ML for Process Industry Series

Machine Learning in Python for Process Systems Engineering

Achieve operational excellence using process data

2024 Edition

Ankur Kumar, Jesus Flores-Cerrillo

`

Machine Learning in Python for Process Systems Engineering

Achieve operational excellence using process data

Ankur Kumar
Jesus Flores-Cerrillo

`

Dedicated to our spouses, family, friends, motherland, and all the data-science enthusiasts

`

Machine Learning in Python for Process Systems Engineering

www.MLforPSE.com

First published: February 2022
Revision 1 published: June 2023
Revision 2 published: January 2024

`

About the Authors

Ankur Kumar holds a PhD degree (2016) in Process Systems Engineering from the University of Texas at Austin and a bachelor's degree (2012) in Chemical Engineering from the Indian Institute of Technology Bombay. He currently works at Linde in the Advanced Digital Technologies & Systems Group in Linde's Center of Excellence, where he has developed several in-house machine learning-based monitoring and process control solutions for Linde's hydrogen and air-separation plants. Ankur's tools have won several awards both within and outside Linde. Most recently, one of his tools, PlantWatch (a plantwide fault detection and diagnosis tool), received the 2021 Industry 4.0 Award by the Confederation of Industry of the Czech Republic. Ankur has authored or co-authored several peer-reviewed journal papers (in the areas of data-driven process modeling and monitoring), is a frequent reviewer for many top-ranked Journals, and has served as Session Chair at several international conferences. Ankur served as an Associate Editor of the Journal of Process Control from 2019 to 2021, and currently serves on the Editorial Advisory Board of Industrial & Engineering Chemistry Research Journal. Most recently, he was included in the 'Engineering Leaders Under 40, Class of 2023' by *Plant Engineering* Magazine.

Jesus Flores-Cerrillo is currently an Associate Director - R&D at Linde and manages the Advanced Digital Technologies & Systems Group in Linde's Center of Excellence. He has over 20 years of experience in the development and implementation of monitoring technologies and advanced process control & optimization solutions. Jesus holds a PhD degree in Chemical Engineering from McMaster University and has authored or co-authored more than 40 peer-reviewed journal papers in the areas of multivariate statistics and advanced process control among others. His team develops and implements novel plant monitoring, machine learning, IIOT solutions to improve the efficiency and reliability of Linde's processes. Jesus's team received the Artificial Intelligence and Advanced Analytics Leadership 2020 award from the National Association of Manufacturers' Manufacturing Leadership Council.

`

Note to the readers

Jupyter notebooks and Spyder scripts with complete code implementations are available for download at https://github.com/ML-PSE/Machine_Learning_for_PSE. Code updates, when necessary, will be made and updated on the GitHub repository. Updates to the book's text material will be made available on Leanpub (www.leanpub.com) and Google Play (https://play.google.com/store/books). We would greatly appreciate any information about any corrections and/or typos in the book.

`

`

Table of Contents

`

`

`

`

`

Preface

Everyday we hear stories about new feats being achieved by machine learning (ML) and artificial intelligence (AI) researchers that have the potential to revolutionize our world. Through humanoid robots, speech recognition, computer vision, driverless cars, fraud detection, personalized recommendations, and automated health diagnosis, machine learning has already become an integral part of our daily life. Moreover, away from the glitz of these ‘visible’ high-tech products, machine learning has also been making silent advances in process industries (chemical industry, biopharma industry, steel industry, etc.) where ML-based solutions are being increasingly used for predictive equipment maintenance, product quality assurance, process monitoring, fault diagnosis, process control, and process optimization. With increasing global competition and stricter product quality standards, industrial plants are relying upon machine learning tools (such as reinforcement-learning-based auto-adaptive process controller) to provide them the winning edge over competitors.

Perhaps you are reading this book because you too have been inspired by the capabilities of machine learning and would like to use it to solve problems being faced by your organization. However, you might be struggling to find a definite guide that can help you decide which specific methodology to choose among the myriad of available methodologies. You may have come across a nice research article that showcases an interesting process systems application of a ML method. However, you might be facing difficulties trying to understand the intricate details of the algorithm. We won’t be surprised if you have struggled to find a data-science book that caters to the needs of a process systems engineer, considers unique characteristics of industrial process systems, and uses industrial-scale process systems for illustrations. We, the authors, have been in that phase. A process engineer will arguably find it more relevant and useful to learn principal component analysis (PCA) by working through a process monitoring application (the most popular application area of PCA in process industry) and learning how to compute the monitoring metrics. Similar arguments could be made for several other popular ML methods. There is a gap in available machine learning resources for industrial practitioners and this book attempts to cover this gap.

In one sense, we wrote this book for our younger selves; a book that we wish had existed when we started experimenting with machine learning techniques. Drawing from our years of experience in developing data-driven industrial solutions, this book has been written with the focus on de-cluttering the world of machine learning, giving a comprehensive exposition of ML tools that have proven useful in process industry, providing step-by-step elucidation of implementation details, cautioning against the

`

pitfalls and listing various tips & tricks that we have encountered over the years, and using dataset from industrial-scale process systems for illustrations. We strongly believe in ‘learning by doing’ and therefore we encourage the readers to work through in-chapter illustrations as they follow along the text. For reader’s assistance, Jupyter notebooks with complete code implementations are available for download. We have chosen Python as the coding language for the book as it convenient to use, has large collection of ML libraries, and is the de facto standard language for ML. No prior experience with Python is assumed. The book has been designed to teach machine learning from scratch and upon completion, the reader will feel comfortable at using ML techniques.

Machine learning will continue to play significant role in unleashing the next wave of productivity improvements in process industry. We hope that this book will inspire its readers to develop novel ML solutions to challenging problems at work. We wish all the best to the budding process systems data scientist.

Who should read this book

This book provides a comprehensive step-by-step exposition of several popular machine learning techniques that have proven useful in process industry. Industry-relevant illustrations have been included in a ‘learn-by-doing’ format. Appropriate references to advanced treatment of specific topics are provided in each chapter. Therefore, the book will be useful to process technologists from industry, undergraduate and graduate students, ML researchers, as well as a general data-science enthusiast. If you belong to any of the following categories, you will enjoy reading this book.

1) Process systems engineers or data scientists seeking to hone their ML skills
2) Process systems engineers looking for ML solutions to specific problems (such as monitoring a high-dimensional multimode process)
3) Budding process systems data scientists taking their first step into the ML world
4) Budding data scientists (with no Python experience) looking to learn Python by working with real industrial datasets
5) Process systems engineers who frequently use commercial data-science software (IBM SPSS, Aspen Mtell) and are interested in learning the technical details for an improved understanding of the methodologies.

Pre-requisites

No prior experience with machine learning or Python is needed. Undergraduate-level knowledge of basic linear algebra and calculus is assumed.

`

Book organization

The book follows a holistic and hands-on approach to learning ML where readers first gain conceptual insight and develop intuitive understanding of a methodology, and then consolidate their learning by experimenting with code examples. Every methodology is demonstrated by using simple process system or numerical example to illustrate major characteristics of the method and then by implementing on industrial-scale processes.

The book has been divided into four parts. **Part 1** provides a perspective on the importance of ML in process systems engineering and lays down the basic foundations of ML. **Part 2** provides in-detail presentation of classical ML techniques and has been written keeping in mind the various characteristics of industrial process systems such as high-dimensionality, non-linearity, multimode operations, etc. **Part 3** is focused on artificial neural networks and deep learning. While deep learning is the current buzzword in ML community, we would like to caution the reader against the temptation to deploy a deep learning model for every problem at hand. Often, simpler classical models can provide as good, if not better, results as those from neural net models. For example, partial least squares (PLS) are still the most popular models for soft sensor development in process industry due to its simplicity and powerful capabilities of handling noisy and correlated data. **Part 4** covers the important topic of deploying an ML solution over web.

It was a deliberate decision to not divide the book in terms of supervised / unsupervised / reinforcement-learning categories or application areas (process modeling, monitoring, etc.). This is because several methods overlap these categories which make it difficult to put them under a specific category. For example, SVM and SVR methods fall under supervised category while the related SVDD method falls under unsupervised category. Similar situation holds for PCA/PCR/PLS methods. A reader who is interested in a specific application area may use the table of contents as a guide to relevant sections in Parts 2 and 3. Care has been taken to title the subsections in different chapters appropriately.

Symbol notation

The following notation has been adopted in the book for representing different types of variables:

- lower-case, bold-face letters refer to vectors ($\boldsymbol{x} \in \mathbb{R}^{m\times 1}$) and upper-case, bold-face letters denote matrices ($\boldsymbol{X} \in \mathbb{R}^{n\times m}$)
- individual element of a vector and a matrix are denoted as x_j and x_{ij}, respectively.
- any *i*th vector in a dataset gets represented as subscripted lower-case, bold-faced letter ($\boldsymbol{x_i} \in \mathbb{R}^{m\times 1}$)

`

Part 1

Introduction & Fundamentals

Chapter 1

Machine Learning for Process Systems Engineering

Imagine yourself in the shoes of an operator or engineer responsible for uninterrupted and optimal operation of an oil refinery. Keeping an eye on each of the 1000s of process measurements being made every second to look for process abnormalities or opportunities for plant performance improvement is akin to finding a needle in a haystack. The task is undoubtedly overwhelming and is the primary reason why plant managers often complain about having *'too much data but little knowledge and insight'*.

However, unlike humans, computers can be programmed to parse through large amounts of data in real-time, extract patterns, trends, and assist plant personnel in making informed business and operational decisions. This practice of learning about systems from data or machine learning has become an indispensable tool in process operations in the age of increasing global competition and stricter product quality standards.

This chapter provides an overview of how the power of machine learning is harnessed for process systems engineering. Specifically, the following topics are covered

- Unique characteristics of process data
- Types of ML systems and typical workflow to convert data into insights
- Classical applications of ML techniques in process industry
- Common ML solution deployment infrastructure employed in industry

Let's now tighten our seat-belts as we embark upon this exciting journey of de-mystifying machine learning for process systems engineering.

`

1.1 What are Process Systems

Process systems refer to a collection of physical structures that convert raw materials (wood, natural gas, crude oil, coal, etc.) into final consumer products (paper, fertilizers, diesel, energy, etc.) or intermediate products which are then used to manufacture other commodity materials. These process systems can range from a simple water-heating system to complex oil refineries. Figure 1.1 shows an example (petrochemical) plant comprising several processing units such as distillation columns, heat exchangers, pumps. Process industry is a broad category that encompasses, amongst others, chemical industry, bioprocess industry, power industry, pharmaceuticals industry, steel industry, semiconductor industry, and waste management industry.

Figure 1.1: Petrochemical plant image (obtained from pixabay.com)

In process industry, the task of optimizing production efficiency, controlling product quality, monitoring the process are categorized as process systems engineering (PSE) activities. These tasks often require a mathematical model of the plant. The traditional practice has been to use first principles mathematical description of physio-chemical phenomena occurring inside each process unit to mathematically characterize the whole plant. However, as you may already know, building such fundamental models are time-consuming and difficult for complex systems. Machine learning (ML)-based methods provide a convenient alternative where process data are used to build empirical plant model which can be used for optimization, control, and monitoring purposes. Availability of large amount of sensor data has further boosted the trend of incorporating ML techniques for PSE and demand for process data scientists.

`

Let’s take a look at the kind of data generated in process industry. Majority of the process data include process stream flowrate, temperature, pressure, level, power, and composition measurements as shown in figure 1.2. Additionally, obtaining vibration signals from rotating equipment (motors, compressors), infrared or visual images, and spectra data are also common now-a-days. These indirect data are frequently utilized for predictive equipment maintenance and product quality control.

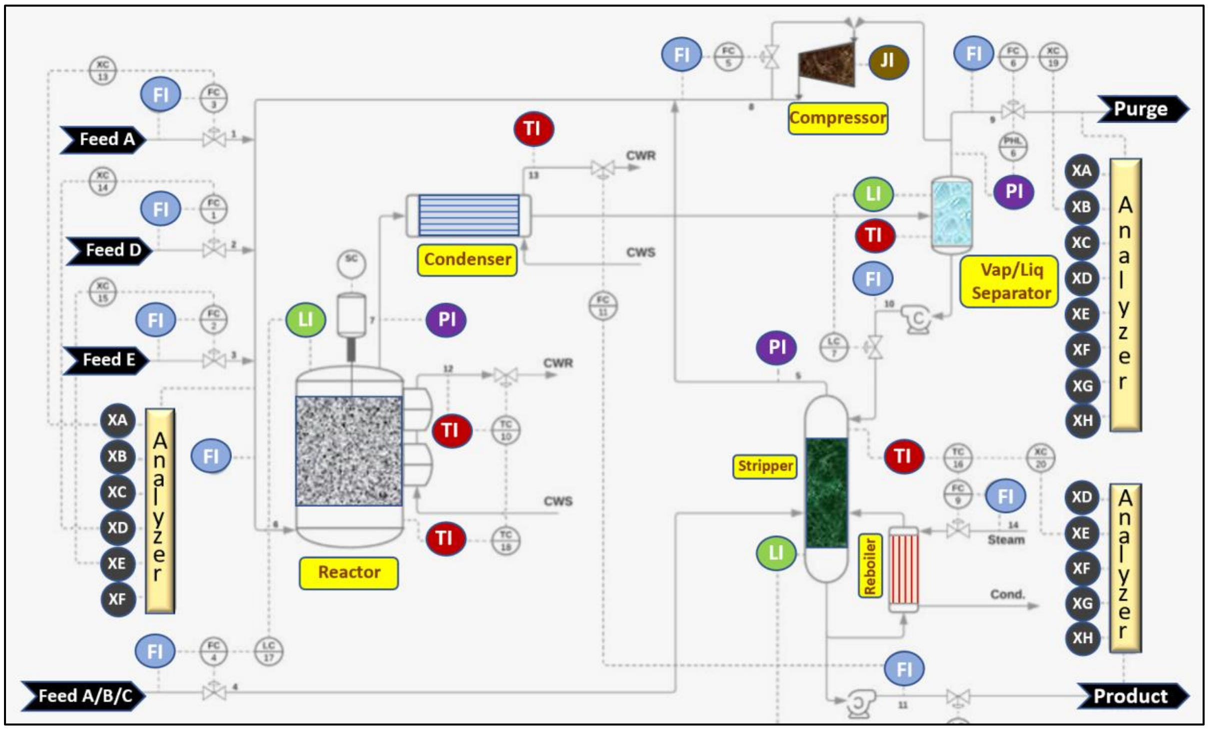


Figure 1.2: A process flowsheet[1] with typical flow (FI), temperature (TI), pressure (PI), composition (Analyzers), level (LI), power (JI) measurements.

Characteristics of process data

Industrial process data often exhibit characteristics which pose challenges to a process data scientist. Choosing an ML technique appropriate to the data is a pre-requisite for a successful project implementation. These characteristics include the following:

- *Dynamic*: Process plants rarely operate at a perfect steady-state i.e., at fixed values of process inputs and outputs. Plants are often subject to random or systematic disturbances such as changing ambient conditions, process feed quality, and product demand which are handled by control system by manipulating process variables. This leads to fluctuations around a steady-state point or plants moving from one steady-state to another.

[1] Adapted from the original flowsheet by Gilberto Xavier (https://github.com/gmxavier/TEP-meets-LSTM) provided under Creative-Commons Attribution 4.0 International License (https://creativecommons.org/licenses/by/4.0/).

`

- *Time-varying*: Correlations between process variables change over time due to gradual changes in process parameters. For example, heat transfer coefficient in a heat-exchanger may change due to fouling or catalyst activity may degrade due to aging.

- *Batch vs continuous*: While continuous processes are dominant in process industry, batch processes are also frequently employed, for example in semiconductor and drug manufacturing. Unlike continuous processes, batch processes are of finite durations and offer its own set of unique challenges such as batch-to-batch variations and inherent non-steady-state operation. Time itself becomes a crucial dimension for accurate modeling of temporal evolution of process variables within a batch.

- *Multimode operations*: Multimode characteristics show up when a process plant operates primarily around a few distinct steady states. For example, a plant producing species a, b, and c may have different production recipes (set-points of process variables) for each of these recipes. Depending on the production schedule, the plant will switch from one recipe to another. Deploying a single global ML model for a multimode process often leads to lower accuracy compared to that obtained from building local models for each mode.

- *Discrete/Discontinuous*: Depending upon the production load, certain equipment may be switched on or off causing step changes in process behavior. For example, switching on an extra turbine or compressor causes step increase in power consumption.

- *Nonlinear*: The complex physio-chemical phenomena (chemical reactions, vapor-liquid equilibrium, etc.) occurring inside process units are most often nonlinear. While linear models often provide good approximations when processes operate around a single steady-state, non-linear models become necessary when processes experience large fluctuations in the operating conditions.

- *High dimensionality*: Modern process plants make hundreds of process-critical measurements. Considering that process plants make process adjustments in real-time, high dimensionality becomes an issue when a computationally intensive methodology is adopted.

- *Multirate sampling*: Not all process measurements are made at the same frequency. While temperature or pressure readings are sampled every second or minute, composition/analyzer measurements are often sampled at much lower frequency (once an hour or a day).

In this book, we will study several ML methods in detail which have been designed to handle these different varieties of process systems. Let us first understand what we mean by machine learning.

`

1.2 What is Machine Learning

At its core, machine learning simply means using computer programs and data for finding relationship between different components of a system and discovering patterns that were not immediately evident. This characteristic of using data to learn about the system makes machine learning interesting and different. System specific knowledge is not explicitly embedded into a ML program; the program extracts the knowledge from data. Figure 1.3 compares ML approach with a non-ML approach for distillation column modeling: ML approach does not require system specific data such as number of stages or the type of packing.

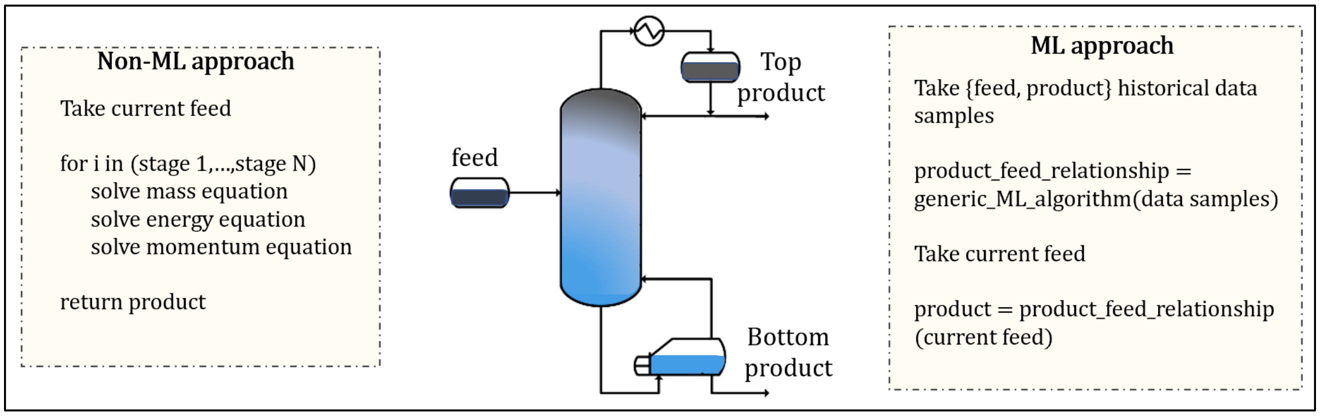

Non-ML approach
Take current feed
for i in (stage 1,...,stage N)
solve mass equation
solve energy equation
solve momentum equation
return product
feed
Top product
Bottom product
ML approach
Take {feed, product} historical data samples
product_feed_relationship = generic_ML_algorithm(data samples)
Take current feed
product = product_feed_relationship (current feed)

Figure 1.3: Computer program using first-principal approach (left) and ML approach (right) for modeling a distillation column

We often marvel at the accuracy of recommendations made by Netflix or amazon for potential shows or products. These companies do not possess explicit information about our personal preferences or psychology (whether we like sci-fi movies or not). The data does all the trick! ML algorithms process past purchase data to discern the likes and dislikes of its customers and make recommendations. In process industry, manufacturers use ML methods to determine optimal equipment maintenance schedule using past maintenance and operating conditions data. Here again, data-based analysis provides considerable convenience over the alternative method of complex metallurgical analysis.

The reliance on data alone to obtain reasonably good system approximation is one of the major reasons behind ML's growing popularity. The barrier of requiring specific domain knowledge to be able to analyze a system has been circumvented by machine learning. ML algorithms are also universal in nature. For example, we can use the same data-clustering algorithm for analyzing demographics data, factory data, or economic data to obtain actionable knowledge. These properties combined with the surge in the amount of data

`

collected and the dip in the cost of computational resources had led machine learning to revolutionize several industries.

The increasing popularity of machine learning does not mean domain knowledge is redundant. Domain expertise often helps to decide which ML algorithm to use and proper infusion of domain knowledge into ML methodology can sometimes increase model accuracy considerably.

"Why don't we just build a model using first-principles and get very accurate models? Why rely on data?" This is a valid question. There is no doubt that fundamental models have better accuracy and generalization capability, however, developing fundamental models are often time consuming and require expert resources. These models can sometimes be too complex to execute in real-time. Adopting ML methodology can help getting around these difficulties.

Machine learning workflow

Figure 1.4 shows the typical tasks involved in a machine learning project. The tasks are categorized into offline computations and online/real-time computations. In online computations, process data are parsed through the process model to provide real-time insights and results. Note that the models could be process input-output models, process monitoring models (process normal/abnormal classification), mode categorization models, or fault classification models.

The models are built offline using historical process data. This offline exercise is performed once or repeated at regular intervals for model update. Brief description of the essential steps performed are provided below:

- *Sample and variable selection*: One does not simply dump all the available historical data and sensor measurements into model training module. Only the portion of historical data that best represents the current process behavior or the behavior of interest is utilized. For process systems, it is common to use data over the past couple of years as training data. If steady-state models are being build, then data from steady-state operation periods are used.

  Input variable selection warrants judicious consideration as well. While including too many model inputs leads to overfitting and high computational complexity, leaving out important variables leads to underfitting. The basic principle is to include only those inputs that are known to influence the model outputs. Specific algorithms for variable selection are covered in Chapter 4.

`

- *Data cleaning*: "*Garbage in, garbage out*" is an age-old principle in computer simulations. The same holds for ML model training. Your model will be practically useless if training data is not 'clean'. For example, process monitoring model won't be able to detect process abnormalities accurately if it has been trained with outlier-infested training data. Data cleaning includes, amongst others, identification and removal of outliers, and removal of noise effects. Detailed algorithms are covered in Chapter 4.

- *Model training and validation*: Model training imply estimating the parameters of the chosen ML model, for example, the coefficients in a linear regression model. Model validation is employed for finding optimal values of model hyperparameters, for example, regularization coefficient in ridge regression. At the end of this step, the coveted process model is obtained. Several best practices for model training and validation are covered in Chapter 3.

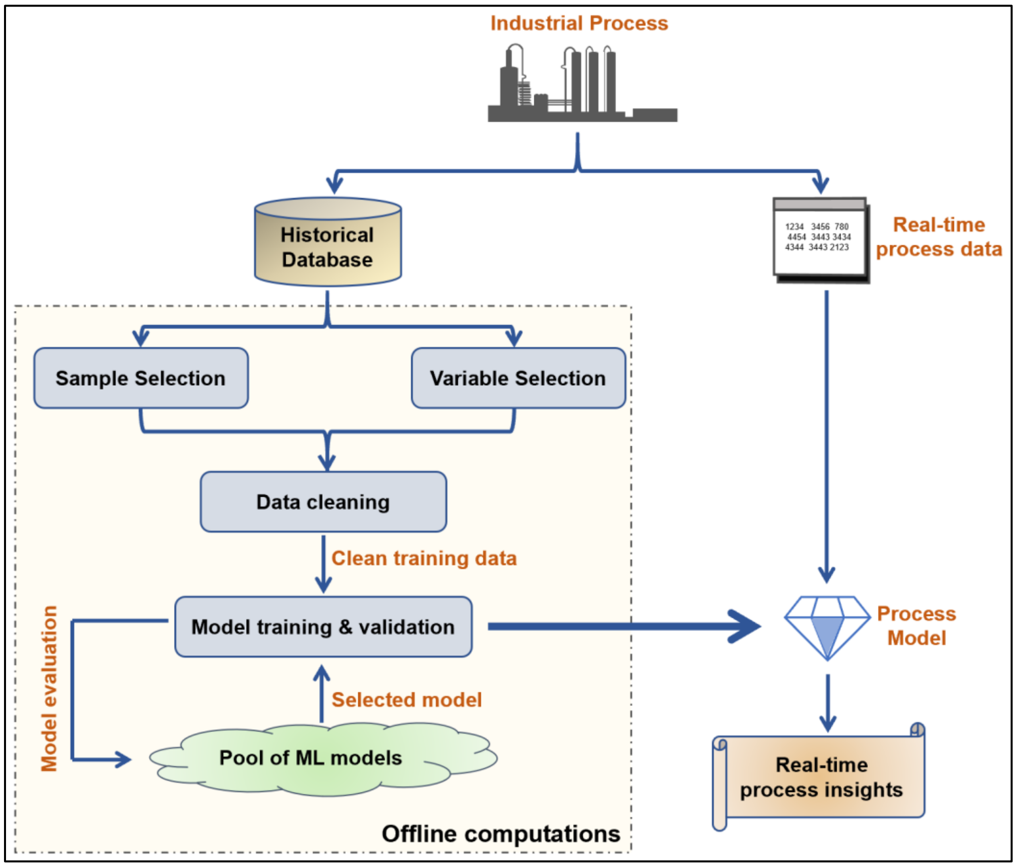


Figure 1.4: Steps involved in a typical ML-based methodology

Distinct from the offline-online paradigm, there is another approach employed in process industry, especially for nonlinear and multimode processes. It is called just-in-time learning or lazy learning. As shown in Figure 1.5, the model building exercise is carried out online as well. When new process data comes in, relevant data are fetched from the historical dataset

`

that are similar to the incoming samples based on some nearest neighborhood criterion. A local model is built using the fetched relevant data. The obtained model processes the incoming samples and is then discarded. A new local model is built when the next samples come in.

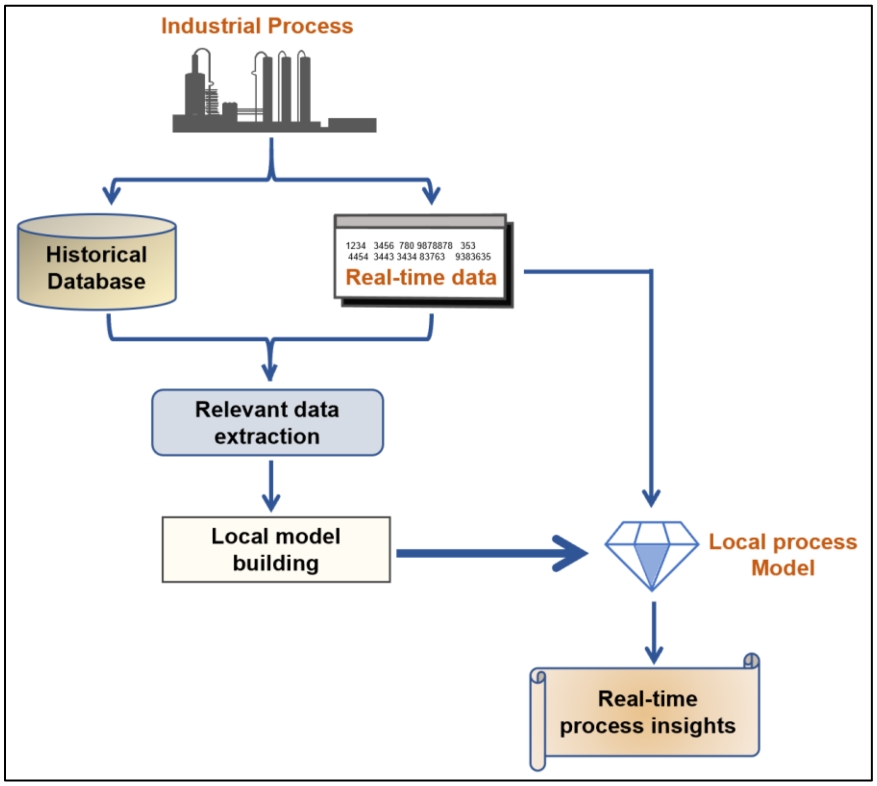


Figure 1.5: Steps involved in a just-in-time learning methodology

Types of machine learning systems

Although there is a plethora of ML models available, all of them can be organized into three broad categories, namely, supervised learning, unsupervised learning, and reinforcement learning. The categorization is based on the nature of model output associated with each sample in the training dataset. Note that there is another category, semi-supervised learning – this method, however, is used less often compared to the other mentioned methods. Figure 1.6 gives a sneak peek into some of the ML algorithms and application areas corresponding to supervised and unsupervised learnings that we will cover in this book.

`

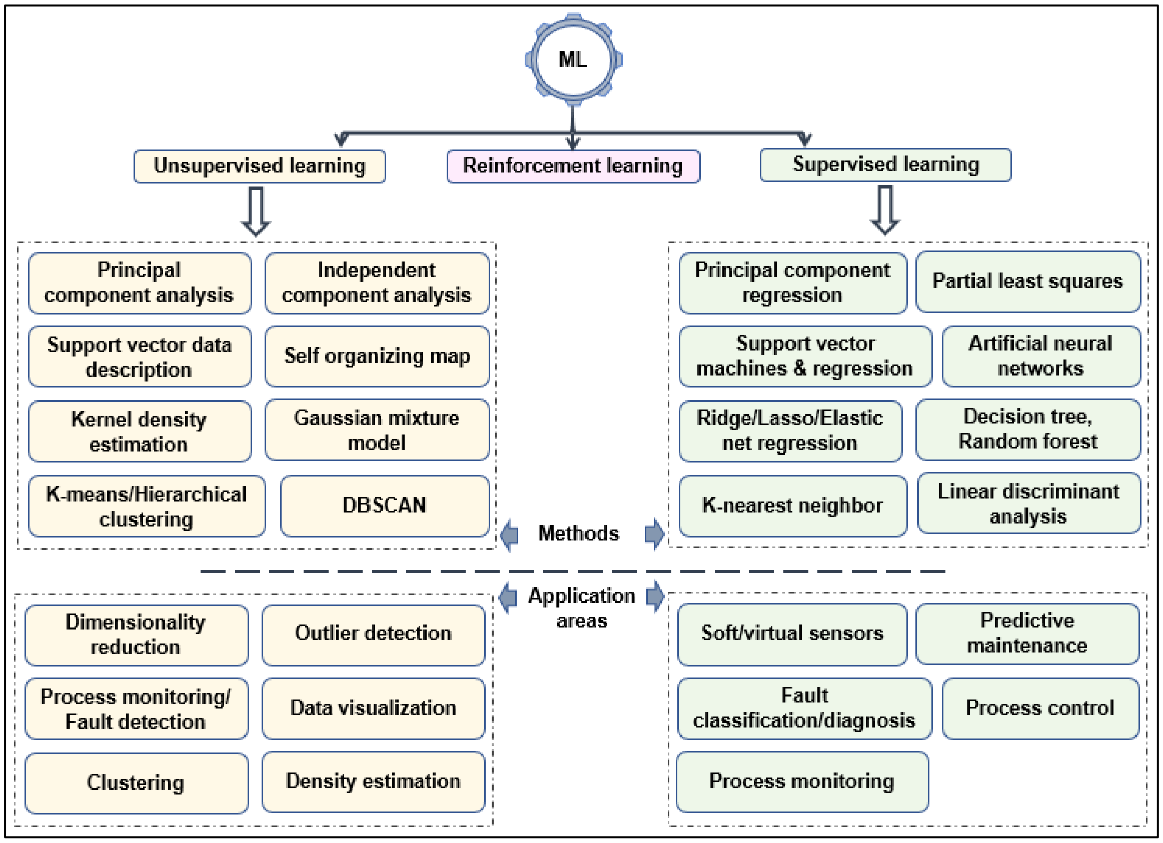

ML
Unsupervised learning
Reinforcement learning
Supervised learning
Principal component analysis
Independent component analysis
Support vector data description
Self organizing map
Kernel density estimation
Gaussian mixture model
K-means/Hierarchical clustering
DBSCAN
Principal component regression
Partial least squares
Support vector machines & regression
Artificial neural networks
Ridge/Lasso/Elastic net regression
Decision tree, Random forest
K-nearest neighbor
Linear discriminant analysis
Methods
Application areas
Dimensionality reduction
Outlier detection
Process monitoring/ Fault detection
Data visualization
Clustering
Density estimation
Soft/virtual sensors
Predictive maintenance
Fault classification/diagnosis
Process control
Process monitoring

Figure 1.6: Classification of machine learning methods

Let's now try to understand these categories a bit more clearly.

Supervised learning

Supervised learning is used when training data includes sets of inputs and associated outputs. As illustrated in Figure 1.7, supervised learning models learn the input-output relationship and use this relationship to predict the unknown output for a new input. The outputs can be discrete values (for classification problems) or continuous values (for regression problems).

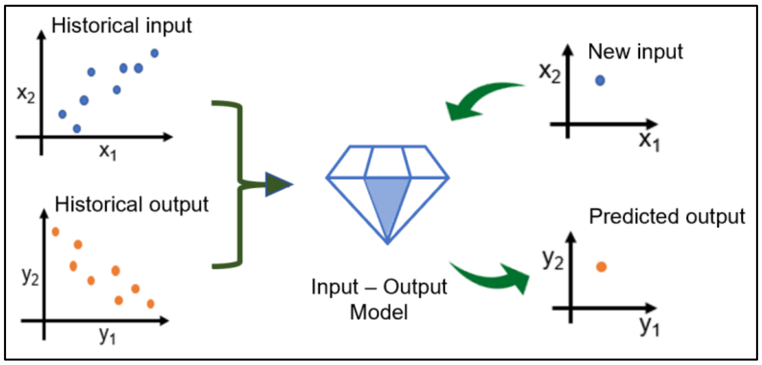


Figure 1.7: Supervised learning scheme

`

Unsupervised learning

Unsupervised learning is used when training data is not divided into inputs and outputs. The primary purpose is to find hidden pattern, if any, in data. An example situation is illustrated in Figure 1.8, where an unsupervised learning model finds prevailing structure (distinct clusters) in historical data and buckets them into different groups. The model can now be used to assign any new incoming sample into either of the groups. Unsupervised learning is often used in conjunction with supervised learning. For example, in Figure 1.8, separate local models can be built via supervised learning for data in different clusters. As you would have guessed correctly, data would need to be separated into inputs and outputs before application of supervised learning.

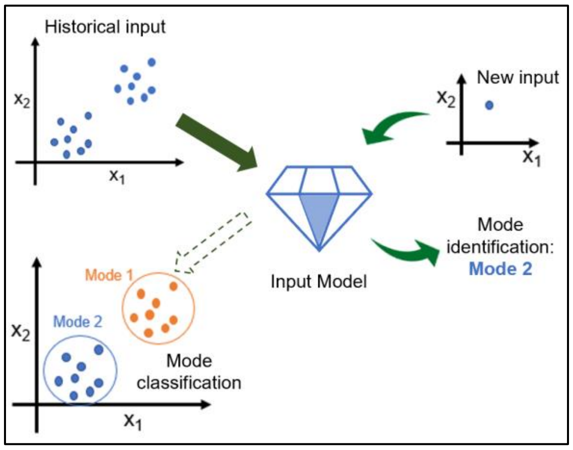


Figure 1.8: Unsupervised learning scheme

Reinforcement learning (RL)

Unlike supervised and unsupervised learning where there is one-time interaction between the system (environment) and model (ML agent) during training, in reinforcement learning the agent continuously interacts with the environment to generate training data to 'learn' an optimal strategy for accomplishing a task. The actions decided as per the learnt strategy are such that the long-term rewards are maximized. For example, consider a simple task of controlling water level in a container at a fixed height during rainy days.

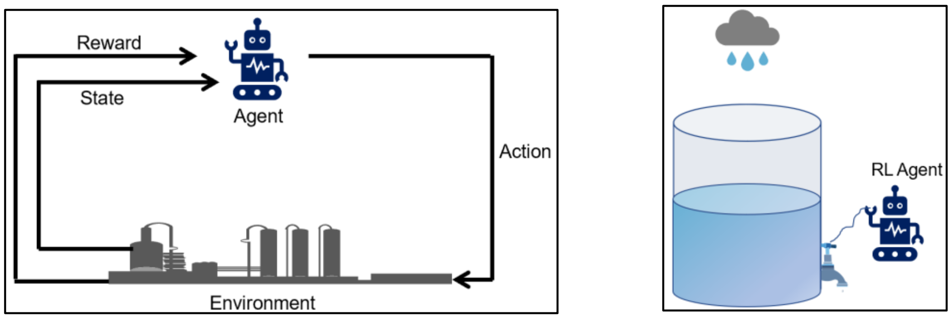


Figure 1.9: Reinforcement learning scheme and simple application setup

`

The RL agent takes actions to adjust the tap opening according to the current system state to maintain the level at some setpoint. The trivial policy of opening and closing the tap completely upon any level change can lead to high water level fluctuations. Therefore, during training the agent learns the best control policy automatically by just interacting with its environment.

1.3 Machine Learning Applications in Process Industry

Product quality control, safe work environment, optimal operations, and sustainable operations are the primary objectives in any industrial plant. Today, ML-based solutions are being utilized for all these purposes. Surrogate models are developed for online prediction of key quality variables and optimizing plant productivity, process monitoring and fault diagnosis models are developed for real-time tracking of process operating conditions, data mining is used for alarm management, data clustering is used for operation mode identification, and the list goes on.

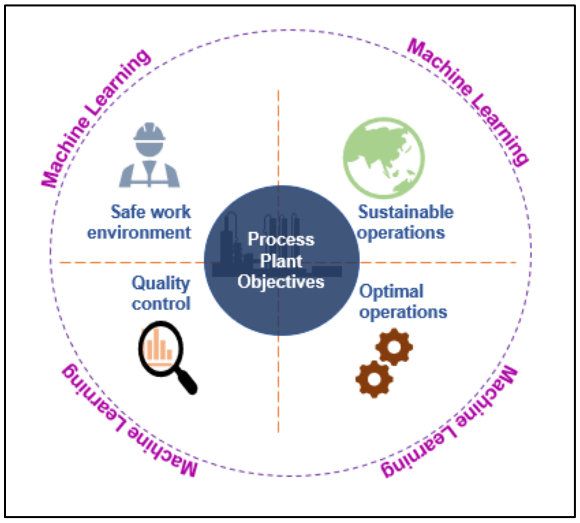


Figure 1.10: Use of machine learning to solve process plant objectives

Several success stories on machine learning application in process industry are publicly available. Shell[2] used recurrent neural networks (RNNs) for early prediction of valve failures, Saudi Aramco used ML tools for alarm analytics and predictive maintenance of turbines[3], a polymer manufacturing company used ML-based feature extractions[4] for troubleshooting quality control issues. In recent times, there has been a proliferation in the number of

[2] https://www.aiche.org/conferences/aiche-spring-meeting-and-global-congress-on-process-safety/2018/proceeding/paper/37a-digital-twins-predicting-early-onset-failures-flow-valves
[3] https://pubs.acs.org/doi/abs/10.1021/acs.iecr.8b06205
[4] https://www.yokogawa.com/at/library/resources/references/successstory-sumitomoseika-chemicals-en/

`

commercial data-analytics software (Aspen Mtell, IBM SPSS) or services offerings for process industry. This is just a testament to the growing trend of ML-driven process control and operations.

As process data scientists, we should be proud of the fact that process industry has always been a pioneer in utilizing process data for plant operations. Model predictive control (MPC) is a classic example which uses data-based model for process control. It has been used as a standard supervisory controller long before Big Data and ML became the buzzwords. Partial least squares (PLS), a popular dimensionality reduction-based soft sensing method, has long been used for online product quality predictions. The new craze about machine learning has only imparted a renewed push to explore non-traditional applications of ML in process industry.

Decision hierarchy levels in a process plant

Before we look at some specific examples of ML applications for process systems, let us first familiarize ourselves with the typical decision-making hierarchy in a process plant. Once you understand this, you will be able to easily identify the avenues where ML renders itself useful in the world of process operations and appreciate how data science has percolated all levels of process decision hierarchy.

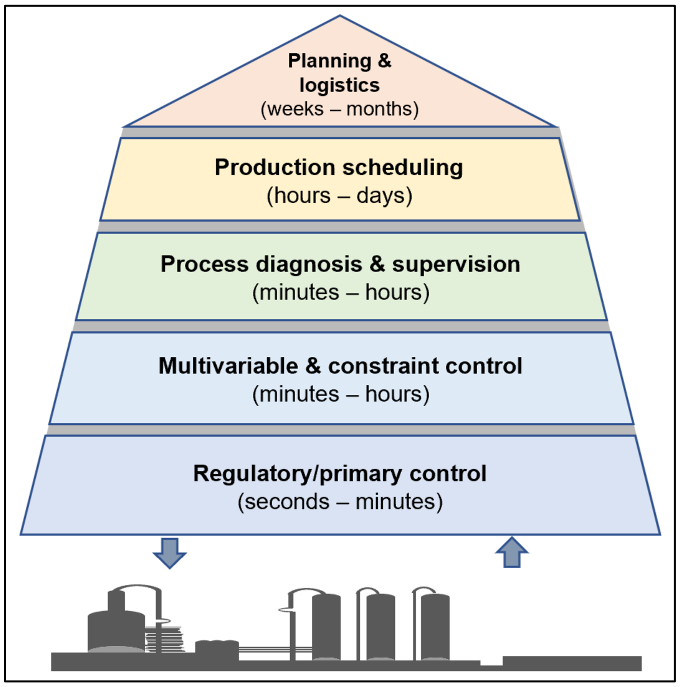


Figure 1.11: Industrial process control/decision-making hierarchy. Interval ranges in brackets show the timescales at which corresponding decisions are made.

`

At the base level, in Figure 1.11, resides the basic regulatory control layer which primarily comprises of control valves; these valves are used to modulate the flow of process streams and indirectly the pressures and temperatures as per process requirements. These requirements are in turn determined by the multivariable control layer which usually consists of MPC and RTO (real-time optimization) modules. This layer determines the base layer requirements using multivariable relationships between plant variables to ensure optimal performance of the plant. This layer is also responsible for ensuring that plant operations remain safe by ensuring safety-critical variables remain within stipulated bounds. The process diagnostics layer, if present, ensures reliability of the process through timely fault detection and diagnosis. This layer may also perform the task of controller performance assessment.

The production scheduling layer has models and methods to determine resource allocation and short-term production schedules considering external influences such as electricity prices or raw material price variations. Time-based or predictive equipment maintenance decisions may also be made in this layer. Results from this layer are communicated to the multivariable control layer. The top-most layer, planning and logistics, make enterprise-wide decisions. An enterprise operating multiple facilities use this layer to determine site-wise production targets based on the enterprise's strategic goals.

Application areas

After familiarization with the process operation decision hierarchy, we are now well-equipped to see the broad application areas of ML in a process plant. We have already seen some of the application areas in Figure 1.6. In this section we will use a furnace (Figure 1.12) as an example system to investigate these in more details.

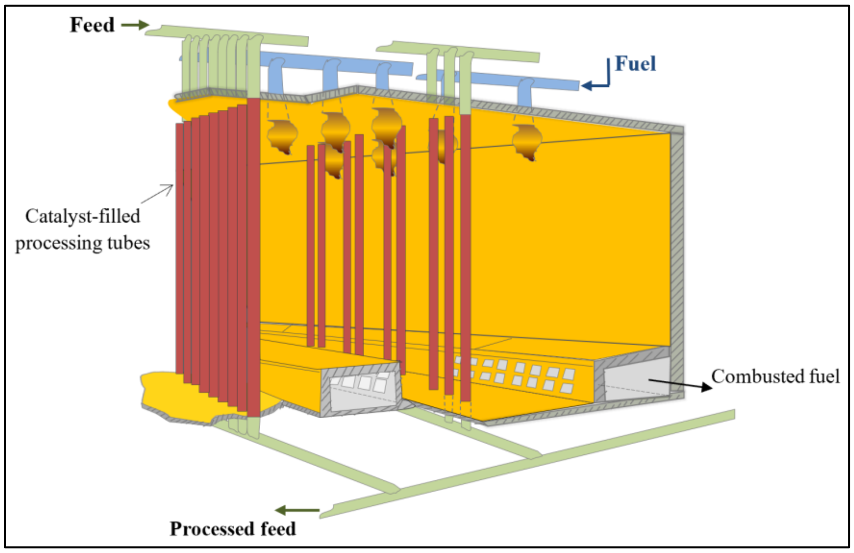


Figure 1.12: Furnace system with catalyst-filled tubes

`

The furnace system consists of several catalyst-filled tubes suspended vertically in a natural-gas fired furnace. Unreacted gas stream enters the tubes from the top, undergoes chemical reactions as they flow down the tubes and exit at the bottom. The heat from fuel combustion provides energy for the chemical reactions.

Soft Sensing

Soft sensors, also called virtual sensors or inferential sensors, are mathematical models used to estimate the values of unknown process variables using available measurements. Soft sensor models can be first principles-based or data-based and are employed when physical sensors are very costly or real-time measurements are not possible (in case of composition measurements).

Usage of data-based soft sensors are popular where process systems are too complex to build mechanistic models, or the mechanistic models are too complex to provide required estimates in real-time. For example, for the furnace system, one can build a computational fluid dynamic (CFD) model to predict the species mole fractions in the processed gas stream as a function of process inputs – fuel, air, unprocessed gas. This estimate can be used by multivariable control layer to adjust input flows accordingly, for example, increase fuel if conversion is low. However, CFD models have large execution times. As an alternative, data-based models can utilize past process data to estimate an appropriate relationship and provide compositions in real-time.

As a process modeler, you should pay careful attention that the training data is sufficiently rich in information, otherwise, a poor/low-accuracy model will result. One way to ascertain data-richness is to check if process inputs show adequate variability.

Partial least squares (PLS), principal component regression (PCR), support vector regression (SVR) are some of the popular ML method choices for estimating process quality variables. In recent times, artificial neural networks (ANNs) have also seen increased usage.

Process Monitoring

Process monitoring/fault detection/abnormality detection is among the most popular application of ML in process industry. The ML model flags an alarm when current process data shows inconsistency with historical behavior as estimated from historical data. These discrepancies could be indications of severe process faults. In the furnace system, there are

`

White-box/grey-box/black-box models

For soft sensing, terms like white-box, grey-box, black-box are often used. Figure 1.13 clarifies the difference between the terms. White-box models utilize fundamental mass/energy/momentum conservation laws to relate model inputs and outputs. Black-box methods build estimation models using only process data. Grey-box methods combine the two approaches to generate a hybrid model.

Consider our furnace system again. CFD model would be the white-box model. PLS model relating furnace inputs to product stream composition would fall in black-box model category. However, to balance the trade-off between model accuracy and computational expense, a hybrid model can be built. Mechanistic model of radiation energy transfer is the most complex part of the furnace model. One can build a black-box model to estimate the amount of radiation energy supplied to the tubes. This model can then be combined with the mechanistic model of tube to predict product composition.

Hybrid models tend to have better extrapolation accuracies compared to black-box models and are preferred when the amount of training data is low.

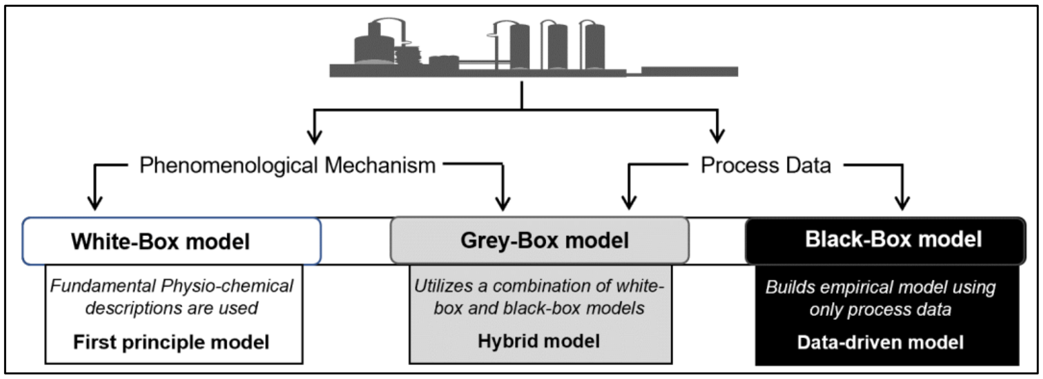

Phenomenological Mechanism
Process Data
White-Box model
Grey-Box model
Black-Box model
Fundamental Physio-chemical descriptions are used
First principle model
Utilizes a combination of white-box and black-box models
Hybrid model
Builds empirical model using only process data
Data-driven model

Figure 1.13: Soft sensing methodology spectrum

hundreds of tube temperature measurements. Unfavorable conditions like catalyst damage or tube leaks may induce abnormal increase in some of the temperatures. However, it is impractical to monitor all these temperatures manually all the time. Instead, process monitoring models could be built using historical data that provides early warning for temperature upsets so that timely corrective actions can be taken.

`

Principal component analysis (PCA) is the most popular method employed for process monitoring. PLS, independent component analysis (ICA), support vector data description (SVDD), self-organizing maps are some other commonly used methods. We will study all these methods and their applications for process monitoring in the later chapters.

Fault Classification

Once a process fault has been detected, the next challenge lies in timely identification of root-cause of the issue or identification of the process variables that are responsible for the process upset. This exercise is called fault diagnosis. If historical data on past failures are available, a classification model could be built to determine the specific fault.

For furnace system, refractory damage, tube leaks, burner malfunctions, catalyst damage can all cause temperature upsets. These different faults tend to impact tube temperatures differently; these differences are exploited by a fault classification model to identify the current fault. In this book, we will study how methods like ANNs, support vector machine (SVM) and linear discriminant analysis (LDA) can help in fault classifications.

Process Optimization & Control

Optimizing a complex system can be computationally expensive. It is not uncommon to find practitioners developing ML-based surrogate models from data to optimize the system offline or in RTO[5] layer. Surrogate models are also used in MPCs for highly non-linear systems. Reinforcement learning models are being tried in the regulatory control layer or for adaptively tuning regulatory controllers[6].

Data Clustering & Mining

Data clustering and mining methods are frequently used for activities like alarm management, operating mode characterization, pattern recognition, etc. For the furnace system, clustering models can be used to find the which set of tubes tend to show similar temperatures. Data mining models could be used to find the effect of process conditions on tube lifespan.

Predictive Maintenance

Predictive maintenance is another very popular usage of ML in process industry. Predictive maintenance models are built to determine the time to failure of any equipment or detect

[5] Real-Time Optimization and Control of Nonlinear Processes Using Machine Learning, Zhang et. al., Mathematics, 2019

[6] Adaptive PID controller tuning via deep reinforcement learning, US patent 187631, 2019

`

patterns in process data that could signal an impending process failure. Detailed planned maintenance can be carried out if failure times are known in advance. In the furnace system, tube leak occurrences may be preceded by a specific pattern in temperatures of neighboring tubes. These patterns are identified during model training and then utilized for real-time failure predictions. Advance warning can help plant operators plan plant shutdown properly.

Forecasting

Uncertainty in product demands and prices of raw materials lead to poor production scheduling. Advanced ML forecasting models are built to determine optimal production plan to maximize resource utilization and minimize production costs. In furnace system, frequent furnace temperature swings have detrimental effect on tube lifespan. If accurate monthly product demand is known in advance, then the furnace can be operated at a steady state throughout the month while using product storage to handle momentary spikes in demand.

Choosing the right ML algorithm

One of the trickiest tasks in a machine learning project is selection of modeling algorithm. Even experienced ML practitioners often recommend trial-and-error approaches. However, with experience, understanding of underlying details of ML algorithms, and process knowledge you can narrow down the trial candidates.

Let's take a sneak peek into the model selection process for the task of process monitoring. For monitoring, if the system is linear, PCA or PLS models should be tried first. If variables are non-gaussian distributed, kernel density estimation (KDE) or SVDD can be used for control-limit determination. For nonlinear systems, kernelized PCA/PLS, ANN, SVMs may be explored. SVMs are preferred when training data are limited. If process exhibits multiple operation mode, then mixture modeling like gaussian mixture model (GMM) can provide better monitoring performance.

1.4 ML Solution Deployment

While working on any project, you are most likely to experiment with several ML algorithms on your personal laptop/computer. After having done all the hard work and having decided the final form of your ML workflow, you might find yourself asking the following questions:

`

- Where does my ML tool reside where it can run uninterrupted?
- How does an end-user access the results of my tool?

The answer to the above questions is summarized in Figure 1.14 which shows a common architecture followed for deploying a Python machine learning solution from scratch within an enterprise network. You will install Python and transfer your tool (Python files) in a tool server machine (a virtual or physical machine where the ML tool would run). The tool server will be configured to execute the tool continuously (as a windows service) or on a schedule. During execution, your ML tool will fetch historical or real-time plant data and store the ML results in a database (MS SQL, MYSQL, etc.).

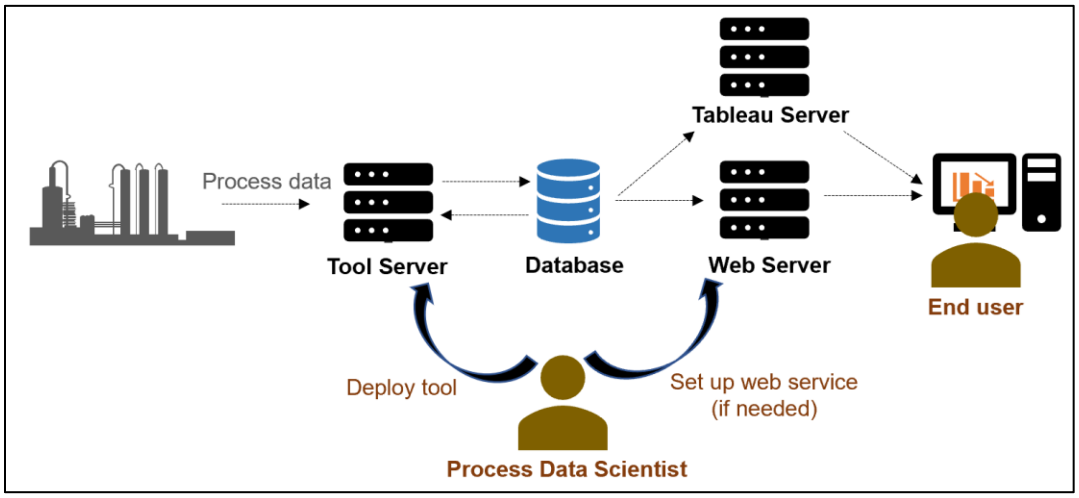


Figure 1.14: ML solution deployment

Let's tackle the second question now. The end-users such as plant operators can access tool's results via a web browser. The user interface could be either built using third-party visualization software (Tableau, Sisense, Power BI) or completely custom-built using front-end web frameworks like bootstrap. If building custom website, then you will also need to setup a web server (using Python, .Net, etc.) which will serve the user-interface webpage when requested through web browser. The user interface communicates with the database to display appropriate data to the end-user. The web server may be configured to execute your tool on demand as well. The web server may be hosted on a separate machine or on the tool server machine itself.

That is all it takes to deploy a ML solution in a production environment. If all this IT stuff has overwhelmed you, don't worry! It is simpler than it seems and in Chapter 14 we will build and deploy an end-to-end solution following this architecture.

`

1.5 The Future of Process Data Science

It is not an exaggeration to say that this is a wonderful time to be a process data scientist. Process industry is witnessing higher and higher product demands due to increasing population and growing lifestyle globally, but there is also a push to run production facilities more efficiently and sustainably. Consequently, adoption of Industry 4.0, which mandates utilizing process data for process improvements all along the production chain, is on the rise. There is palpable interest among process industry executives to implement ML-based solutions and the responsibility to show that the ML hype is true has fallen on the shoulders of process data scientists.

It's a foregone conclusion that ML is a powerful tool for PSE. Process data hold tremendous power if they are put to use in the right way. However, blind application of ML often leads to discouraging results. As a process data scientist with expert process knowledge and ML skills, you are in a unique position to combine process systems knowledge and power of ML to unleash the true potentials of data science in process industry. Let's cheer to your bright career prospects as a process data scientist and continue our journey to now learn the intricate details of ML algorithms.

Summary

In this chapter we tried to get a conceptual understanding of where ML fits in the world of process industry. We looked at the different types of machine learning workflows and methodologies. We also explored some application areas in process industry where ML has proved useful. We hope that you got the chapter's overarching message that process data science has already proven to be an indispensable tool in process operations to turn data into knowledge and support effective decision making. In the next chapter we will take the first step and learn about the environment you will use to execute your Python scripts containing ML code.

`

`

Part 2

Classical Machine Learning Methods

`

Chapter 5

Dimension Reduction and Latent Variable Methods (Part 1)

It is not uncommon to have hundreds of process relevant variables being measured at manufacturing facilities. If you are looking to build a machine learning model using these process variables, then the high dimensionality (number of model variables) will present you several unique challenges related to algorithmic issues (due to collinearity among variables), difficulty in visualizing data, large computational costs, and slow model training. However, this problem, referred to as the *curse of dimensionality*, should not dampen your spirits! Conservation laws such as mass balances, thermodynamics constraints, enforced product specifications, and other operational restrictions induce correlations among the process variables and make it appear as if the measured variables are all derived from a small number of hidden (un-measured) variables. Latent variable-based methods reduce process dimensionality by finding these hidden latent variables.

PCA and PLS are among the most popular latent variable-based statistical tools and have been used successfully in several process monitoring and soft sensing applications. This chapter provides a comprehensive exposition of the PCA and PLS techniques and teaches you how to apply them on process data. Specifically, the following topics are covered

- Introduction to PCA and PLS
- Process modeling and monitoring via PCA and PLS
- Fault diagnosis for root cause analysis
- Nonlinear and dynamic variants of linear PCA and PLS

`

5.1 PCA: An Introduction

Principal component analysis (PCA) is a multivariate technique that transforms a high-dimensional set of correlated variables into a low-dimensional set of uncorrelated (latent) variables with minimum loss of information. Consider the 3-dimensional data in figure 5.1. It is apparent that although the data is three dimensional, the data-points mostly lie along a 2-D plane; and even in this plane, the spread is much higher along a particular direction. PCA converts the original *(x,y,z)* space into a 2-D principal component (PC) space where the 1st PC (PC1) corresponds to the direction of maximum spread/variance in data and the 2nd PC (PC2) corresponds to the direction with highest variance among all directions orthogonal to 1st PC. Depending upon modeling requirements, even the 2nd PC may be discarded, essentially obtaining a 1-D data while losing out some information. Also, as we will see soon, it is straightforward to recover original data from data in PC space.

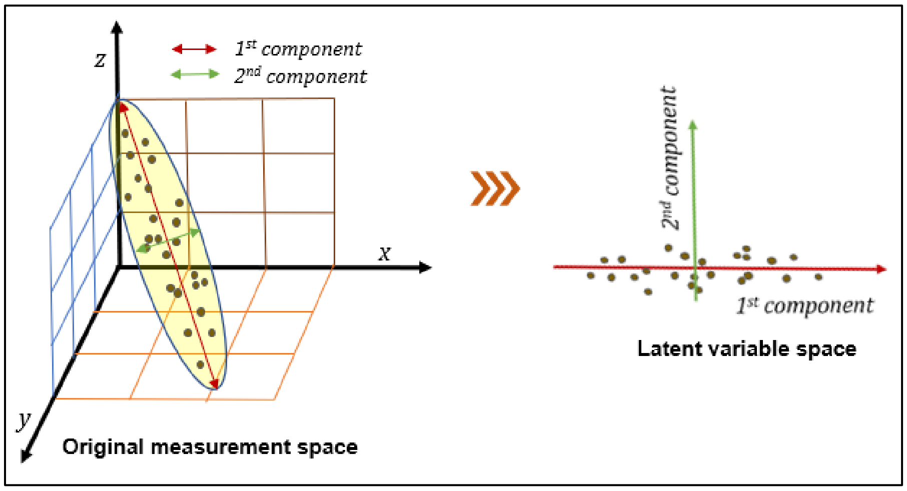


Figure 5.1: PCA illustration

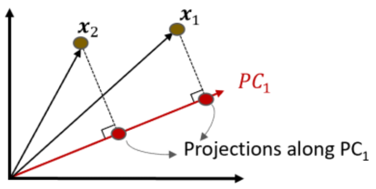
In ML world, it is common to find application of classification and clustering techniques in the PC space. In process industry, process modeling (via principal component regression (PCR)) and monitoring are common application of PCA[7]. In PCR, the process outputs are regressed onto the principal component values of the input data. By doing so, the problem of ill-conditioning frequently encountered in classical multiple linear regression (MLR) due to high input space dimensionality and high degree of correlation among input variables is circumvented. PCA is also frequently utilized for process visualization. For many applications, two or three PCs are adequate for capturing most of the variability in process data and therefore, the compressed process data can be visualized with a single plot. Plant operators and engineers use this single plot to find past and current patterns in process data.

[7] The popularity of latent-variable techniques for process control and monitoring arose from the pioneering work by John McGregor at McMaster University.

`

Mathematical background

Consider a data matrix $\boldsymbol{X} \in \mathbb{R}^{N\times m}$ consisting of *N* samples of *m* input variables where each row represents a data-point in the original measurement space. It is assumed that each column is normalized to zero mean and unit variance. Let $\boldsymbol{v} \in \mathbb{R}^{m}$ represent the 'loading' vector that projects data-points along PC1 and can be found by solving the following optimization problem



$$\max_{\boldsymbol{v}\neq 0} \frac{(\boldsymbol{Xv})^{\mathrm{T}}\boldsymbol{Xv}}{\boldsymbol{v}^{T}\boldsymbol{v}} \qquad \text{eq. 1}$$

It is apparent that eq. 1 is trying to maximize the variance of the projected data-points along PC1. Loading vectors for other PCs are found by solving the same problem with the added constraint of orthogonality to previously computed loading vectors. Alternatively, loading vectors can also be computed from eigenvalue decomposition of covariance matrix (***S***) of ***X***

$$\frac{1}{N-1}\boldsymbol{X^T}\mathbf{X} = \boldsymbol{S} = \boldsymbol{V\Lambda V^T} \qquad \text{eq. 2}$$

Above is the form you will find more commonly in PCA literature. The columns of eigenvector matrix $\boldsymbol{V} \in \mathbb{R}^{m\times m}$ are the loading vectors that we need. The diagonal eigenvalue matrix $\boldsymbol{\Lambda}$ equals *diag{*$\lambda_1, \lambda_2, \ldots, \lambda_m$*}*, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_m$ are the eigenvalues. Infact, λ_j is equal to the variance along the *j*th PC. If there is significant correlation in original data, only the first few eigenvalues will be significant. Let's assume that *k* PCs are retained, then, the first *k* columns of $\boldsymbol{V}$ (which corresponds to the first *k* λ_s) are taken to form the loading matrix $\boldsymbol{P} \in \mathbb{R}^{m\times \mathrm{k}}$. Transformed data in the PC space can now be obtained

$$\text{Projected values along j}^{th}\text{ PC} \longleftarrow \boldsymbol{t}_j = \boldsymbol{Xp}_j \; \boldsymbol{or} \;\; \boldsymbol{T} = \boldsymbol{XP} \qquad \text{eq. 3}$$

The m dimensional *i*th row of *X* has been transformed into *k* (< *m*) dimensional *i*th row of *T*. $\boldsymbol{T} \in \mathbb{R}^{N\times k}$ is called score matrix and the *j*th column of *T* (t_j) contains the (score) values along the *j*th PC. The scores can be projected back to the original measurement space as follows

$$\hat{\boldsymbol{X}} = \boldsymbol{TP}^{T} \qquad \text{eq. 4}$$

Note that because we discarded the loading vectors corresponding to insignificant λ_s, $\hat{\boldsymbol{X}} \neq \boldsymbol{X}$. The difference $\boldsymbol{E} = \boldsymbol{X} - \hat{\boldsymbol{X}}$ is referred to as residual matrix as each row is the residual or error vector for a data-point. Overall, the PC space captures the systematic trends in process data and the residual space primarily describe the noise in data.

`

Dimensionality reduction for Polymer Manufacturing Process

Let us now see the powerful dimensionality reduction capability of PCA in action. We will use data from a polymer manufacturing facility. The dataset contains 33 variables and 92 hourly samples (Figure 5.2).

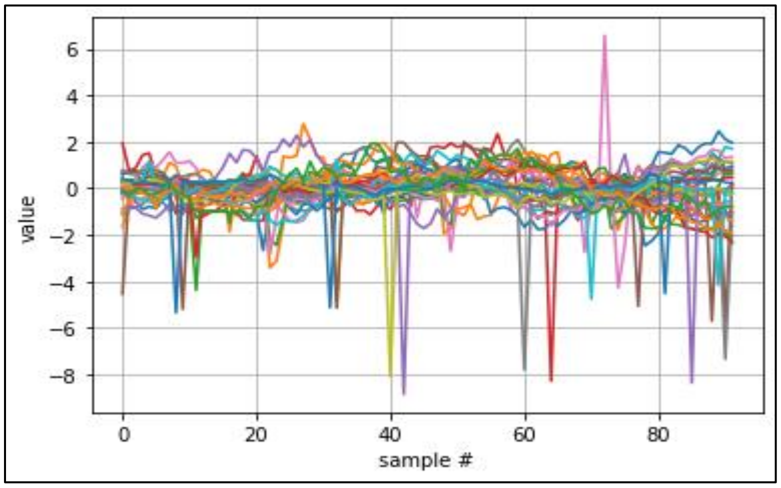


Figure 5.2: Process data from a polymer manufacturing plant. Each colored curve corresponds to a process variable

For this dataset, it is reported that the process started behaving abnormally around sample 70 and eventually had to be shut down. Therefore, we use samples 1 to 69 for training the PCA model using the code below. The rest of the data will be utilized for process monitoring illustration later.

```
# import requisite libraries
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# fetch data and separate training data
data = pd.read_excel('proc1a.xls', skiprows = 1, usecols = 'C:AI')
data_train = data.iloc[0:69,]

# normalize data
scaler = StandardScaler()
data_train_normal = scaler.fit_transform(data_train)

# PCA
pca = PCA()
score_train = pca.fit_transform(data_train_normal)
```

`

After training the PCA model, loading vectors/principal components can be accessed from transpose of the components_ attribute of pca model. Note that we have not accomplished any dimensionality reduction yet. PCA has simply provided us an uncorrelated dataset in score_train. To confirm this, we can compute the correlation coefficients among the columns of score_train. Only the diagonal values are 1 while the rest of the coefficients are 0!

```
# confirm no correlation
corr_coef = np.corrcoef(score_train, rowvar = False)
>>> print('Correlation matrix: \n', corr_coef[0:3,0:3]) # printing only a portion

Correlation matrix:
 [[ 1.00000000e+00  8.24652750e-16 -1.88830953e-16]
 [ 8.24652750e-16  1.00000000e+00  2.36966153e-16]
 [-1.88830953e-16  2.36966153e-16  1.00000000e+00]]
```

For dimensionality reduction we will need to study the variance along each PC. Note that the sum of variance along the *m* PCs equals the sum of variance along the *m* original dimensions. Therefore, the variance along each PC is also called explained variance. The attribute explained_variance_ratio gives the fraction of variance explained by each PC and Figure 5.3 clearly shows that not all 33 components are needed to capture all the information in data. Most of the information is captured in the first few PCs itself.

```
# visualize explained variance
import matplotlib.pyplot as plt

explained_variance = 100*pca.explained_variance_ratio_ # in percentage
cum_explained_variance = np.cumsum(explained_variance) # cumulative % variance explained

plt.figure()
plt.plot(cum_explained_variance, 'r+', label = 'cumulative % variance explained')
plt.plot(explained_variance, 'b+', label = 'variance explained by each PC')
plt.ylabel('Explained variance (in %)'), plt.xlabel('Principal component number'), plt.legend()
```

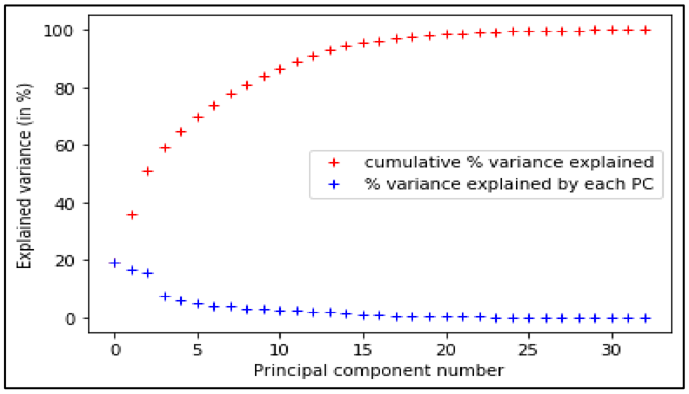


Figure 5.3: Variance explained by principal components

`

A popular approach for determining the number of PCs to retain is to select the number of PCs that cumulatively capture atleast 90% (or 95%) of the variance. The captured variance threshold should be guided by the expected level of noise or non-systematic variation that you do not expect to be captured. Alternative methods include cross-validation, scree tests, AIC criterion, etc. However, none of these methods are universally best in all the situations.

```
# decide # of PCs to retain and compute reduced data in PC space
n_comp = np.argmax(cum_explained_variance >= 90) + 1
score_train_reduced = score_train[:,0:n_comp]

>>> print('Number of PCs cumulatively explaining atleast 90% variance: ', n_comp)

Number of PCs cumulatively explaining atleast 90% variance: 13
```

Thus, we have achieved ~60% reduction in dimensionality (from 33 to 13) by sacrificing just 10% of the information. To confirm that only about 10% of the original information has been lost, we will reconstruct the original normalized data from the scores. Figure 5.4 provides a visual confirmation as well where it is apparent that the systematic trends in variables have been reconstructed while noisy fluctuations have been removed.

```
# confirm that only about 10% of original information is lost
from sklearn.metrics import r2_score

V_matrix = pca.components_.T
P_matrix = V_matrix[:,0:n_comp]

data_train_normal_reconstruct = np.dot(score_train_reduced, P_matrix.T)
R2_score = r2_score(data_train_normal, data_train_normal_reconstruct)

>>> print('% information lost = ', 100*(1-R2_score))

% information lost = 9.0469
```

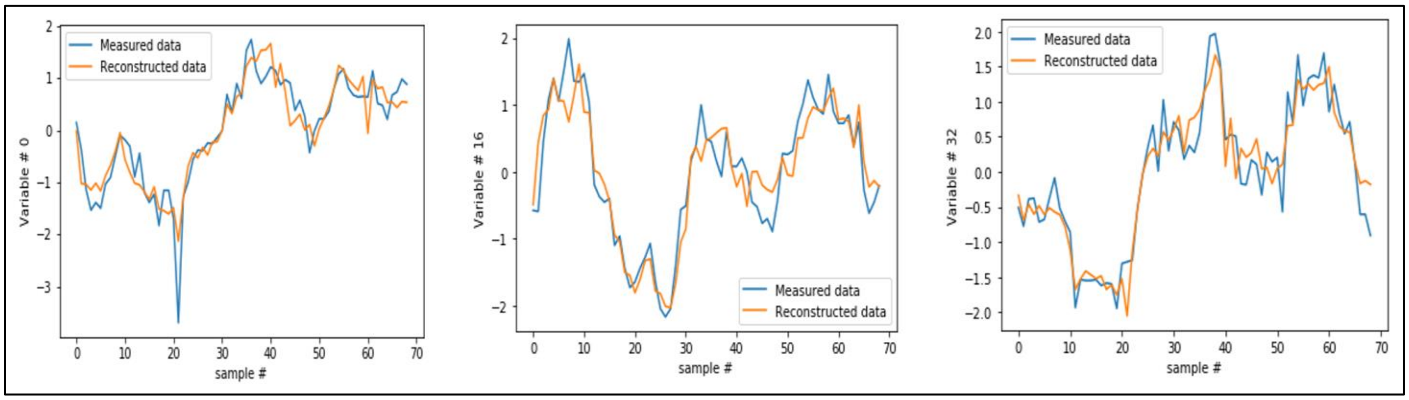


Figure 5.4: Comparison of measured and reconstructed values for a few variables

`

The 90% threshold could also have been specified during model training itself through the n_components parameter: pca = PCA(n_components = 0.9). In this case the insignificant PCs are not computed and the score_train_reduced matrix can be computed from the model using the inverse_transform method.

```
# alternative approach
pca = PCA(n_components = 0.9)
score_train_reduced = pca.fit_transform(data_train_normal)

data_train_normal_reconstruct = pca.inverse_transform(score_train_reduced)
R2_score = r2_score(data_train_normal, data_train_normal_reconstruct)

>>> print('% information lost = ', 100*(1-R2_score))

% information lost = 9.0469
```

5.2 Process Monitoring via PCA for Polymer Manufacturing Process

In Figure 5.2, we saw that it was not easy to infer process abnormality after 69th sample by simply looking at the combined time-series plot of all the available variables. Individual variable plot may provide better clues, but continuously monitoring all the 33 plots of individual variables is not a convenient task.

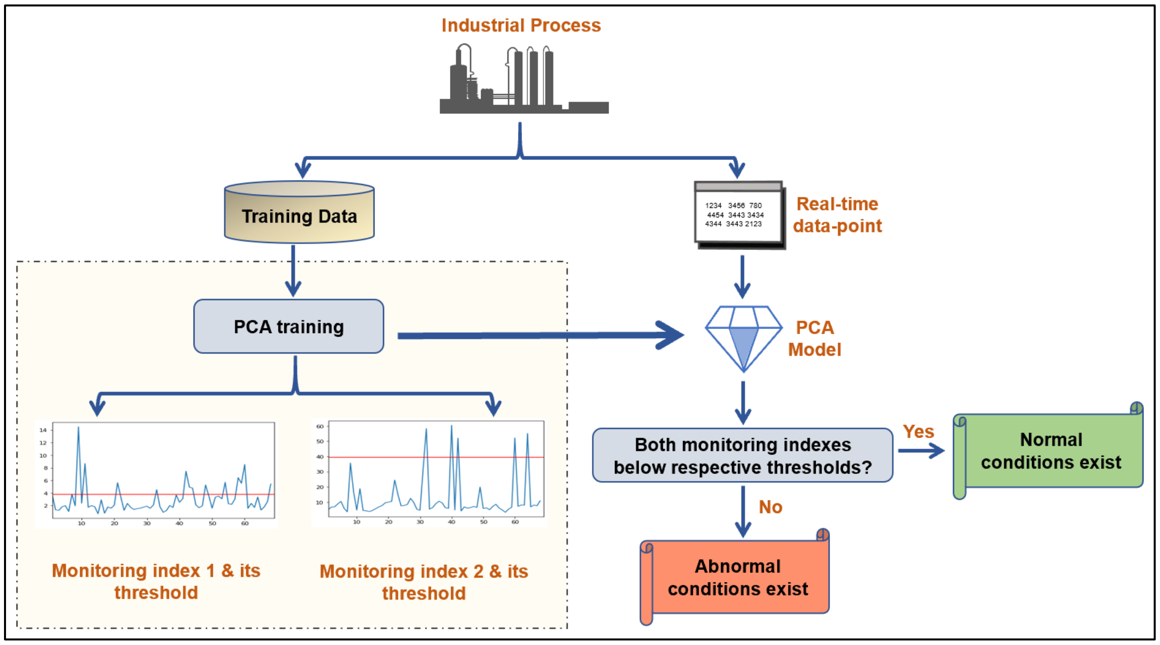

Industrial Process
Training Data
Real-time data-point
PCA training
PCA Model
Both monitoring indexes below respective thresholds?
Yes
Normal conditions exist
No
Abnormal conditions exist
Monitoring index 1 & its threshold
Monitoring index 2 & its threshold

Figure 5.5: PCA-based process monitoring workflow

`

PCA makes the monitoring task easy by summarizing the state of any complex multivariate process into two simple indicators or monitoring indices as shown in Figure 5.5. During model training, statistical thresholds are determined for the indices and for a new data-point, the new indices' values are compared against the thresholds. If any of the two thresholds are violated, then presence of abnormal process conditions is confirmed.

Process monitoring/fault detection indices

`

Part 3

Artificial Neural Networks & Deep Learning

`

Chapter 13

Reinforcement Learning

The ML algorithms we have learnt till now rely upon supply of all modeling-relevant input/output data prior to model training. In contrast, reinforcement training (RL) takes ML a step further and is designed to collect the required training data by itself through interactions with the physical system that it is trying to learn about. Through trial-and-error, an RL model learns what actions to take to accomplish any given task. During training, actions that result in favorable results/rewards get reinforced and after multiple interactions, the model eventually learns an optimal action plan/policy! Sounds impressive, right?

RL mimics how we, humans, learn things (such as riding a cycle) through trial & error and environment interactions. This concept opens up a plethora of potential RL applications. You have probably already heard or seen some of the remarkable feats achieved by RL models such as computers playing games better than the best human players or humanoid robots learning how to run, etc. In this chapter, we will focus on process industry-related applications of RL, specifically for process control.

RL is a very broad and constantly evolving field. There are a lot of RL-specific terminology and concepts. We will declutter the world of RL in this chapter, and you will learn how to setup and solve an RL problem. Specifically, this chapter covers the following topics

- Introduction to RL and RL agents as process controllers
- Introduction to Q-learning, deep RL, and Actor-Critic architecture
- Introduction to DDPG algorithm for handling process industry-relevant problems
- Tank level control using an RL agent

`

13.1 Reinforcement Learning: An Introduction

RL is the branch of ML wherein an agent repeatedly interacts with its environment to learn the best way to accomplish a task or the optimal action policy. Figure 13.1 shows the basic setup of reinforcement learning. As shown, the RL agent receives information about the current state ($\mathbf{s}_t$) of the environment based on which an action ($\mathbf{a}_t$) is decided (as per agent's action policy). As a result of the action, the environment moves to a new state ($\mathbf{s}_{t+1}$) and generates a scalar reward (r_{t+1}) indicating how good was the taken action. Before the agent takes another action, the learning algorithm uses the information ($\mathbf{s}_t$, $\mathbf{a}_t$, r_{t+1}, $\mathbf{s}_{t+1}$) to improve its policy and then the cycle continues. Eventually, an optimal policy is obtained that maps environment states to optimal actions such that the total rewards earned till task completion is maximized. Once trained, the learning process can be stopped, and the policy function can be deployed.

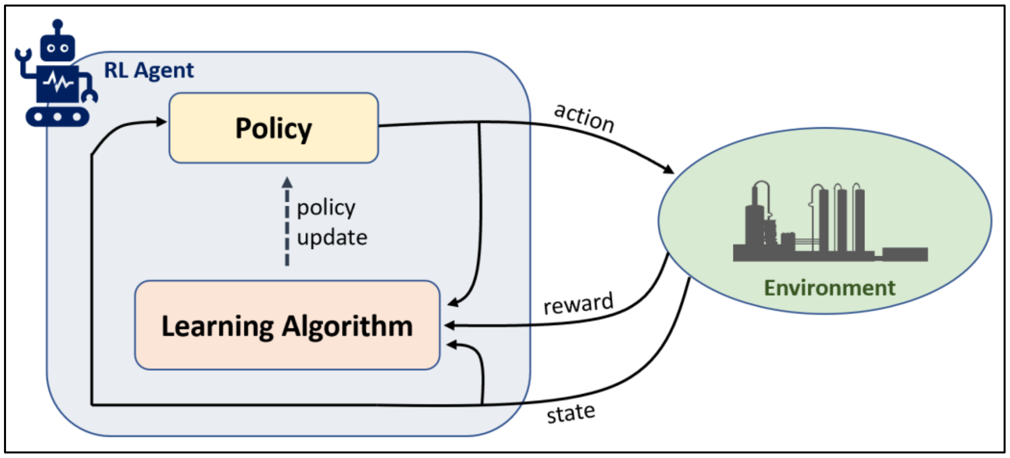


Figure 13.1: Reinforcement learning setup depicting an agent's interactions with its environment

A simple real-life analogy could be the task of finding the optimal driving route from your office to home in a new town. Here, you are the agent and environment would comprise of your car, city's roads and highways network, traffic, weather, your geospatial location, and basically everything excluding you. The total reward to maximize could be the negative of the time taken to reach home (less driving time is better). During driving, depending on the environment state, you would take decisions on whether to take any highway exit or make any turn or not. Assuming no internet (and google maps!), being new in town, you would not know if taking those exits or turns will help you reach home faster and therefore you will explore different possible routes. After several trials, you would gain a good understanding of the town and eventually would be able to take the most optimal action for any given environment state at any point during driving. RL follows the same methodology to find the optimal mapping using some systematic (and very smart) learning algorithms.

`

RL for process control

The RL framework of observing an environment's current state and deciding upon corresponding optimal actions fits quite naturally to the setup of process controllers in process industry. You might already know that these controllers are tasked with keeping process operations optimal by keeping critical process variables (product purity, fluid level, temperatures, etc.) at optimal values through adjustments in manipulated variables (stream flows, valve openings, etc.). Figure 13.2 shows the typical setup of a process controller and the division of this setup into agent/environment framework. Rewards are designed as some suitable combination of reference signal, environment state, and controller action.

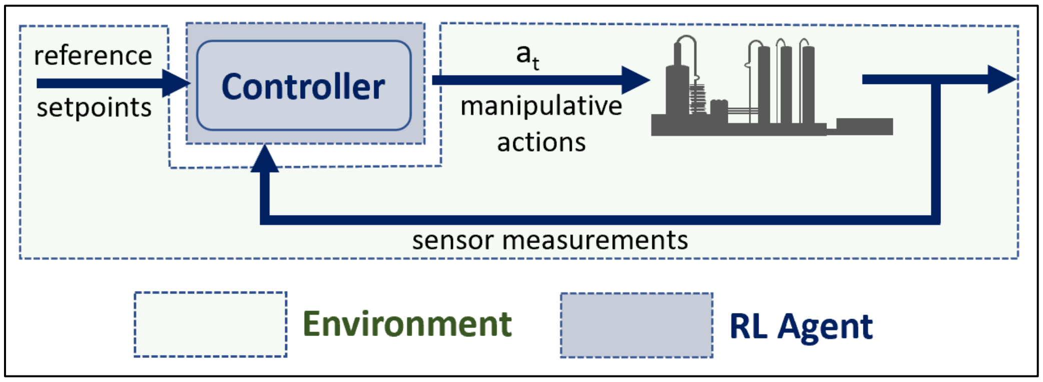


Figure 13.2: Process control system in RL framework

In process industry, although PID and MPC controllers are well-established, their shortcomings are well-known. While PID controllers perform unsatisfactorily for complex nonlinear systems, MPCs solve optimization problems using process models which makes online action computation infeasible for large-scale nonlinear systems. Moreover, both these controllers suffer performance degradation issues (due to changing process conditions, process drift) over time, necessitating regular maintenance. Controller maintenance entails re-identification of process models which can be time and resource intensive and may require interference to normal plant operations for training data collection.

Given the aforementioned issues with the current state-of-the-art for the process controllers and the recent successes of RL, interest in leveraging RL technology for process control has been reignited. Several recent studies[8] have demonstrated how RL-based controllers can provide superior performance. Not requiring online optimization (because optimal action policy is pre-computed), easy adaptation of action policy under changing environment by

[8] Cassol et. Al., Reinforcement learning applied to process control: A Van der Vusse reactor case study, Computer Aided Chemical Engineering, 2018
Rajesh Siraskar, Reinforcement learning for control of valves, Machine Learning with Applications, 2021
Ma et. Al., Continuous control of a polymerization system with deep reinforcement learning, Journal of Process Control, 2019

`

continued learning with new process data are some characteristics that make RM-based controllers very promising.

You may be slightly concerned about training an RL agent in the real plant environment because the generated actions in early stages of training can be 'very bad' and even unsafe. Moreover, for complex system, thousands of interactions may be required to reach even a reasonably good policy; plant managers will certainly never agree to this! These are valid concerns and therefore, the common practice is to use a sufficiently accurate model of the plant and train RL agent offline in a simulated environment. Once offline learning is complete, the learning process can be turned off and RL agent deployed in real plant. There are, however, a few good reasons to keep learning on (continually or sporadically) post-deployment. First, your plant model will probably not be 100% accurate. Therefore, the RL agent may use some online interactions to fine-tune its policy. Second, as alluded to before, the plant behavior may change over time and the agent will need to tweak its policy to re-adjust to changes in its environment.

This discussion also brings us to the point of model-free vs model-based RL. Model-free RL doesn't use any environment/plant model during training and learns its policy based solely on its interactions with real environment. Model-based RL, on the other hand, uses a model either for simulating the environment or assisting the learning algorithm.

13.2 RL Terminology & Mathematical Concepts

We have showered enough praises on RL. Let's now get down to understanding how RL actually works. For this, we will first learn some RL terminology and concepts. A quick disclaimer here that some of these new concepts may seem 'abstract' and not immediately useful. But, as you read through this chapter, we promise that all dots will connect and their utility will become 'obvious'.

Environment and Markov decision process

While informally an environment is anything that an RL agent interacts with, in a formal setting, it is represented as a Markov decision process (MDP). A MDP is named such because it exhibits Markov property, i.e., the (probability of) transition from state $\mathbf{s}_t$ to $\mathbf{s}_{t+1}$ depend only

`

on the most recent state (**s**t) and action (**a**t). Such memoryless characteristic is important in RL because it allows the agent to only consider the current state when deciding the next optimal action and not worry about what actions were previously taken to reach **s**t.

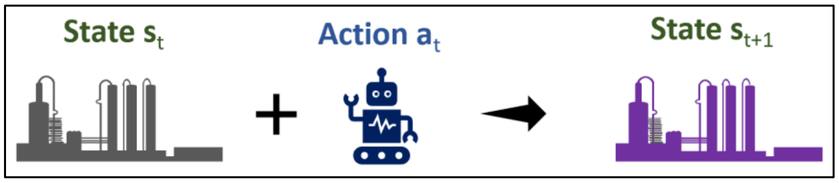
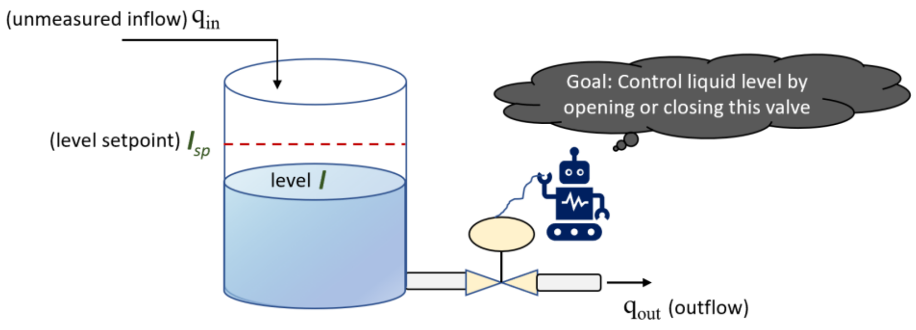


Figure 13.3: Transition in an MDP depend only on current state and action

In an MDP, selection of features that completely characterize the state of the environment and enable the agent to act on its basis becomes crucial. For example, consider the following problem of controlling the liquid level in a tank


(unmeasured inflow) q_{in}
(level setpoint) l_{sp}
level l
Goal: Control liquid level by opening or closing this valve
q_{out} (outflow)

Here, *l*, *lsp*, and the rate of change of *l* could be used to define the environment state. *lsp* can also be substituted with *l* - *lsp* as an alternative formulation. Depending on your problem formulation, data from the past may also be included in the state vector to facilitate the agent's decision making.

`

`

Part 4

Deploying ML Solutions Over Web

`

Chapter 14

Process Monitoring Web Application

You have done all the hard work to obtain a ML model that meets performance criteria and now it's time to deliver the solution / model to its end-users who will be using the model's results on a regular basis. But how do you do it? If the end-users are non-technical (from data-science perspective) like plant operators, you cannot ask them to have their own Python installation to run the ML model. Under such circumstances, deployment over web is a good and frequently employed solution for delivering ML results.

For web deployment, Python provides several frameworks (like Django, Flask, CherryPy) for developing simple to complex web application fast. Very often, all you may want is a quick prototype that allows you to collect user feedback. Keeping this in mind, we will show you how to build a light-weight web application from scratch and demonstrate how easy it is to do so. Specifically, we will build a process monitoring tool that provides 24 X 7 fault detection & diagnosis (FDD) results to plant operators over web. During the course of building this solution, we will learn the following topics

- Deploying Python applications using CherryPy's web server
- Embedding ML models into a web application
- Saving ML models for later reuse
- Building front-end user interfaces using HTML, CSS, Javascript

---

`

14.1 Process Monitoring Web App: Introduction

Figure 14.1 shows the user interface that we will build in this chapter to display the results in real-time from a process monitoring ML model. The user interface is to be accessed via a web browser and provides two crucial information to the plant operators. First, the fault detection component communicates to the operators the state of the process and alerts them if any process abnormality is detected. Second, the fault diagnosis component identifies the faulty variables responsible for process abnormality. The monitoring model will be built using PCA and the methodology introduced in Chapter 5 will be employed for FDD. The process system is the same as that in Chapter 5, i.e., the polymer processing plant with 33 variables.

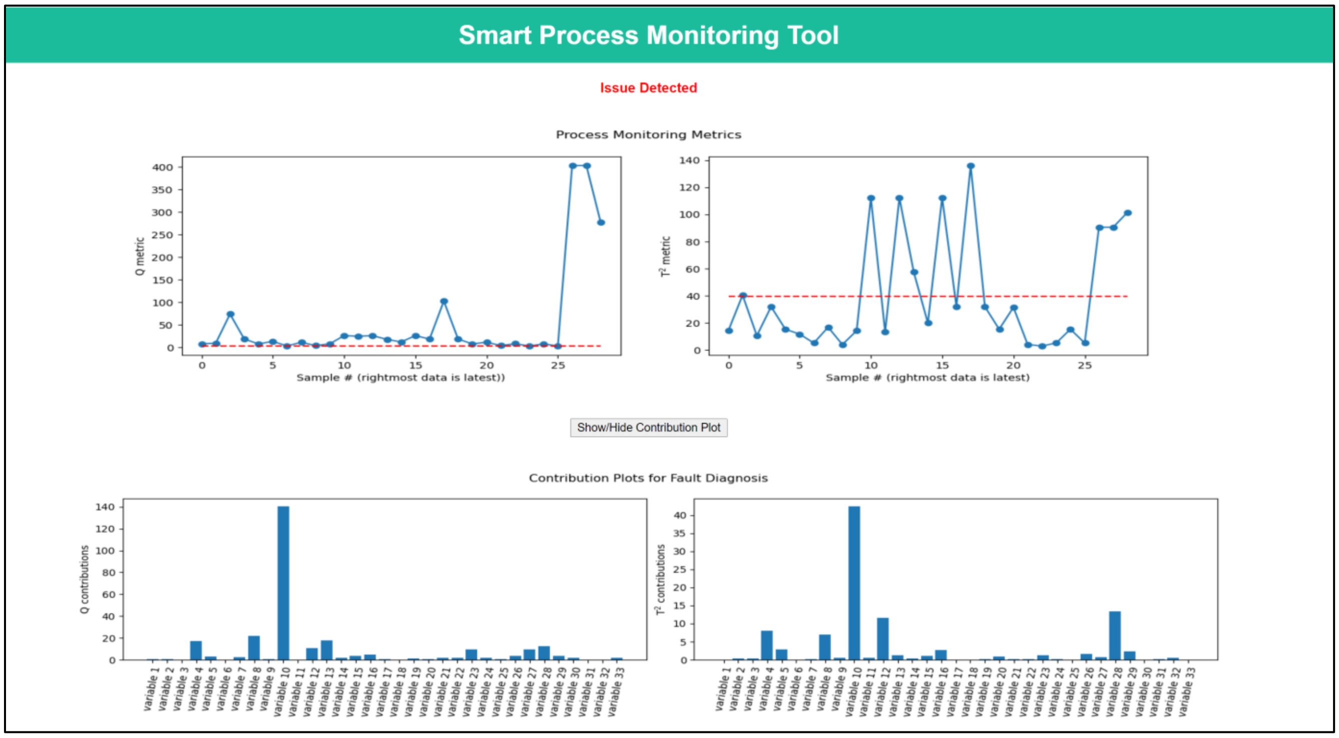

Smart Process Monitoring Tool
Issue Detected
Process Monitoring Metrics
Q metric
T² metric
Sample # (rightmost data is latest))
Sample # (rightmost data is latest)
Show/Hide Contribution Plot
Contribution Plots for Fault Diagnosis
Q contributions
T² contributions

Figure 14.1: Process monitoring web app user interface

A web application primarily has 2 main parts: front-end and back-end. Front-end (or client-side) is the part that end-users see and interact with directly through web browsers. The back-end works behind the scenes to deliver information to the browsers. When a user enters your website's URL in browser, a request is sent to the back-end which parses this request, processes it, and sends back a response which is displayed on the front-end. Illustration below shows the data transmission scheme that we will employ

`

`