Transfer Learning and Domain Adaption

Anshul, Isha, Poorvi, Nutan

January 2021

1 Meeting logs

Jan 9, 2021

1.1 Paper 1

A theory of learning from different domains, by Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, Jennifer Wortman Vaughan. This introduced the theoretical notion of transfer learning and gave a mathematical language for speaking about theoretical bounds on transfer learning.

Question. Consider this recent paper A Theory of Universal Learning by Olivier Bousquet, Steve Hanneke, Shay Moran, Ramon van Handel, Amir Yehudayoff which takes a modern view at PAC learning and proves some bounds. Can we use ideas from this paper and the above paper on transfer learning to give a modern theoretical view of transfer learning?

1.2 Paper 2

In domain adaptation we may have $P(X_S) \neq P(X_T)$, but $P(Y_S|X_S) = P(Y_T|X_T)$. In transfer learning $P(X_S, Y_S) \neq P(X_T, Y_T)$.

Question Can idea of reduction work for transfer learning? Concretely, suppose we have source and target tasks S and T such that $D_S \subseteq D_T$. In fact say, T has exactly 1 extra input feature compared to S. In this case, can we say that if the newly added feature is "close" to some of the existing features, then transfer learning (without much additional training) can work. Towards this, we would like to do the following.

- Take some older data set D_S from "input feature re-weigting" papers.
- Build models based on them. Call this model M_S.
- Now, create a manufactured data set, say D_T, which has 1 new feature.
- Check: If the newly added feature is "close" to the existing features, then does transfer learning work?

Jan 16, 2021

Paper on What is being transferred in transfer learning? by Behnam Neyshabur, Hanie Sedghi, Chiyuan Zhang, being read and presented by Isha.

- Get the data sets and code for the paper.
- Confirm the observations from section 3.1.
- Check whether we can analyse feature reuse using blanking out of blocks instead of shuffling?
- Come up with other tests that complement sections 3.2 to 3.5.

Can we come up with meaningful ways of understanding transfer learning through NLP projects?

[Poorvi to plug in the details of the paper about CMD and Anshul to plug in the details about the paper he mentioned.]

Papers read by Poorvi:

- Central Moment discrepancy for Domain Invariant Representation Learning by [**?**]
- Moment Matching for Multisource Domain Adaption[**?**]

The paper mentioned by Poorvi comes with its repository and data.[**?**]. A collection of relevant journals and code bases can be found here. [**?**]

- Get it working by the next time (if possible).
- See if CMD helps when we have one or more new input columns to learn.

Paper presented by Anshul - Learning with previously unseen features [**?**]. The paper introduces a problem statement very close to the reduction problem mentioned, by considering a prediction task on an array of sensors such that the target array has a few extra sensors. The paper proposes a self-supervised kind of approach to the task, however, we could leverage the datasets used and related work in our solution.

Jan 23, 2021

Papers presented by Anshul -

- A theory of universal learning [**?**] - This paper tries to modify the "worst case" learning rates presented in PAC learning to a more practical setting. Learning rate here refers to the relation between the number of train samples and the expected loss. They argue that the PAC learning bounds are too restrictive since they require universal constants. Further, under the PAC paradigm the worst case distribution might be different for each sample size, which is not practically useful. Instead we should fix a distribution and look at the bound according to it. They propose an alternate method of characterizing learning bounds through data/class dependent constants, and show that under this new method better learning rates are obtained. The crux of their bound involves computing a VC-L tree, a hybrid of a Littlestone tree where each node is a set, hence the depth will have a relation to the VC dimension as well (since the function class needs to shatter the sets at each level). Based on whether or not such a tree is infinite for a given class of functions, the authors give bounds on the learning rates.
  Questions Proposed - Need to look at how exactly the distribution dependence is captured in the proof of learning rate bounds with the VC-L tree, since we can then modify such proofs for the case of transfer learning.

- Minimax Lower Bounds for Transfer Learning with Linear and One-hidden Layer Neural Networks [**?**] - The paper talks about a lower bound on the best worst case test performance of transfer learning for neural networks. The quantity under consideration is close to the quantity under consideration from [**?**], and we also notice a similarity in the learning curves plotted.
  Questions Proposed - Taking inspiration from the bounds proposed in this paper can we come up with a more general quantity we would like to obtain performance guarantees on using the universal learning framework? At the outset certain things seem to be different in the two approaches, mainly the fact that universal learning looks at all possible distributions while fixing a concept class, while we may be interested in having a fixed target distribution and certain source distributions which are close to it etc.

  We propose getting learning rate bounds on the following objective as a -

  $$inf_{\hat{h}_n} sup_{P_t \in \Delta} \mathbb{E}_{P_s^{n_s}, P_t^{n_t}} [\mathbb{E}_{x \sim P_t} l(\hat{h}_n(x), f(x))]$$

  This is akin to saying that we fix a source distribution and then consider target distributions which are "close" to the source distribution based on some distance measure, and then we look at the best worst case learning rates of the target distribution error as a function of the number of samples of source and/or target data. From a practical perspective this makes sense since in usual transfer learning scenarios the source distribution (in fact

the source model) is fixed and we try to investigate how well knowledge can be transferred to other domains.

Jan 23, 2021

Poorvi -

- CMD and MMD [**?**] - The benefit of moment matching only till a few moments instead of like what's done in MMD. Office dataset, amazon dataset, MNIST datset.
  Papers read: Central Moment discrepancy for Domain Invariant Representation Learning[**?**]; Moment Matching for Multisource Domain Adaptation [**?**];
  Relevant parts of Robust Unsupervised Domain Adaptation for Neural Networks via Moment Alignment[**?**] and Unsupervised Domain Adaptation with Adversarial Residual Transform Networks[**?**]
  Questions Proposed Use this to get couloured MNIST dataset with a different background colour added as a new feature.

- Bert models. Fine tune to get a pre-trained model working for our task of transfer learning.

Jan 23, 2021

Isha - Time spent on analysing and running the code of the paper[**?**] [**?**]. Stuck in problems with GPU to run on a large dataset like image-net. Shifted to mnist dataset, as small datatset to train for.

Jan 30,2021

Anshul -
TODO - Get a statement about transfer learning that we would like to prove. It should be in the framework of the universal learning paper, ideally should consist of getting a bound on the learning rate of a target distribution given a fixed source distribution, and the conditions under which these learning rates hold. Since the seminal Shai Ben-David paper was able to get bounds for transfer learning using the PAC/VC dim framework, we might be able to get an analogue using the universal learning framework.

Jan 30, 2021

Poorvi - **TODO**- Get the CMNIST Data ready. Train a resnet for MNIST dataset. Use this trained resnet to test for the (MNIST data added with a few cmnist data).
Also, try perturbing the last layer and retraining with the first few (50-100 or 10% say) cmnist images added to the mnist data. Keep adding more and see the results.

CMNIST training

- Trained for 2 MNIST class-classification 1 and 7.
- Got very great test accuracies 99.722.
- Code for generating colored mnist images using RGB backgrounds.
- Added 100 images in the tra

Results

Jan 30, 2021

Isha - Code [**?**] Modified the code for mnist data, just some connectivity issues left to resolve.

Feb 7, 2021

TODO- Tvarit wanted us to work with tabular data

1.2.1 Anshul

Notation & Setting - Let $\mathcal{H}$ be a hypothesis class. Let our source and transfer domains be $P_S = (\mathcal{D}_S, f_S)$ and $P_T = (\mathcal{D}_T, f_T)$, where f is a binary labelling function. Let $l(y_1, y_2)$ be a loss function $(0-1$ loss) incurred for the labels. We also use the $\mathcal{H}\,\Delta\,\mathcal{H}$ divergence as defined by Ben-David et. al. as $d_{\mathcal{H}}(\mathcal{D}, \mathcal{D}') = 2\sup_{h\in\mathcal{H}} |Pr_{\mathcal{D}}(h(x)) - Pr_{\mathcal{D}'}(h(x))|$. We fix our source distribution to be P_S, and define $\Delta_S = \{P_T : d_{\mathcal{H}\,\Delta\,\mathcal{H}}(P_T, P_S) \leq \epsilon\}$. We also have a learning algorithm which outputs $\hat{h}_{n_S,n_T}$ after looking at some source and target samples
Form of inequality - Then, following the universal learning framework, our inequality is of the form -

$$\forall P_T \in \Delta_S \exists C, c \mathbb{E}_{S,T} \mathbb{E}_{x\sim P_T}[l(\hat{h}_{n_S,n_T}(x), f_T(x))] \leq C R_S(cn_T, n_S)$$

where R_S is a function of the number of source and target samples. In the Ben-David framework, this term also
Assumptions - The universal learning framework depends on realizability, while the transfer learning theory usually doesn't. In this case, we extend realizability to define a P_S, P_T to be realizable for a hypothesis class $\mathcal{H}$ if $f_S \in \mathcal{H}$

and $f_T \in \mathcal{H}$. This is consistent with the assumption in the [**?**] work, where the source and target are taken to be NN. An interesting question then is how does the joint optimal [**?**] function change in this case.
We may also need to refine our definition of $d_{\mathcal{H}}$ to be deterministic, as we see later.
Prior Work forms- Most prior work has not really concerned itself with learning algos, instead giving general bounds on the best achievable error for any hypothesis on the target domain given its error on source domain. Translated into this setting, the learning bounds have been of the form $\sqrt{\frac{log(m)}{m}}$, or $(\frac{C}{m})^x$.
Proof sketch in Universal Learning - The overall idea of the proof sketch of exponential learning rates involves looking at the finiteness of Littlestone trees, using that to get a winning strategy for the learner/adversary, and the winning strategy also gives a way to learn a classifier. The learner's strategy aims to methodically restrict the set of valid hypothesis till only one remains, by outputting the opposite of what the Littlestone tree tells it to. The constructed algo first gets an estimate for the number of examples needed for getting a small error with 1/4 probability, then they divide their train data into n/t batches and claim that a majority voting algo will suffice for exponential learning rate (see paper). We hence need to define a similar adversarial strategy for our scenario
Modifications in tools- We propose a modified Littlestone tree for the TL scenario.
As a motivation, we see that the proof of exponential learning rate relies on the Littlestone tree to come up with an adversarial strategy, and consequently a learning algorithm which is guaranteed to give zero error on a large enough sample in a probabilistic realizable case (Lemma 4.3 of the paper). Hence, we define our TLL tree as a normal Littlestone tree with nodes and labels such that for each path $(x_1, y_1, x_2, y_2, ..., x_n, y_n)$, $\exists i, h_1, h_2$ s.t. $\forall k \leq i, y_k = h_1(x_k)$ and $\forall k > i, y_k = h_2(x_k)$. In other words, on each path we are allowed to switch functions exactly (or upto?) once. (We may need to fix a single i for a tree, since that represents the transition between source and target.

Notice that an infinite Littlestone tree implies the existence of an infinite TLL tree, while the converse may not be true if i is not fixed?
For a second modification, we look at another variant such that for each path $(x_1, y_1, ..., x_n, y_n)$ we define a set $\{h_1, h_2, ..., h_k\}$ functions where the maximum number of discrepancies between any two functions is p, such that for each i in the path, there is atleast one function h consistent with the label i. **Strategy**- The strategy for a Littlestone game begins by defining $\mathcal{H}_{x_1,y_1,x_2,y_2,...,x_t,y_t} = \{h : h(x_i) = y_i \forall i \leq t\}$. Then, the learner tries to make sure that $\mathcal{H}_{x_1,1-\hat{y}_1,x_2,1-\hat{y}_2,...,x_t,1-\hat{y}_t}$ is empty, so that the only feasible labelling is $(x_1, \hat{y}_1, x_2, \hat{y}_2, ..., x_t, \hat{y}_t)$, where $\hat{y}_i$ is the learner's learnt guess at time i. The learner essentially goes one level down the tree and changes its function each time it makes a mistake. The strategy is successful due to the realizability and finiteness of Littlestone tree.
Can we naively extend this strategy to define $\mathcal{H}_{x_1,y_1,x_2,y_2,...,x_t,y_t} = \{(h_1, h_2) : h_1(x_k) = y_k \exists i \forall k \leq i, h_2(x_k) = y_k \forall k > i\}$. Our learning algo may actually then

have to do more than what is required, since for the purpose of TL we only care that the second function is found, but our algo will have to look at h_1, h_2, i.
Can we also leverage the fact that the two domains have a bounded $d_{\mathcal{H}}$, and connect it with the maximum number of different predictions that h_1, h_2 might make? For example, each time you make a mistake you can descend the tree or switch the function, which would work upto some k mistakes after which you have to switch in any case? This may make the algo more involved, with some backtracking as well...
Questions to discuss / PoA

- Come up with a strategy to win the game for the learner for the finite TLL case. Also come up with a batch learning type algo using this strategy's classifier. The strategy may have to be probabilistic if the $d_{\mathcal{H}}$ is used. We need to look more carefully at the strategy already described in the universal learning paper's section 4, to come up with our own strategy and structure in tandem
- Prove an equivalent of Lemma 4.3 of universal learning, i.e. that the error on test set goes to 0 as number of samples go to infinity.
- Finally get learning rates for TLL tree cases.

1.2.2 Poorvi

get model after black and white images. and then train it with the colored one separately (no shuffling)

test the current network with colored images first.

testing–¿ ¡8-% then separate training. bg redundant. mnist maybe an easy dataset.

did for cat dog example

last layer perturbations. not for now.

Nasa turbafan dataset. check it out.

Description: Engine degradation simulation was carried out using C-MAPSS. Four different were sets simulated under different combinations of operational conditions and fault modes. Records several sensor channels to characterize fault evolution. The data set was provided by the Prognostics CoE at NASA Ames. Format: The set is in text format and has been zipped including a readme file.

Tvarit: Ai solutions.

1.3 Isha

Written code to Blur the image for imagenet data to figure out the role of feature reuse after bluring the image instead of shuffling.

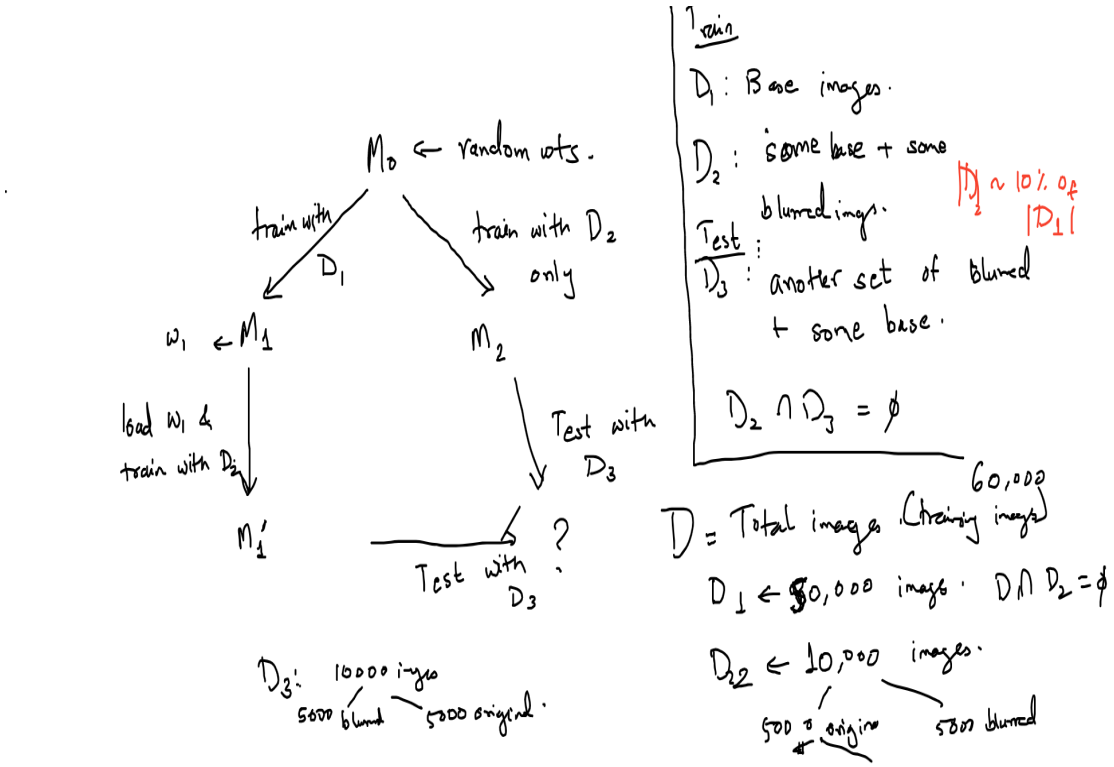
Feb 21, 2021

1.3.1 Isha

- Experimenting on blured-dataset(mnist) and the shuffled one(mnist).
- Model is simple linear model
- The (blured)data-set used to train contains about 60k images.
- Accuracies of rand-int and pre-trained weights does not show much difference. Link to accuracy:.

TO DO for next week:

- Take mixed dataset blured+base, for test and train
- Take small train data-set for each digit.
- try with a CNN architecture, so that it can capture the edges and lines nicely.
- Also capture the accuracies for each digit(percent of each image getting classified.)



1.3.2 Poorvi

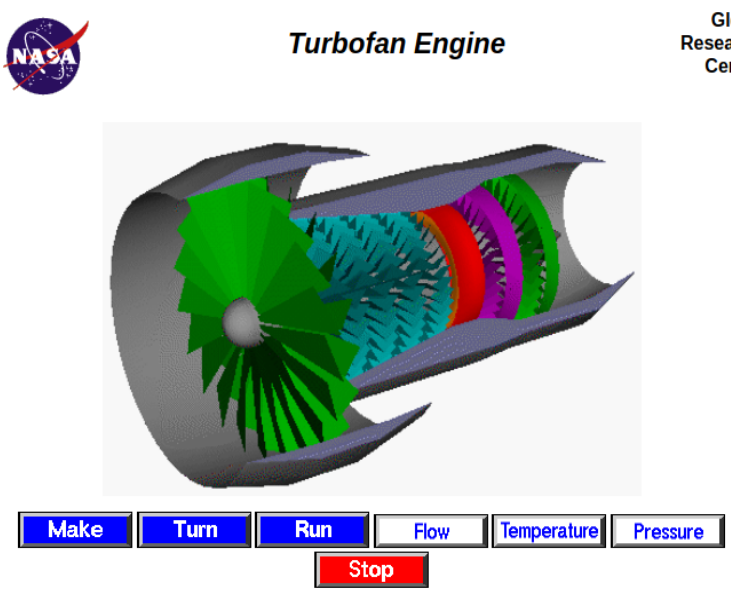
21-Feb-2021

did for cat dog example white dog vs white cat test it for black ones try for 10 classes mnist nasa turbofan data look into this

March 7, 2021

1.3.3 Poorvi

Turbofan is a jet engine in which a turbine-driven fan provides additional thrust.



A turbofan engine is the most modern variation of the basic gas turbine engine. As with other gas turbines, there is a core engine, which is surrounded by a fan in the front and an additional turbine at the rear. The fan and fan turbine are composed of many blades, like the core compressor and core turbine, and are connected to an additional shaft. All of this additional turbomachinery is colored green on the schematic. As with the core compressor and turbine, some of the fan blades turn with the shaft and some blades remain stationary. The fan shaft passes through the core shaft for mechanical reasons. This type of arrangement is called a two spool engine (one "spool" for the fan, one "spool" for the core.) Some advanced engines have additional spools for even higher efficiency.[**?**]

Jet Engine Remaining Useful Life (RUL) Prediction:

Aim: To perform predictive maintenance (PM) on a commercial turbofan en-

gine by a data-driven approach, i.e data collected from the operational jet engines is used to PM modeling. [**?**] We therefore need to build a model to estimate the RUL of a jet engine based on run-to-failure data of a fleet of similar jet engines.

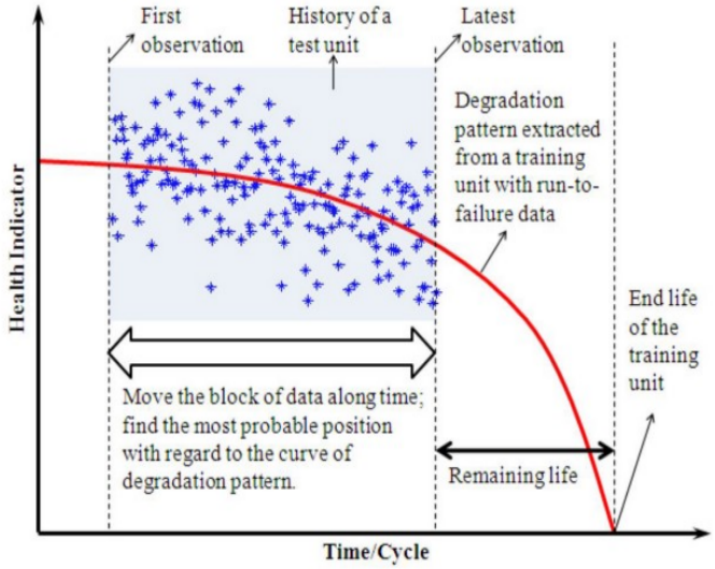
Dataset:

- The data set includes time-series measurements of various pressures, temperatures, and rotating equipment speeds that are feasible for the jet engine.
- All engines are of the same type, but each engine starts with different degrees of initial wear and variations in the manufacturing process, which is unknown to the user.
- There are **3** optional settings that can be used to change the performance of each machine: altitude (0–42K ft.), Mach number (0–0.84), and throttle resolver angle (TRA) (20–100).
- **6** different flight conditions were simulated that comprised of a range of values for these 3 operational conditions:
- Each engine has 21 sensors collecting different measurements related to the engine state at runtime =¿ 21 output attributes.
- Collected data is contaminated with sensor noise.
- Over time, each engine develops a fault, which can be seen through sensor readings. The time series data stops for each engine when a failure has occurred for that particular engine. Hence the actual RUL is known based on the length of the data.

Problem statement: To come up with a machine learning model to predict RUL based on time-series data of sensor measurements typically available from aircraft gas turbine engines.
Strategy:

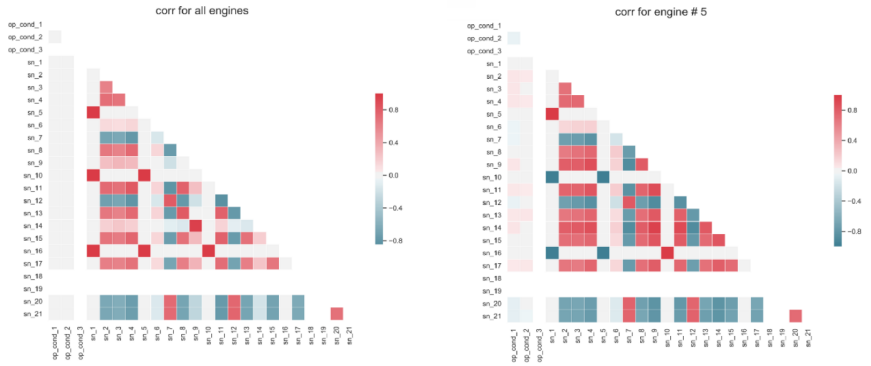
- Since the data is in a form of a time trajectory of many (21) sensor data, then there will be a need to fuse these sensors into a condition indicator or a health index that help in identifying the occurrence of a failure.
- the model in testing mode will compare how similar/ correlated the testing fused signal is to the training fused signal. Based on this similarity comparison, a prediction is made.
- Since the training data composed of run-to-failure trajectories, whereas the testing data contains trajectory up to undefined health state, then the training process will include training the model on a portion of the trajectory before the failure has occurred to simulate the real use of the model in online prediction mode.

- Therefore RUL of the testing unit is estimated based on the actual life of a training unit that has the most similar degradation pattern.

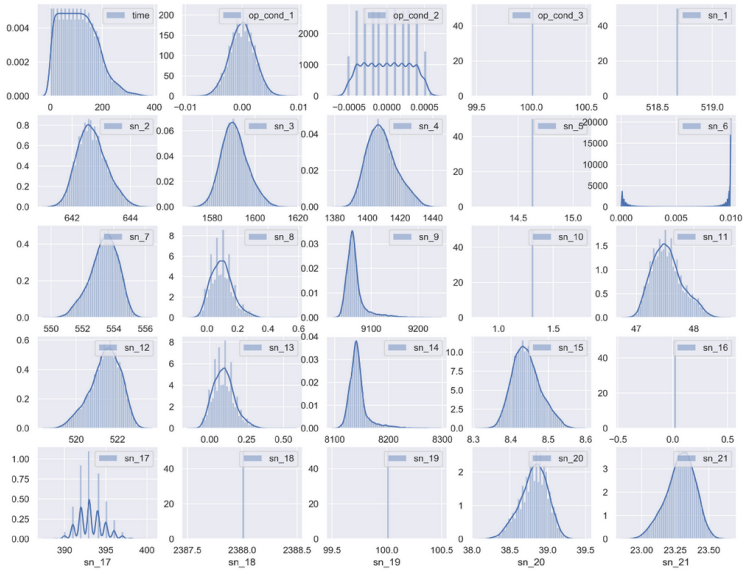


Data files

- consider a specific training file: 'train-FD001.txt'. this specific data set contains 100 run-to-failure engine simulation (corresponding to 100 different engine).
- the first column specifies the engine number, second represents the cycle number, third represents the operational condition in that cycle, and then there are 21 other columns each of which correspond to the 21 sensor measurements respectively.
- considering the correlation between the sensors



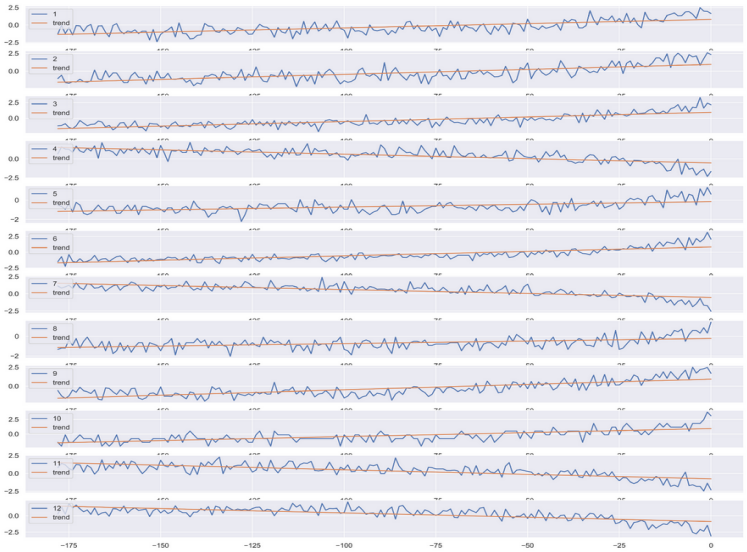
- From the first look, it seems that some sensors are very correlated with each other. This seems to hold for all engine together and a single-engine data as well. This might cause issues during modeling and I might need to delete/fuse some sensor data.
- Some columns need to be removed since they are all white. this might indicate non-changing values too. For instance the coorelation between any 2 of sn1, sn5, sn10, sn16 is 1; which means that I can remove some columns for the analysis
- If we look at the time series plot and distribution of various data columns



- main takeaways:
  - the distribution of almost all variables is single skewed gaussian
  - opcond2 and sn17 seem to be discrete variables and not continuous (maybe these need to be deleted to keep only continuously varying variables for modeling)
  - the observations above holds when plotting all engines and when the plot is made for a specific engine. (means all engine are very similar in their output response)
  - columns to be removed from analysis since they do not change with time: ['op_cond_3', 'sn_1', 'sn_5', 'sn_6', 'sn_10', 'sn_16', 'sn_18', 'sn_19']
  - extra columns to be removed because of high correlation is one of sn9 and sn14

Algorithm:
Linear Trending the trend is found simply by fitting a linear model to each sensor. I order then the sensors based on their absolute value of the linear slope (after normalization of the sensor values).

We need to check trend of the remaining sensors as a function of operating cycles. The higher trend probably means better predictability of events at the end of life for the engine.



the order of trend slope magnitude is ['sn_11' 'sn_4' 'sn_12' 'sn_7' 'sn_15' 'sn_21' 'sn_20' 'sn_17' 'sn_2' 'sn_3' 'sn_13' 'sn_8']

Dimensionality Reduction PCA and fusing sensors before fitting the model

PCA results considering the above 12 sensors:

First PC: 74%

second PC: 4.1%

third PC: 3.5%

now using a subset of the data that has only the highest 6 sensors in terms of the linear trend slope,PCA resullts:

first PC: 81.6%

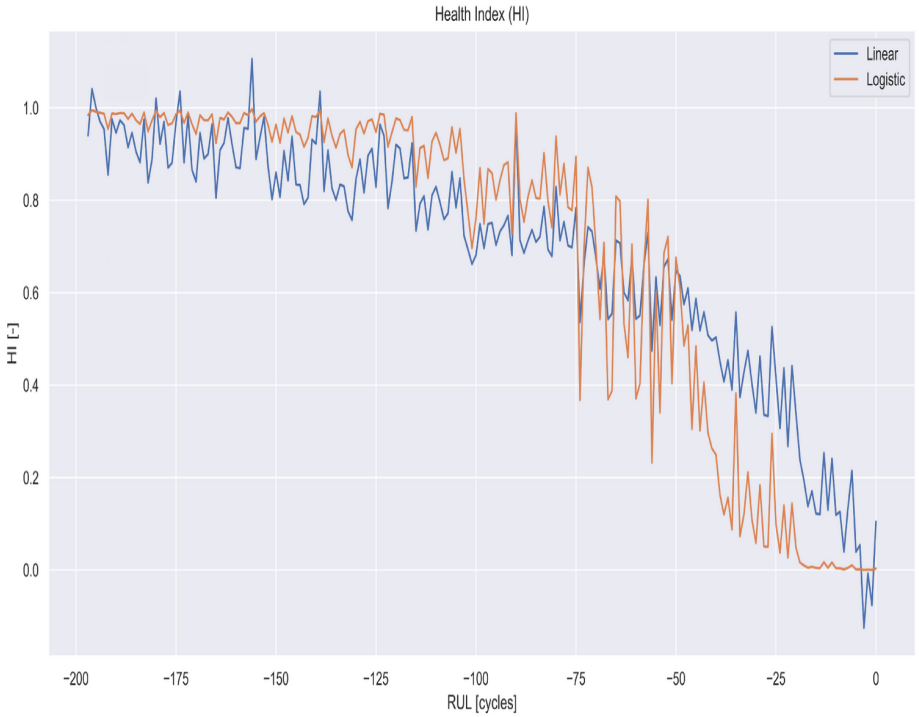
second PC: 5%

third PC: 4.4%

To create a fused health index (HI) sensor, first, we need to train the extreme data (data at the beginning of the engine cycle and at the end of engine cycle life).

beginning of life gets values of 1, while the end of life gets values of 0. so the model takes the sensor values and finds a fused signal that gives the health

indication HI.

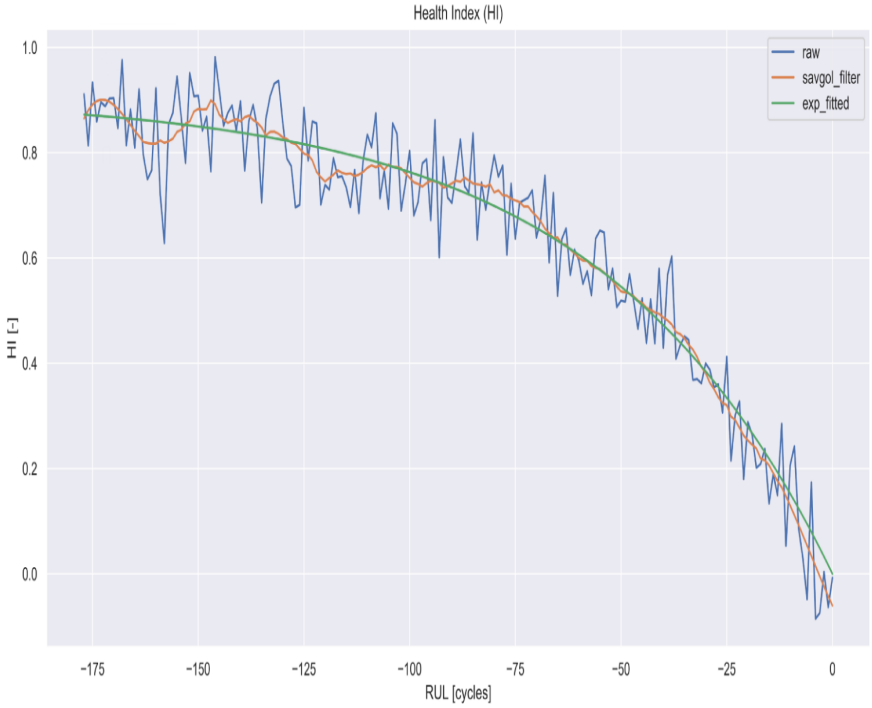


I'll call engine with prefect health as the one which has (RUL_hgih=300) or more cycles until its failures. zero health is considered as the last (RUL_low=5) cycles of each engine operation.

The above figure shows the results of fusing all sensors into one signal, the Health Index (HI). the result above is for a representative engine for fusion using linear and logistic regression model. HI can be obtained by other fusing techniques also.

Observation: Logistic regression give value strictly between [0,1] while the linear regression model does not guarantee that. Since the HI is noisy, a single smoothing will be applied Savitzky-Golay (Sav_gol) filter is used in the remaining analysis on the decided linear regression model.

Fitting the model: the way we are going to use the HI in the model is to create a simple exponential model y = a[exp(b*t)-1] = HI for each engine and then use this in predicting RUL using similarity between the fitted HI and the raw HI for new engine. the plot below shows the curve for exponentially fitted HI for a representative engine along with the raw HI and the filtered HI.



Predicting new engine RUL:

Inputs:

the prediction of new engine RUL takes as input:

1. a fused sensor with observation equals the current life cycle for the new test engine 2. fitted exponential model of the HI for all trained engines

the model then tries to compare the observation of the fused sensor for the new engine with the HI models (exponential fitted curves) (using sum of squared differences method)

14/3/21: Poorvi

time series as images 2 sensors 1 for training, new comes up, see correlation b/w them (see if nn can help get this by analysing the 2 sensor data..subquestion)

say it's not related. can we bypass HI here, and use the data from s2 to predict RUL.

reduce the problem to a transfer learning setup.

say we have 1 sensor. that itself acts like a health indicator. we plot sensor measurement for every cycle (average measurement of many similar engines). For predicting we best fit this curve with the present testing data and find the RUL by subtracting from the expected failure time.

now we add one more sensor, implying the plot is now a 3d plot where you plot both the sensor measurements for every cycle and then fit the testing data by reducing some kind of loss function. **P-value**: Null hypothesis states that there is no difference between 2 tests after changing some particular parame-

ter/ variable. P-value of that variable is defined as the probability of the null hypothesis being true. For the sensor to be relevant, the p value needs to be low and the trend needs to be high.

if this was extended to n already present features. adding one new feature would be like adding one more dimension to the already present HI indicator. The loss function that needs to be minimized would therefore be a function of just HI and the new introduced sensor. So, say the testing data was initially fit in a 2d curve, adding a new dimension introduces new errors wrt the new sensor.

If we know how the new sensor is gonna effect the health index using the trend concept which we saw earlier, we can just use that factor, which will be related to the p value of the new sensor for the function HI.

P-value: Null hypothesis states that there is no difference between 2 tests after changing some particular parameter/ variable. P-value of that variable is defined as the probability of the null hypothesis being true. For the sensor to be relevant, the p value needs to be low and the trend needs to be high.

So if we know the p value of the new sensor introduced, we can include that factor and now try minimizing the new loss function= sum over cycles [(HI(test) - HI(train))$\hat{2}$ + (Snew(test) - Snew(train))$\hat{2}$].

Coming to nn models one way to think of this as a transfer learning model is to consider these time series data as images. Every 21 senors' measurement datapoint at a particular cycle represents 1 particular image and the label given to it is the RUL at that particular cycle. (i.e failure time cycle - current time cycle). If a new sensor is introduced the image would be basically altered as it would now be corresponding to a 22 sized measurement. This might be considered as a new set of training images and the same nn model which was trained before could be used for testing the new set of images corresponding to 22 sensors.

Implementation(not yet tried) https://github.com/kpeters/exploring-nasas-turbofan-dataset/blob/master/$6_R eproducible_r esults_p rimer.ipynb$

https://towardsdatascience.com/predictive-maintenance-of-turbofan-engines-ec54a083127

https://towardsdatascience.com/tagged/exploring-nasa-turbofan

21 attributes correspond to 21 dimensional pt to be represented as an image. 100 of the continuous images with RUL x to x+99 belong to 1 class. Now for predicting the rul, we were finding the best fit (least mse loss) with the training set and then predicting the rul by subtracting from the last pt of overlap in the HI vs time cycle fit.

For images: 100 images belong to 1 class of RUL which is the average of the RUL value of the middle image. so say, there are n classes. For fitting we follow the exact same procedure as we did before and find the mse loss for 21 attributes (for now just taking the average of errors).

Adding a new sensor is like changing the image features and hence changing the entire dataset. Using the transfer learning setup we suppose make a classifier model for predicting the RUL in the first case we use the same model here. The

mse loss is different in this case and hence the best fit might happen somewhere else.

28/3/21 Wrote code for nn corrsponding to 21 sesnors and prediction of RUL.

11/4/21 Poorvi Paper: Encoding Time Series as Images for Visual Inspection and Classification Using Tiled Convolutional Neural Networks

- encode time series data as different types of images, namely, Gramian Angular Fields (GAF) and Markov Transition Fields (MTF)
-