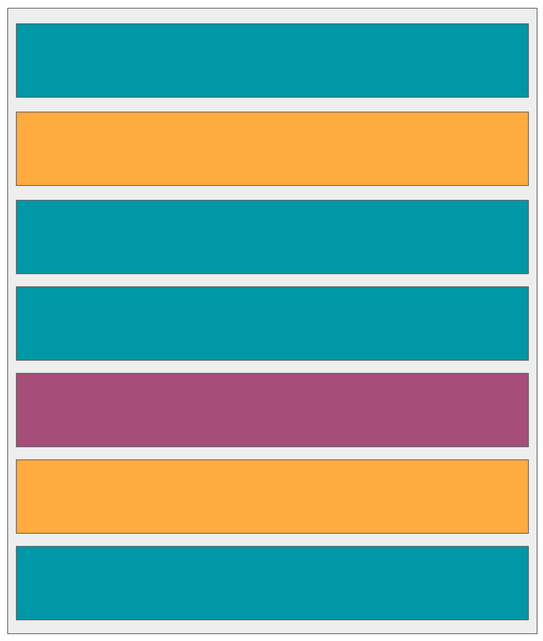
Applying Software Engineering Principles To A Machine Learning Algorithm: Lessons learned

Université de Montréal · geodes

Thomas Schweizer
thomas.schweizer@umontreal.ca

Michalis Famelis
famelis@iro.umontreal.ca

MOTIVATION

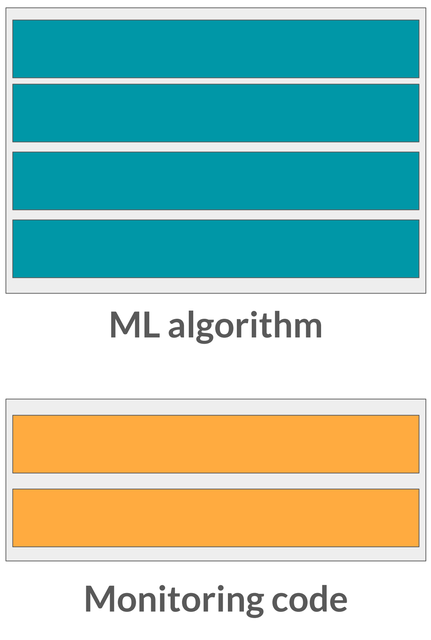


Original ML code

The machine learning (ML) research algorithm worked but had a few challenges, hindering productivity

Software engineering (SE)

We improved the code using a common software engineering technique



ML algorithm

Monitoring code

Boilerplate code

Challenges

READABILITY MAINTAINABILITY REUSABILITY

TESTABILITY PRODUCTIVITY

Decoupling concerns using refactorization

SIMPLE EFFECTIVE QUICK

Benefits

READABILITY

MAINTAINABILITY

REUSABILITY

INCREASE IN AGILITY & TESTABILITY

BETTER PRODUCTIVITY

ML AND *SE* CAN HELP EACH OTHER

ML → SE Software repository mining, Integrated development environment, etc..

SE → ML ∅

ML DEVELOPERS ARE SPECIALIZED PROGRAMMERS WITH THEIR OWN CONCERNS

OBSERVABILITY

TESTABILITY & VERIFICATION

MONITORING

POST-PROCESSING

DEPLOYMENT

LESSONS LEARNED

- The developer was able to increase his productivity
- Unique trade-offs for each project and stakeholders
- Small changes can already yield great productivity increase
- Extensive industry evidence in other domains

REMAINING CHALLENGES

- Environnement portability (Cloud / Local)
- Processor portability (CPU / GPU)
- Reproducing experiments (Gold Standard)

FURTHER HORIZONS

- *Does changing the structure of a machine learning algorithm affect the results [1] ?*
- *What are the unique architectural concerns in ML and how can we address them ? Model Driven Engineering for ML ?*
- *What errors can we find automatically in ML algorithms by using SE ?*

[1] Mitliagkas, Ioannis et al. “Asynchrony begets momentum, with an application to deep learning.” 2016. 54th Allerton CCC conference: 997-1004

Acknowledgements: Rémi Le Priol, Ioannis Mitliagkas and, Marios-Eleftherios Fokaefs