want to achieve is not always directly measurable, ex. recommending videos on YouTube to maximize the revenue of the business in the long term. Thus it is important to learn from domain experts about how different proxy-metrics interact with the true metric.

d) Some of the practices are already used in my project (using cross-validation, and logging results with model version and input data). Some of the practices are more difficult to apply in my project, such as automating hyperparameter optimization and model selection due to having more complicated search spaces and computations taking significantly longer. However, adopting practices such as using error validation and categorization could help improve reliability.

Prevalence of Code Smells in Reinforcement Learning Projects

a) In (Cardozo et al., 2023), the authors study the prevalence of code smells in 24 popular reinforcement learning (RL) based Python projects. The authors find that RL repositories contain many code smells - even those written by experienced RL engineers. Cardozo et al. (2023) argues that RL has some intrinsic complexity that lead to these code smells and improved tooling and abstractions are necessary to get rid of them. As discussed in the lectures, good AI engineering is intimately connected to good software engineering practices and we must therefore try to understand how these code smells can be avoided.

b) The field RL is closely connected to the field of multi-armed bandits. In RL, we extend the MAB framework and assume that the environment has a state that changes based on the actions taken by the agent. Equivalently, the MAB problem can be seen as a subset of RL where only the initial state exists. Therefore, the lessons learnt on good software practices for RL can certainly be applied to MAB.

c) The paper finds that the following code smells are the most prevalent: Long Method, Large Class, Multiply-Nested Container and Long Parameter List. If our research was applied in a larger software project, we should take proactive measures to try to avoid these code smells from the start. As was pointed out in (Cabral et al., 2024), data scientists may have a variety of STEM backgrounds and can lack strong software engineering foundations. Therefore, one should ensure that ML experts and software engineers interact and collaborate within the project to learn best practices from their respective domains.

   The paper also notes that some smells, especially Multiply-Nested Container, arise from the complex state-action spaces in RL. Multi-armed bandits does not make use of states so this issue may not be as prevalent. However, the action space can still be complicated and one should be careful in how it is used in the code.

d) Revisiting my code, I recognize many of the smells discussed in the paper. Thus, this paper will help me refactor the code and establish more sound coding practices.

Bibliography

Cabral, R., Kalinowski, M., Baldassarre, M. T., Villamizar, H., Escovedo, T., & Lopes, H. (2024). Investigating the Impact of SOLID Design Principles on Machine Learning Code Understanding. *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI*, 7–17. https://doi.org/10.1145/3644815.3644957

Cardozo, N., Dusparic, I., & Cabrera, C. (2023). Prevalence of code smells in reinforcement learning projects. *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*, 37–42.

Feldt, R., Oliveira Neto, F. G. de, & Torkar, R. (2018). Ways of applying artificial intelligence in software engineering. *Proceedings of the 6th International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering*, 35–41.

Grote, M., & Bogner, J. (2023). A Case Study on AI Engineering Practices: Developing an Autonomous Stock Trading System. *2023 IEEE/ACM 2nd International Conference on AI Engineering–Software Engineering for AI (CAIN)*, 145–157.

Martínez-Fernández, S., Bogner, J., Franch, X., Oriol, M., Siebert, J., Trendowicz, A., Vollmer, A. M., & Wagner, S. (2022). Software engineering for AI-based systems: a survey. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *31*(2), 1–59.