Winter Term 21/22

Adversarial Self-Supervised Learning with Digital Twins

Project Scope

Prof. Dr. Holger Giese (holger.giese@hpi.uni-potsdam.de)

Christian Medeiros Adriano (christian.adriano@hpi.de) - **"Chris"**

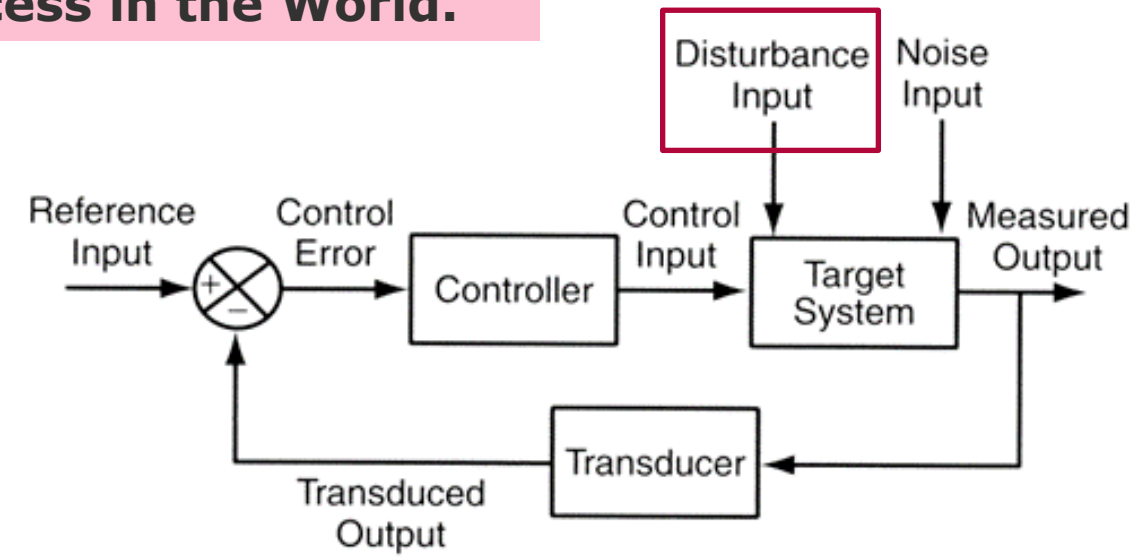
He Xu (he.xu@hpi.de)

Infrastructure to run experiments

“Success in the Lab is not guarantee of success in the World.”

Feedback loop models

"When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one." **Vladimir Vapnik**

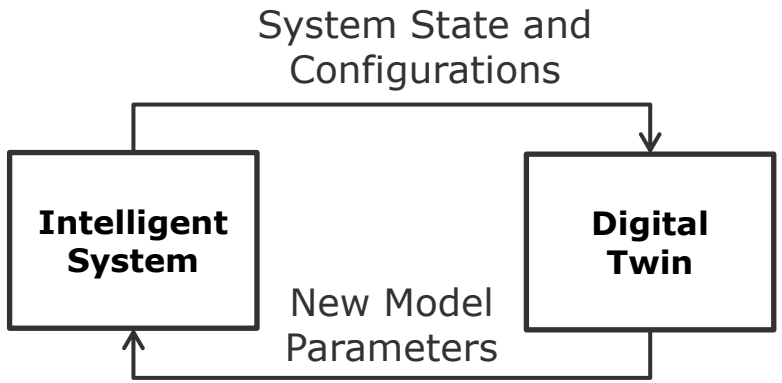


Simulation models

"Thinking is acting in an imagined space" **Konrad Lorenz**

"Perception is a generative act" – [Gross et al. 1999]

" Consciousness is a controlled hallucination " - [Seth et al. 2000]



Operating in Sparse, Safety Critical, Uncertainty World

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

- Operation produces few if any training examples (relevant events are rare)
- Predictive performance is necessary but not sufficient
- Simulation requirements for robustness are functions of the operational context, hence:
  - Requirements are domain-specific, which make them more challenging to automate
  - Their reification happens outside the machine learning algorithms

Requirements Engineering Questions

1. Which simulation requirements do we need?
2. How do we verify that these simulation requirements are being met?
3. How to incorporate these simulation requirements into training?
4. How do we transfer the simulation outcomes to production?

Work-Packages

1- Hidden Markov Model

Study how to move the HMM implemented in the Python side to the Java side

2- Failure Inject Mechanism

Study how to generate failure injects that reflect that failure propagation patterns

3- Reinforcement Learning

Study how to replace the Supervised Learning controller (Regression) with a Self-Supervised One (RL)

4- Monitoring

Study how to visualize the utility, risk, and mechanism gap between the Real and Simulated (Digital-Twin)

5- Adversarial Tests

Study how to generate stress tests that show how policies that have equivalent predictive outcome at training present distinct outcomes at adversarial test sets

Research Problems

RP.1 Underspecification under Adversarial Failure Propagation

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Goal: Show that (for testing data) different prediction models can successfully learn safe recovery-zones. **However**, some of these models fail when subject to out-of-distribution data.

- recovery-zones are learned off-line by a safe policy π_{safe}
- failure repair is learned on-line by an optimal task policy π_{task}

Insights [Thanajeyan et al. 2021]:

- Train models with distinct hyper-parameterization (prior-knowledge)
- Choose failure traces that produce tight rankings (utility of components are very similar)
- Increase variance in utility from training/validation to testing. For that, train the models with failure propagation traces that are smaller (i.e., lower uncertainty) than the traces in the testing data (i.e., higher uncertainty).

Main Work-packages:

- Reinforcement Learning Agent (hyper-parameters)
- Failure Propagation (shift in the threshold of propagation)

RP.2 Convergence under adversarial unsafe actions

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Goal: Show different rates of synchronization between Production and Simulation can lead to:

- excessive cost of training and redeployment
- increase in the risk of under-performance

Insights [Seurin et al. 2019]:

- Redefine optimality in terms of expected utility, which allows us to adjust the utility (reward) to the probability (risk) of achieving it.
- Compare the convergence of a risk-averse (e.g., Monte-Carlo, off-policy) with risk-driven (e.g., bootstrapping, on-policy).
- Evaluate how the agent learns unsafe actions, for instance, removes an instance when there is high uncertainty about the average utility of a component and/or increase the penalty (negative reward) when trying to fix the wrong component.

Main Work-packages:

- Reinforcement Learning Agent (e.g., Sarsa vs Q-Learning)
- Utility Model (Ridge-Regression)

Lack of Robustness in AI Systems

Robustness comprise three requirements:

(1) generalizability to OOD datasets,

(2) insensitivity to model underspecification, and

(3) protection against unsafe action-state spaces.

Traditional methods

Methods to find optimal bias-variance trade-off, cross-validation, regularization, ensemble methods, sparse representation learning (deep-neural nets)

Methods to generate more and better data

Methods to train larger models, over-parameterized models, foundation models, etc.

Methods to train models quicker and online

Our approach - adversarial training

Lack of robustness when models are exposed to shifts that force system to make predictions on out of the distribution (OOD) datasets, i.e., datasets not used during training and testing. OOD datasets are generated in many situations that are beyond the control of the training methods, e.g., production environments are more complex and evolve in unpredictable ways.

The alternative is to let agents to continue learning after deployment. The approaches for that involved detecting distribution shifts and automatically triggering training. However, a shift in distribution involves not one new data point but various ones, which might lead the agent to underperform in unpredictable ways. After detecting the shift, another problem arises, how to collect the data needed to retrain the agent? For IID data, that might simply involve enriching the existing dataset with the characteristics of the new detected shifts. This might involve applying transformation that preserve meaning, e.g., image rotations, sentence paraphrasing, or generating data via bootstrapping and or importance sampling techniques.

However, in RL the data is non-IID, i.e., the sequence of the training data matters. This sequence is produced by the agent balancing exploitation and exploration. The latter when done in OOD datasets could lead the agent to take unsafe actions, for instance, leading the system into an invalid state that can cause physical or financial harm to the system owners. In other words, learning online in the production environment prevents the agents from learning from their mistakes.

Research Goals

Can we learn to:

- detect when a new model needs to be re-trained?
- determine the feature and outcome space that requires re-training?
- generate the type of data and how much is needed for re-training?

Approaches to training

- Learn which action-states are possibly unsafe
- Learn how to recover from unsafe action-states

Adversarial training - Create situations for unsafe action-states

- Distribution shift in failure propagation sequences
- Distribution shift in expected utility of repair actions

Design Questions

Simulation to detect Underspecification

Think about answers to the for the following questions:

1. How to generate test-data?
2. How to generate distribution shifts? How much shift (how to measure it)?
3. How to train distinct models that perform equally well on test data?
4. How do you know that two models are distinct?
5. How do you know that they perform equally well on the test data?
6. Preliminary assignment and ideas for the work-packages

Suggested Deadline = Nov 16 (present during lecture)

END