Data-based process optimization (PO) and data-driven predictive maintenance (PdM)

Rufat Babayev

Data Scientist at evosoft GmbH
Group "Data Intelligence"

14.04.2023

Table of Contents

Goals

Modern machine tools deliver a large amount of data containing information about the manufacturing process. Using machine learning, the **various available data sources** can be efficiently combined and analyzed. However, solutions in this context have to fit a variety of different machines, so the designed algorithms and analytical solutions have to be **generally applicable**.

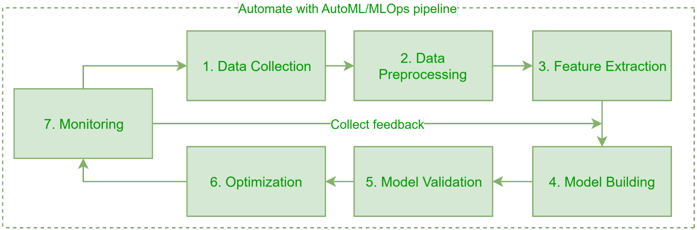
- During the project, the team will work on the next level of AI-based process optimization for machine tools.
- The team will be researching the **automated** evaluation of time series and image-based data.
- The team will then evaluate approaches to **predict** machining/processing conditions and optimize process parameters.
- For the final evaluation, the developed algorithms will be implemented on industrial machine tools.

Motivation

- For advanced systems such as airplanes, railways, power plants, and machine tools; process optimization and maintenance is a crucial concern because it ensures the systems' **efficiency, productivity, reliability, and safety** throughout their operational lifetimes.
- Leveraging the potential of advanced sensor capability, IoT technology, and data analytics algorithms, process optimization and maintenance in the age of Industry 4.0 has experienced a rapid shift from "reactive" to **"proactive"**: instead of performing optimization and maintenance only when failure has already happened, the state-of-the-art strategy is to proactively predict system failures and optimize/schedule necessary steps **"just-in-time"** [1]. Data-based and predictive strategies emerged.
- In turn, **data-based process optimization (PO)**, and **data-driven predictive maintenance (PdM)** became essential components of modern industrial systems, especially in the context of Industry 4.0.

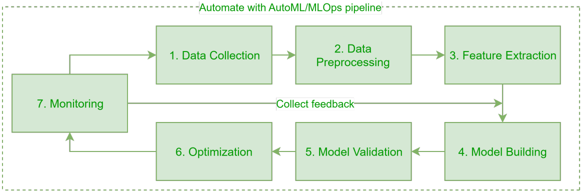
Data-based PO for machine tools (1)

Data-based process optimization in the manufacturing domain involves using machine-generated data to **improve machining processes (process efficiency and performance), enhance product quality, and reduce waste**. The key steps for optimizing manufacturing processes include:



Data collection: Get data from various sources, such as **machine logs, sensor data (e.g., temperature, vibration, and pressure), computer numerical control (CNC) programs**, and other process parameters.
Data preprocessing: Clean and preprocess the data to remove noise and handle missing values and inconsistencies.

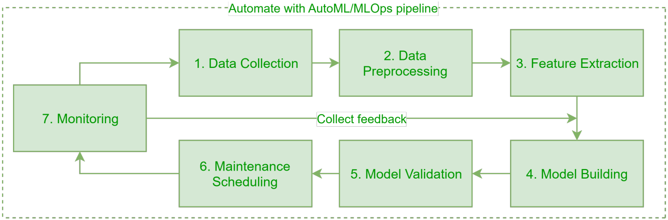
Data-based PO for machine tools (2)



Feature extraction: Identify relevant features and patterns in the data, such as **tool wear, cutting forces, and surface roughness**, which can provide insights into process efficiency and performance.
Model building: Develop ML models to predict and optimize process performance based on the extracted features. Some popular models for manufacturing applications include Random Forest, SVM, and **NNs**.
Model validation: Evaluate the accuracy and robustness of the models using **historical data or real-time data streams**.
Optimization: Apply the validated models to optimize process parameters, such as **cutting speed, feed rate, and depth of cut**, among others.

Data-driven PdM for machine tools

Data-driven PdM involves using data analytics and ML techniques to monitor the health of machine tools and **predict potential failures** before they occur. PdM can help **minimize downtime, reduce maintenance costs, and extend the life of machine tools**. The key steps in data-driven PdM for machine tools are:



Feature extraction: Identify features that can provide insights into the **health state and potential failure modes** of the machine tools.

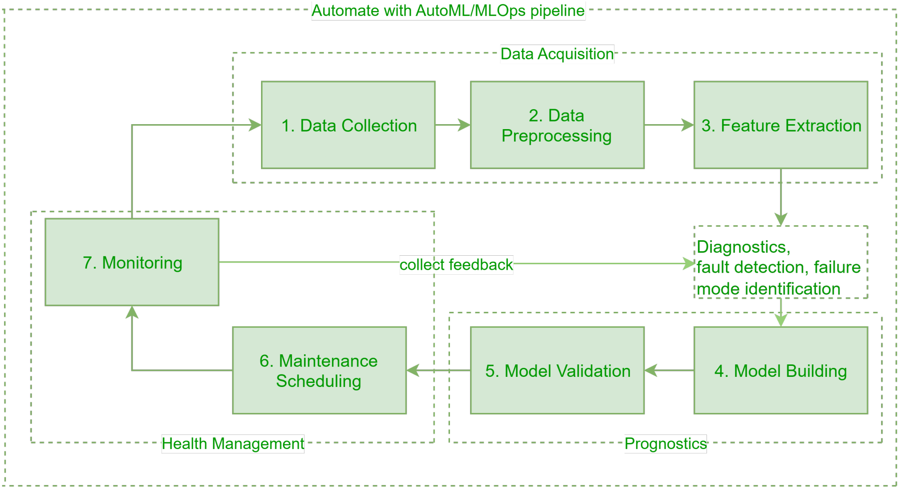
Maintenance scheduling: Use the predictive models to schedule maintenance activities based on **predicted tool health and remaining useful life (RUL)**.

Case Study: Estimate the system's "RUL" by PdM (1)

- In practice, PdM is typically achieved by first using sensors to monitor the system's health state constantly.
- Subsequently, data analytics algorithms are employed to predict the system's RUL based on up-to-date measurements.
- Finally, a maintenance schedule is devised to maintain the system in its originally intended function [1].
- The core of PdM is prognostic techniques, which enable predicting the degradation trend of an in-service system given the real-time measured data. Here, ML models are commonly built to identify the characteristics of the current health state of the system and to predict the remaining time until system failure occurs [1].
- Let's take a look at common data-driven models for prognostics.
  - First, we will briefly review the general steps involved in PdM.
  - Second, we will classify systems into categories based on their characteristics, and discuss popular ML models for each category.
  - Finally, we will talk about the challenges faced by delivering reliable prognostic analysis, and provide a discussion and conclusion.

Case Study: Estimate the system's "RUL" by PdM (2)

PdM's general steps include **data acquisition, diagnostics, prognostics, and health management**, as demonstrated in the diagram below.

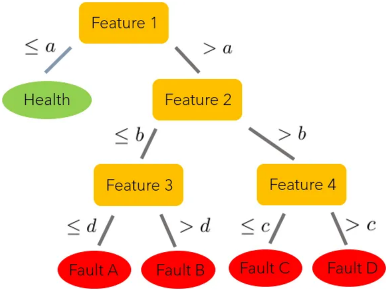


Case Study: Estimate the system's "RUL" by PdM (3)

Data acquisition:

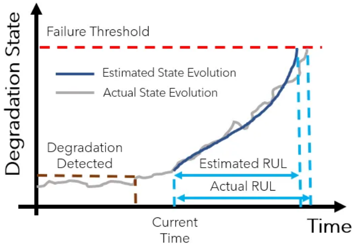
- We have briefly talked about data acquisition. Let's elaborate more on the feature extraction.
- To extract features, signal processing techniques are usually employed to transform raw data into features in a different domain (e.g., **time, frequency, time-frequency**). Since PdM mainly encounters non-stationary signals, **time-frequency analysis tools** are handy to extract features for diagnostic and prognostic purposes. Under this category, **short-time Fourier transform, wavelet package decomposition, empirical mode decomposition, and Hilbert-Huang transform** are the most popular approaches [1].
- An additional step after feature extraction is **feature reduction**. This is the case as the extracted features are usually too numerous to be exploited in practice. Popular dimensionality reduction methods, such as **principal components analysis (PCA), kernel-PCA, Isomap**, etc., are usually employed to eliminate redundant features.

Diagnostics: It focuses on determining the current health status of a system. It deals with fault detection and failure mode identification based on the extracted features from existing data. Fault diagnostics are usually formulated as a classification problem. As a result, popular classification methods, such as **k-nn, SVM, decision trees, RF**, are widely adopted to predict the system health state labels given the observed feature values [1]. A simple illustration of applying the decision tree model to classify the system failure mode is given below.



Case Study: Estimate the system's "RUL" by PdM (5)

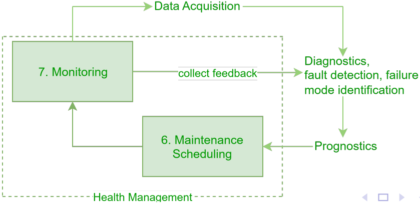
Prognostics: Predicts the monitored system's future state and estimate the system's RUL (i.e., how long it will take until system failure occurs) based on historical data and the current health status from diagnostics.



Since prognostics predicts the time at which a system will no longer perform its intended function, it provides users with the opportunities to **mitigate failure risk** while extending the system's useful life [1]. Naturally, data-driven approaches are heavily investigated to estimate the RUL. As a result, an array of ML strategies have been proposed for various use cases. Later, we will review some of them for RUL prediction. [1].

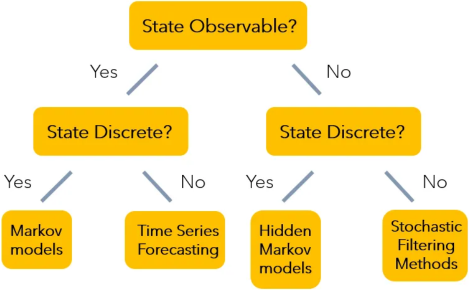
Case Study: Estimate the system's "RUL" by PdM (6)

Health Management: After detecting the system fault and estimating the system's RUL, some actions are taken based on obtained results. The main goal of health management is to manage the maintenance and logistics in an optimal manner, i.e., achieving increased availability, reliability, and safety, also reduced maintenance and logistics costs. Health management is usually formulated as a **constrained optimization problem**, where global optimization algorithms are used to derive the best **maintenance scheduling** [1]. An additional step in health management is **monitoring**. Here, we collect feedback and new data to refine and update the ML models as needed. We also track the effectiveness of the maintenance activities and adjust the schedule and strategies as needed.



Case Study: Estimate the system's "RUL" by PdM (7)

Classification of System Characteristics: Prognostic methods generally differ according to the type of system considered. Therefore, it would be a good idea to first classify various systems based on their characteristics before discussing specific methods under individual categories.
System Characteristics: In general, we can categorize a system based on whether its state is directly observable or indirectly observable, as well as whether its state is modeled as a discrete process or a continuous process. The following decision tree illustrates this classification scheme [1].



Case Study: Estimate the system's "RUL" by PdM (8)

Direct or Indirect Observable? The 1^{st} criterion is the **observability of the system's state**. In some cases, the monitored data can **directly** describe the system's underlying state, such as **wear** and **crack size**. However, in other cases, the data can **indirectly** indicate the system's state, such as **vibration and oil-based monitoring** for rotatory machines.

Discrete or Continuous State Evolution? The 2^{nd} criterion is **how we model the state evolution of the system**. For some cases, we assume the system evolves on a finite state space $\boldsymbol{\Phi} = \{0, 1, \ldots, N\}$, where 0 corresponds to the perfect healthy state and N represents the failed state. Those discrete states can be derived based on meaningful operational conditions, e.g. "Good", "Minor defects only", "Maintenance required", or they can be derived from applying clustering techniques to training data. For other cases, it may make more sense to model the system evolution as a continuous process. For example, the battery's internal resistance, which commonly served as a health indicator for lithium-ion batteries, degrades continuously when going through a sequence of charge-discharge cycles.

Case Study: Estimate the system's "RUL" by PdM (9)

Prognostic algorithms: Here, we will discuss some of the commonly employed ML methods for prognostics - predicting the system's RUL. We organize our discussion according to the categories from the previous part. **Markov Models:** Are useful for systems whose states are **directly observable** and evolve **discretely**. Markov methods model the system degradation as a stochastic process that jumps between a finite set of states $\Phi = \{0, 1, \ldots, N\}$, where 0 is the healthy and N is the failed state. The sequence of the states is a **Markov chain**. The assumption of the Markov model is that the future state depends on the current state. This is known as a **Markov property**. Under Markov, RUL is defined as the time the degradation will take from the current state to N for the first time. Of course, to calculate RUL using Markov methods, we need to know the number of states and the transition probability matrix A between states, where A_{ij} denotes the transition probability from a state i to j. In practice, they are estimated from the training data. For determining the number of states, a K-means clustering algorithm is usually employed [1].

Case Study: Estimate the system's "RUL" by PdM (10)

TS Forecasting: Is useful for systems whose states are **directly observable** and evolve **continuously**. Here, RUL is essentially the estimation of the measured TS data to reach a predefined threshold. **Exponential smoothing method** forecasts new observations as a weighted average of past observations, with the weights decreasing exponentially back in time. As well as **RNNs**, in particular, LSTM models.

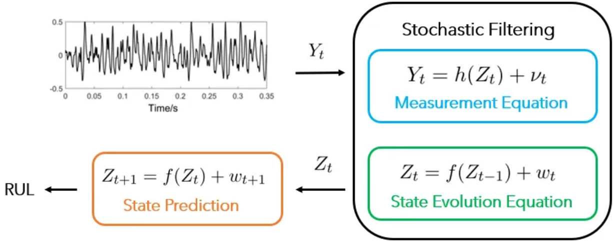
Hidden Markov Models (HMM): are useful for systems whose states can only be **indirectly observed** and evolve **discretely** [1].

$$\begin{array}{ccccccc} \rightarrow & Z_{t-1} & \rightarrow & Z_t & \rightarrow & Z_{t+1} & \rightarrow \\ & \downarrow & & \downarrow & & \downarrow & \\ & Y_{t-1} & & Y_t & & Y_{t+1} & \end{array}$$

There is an observable process $\boldsymbol{Y_n}$ which accounts for indirect observations obtained from sensor measurements and a system degradation process $\boldsymbol{Z_n}$ whose states are hidden and evolves according to a Markov chain on a finite state space; $\boldsymbol{P(Y_n|Z_n = i)}$ is relationship between them.

Case Study: Estimate the system's "RUL" by PdM (11)

Finally, **stochastic filtering methods** are useful for systems whose states can only be **indirectly observed** and evolve **continuously**.



The **measurement equation** bridge the gap between the measured feature values and the internal system state. Here, $\boldsymbol{h(.)}$ denotes the measurement model, and $\boldsymbol{\nu}$ represents the measurement noise. Meanwhile, we have a **state evolution equation** to describe the system degradation process. Here, $\boldsymbol{f(.)}$ denotes the degradation model, and $\boldsymbol{w}$ represents the model uncertainty. This uncertainty term is induced because the degradation model can only partially reflect the true physical process [1].

Case Study: Estimate the system's "RUL" by PdM (12)

Challenges In Prognostics: Despite the rapid advances in prognostic algorithms, performing reliable prognostics is not always easy in reality. There are some challenges that may prevent us from achieving the goal [1]:

- **Sensor reliability and failures**, as sensors may operate in a hostile environment;
- **Feature extraction**, as it is a non-trivial task to isolate features that are related to the degradation process for complex systems;
- **Data availability**, as employing ML techniques for prognostics usually requires a large amount of training data (especially run-to-failure data), which is not readily accessible from in-service systems due to time and cost.

Case Study: Estimate the system's "RUL" by PdM (13)

Besides the mentioned problems, uncertainty encountered in the prognostics constitutes another major challenge for obtaining a reliable RUL estimation. Prognostic uncertainties may originate from [1]:

- **Input data:** Sensor data may contain a significant level of noise. Also, environmental/operational conditions are constantly changing.
- **Model:** Due to limited training data, the constructed data-driven models may fail to capture the true system degradation process accurately, thus producing modeling errors and uncertainties.

Since these uncertainties can lead to significant deviation of prognostics results from the actual situation, developing a **systematic uncertainty management framework** is crucial for delivering meaningful RUL predictions [1].

How can we address the challenges in prognostics?

- **Sensor reliability and failure problems:** Sensor fusion, redundant sensors, regular calibration and maintenance of sensors.
- **Non-trivial feature extraction:** Advanced or ensemble feature selection techniques like *Recursive Feature Elimination (RFE), LASSO*, etc. *CNNs and RNNs* can automatically learn feature representations from raw data.
- **Data availability problems:** transfer learning, data augmentation, data sharing & collaboration.
- **Systematic Uncertainty Management Framework:** Libraries for Probabilistic/Bayesian learning and optimization under uncertainty like *PyMC3, GPy, TensorFlow Probability, Stan*.

Latest methods for data-based PO & data-driven PdM (1)

Advancements in data analytics, ML, and AI have led to new methods and techniques for PO and PdM in the context of machine tools. Some of these latest methods include:

- **Deep learning:** Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) can be used to analyze complex data patterns, such as time-series data, images, and audio signals, which are common in machine tool applications.
- **Transfer learning:** In cases where data is scarce or not diverse enough, transfer learning can be employed. Pre-trained models can be fine-tuned to specific manufacturing applications, reducing the amount of training time and computational resources.
- **Reinforcement learning:** Techniques like Q-learning and Deep Q-Networks (DQN) can be used to develop intelligent agents that can optimize process parameters and control strategies in real-time. These agents learn by interacting with the environment (e.g., machine tools) and receiving feedback (rewards or penalties) based on their actions.

Latest methods for data-based PO & data-driven PdM (2)

- **Unsupervised learning:** The methods like clustering and autoencoders can help identify patterns, trends, and anomalies in large datasets without labeled data. For example, clustering can be used to group similar machining conditions or tool wear patterns, which can then be analyzed to optimize process parameters. Autoencoders, on the other hand, can learn a compact representation of the data, enabling efficient analysis and anomaly detection.
- **Edge computing:** Distributing data processing and analytics tasks across edge devices (e.g., sensors, controllers) can improve response times, reduce network traffic, and increase the overall efficiency of manufacturing systems.

By leveraging these advanced methods, data-based PO and data-driven PdM can significantly enhance the performance and reliability of manufacturing systems.

Conclusion

How can we automate each step individually or all steps together in data-based PO and data-driven PdM?

Data collection: *Apache Kafka, Apache NiFi, Logstash, Fluentd* for data ingestion from different types of manufacturing systems or machine tools.

Data prepossessing and ETL: *Apache Beam, Apache Nifi, Talend, or Microsoft Azure Data Factory* besides *pandas, NumPy, and scikit-learn*.

Feature Extraction: *scikit-learn, TensorFlow, PyTorch, and Featuretools*.

Model building, validation, and optimization/maintenance scheduling: AutoML Frameworks such as *H2O.ai, AutoPyTorch, TPOT, Auto-sklearn, AutoGluon* besides *scikit-learn, TensorFlow, PyTorch*.

MLOps pipelines: *MLflow, Kubeflow, or TFX (TensorFlow Extended)* to orchestrate the pipeline, track experiments, manage artifacts, and deploy models. And CI/CD tools like *Jenkins, GitLab CI/CD, or GitHub Actions* to automate the process further.

Deployment and Integration: Deploy and integrate the final solution to the existing platform.

References

[1] Towards Data Science. Data-driven predictive maintenance in a nutshell. https://tinyurl.com/t5fs4npe, 2021. Accessed: 2023-06-29.