Predicting Machine Failures with Reinforcement Learning

Dalton Schutte

What & Why

- **What:**
  - How effectively can reinforcement learning agents predict machine failures?
- **Why:**
  - Reinforcement Learning (RL) is defined on Markov Decision Processes (MDP) and time series data provides an interesting state space
  - RL is commonly used in process control but is, arguably, most famous due to recent success in games such as Go
  - Additionally, I have a lot of personal interest in RL and believe this will be an interesting use-case to explore

Data

Three datasets provided in *Predicting Machine Failures from Multivariate Time Series: An Industrial Case Study* [3]

- Wrapping Machine
  - >670K time steps with 16 sensor observations each
- Blood Refrigerator
  - ~140K time steps with 12 sensor observations each
- Nitrogen Generator
  - ~86K time steps with 4 sensor observations each

How

- Metrics will be F_1 score, recall, precision, and mean time-from-event (measures the mean number of time steps between the labeled event and when the method predicted the event)
- Baselines will come from standard techniques in Statistical Process Control (SPC) [2]
  - Control charts
- Competing methods will be RL agents using a mix of Q-Learning and Policy Optimization [1,7]
  - Reward will be based on the inverse temporal distance between the agent "stopping the machine" (returning the value that means the signal has decayed)
  - Value functions and policies will be a variety of neural networks (GRU, LSTM, transformer, etc.)

How

- Monte Carlo Tree Search [5,6] is the algorithm underlying AlphaGo and its application to this type of problem is absent in the literature
    - May yield interesting results
- Additional comparisons shall be drawn against other methods from time series analysis and deep learning
    - ARIMA*
    - EWA*
    - LSTM
- Time permitting:
    - Cooperative multi-agent RL will be investigated [7]
        - Randomly split variables into overlapping sets that will be used to train *k* RL agents
    - Graph-enabled RL will be investigated [4]

* A threshold will be determined that will trigger a “stop machine” response

References

1. Gao, X. Deep reinforcement learning for time series: playing idealized trading games. Preprint at http://arxiv.org/abs/1803.03916 (2018).
2. Guthrie, W. F. NIST/SEMATECH e-Handbook of Statistical Methods (NIST Handbook 151). National Institute of Standards and Technology https://doi.org/10.18434/M32189 (2020).
3. Pinciroli Vago, N. O., Forbicini, F. & Fraternali, P. Predicting Machine Failures from Multivariate Time Series: An Industrial Case Study. *Machines* **12**, 357 (2024).
4. Shaik, T. *et al.* Graph-enabled Reinforcement Learning for Time Series Forecasting with Adaptive Intelligence. Preprint at http://arxiv.org/abs/2309.10186 (2024).
5. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *Nature* **529**, 484–489 (2016).
6. Silver, D. *et al.* Mastering the game of Go without human knowledge. *Nature* **550**, 354–359 (2017).
7. Sutton, R. S. & Barto, A. G. *Reinforcement Learning: An Introduction*. (MIT Press, Cambridge, Mass, 1998).

Appendix - Reinforcement Learning

- Method where an Agent learns how to interact with an environment by acting on the state of the environment and receiving feedback as the environment changes as a consequence of that action
- Components:
  - RL Agent: a parameterized policy (or value function) that takes the environment state as input and outputs an action
    - I will explore Q-Learning and Policy Optimization
    - I will not explore model-based RL
  - Environment: an object with an observable state and some mechanisms to change the state based on a received action and return a reward based on that change
    - My environment will be set of all time series for each machine
    - Each state will be the set of observations at a given time step

Appendix - Reinforcement Learning

- A Markov Decision Process is a tuple *(S, A, R, p)* where:
  - *S* (State space) := is the set of allowable states
    - In our setting, *S* will be equal to the real numbers
  - *A* (Action space) := is the set of actions the RL agent can choose from at each time step
    - *A* will be *{0=*do not stop machine,*1=stop machine}*
  - *R* (Rewards) := is the reward function
    - *R* will be $1/|t_{pred}-t_{label}|$ where t_{pred} is the time step the RL agent returned 1 and t_{label} is the time step provided by the data
    - Subject to change, reward function crafting/learning is a deep area in its own right
  - *p* (Transition Probabilities) := the probability that maps the current state and action to the next state
    - Implicitly provided by the environment