Improve usage of unsupervised data for the definition of RUL-based maintenance policies

Davide Angelani, Yellam naidu Kottavalasa, Usha padma Rachakonda[1]

[1]Department of Computer Science - Informatics and Engineering, Bologna, Italy
[2]Academic Year 2021-2022

Abstract

In the recent years, industries such as aeronautical, railway, and petroleum has transitioned from corrective or preventive maintenance to condition based maintenance (CBM). One of the enablers of CBM is Prognostics which primarily deals with prediction of remaining useful life (RUL) of an engineering asset. Besides physics-based approaches, data driven methods are widely used for prognostics purposes, however the latter technique requires availability of run to failure datasets. In this project we show a method to inject external knowledge into the deep learning model to exploit data about the normal functioning of the machines. We apply our approach on the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) model developed at NASA.

1 Introduction

Remain useful Life refers to the remaining time of a machine before it requires a replacement. The prediction of the RUL helps to delay maintenance operations until they are really needed. RUL predictions have the potential to prevent critical failures, and hence, becomes an important measurement to achieve the ultimate goal of zero-downtime performance in industrial systems. Therefore, it leads the company to save money by optimizing operational efficiency and avoid unplanned downtime. To produce robust and trustworthy prediction we need a lot of data about the lifespan of the machine until its failure. The problem is that to produce data in large quantity is generally expensive and time consuming. Despite the small amount of run-to-failure experiments, we have a lot of data about the normal functioning of the machine. We propose a semi-supervised approach for estimating the Remain Useful Life. In particular, we inject external knowledge into the model by the mean of constraints. In this way we train our model using the traditional loss function with the supervised data and apply the constraints on the unsupervised data. The report is divided in this way: first we discuss about the theory behind our method by showing how the Semantic-based regularizer is defined and how it is optimized by the mean of the Lagrangian Duality Framework, then we show the property of the RUL curve we want to enforce and the experiments on the CMAPSS data set we carry out. Finally we discuss about the results and the conclusion we draw.

2 Semantic-based regularizer

To exploit our external knowledge about the rul estimation, we need a way to model the constraint, so that we can inject it into our model. We use the method proposed by Diligenti, Gori, and Saccà 2017. This novel approach consists of extending the learning task. In this manner, we define a new loss function composed of the traditional loss function plus the penalty term λ which measure the penalty introduced by violating the constraint.

$$f_\lambda(y) = f(y) + \sum_{i=0}^{m} \lambda_i g_i(y) \tag{1}$$

Applying this custom loss function defined by the equation 1, we can exploit the external knowledge we are trying to inject in a semi-supervised fashion. We can use the supervised data to compute the traditional loss and both supervised and unsupervised data with the regularizer which quantify the constraints violation.

3 Lagrangian Dual Framework

Since using a validation set to choose the right regularizer term is not a viable option due to the scarcity of supervised data, we follow the approach proposed in Fioretto et al. 2020. They use the Lagrangian Duality to optimize the penalty term. If we consider the following optimization problem

$$min f(y) \qquad \text{subject to} \qquad g_i \leq 0 \qquad \forall i \in 1, \ldots, m \tag{2}$$

The Lagrangian relaxation of the problem is

$$f_\lambda(y) = f(y) + \sum_{i=0}^{m} \lambda_i g_i(y) \tag{3}$$

If we take the maximum with respect to λ of this function we recover the original problem 2. So our optimization problem become

$$\max_{\lambda} f_\lambda(y) \tag{4}$$

We still need to solve the original problem in which we want to minimize $f(y)$. This mean we have to find

$$\min_{x} \max_{\lambda} f_\lambda(y) \tag{5}$$

Then, if we reverse the order of maximisation we get the dual function

$$\max_{\lambda} \min_{x} f_\lambda(y) = \max_{\lambda} \hat{f}(\lambda) \tag{6}$$

where $\hat{f}\lambda = \min_{x} f_\lambda(y)$. In this way we can get the strongest Lagrangian relaxation of the original problem, which consists in finding the best Lagrangian multipliers λ.

4 Monotone behaviour

Defined the theory behind our model we can now model our constraint which exploit a property of the RUL curve. In particular, after each time step, we can notice that the RUL value decrease

by a number of units corresponding to the time step elapsed. So, after T time step, the RUL will have decreased by T unit. Therefore, the RUL preserves a non-increasing behaviour and we can exploit it by defining the constraint

$$f(\hat{x}_j, w) - f(\hat{x}_i, w) = j - i \qquad \forall i, j = 1 \dots m c_i = c_j \tag{7}$$

We can notice that we are considering pairs of contiguous sample, so we need to apply our constraint on the predictor itself. Since we are not interested at a strict satisfaction, we can use a relaxation of it. Therefore, given the function 1. Our constraint becomes:

$$\lambda(f(\hat{x}_i, w) - f(\hat{x}_j, w) - (j - i))^2$$

5 Experiments

In this section first we describe the data set on which the experiments are carried out and how we manage to simulate the scarcity of the supervised data. Then we show our baseline model and the experiments themselves.

5.1 Dataset

Commercial Modular Aero-Propulsion System Simulation (C-MAPSS), which was developed by NASA Saxena and Goebel 2008. The CMAPSS dataset includes 4 sub-datasets that are composed of multi-variate temporal data obtained from 21 sensors. Each sub-dataset contains one training set and one test set. The training datasets include run-to-failure sensor records of multiple aero-engines collected under different operational conditions and fault modes. Each engine unit starts with different degrees of initial wear and manufacturing variation that is unknown and considered to be healthy. As time progresses, the engine units begin to degrade until they reach the system failures, i.e. the last data entry corresponds to the time cycle that the engine unit is declared unhealthy. On the other hand, the sensor records in the testing datasets terminate at some time before system failure, and the goal of this task is to estimate the remaining useful life of each engine in the test dataset. For verification, the actual RUL value for the testing engine units are also provided.

Dataset	FD001	FD002	FD003	FD004
Training Trajectories	100	260	100	248
Test Trajectories	100	259	100	249
Operating Conditions	1(sea level)	6	1(sea level)	6
Fault Modes	HPC	HPC	HPC, Fan	HPC, Fan

Table 1: CMAPSS dataset details

The data is arranged in an N $\times$ 26 matrix where N is the number of data points in each subset. The first two variables represent the engine and cycle numbers, respectively. The following three variables are operational settings which correspond to the conditions in Table 1 and have a substantial effect on the engine performance. The remaining variables represent the 21 sensor readings that contain the information about the engine degradation over time.

5.1.1 Dataset generation

To simulate the scarcity of the data at various level, we define a series of ratios in which the data set will be splitted in supervised training set, unsupervised training set and test set. The partitions are defined by the mean of the machines, in this way we maintain together the sample about the same machine. The size of the test set is fixed to 12% for all the experiments. The table 2 shows how we have partitioned the data set.

%	no.of supervised machine/samples	no.of unsupervised machine/samples	no.of test machine/samples
3%/ 75%	7/1548	186/45470	56/14231
23%/ 55%	57/13367	136/33651	56/14231
43%/ 35%	107/25853	87/21505	55/13891
63% / 15%	156/38215	37/8803	56/14231

Table 2: The table shows the number of machine and sample divided in supervised set, unsupervised set and test set for each split of the data set

5.2 Baseline model

We have defined our baseline model as a network of two hidden layer of 32 neurons. The figure 1 show our model.

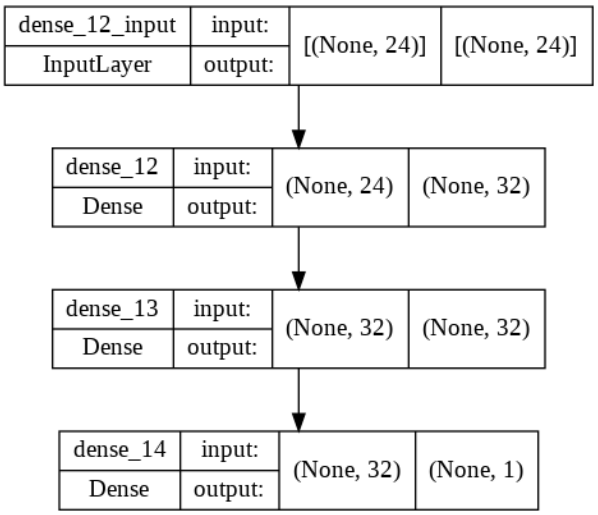


Figure 1: Summary of our baseline model

5.3 Experiments

For each split, shown in 2, we have trained four different model. The first one is the baseline model, the second one exploit the constraint defined with single fixed multiplier equal to 1, the third use the same loss function as the second but the multiplier is learned using the Lagrangian dual approach. Finally, the fourth model is trained using the same loss function as third model, but we we introduce a multiplier for each of the constraint. This mean that, given a batch of consecutive machine sample, each pair has their own regularizer. While in third model the regularizer is computed as the average of each constraint violation.

6 Results

In this section we show the results we have obtained.

6.1 Cost Model

The goal of RUL estimation is not to achieve the best possible accuracy but to save as much as possible by delaying the maintenance operation when they are really needed. So, for a proper evaluation of our model we define a cost model. Given that whenever a turbine operates for a time step, we gain a profit of 1 unit, a failure costs C units and we never trigger maintenance before s time steps, the time step when we trigger maintenance is given by

$$\min\{i \in I_k | f(x_{ki} < \theta\}$$

a failure occurs if

$$f(x_{ki}) \geq \theta$$

where x_k is the time series for a machine k and I_k is its set of time steps. The whole cost formula is

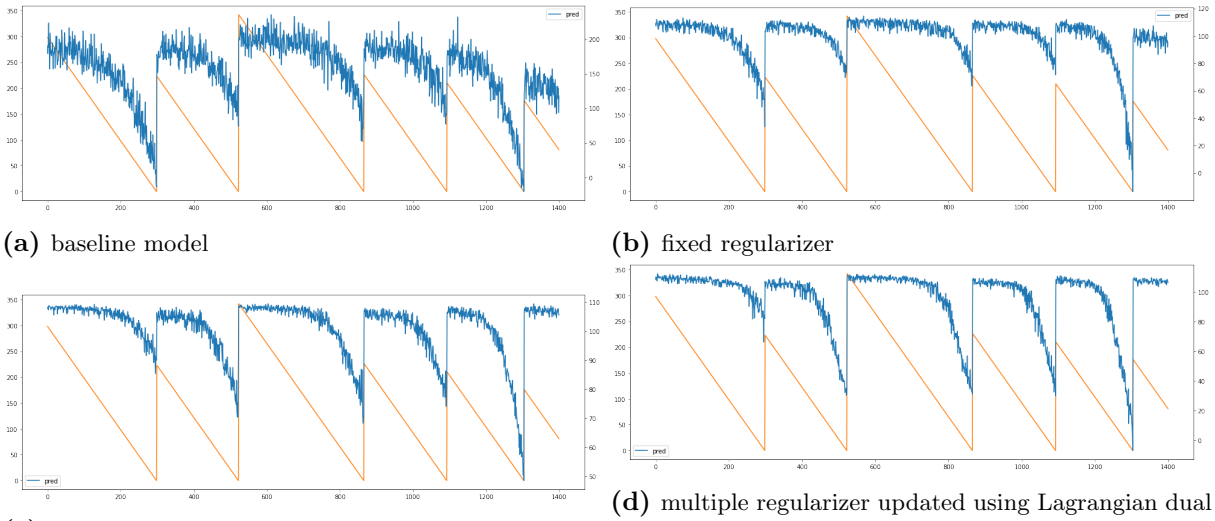
$$cost(f, \mathbf{x}, \theta) = \sum_{k}^{m} \text{op\_profit}(f(x_k), \theta) + \text{fail\_cost}(f(x_k), \theta) \tag{8}$$

where

$$\text{op\_profit}(f(x_k), \theta) = -\max(0, \min\{i \in I_k | f(x_{ki} < \theta\} - s)$$

$$\text{fail\_cost}(f(x_k, \theta) = \begin{cases} C \text{ if } f(x_{ki}) \geq \theta & \forall i \in I_k \\ 0 \end{cases}$$

6.2 3% supervised 75% unsupervised



(a) baseline model

(b) fixed regularizer

(c) regularizer updated using Lagrangian dual approach approach

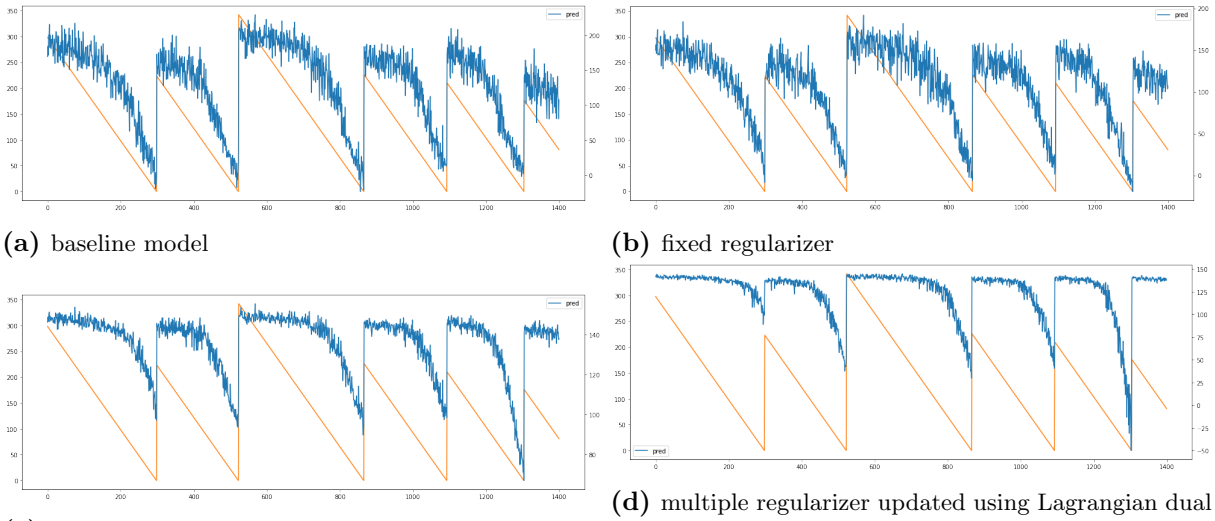
(d) multiple regularizer updated using Lagrangian dual

Figure 2: Accuracy on the test set for the split 3% supervised and 75% unsupervised

	base	fixed multiplier	updated regularizer	multiple regularizer
cost	-417/35952/9971	-361/36747/11817	-29/-2807/-1513	-51/-3903/-1826
average fails	0.00/0.44/0.41	0.00/0.44/0.46	0.00/0.05/0.04	0.00/0.03/0.02
average slack	9/8/10	17/8/7	99/91/97	74/99/110

Table 3: Cost model for the split 3% supervised and 75% unsupervised

6.3 23% supervised 55% unsupervised



(a) baseline model

(b) fixed regularizer

(c) regularizer updated using Lagrangian dual approach approach

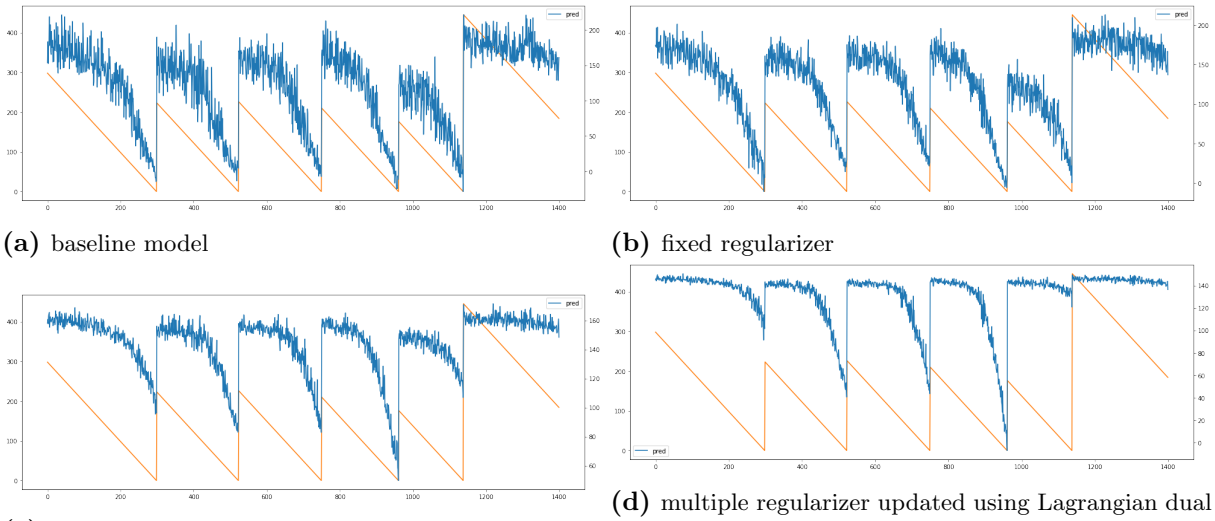
(d) multiple regularizer updated using Lagrangian dual

Figure 3: Accuracy on the test set for the split 23% supervised 55% unsupervised

	base	fixed multiplier	updated regularizer	multiple regularizer
cost	-5389/-11835/-5551	-4853/-13490/-5431	-3834/-9514/-4864	-1132/-4224/-1788
average fails	0.00/0.02/0.02	0.02/0.00/0.02	0.02/0.01/0.00	0.02/0.01/0.00
average slack	18/17/14	18/18/17	36/39/38	106/107/127

Table 4: Cost model for the split 23% supervised 55% unsupervised

6.4 43% supervised 35% unsupervised



(a) baseline model

(b) fixed regularizer

(c) regularizer updated using Lagrangian dual approach approach

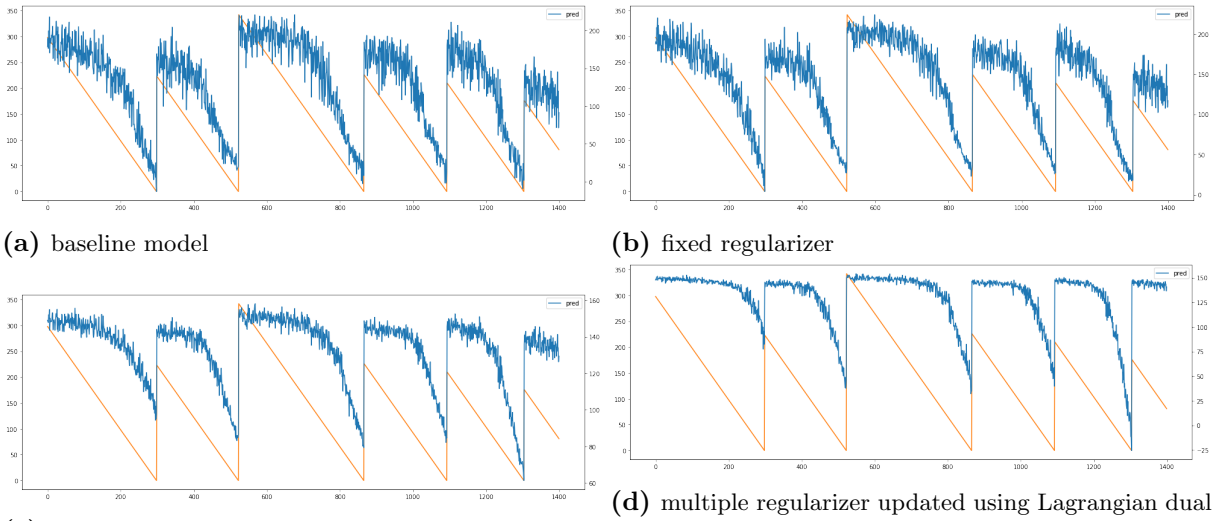
(d) multiple regularizer updated using Lagrangian dual

Figure 4: Accuracy on the test set for the split 43% supervised 35% unsupervised

	base	fixed multiplier	updated regularizer	multiple regularizer
cost	-10561/-9053/-5988	-10141/-7891/-5201	-8545/-6991/-4753	-4099/-4108/-2801
average fails	0.00/0.00/0.00	0.00/0.01/0.02	0.00/0.00/0.00	0.00/0.00/0.00
average slack	14/17/14	18/22/17	72/80/80	23/18/20

Table 5: Cost model for the split 43% supervised 35% unsupervised

6.5 63% supervised 15% unsupervised



(a) baseline model

(b) fixed regularizer

(c) regularizer updated using Lagrangian dual approach approach

(d) multiple regularizer updated using Lagrangian dual

Figure 5: Accuracy on the test set for the split 63% supervised 15% unsupervised

	base	fixed multiplier	updated regularizer	multiple regularizer
cost	-14371/-3029/-5229	-15200/-3857/-6140	-13018/-3122/-5170	-11625/-2802/-4550
average fails	0.00/0.00/0.00	0.00/0.00/0.00	0.00/0.00/0.00	0.00/0.00/0.00
average slack	24/18/20	18/15/15	32/35/32	41/43/43

Table 6: Cost model for the split 63% supervised 15% unsupervised

6.6 Discussion

The figures 2, 3, 4 and 5 show the predictions compared to the true value we have obtained for the various split and the various model. As we can see the predictions obtained with the baseline model (figures 2a, 3a, 4b and 5b) are the noisiest with the highest wobble while the ones obtained with the model trained with the multiple regularizer (figures 2d, 3d, 4d and 5d) are the least noisy. This mean that the method we are using help the model to learn the property of the RUL curve. Then we apply the cost model to each model. Given the predictions on the supervised training set, we compute the threshold θ which minimize the costs. As we can see from the table 3 the baseline model is the worst, with the highest cost on the test set and an high percentage of fails. The third and forth model, on which we apply the constraint, are the ones with the best performance, with the lowest cost and percentage of failure, despite the high average of remaining days to the end-of-life of the machines. Increasing the number of supervised samples available helps the baseline model to get better results and outperform the other models (4, 5 and 6). The problem is that the models on which we apply the constraint tends to have an higher average of remaining days.

7 Conclusion

In this project we show a semi-supervised approach for estimating the Remain Useful Life. The method is based on injecting external knowledge into the model by the mean of constraints using a semantic based regularizer. After defining our learning task expanded with the regularizer, we show how the penalty term is optimized to minimize the constraint violation.
Then we have evaluated this approach on the CMAPSS data set, using multiple ratio supervised/unsupervised for simulating the scarcity of the data. The results we have obtained show that the model with the injected knowledge outperform the baseline model when the percentage of supervised data is very low (3%), while it has poor performance when the percentage of supervised data is higher. The problem is the high average of remaining days to the end-of-life for each machine, this is probably due to the too strictly satisfaction of the constraints and need further investigation.

References

Diligenti, Michelangelo, Marco Gori, and Claudio Saccà (2017). "Semantic-based regularization for learning and inference". In: *Artificial Intelligence* 244. Combining Constraint Solving with Mining and Learning, pp. 143–165. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2015.08.011. URL: https://www.sciencedirect.com/science/article/pii/S0004370215001344.

Fioretto, Ferdinando et al. (2020). "A Lagrangian Dual Framework for Deep Neural Networks with Constraints". In: *CoRR* abs/2001.09394. arXiv: 2001.09394. URL: https://arxiv.org/abs/2001.09394.

Saxena, Abhinav and Kai Goebel (2008). "Turbofan engine degradation simulation data set". In: *NASA Ames Prognostics Data Repository*, pp. 1551–3203.