Writeup

Yannik Pitcan
Ram Akella

September 16, 2019

Research Objective

Deep reinforcement learning methods have not been studied for the preventive maintenance problem, not withstanding claims made by firms such as GE. The first goal is to understand how well such deep reinforcement learning methods can help us implicitly predict quantities such as RUL (remaining useful life) and achieve close to optimal repair policies. The second goal is to investigate how we can combine observations about the system process with prior knowledge and for multiple predictor distributions. In particular, we want to understand how to mix the two models to get a better understanding of our system. One is data-based empirical knowledge and the second is either prior knowledge or expertise-based adaptation of empirical knowledge. This will be combined with real-time observations. Currently, the first task is completed with deep reinforcement learning and classical POMDP implementations for the CMAPPS dataset. The focus from now onwards is the second task.

Example of why this is important

Past work attempted to understand the RUL distribution via fitting a Weibull distribution. Furthermore, the fitted parameters (λ and k) are dependent on input covariates. However, this doesn't fully incorporate the fact that we are dealing with a dynamical system and x_t, y_t are changing with respect to $t - x_t$ is the underlying system state and y_t is the sensor observation.

Detailed Discussion of Two Research Pathways

Currently, the models in place just treat the preventive maintenance problem as an MDP, focusing solely on measurement observations (y_t) while ignoring the potential hidden states (x_t). This leads us to discuss the first research direction.

Path 1: Single Source

This may be split into two parts:

- **Estimation:** System identification – estimate an underlying state and then use model-based approaches to solve for optimal repair strategy to reduce total maintenance cost.
- **Control:** Determining optimal actions without knowledge of underlying states (model-free reinforcement-learning approach)
- **Combining Estimation and Control:** Understanding the benefits of using a deep-RL approach that combines the two.

As before, we create a simulation for a machine running until it reaches failure. The underlying states $(1, 2, 3, 4)$ represent the health level of the machine, with 1 indicating very-low/failing and 4 very-high/perfect condition. A level of 2 denotes low but not failing and 3 moderately high health.

Our state transitions are given by

$$\begin{bmatrix} p_{11} & 0 & 0 & 0 \\ p_{21} & p_{22} & 0 & 0 \\ 0 & p_{32} & p_{33} & 0 \\ 0 & 0 & p_{43} & p_{44} \end{bmatrix}$$

where p_{ij} denotes the probability of the system health moving from level i to j. The above system is for the case where no action is taken. This model assumes gradual degradation.

The actions here are either to repair $a = 1$ or do nothing $a = 0$.

When a repair is done, then

$$\begin{bmatrix} 0 & p'_{12} & p'_{13} & p'_{14} \\ 0 & 0 & p'_{23} & p'_{24} \\ 0 & 0 & 0 & p'_{34} \\ 0 & 0 & 0 & p'_{44} \end{bmatrix}$$

is our transition matrix, where we use p' to denote these are probabilities when we do a repair.

Our observations depend on whether we repair the system or not. Furthermore, the observation space is continuous with the observations distributed by pdfs f_0 and f_1 for actions 'no-repair' and 'repair' respectively.

If we don't repair the system, then define $O(o|s, a = 0) \sim f_0(s)$. Else, if we do repair, then $O(o|s, a = 1) \sim f_1(s)$.

If we use discrete observation buckets, then we have a matrix form

$$O(0) = \begin{bmatrix} NoRepair & 1 & 2 & 3 & 4 \\ 1 & o_{11} & o_{12} & o_{13} & o_{14} \\ 2 & o_{21} & o_{22} & o_{23} & o_{24} \\ 3 & o_{31} & o_{32} & o_{33} & o_{34} \\ 4 & o_{41} & o_{42} & o_{43} & o_{44} \end{bmatrix}$$

and

$$O(1) = \begin{bmatrix} Repair & 1 & 2 & 3 & 4 \\ 1 & o'_{11} & o'_{12} & o'_{13} & o'_{14} \\ 2 & o'_{21} & o'_{22} & o'_{23} & o'_{24} \\ 3 & o'_{31} & o'_{32} & o'_{33} & o'_{34} \\ 4 & o'_{41} & o'_{42} & o'_{43} & o'_{44} \end{bmatrix}$$

Lastly, our reward function takes the following form:

$$\sum_{t=1}^{T} \sum_{s=1}^{4} [c_1(4 - s_t) I(s_t = j) + c_2(s_t) I_M(a_t = 1 | s_t = j)]]$$

where I is an indicator function. $c_1(4 - s_t)$ is the cost of maintenance and $c_2(s_t)$ is the cost of being in a degraded state.

Another way of writing this is as follows

$$R(0) = \begin{bmatrix} NoRepair & Reward \\ 1 & r_1 \\ 2 & r_2 \\ 3 & r_3 \\ 4 & r_4 \end{bmatrix}$$

$$R(1) = \begin{bmatrix} Repair & Reward \\ 1 & r'_1 \\ 2 & r'_2 \\ 3 & r'_3 \\ 4 & r'_4 \end{bmatrix}$$

The overarching goal is to use reinforcement learning methods with POMDPs to give an optimal strategy for repairing the system that minimizes the total repair costs and losses due to non-repair.

The two routes are as follows:

- 1. EM based approach for estimation and then determining an optimal control policy using any POMDP solver. For example, we could use a VAE (variational autoencoder) for time series. This doesn't incorporate control, maximizing a likelihood function, and outputs a particle filter model.
- 2. DVRL (deep variational reinforcement learning), which combines both estimation and control. This is because it's optimizing an objective cost function that also depends on model parameters.

We also incorporate knowledge of the start and end states being in perfect and poor condition respectively, which we discuss in the next section.

Current Status 1

So far, we have studied four different POMDP based techniques for modeling the preventive maintenance problem. We implemented value-iteration, Monte-Carlo Tree Search (POMCP), ADRQN, and DVRL on the CMAPSS dataset. After finishing up these implementations, we are currently analyzing how well deep-RL methods such as DVRL and ADRQN performed versus model-based methods. We also incorporate constraints on the start and end states by a method similar to boosting, where we propagate the errors between our true start and end states and our estimates, re-running DVRL until we reach convergence. The next steps are analyzing how two sources of information (prior knowledge with observations) should be combined to determine optimal repair strategies.

In practice, we often have some existing knowledge about the system process before seeing measurements. This leads us to the second research path.

Path 2: Combining Multiple Sources

Here, we ask how to combine prior knowledge about the data generating process when determining the optimal repair strategy that minimizes costs. For example, if we have a controlled system generating data, this alters the observed machine measurements as opposed to an uncontrolled system. This is because if the machine is deteriorating, we repair and restore the system back to full health so our measurements seen will be different than if we let the machine fail.

The alternative route would be to identify enough of the distribution based on impact on the objective function and disambiguate.

For now, let us assume the data generating process takes on one of the following forms:

- No prior distribution – our data is generated via bootstrapping. For example, with the C-MAPPS data, we could resample from the trajectories.
- One prior
  - Do we trust this prior distribution? Or should we ignore that and just bootstrap? Or just combine both?

- We have two prior distributions and we would like to understand which one generates our data, or how they interact with each other. For example, we can combine two priors via a "weighted average" (partial update) or fully via a Bayesian update.

These three are the possible ways of combining the two models, which we are investigating:

- One generating model at each timestep
- Mixture model
- Hierarchical Bayes

Analogously to the single source model, we have another layer of complexity when studying estimation versus control here. Either we could

- Estimate underlying model and then come up with a control policy using POMDP-RL.
- Perform control and estimation at once using a method such as DVRL again.

This section gives one a glimpse of some of the open problems we are working on.

Sequential Testing

The paper `http://stat.columbia.edu/~jcliu/paper/GSPRT_SQA3.pdf` introduces Sequential Probability Ratio Testing (SPRT) for different families of hypotheses.

When we have two families of composite hypotheses, i.e.,

$$H_0 : f \in \{g_\theta : \theta \in \Theta\} \text{ against } H_A : f \in \{h_\gamma : \gamma \in \Gamma\}$$

, we consider a sequential test where we compute

$$L_n = \frac{\sup_{\gamma\in\Gamma} \prod_{i=1}^n h_\gamma(X_i)}{\sup_{\theta\in\Theta} \prod_{i=1}^n g_\theta(X_i)}.$$

If L_n crosses e^A or e^{-B} for constants $A, B > 0$, then we stop sampling. If $L_n > e^A$, then we reject the null.

We could use this to infer which distribution is generating our data, but the research component is understanding how to use this with particle filters. If we use a variational autoencoder to learn an underlying model, the output is a particle filter. Currently there hasn't been work on how to use SPRT with particle filters.

Example: Mixture Model Setting

What we would like to solve is initially the following: If our data is generated from $\epsilon_t P_0 + (1-\epsilon_t)P_1$, where P_0 and P_1 are the two priors, does $\epsilon_t \to 0$ or to a constant over time? Maybe P_0 is one prior generated from bootstrapping and P_1 is the second prior provided or based on data that incorporates domain knowledge.

In this setting, we can think of our $z_t = \{x_t, y_t\}$ pairs coming from data sources P_0 and P_1.

Define $p(z_t|P_0)$ and $p(z_t|P_1)$ to be the probabilities of observing z_t given P_0 and P_1 respectively at time t. Then

$$p(P_j|z_t) = \frac{\epsilon_t p(z_t|P_0) I(j=0) + (1-\epsilon_t) p(z_t|P_1) I(j=1)}{\epsilon_t p(z_t|P_0) + (1-\epsilon_t) p(z_t|P_1)}.$$

But now, we want to understand

$$p(\{P_{j_1}, \ldots, P_{j_T}\}|\{z_1, \ldots, z_T\})$$

where each P_{jt} is either P_0 or P_1.

The simplistic approach to estimate ϵ would be to do a maximization of the above quantity over all assignments of $\{P_{j1}, \ldots, P_{jT}\}$ to the set $\{P_0, P_1\}^T$. This method, however, is computationally intensive and furthermore does not account for time-dependence in our data.

We propose modeling this as a multiarmed bandit problem, where P_0 and P_1 are the two arms. The goal here is to then use concentration inequalities to understand good approximate repair policies. These will be approximate because there is a dependency between actions taken at different timesteps and we are looking at finite-horizon solutions.

Current Status 2

For this, the first priority is studying how we can combine particle filters (outputs of VAE) can be used with SPRT. Other open problems that will be tackled after this are how to get guarantees on learning multiple source distributions using concentration-type inequalities.