Writeup

Yannik Pitcan
Ram Akella

October 8, 2019

Research Objective

Deep reinforcement learning methods have not been studied for the preventive maintenance problem, not withstanding claims made by firms such as General Electric (GE) [1]. The goal of the current study is to explore the use of such methods for the preventative maintenance problem. The first step to achieving this goal is to understand how well such deep reinforcement learning methods can be leveraged to implicitly predict maintenance metrics such as Remaining Useful Life (RUL) to achieve close to optimal repair policies. The second step is to investigate how to combine two models: one, observations about the system process, and two, prior knowledge of the system. In particular, this study seeks to understand how to mix the observations with the prior knowledge to get a better understanding of the given system. The first model is data-based empirical knowledge and the second model is either prior knowledge or expertise-based adaptation of empirical knowledge. The study also seeks to understand how to combine these models with multiple predictor distributions. Currently, the first step has been realized using deep reinforcement learning and classical POMDP implementations for the CMAPPS dataset. The following paper describes the deep reinforcement learning and POMDP implementation used. Two methods for combining real-time observation with prior knowledge were taken in this research. The first method is fitting a Hidden Markov Model (HMM) to the provided data to create a POMDP, and the created POMDP was solved using model-based approaches. The other method uses deep reinforcement learning techniques [5]. Future efforts beyond what is presented in this paper will focus on the second step of the proposed research.

Dynamical System Consideration

Past work on RL for preventive maintenance have shown promise, but suffered from a few flaws. One flaw was the lack of distinguishment between covariates and sensor readings. For example, past work attempted to understand the RUL distribution via fitting a Weibull distribution. But, the fitted parameters (λ and k) were dependent on input covariates. This dependence shines light on the second flaw, the use of summary values such as RUL to represent the full distribution, when a set of samples (x_t, y_t) from said distribution may have been preferable. The presented research differs from past work in that it models the data as a dynamic system with **latent states** and observations and avoids encapsulating the knowledge of the distributions with a simple summary statistic.

POMDP vs MDP

The use of a Partially Observable Markov Decision Process (POMDP) goes in tandem with distinguishing between covariates and sensor readings. In previous research, when the preventive maintenance problem was treated as an MDP, it ignored the fact that there may be hidden states (for example, system health) that impacted sensor observations.

Using a Modified EM Framework

One of the primary question this research seeks to answer is, how we can determine these hidden states? To our knowledge, nobody has developed methodology to understand latent states when there are extra state constraints in the manner we have. The first part of this section discusses the estimation used and the later subsections discuss the new method proposed plus some experiments validating our methodology.

Let $S = \{1, 2, \ldots, X\}$ be the space of underlying states and O be the space of observations. A HMM $\theta = (\pi_0, P, B)$ is parameterized by π_0, P, and B, which represent the initial state matrix, state transition matrix, and emission matrix, respectively. Let $\pi_0 = [P(s_0 = i)]_{i=1}^{X}$, $P_{ij} = P(s_{t+1} = j | s_t = i)$ and $B_{ij}(\eta) = P_\eta(y_t = j | s_t = i)$.

For estimation, we can use an EM (expectation maximization) framework, treating the hidden states as the latent states in a Hidden Markov Model (HMM) with our measurements as the observations. The latent states represent the health of the machine, where 1 is failing condition and M is perfect condition.

Let $\mathcal{Y} = (Y^{(1)}, \ldots, Y^{(D)})$ be the set of D observation sequences, where each $Y^{(i)} = (y_1^{(i)}, \ldots, y_T^{(i)})$, and assume that each observation is an IID draw.

The technique used was to repeat the following steps until convergence ($\theta = (\pi_0, P, B)$):

- Compute $Q(\theta, \theta^s) = \sum_{x \in \mathcal{S}} \log[P(\mathcal{O}, x; \theta)] P(x | \mathcal{O}; \theta^s)$ for the E-step
- Set $\theta^{s+1} = \arg\max_\theta Q(\theta, \theta^s)$ for the M-step.

Let $\theta^s = (\pi_0^s, P^s, B^s)$.

We can rewrite the auxiliary log-likelihood as

$$\begin{aligned} Q(\theta^s, \theta) &= \mathrm{E}\left[\sum_{k=1}^{N}\sum_{i=1}^{X} I(x_k = i) \log B_{iy_k}(\eta) + \sum_{k=1}^{N}\sum_{i=1}^{X}\sum_{j=1}^{X} I(x_{k-1} = i, x_k = j) \log P_{ij} | y_{1:N}, \theta^{(n)}\right] + const(\theta^s) \\ &= \sum_{k=1}^{N}\sum_{i=1}^{X} \pi_{k|N}(i) \log B_{iy_k}(\eta) + \sum_{k=1}^{N}\sum_{i=1}^{X}\sum_{j=1}^{X} \pi_{k|N}(i,j) \log P_{ij} + const(\theta^s) \end{aligned}$$

Of note is that $\pi_{k|N}(i) = P(x_k = i | y_{1:N}, \theta^s)$ and $\pi_{k|N}(i,j) = P(x_k = i, x_{k+1} = j | y_{1:N}, \theta^s)$ are computed in the E step of the EM framework. The derivation of these probabilities is presented in the Forward-Backward Filtering subsection.

Continuous Emissions

For this work, assume $B_{ij}(\eta) = P_\eta(y_t = j | s_t = i) = f(j)$, where $f \sim N(C(i), R(i))$. Therefore, $\eta = (C(1), \ldots, C(X), R(1), \ldots, R(X))$. The parameters η and P are estimated in the M step.

Then $\pi_{k|N}(i) \log B_{iy_k}(\eta)$ becomes

$$\sum_{k=1}^{N}\sum_{i=1}^{X} \pi_{k|N}(i)\left(-\frac{\log R(i)}{2} - \frac{(y_k - C(i))^2}{2R(i)}\right) = -\frac{1}{2}\sum_{k=1}^{N}\sum_{i=1}^{X} \pi_{k|N}(i)\left(\log R(i) + \frac{(y_k - C(i))^2}{R(i)}\right).$$

$$P_{ij} = \frac{\sum_{k=1}^{N} \pi_{k|N}(i,j)}{\sum_{k=1}^{N} \pi_{k|N}(i)}$$

is the estimate after maximizing $Q(\theta^s, \theta)$ with respect to the constraint that P is stochastic.

And subsequently, the M step updates for C and R yield $C(i) = \frac{\sum_{k=1}^{N} \pi_{k|N}(i) y_k}{\sum_{k=1}^{N} \pi_{k|N}(i)}$ and $R(i)$ does not have a closed form but one can use gradient based methods to find a maximizer.

Forward-Backward Filtering

It is not obvious how to compute $\pi_{k|N}$ and $\pi_{k|N}(i,j)$ at a first glance. To compute them, a forward-backward filtering algorithm is applied. In this algorithm, two passes are made through the data. The first pass goes forward in time while the second pass goes backward in time.

$$\pi_{k|N}(i) = P(x_k = i|y_{1:N}) = P(x_k = i|y_{1:k}, y_{k+1:N}) \propto P(y_{k+1:N}|x_k = i)P(x_k = i|y_{1:k}) = \alpha_{1:k}(i)\beta_{k+1:N}(i)$$

where $\alpha_{1:k}(i) = P(x_k = i|y_{1:k})$ and $\beta_{k+1:N}(i) = P(y_{k+1:N}|x_k = i)$. In shorthand notation, we say

$$\pi_{k|N} \propto \alpha_{1:k}\beta_{k+1:N}.$$

α and β the forward and backward "messages", respectively.
Similarly,

$$\pi_{k|N}(i,j) \propto \alpha_{1:k}(i)P_{ij}B_{jy_{k+1}}\beta_{k+1:N}(j).$$

To compute the forward and backward messages, this is done recursively. For the forward update, we have

$$\begin{aligned}
\alpha_{1:k+1} &= P(x_{k+1}|y_{1:k+1}) \\
&= P(x_{k+1}|y_{1:k}, y_{k+1}) \\
&\propto P(y_{k+1}|x_{k+1}, y_{1:k})P(x_{k+1}|y_{1:k}) \\
&\propto P(y_{k+1}|x_{k+1})P(x_{k+1}|y_{1:k})
\end{aligned}$$

And

$$\begin{aligned}
P(x_{k+1}|y_{1:k+1}) &\propto P(y_{k+1}|x_{k+1}) \sum_{x_k} P(x_{k+1}|x_k, y_{1:k})P(x_k|y_{1:k}) \\
&= P(y_{k+1}|x_{k+1}) \sum_{x_k} P(x_{k+1}|x_k)P(x_k|y_{1:k}) \\
&= P(y_{k+1}|x_{k+1}) \sum_{x_k} P(x_{k+1}|x_k)\alpha_{1:k}.
\end{aligned}$$

And, for the backward update:

$$\begin{aligned}
\beta_{k+1:N} &= P(y_{k+1:N}|x_k) \\
&= \sum_{x_{k+1}} P(y_{k+1:N}|x_k, x_{k+1})P(x_{k+1}|x_k) \\
&= \sum_{x_{k+1}} P(y_{k+1:N}|x_{k+1})P(x_{k+1}|x_k) \\
&= \sum_{x_{k+1}} P(y_{k+1}, y_{k+2:N}|x_{k+1})P(x_{k+1}|x_k) \\
&= \sum_{x_{k+1}} P(y_{k+1}|x_{k+1})P(y_{k+2:N}|x_{k+1})P(x_{k+1}|x_k) = \sum_{x_{k+1}} P(y_{k+1}|x_{k+1})\beta_{k+2:N}P(x_{k+1}|x_k)
\end{aligned}$$

Assume we find the observation at the kth step is j, i.e. $y_k = j$. Then $O_k = diag(B_{*j})$, meaning it is a matrix with diagonal entries given by $P(y_k = j|x_k = i)$. In matrix form, the above recursions take the form of:

$$\alpha_{1:k+1} \propto O_{k+1}P^\top \alpha_{1:k}$$

and

$$\beta_{k+1:N} = PO_{k+1}\beta_{k+2:N}.$$

Adapting for Fixed States

As mentioned before, our state sequences must be constrained to those state sequences that begin with a "perfect" state and end at a "failing" state. The difference between past work and the current work is that both starting and ending states are constrained in the presented research.

There are several ways to carry out this constraint; this work investigates two methods. The first method involves modifying the M step calculation for the expected log-likelihood to add an extra constraint in the Lagrangian. The second method incorporates the constraint in the forward-backward recursion. Ultimately, the second method was chosen for future study, as it was more amenable to a closed form solution and easier to implement.

Since the starting state will be a fully healthy state,

$$\alpha_{1:1} = P(x_1|y_1) = (1, 0, 0, 0).$$

Every time we do the E step in our EM algorithm, we must compute $\alpha_{1:j}$ for $j \in \{1, \ldots, N\}$.

$$\alpha_{1:k} \propto \prod_{i=2}^{k} (O_i P^\top) \alpha_{1:1}$$

in the forward direction.

We then deal with the end state constraint in our backward message updates. Normally, one sets $\beta_{N:N} = (1, 1, 1, 1)$, but since we know with certainty that the final state is a failing state, we set $\beta_{N+1:N} = (0, 0, 0, 1)$.

To see why this works, look at

$$\begin{aligned}
\beta_{N:N} &= P(y_{N:N}|x_{N-1}) \\
&= \sum_{x_N} P(y_{N:N}|x_N, x_{N-1}) P(x_N|x_{N-1}) \\
&= \sum_{x_N} P(y_{N:N}|x_N) P(x_N|x_{N-1}) \\
&= \sum_{x_N} P(y_N, y_{N+1:N}|x_N) P(x_N|x_{N-1}) \\
&= \sum_{x_N} P(y_N|x_N) P(y_{N+1:N}|x_N) P(x_N|x_{N-1}) = \sum_{x_N} P(y_N|x_N) \beta_{N+1:N} P(x_N|x_{N-1})
\end{aligned}$$

And since

$$\sum_{x_N} P(y_N|x_N) \beta_{N+1:N} P(x_N|x_{N-1}) = \sum_{x_N=3} P(y_N|x_N) \beta_{N+1:N} P(x_N|x_{N-1}) = P(y_N|x_N = 3) \beta_{N+1:N} P(x_N = 3|x_{N-1})$$

and

$$P(y_{N:N}|x_{N-1}) = P(y_N|x_N = 3, x_{N-1}) P(x_N = 3|x_{N-1}),$$

this means $\beta_{N+1:N} = (0, 0, 0, 1)$ is our initialization vector.

These computed β and α values, are used to compute $\pi_{k|N}(i)$

And in the M step, we skip parameter updates for P_{ij} when $i < j$ and keep these values fixed at 0. This idea is inspired by Roweis [3].

In summary, an altered method of forward-backward filtering is used to control the first and last states. This method has not been taken in previous studies, and empirically the results are promising when when using this method.

The estimation of HMM parameters is done for each system trajectory, and then the average of these parameters over all trajectories is calculated. For example, this method estimates the transition matrix

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.368893	0.357284	0.654884	0.246592	0.159665	0.319512	0.662172	0.268697	0.196373	0.377861	0.365443	0.604374	0.629463
1	0.445521	0.436801	0.552530	0.297437	0.192327	0.405016	0.570111	0.322218	0.220602	0.458655	0.445462	0.527639	0.548432
2	0.531650	0.501197	0.479200	0.356338	0.238838	0.520376	0.482756	0.371746	0.261436	0.540383	0.504609	0.437612	0.457751
3	0.669016	0.622176	0.336322	0.452984	0.304682	0.684514	0.320877	0.470418	0.314942	0.682230	0.627440	0.301617	0.310948

Figure 1:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0.009337	0.008524	0.006625	0.005004	0.003778	0.006497	0.006709	0.004922	0.003688	0.007666	0.007730	0.007660	0.008378
1	0.009711	0.009147	0.007293	0.005418	0.004399	0.007117	0.007430	0.005291	0.004283	0.008104	0.008290	0.008257	0.009048
2	0.009785	0.009134	0.006723	0.004501	0.003339	0.006753	0.007112	0.004476	0.003221	0.008190	0.008300	0.007945	0.008670
3	0.010450	0.009430	0.007269	0.004895	0.003306	0.008217	0.008183	0.004830	0.003110	0.008772	0.008593	0.009057	0.009645

Figure 2:

for each sample path for a given machine and then takes the average of all these transition matrices to obtain a state transition matrix for that machine.

The same was done for emission means, and emission covariance matrices.

Figures 1 and 2 show the emission matrices. For emissions, two matrices are used – one representing the means at each state (Figure 1) and one with variances (Figure 2).

The estimated transition matrix is $\begin{pmatrix} 0.66 & 0.34 & 0. & 0. \\ 0. & 0.63 & 0.37 & 0. \\ 0. & 0. & 0.77 & 0.23 \\ 0. & 0. & 0. & 1. \end{pmatrix}$.

Other Experiments

We also test our new estimation method with a custom hidden markov model. In this case, we simulate a four state Gaussian HMM which starts at state 0 and has the transition matrix

$$\begin{pmatrix} 0.5 & 0.5 & 0. & 0. \\ 0. & 0.5 & 0.5 & 0. \\ 0. & 0. & 0.5 & 0.5 \\ 0. & 0. & 0. & 1. \end{pmatrix}$$

The emission distribution at state s is normal with mean $2s$ and variance 0.5. We generate a sample path of length 500 from this and then re-estimate the underlying parameters. With our new approach, we get improved estimates of the transition matrix as

$$\begin{pmatrix} 0.347 & 0.653 & 0.000 & 0.000 \\ 0.000 & 0.652 & 0.348 & 0.000 \\ 0.000 & 0.000 & 0.505 & 0.495 \\ 0.000 & 0.000 & 0.000 & 1.000 \end{pmatrix}$$

with estimated mean emissions $(1.24, 3.40, 5.07, 6.09)$ and variances $(0.63, 0.62, 0.38, 0.21)$.

Previously, with standard HMM fitting without our new forward-backward updates, we were not estimating that state 3 was an absorbing state as the predicted probability of staying at 3 was not 1. Furthermore, when using our state constrained method, we have more stable estimates of the covariance. When a state constraint wasn't used, our variances were $(.63, 66.95, 200.52, 400.30)$ and the variances here were extremely off from the original model variances, unlike in our fitting with constraints. This is a clear indicator that our constrained EM approach is superior to a standard Baum-Welch, especially when working with systems running to failure.

Variational Autoencoders

The prior two approaches involved treating the preventive maintenance problem as a HMM. Conversely, deep learning techniques such as variational autoencoders (VAE) can be used to tackle this problem. Here, the VAE attempts to reconstruct output parameters for a chosen distribution. The goal

in this method is maximize the sum of log marginal likelihood terms $\sum_{n=1}^{N} \log p_\theta(o^{(n)})$ for observations $(o^{(n)})_{n=1}^{N}$ where $p_\theta(o) = \int p_\theta(o|s)p_\theta(s)\,ds$. This is intractable in practice, so instead the sum of ELBOs is maximized, where each ELBO term is a lower bound of the log marginal likelihood.

$$ELBO(\theta, \phi, o) = \mathrm{E}_{q_\phi(s|o)}[\log \frac{p_\theta(o|s)p_\theta(s)}{q_\phi(s|o)}]$$

for a family of encoders $q_\phi(s|o)$ parameterised by ϕ. One can see the analogy between this and the EM approach discussed earlier. This study uses a surrogate expression, ELBO, instead of the log marginal distribution.

Deep Variational RL

Lastly, estimation and control are incorporated by using deep variational reinforcement learning. At a high level, DVRL does not only estimate the underlying model, but it also attempts to find an optimal policy simultaneously. So instead of learning a generative model that maximizes just the ELBO functions, the newly developed method also optimizes an expected return. The proposed method was implemented on CMAPSS data using Tensorflow [2]. Tensorflow was chosen over other machine learning libraries because of the need to restrict the state sequences from model estimates. Tensorflow, although more difficult to prototype in, leads itself to customization far easier than most other tools available.

Detailed Discussion of Two Research Pathways

Currently, the models in place 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 to the first research path.

Path 1: Single Source

The first research path proposed is the use of single source prior knowledge. This path takes the form of:

- **Estimation:** Estimating 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.

To test this research approach, a machine running until failure was simulated. The underlying states $(1, 2, 3, 4)$ represent the health level of the machine, with 1 indicating very-high/perfect and 4 very-low/failing condition. A level of 3 denotes low but not failing and 2 moderately high health.

The state transitions are given by

$$\begin{bmatrix} p_{11} & p_{12} & 0 & 0 \\ 0 & p_{22} & p_{23} & 0 \\ 0 & 0 & p_{33} & p_{34} \\ 0 & 0 & 0 & p_{44} \end{bmatrix}$$

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

$$\begin{bmatrix} p'_{11} & 0 & 0 & 0 \\ p'_{21} & p'_{22} & 0 & 0 \\ p'_{31} & p'_{32} & p'_{33} & 0 \\ p'_{41} & p'_{42} & p'_{43} & p'_{44} \end{bmatrix}$$

is the transition matrix, where p' are the probabilities of when a repair is executed.

The observations depend on whether or not the systems is repaired. 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 no repair is executed, then define $O(o|s, a = 0) \sim f_0(s)$. Else, if there is a repair, then $O(o|s, a = 1) \sim f_1(s)$.

Lastly, the reward function takes the following form:

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

with I as 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 where c_1 and c_2 are constants that we set. In our case, we will set these to $c_1 = 50$ and $c_2 = -50$.

Another way of writing this is as follows:

Table 1: Reward table.

State	**Repair**	**Not Repair**
1	100	-50
2	50	0
3	0	50
4	-50	100

For example, the reward for repairing in state 2 is 50.

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.

Two methods can be used to realize this approach to the given research problsm:

1. Using an EM-based approach for estimation and then determining an optimal control policy using any POMDP solver. For example, using a VAE (variational autoencoder) for time series. This approach doesn't incorporate control, instead maximizing a likelihood function and outputting a particle filter model.

2. A DVRL (deep variational reinforcement learning) approach, which combines both estimation and control. This approach optimizes an objective cost function that depends on model parameters.

Both of these approaches also incorporate knowledge of the start and end states being in perfect and poor condition respectively, which is discussed in the following section.

Current Status 1

So far, the authors 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 realizing these implementations, we are currently analyzing the performance of deep-RL methods such as DVRL and ADRQN versus model-based methods. We have also incorporated the constraints on the start and end states discussed in this paper via an adjustment

to the E-step in the EM algorithm for HMM fitting. The next steps are to analyze how two sources of information (prior knowledge with observations) should be combined to determine optimal repair strategies.

Results from Separating Estimation and Control

After doing HMM estimation we used SolvePOMDP, a Java program that solves POMDPs, to execute value iteration. With our cost function defined from earlier, we saw that expected reward from the optimal policy was 158. Furthermore, in this policy, one starts off not repairing the system, but then chooses the repair action repeatedly in the later iterations.

In practice, there is often some existing knowledge about the system process before seeing any real-time measurements. This knowledge leads to the second research path.

Path 2: Combining Multiple Sources

Here, this study asks how to combine prior knowledge about the data generating process when determining the optimal repair strategy that minimizes costs. For example, if given a controlled system generating data, this alters the observed machine measurements as opposed to an uncontrolled system. This alteration is because if the machine is deteriorating, a repair and restoration the system back to full health is executed such that the measurements seen will be different than if machine fails.

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

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

- No prior distribution – data is generated via bootstrapping. For example, with the C-MAPPS data, data is resampled from the trajectories.
- One prior distribution:
  - Can this prior distribution be trusted, or should it be ignored and just bootstrap a distribution; or, should the distributions just be combined?
- If there are two prior distributions, which one generates our data, or how do they interact with each other? For example, they can be combined a "weighted average" (partial update) or fully via a Bayesian update.

These three possible ways of combining the two models are under investigation:

1. One generating model at each timestep
   - In this setting, only one of P_1 or P_2 generates data.
2. Mixture model
3. Hierarchical Bayes

Analogously to the single source model, there is another layer of complexity when studying estimation versus control here. The analysis either:

- Estimates the underlying model and then comes up with a control policy using POMDP-RL.
- Performs control and estimation at once using a method such as DVRL again.

This section gives one a glimpse of some of the open problems the authors are addressing.

Sequential Testing

Thomas Kolarik's paper "Using SPRT as stopping condition" [4] introduces Sequential Probability Ratio Testing (SPRT) for different families of hypotheses.

When there are 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.

This is then used to infer which distribution is generating the data. The current research component is understanding how to use this with particle filters. By using a variational autoencoder to learn an underlying model, the output becomes a particle filter. Currently, there has not been any published research on how to use SPRT with particle filters.

Example: Mixture Model Setting

What this study proposes to solve is initially the following: if the 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, the $z_t = \{x_t, y_t\}$ pairs can be thought of as 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, this study seeks 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 the data.

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

References

[1] Predix Platform, online: https://www.ge.com/digital/iiot-platform/machine-learning-analytics, last access 10-2-2019.

[2] Abadi, MartĂn, et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." *arXiv preprint arXiv:1603.04467* (2016).

[3] S. Roweis. Constrained Hidden Markov Models. Advances in neural information processing systems, 2000.

[4] T. Kolarik, "Use SPRT as stopping condition," online: http://stat.columbia.edu/ jcliu/paper/GSPRT_SQA3.pdf

[5] Igl, Maximilian, et al. "Deep Variational Reinforcement Learning for POMDPs," JMLR: http://proceedings.mlr.press/v80/igl18a/igl18a.pdf

[6] Cassandra, A. R.; Littman, M. L.; and Zhang, N. L. 1997. Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes. In UAI.