Artificial Intelligence in Industry

Last update: 23 October 2024

Academic Year 2024 – 2025

Alma Mater Studiorum · University of Bologna

Contents

1 Preliminaries

Problem formalization Defines the ideal goal.

Problem formalization

Solution formalization Defines the actual possible approaches to solve a problem.

Solution formalization

Occam's razor Principle for which, between two hypotheses, the simpler one is usually correct.

Occam's razor

> **Remark.** This approach has less variance and more bias, making it more robust.

2 Low dimensional anomaly detection: Taxi calls

Anomaly Event that deviates from the usual pattern.

Anomaly

Time series Data with an ordering (e.g., chronological).

Time series

2.1 Data

The dataset is a time series and it is a `DataFrame` with the following fields:

`timestamp` with a 30 minutes granularity.

`value` number of calls.

The label is a `Series` containing the timestamps of the anomalies.
An additional `DataFrame` contains information about the time window in which the anomalies happen:

`begin` acceptable moment from which an anomaly can be detected.

`end` acceptable moment from which there are no anomalies anymore.

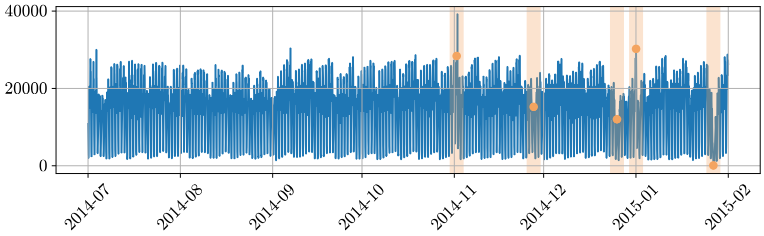


Figure 2.1: Plot of the time series, anomalies, and windows

2.2 Approaches

2.2.1 Gaussian assumption

Assuming that the data follows a Gaussian distribution, mean and variance can be used to determine anomalies through a threshold. z-score can also be used.

2.2.2 Characterize data distribution

Classify a data point as an anomaly if it is too unlikely.

Problem formalization Given a random variable X with values x to represent the number of taxi calls, we want to find its probability density function (PDF) $f(x)$.

An anomaly is determined whether:

$$f(x) \leq \varepsilon$$

where ε is a threshold.

> **Remark.** A PDF can be reasonably used even though the dataset is discrete if its data points are sufficiently fine-grained.

> **Remark.** It is handy to use negated log probabilities as:
>
> - The logarithm adds numerical stability.
> - The negation makes the probability an alarm signal, which is a more common measure.
>
> Therefore, the detection condition becomes:
>
> $$-\log f(x) \geq \varepsilon$$

Solution formalization The problem can be tackled using a density estimation technique.

2.2.2.1 Univariate kernel density estimation

Kernel density estimation (KDE) Based on the assumption that whether there is a data point, there are more around it. Therefore, each data point is the center of a density kernel.

Kernel density estimation (KDE)

Density kernel A kernel $K(x, h)$ is defined by:

- The input variable x.
- The bandwidth h.

Gaussian kernel Kernel defined as:

$$K(x, h) \propto e^{-\frac{x^2}{2h^2}}$$

where:

- The mean is 0.
- h is the standard deviation.

As the mean is 0, an affine transformation can be used to center the kernel on a data point μ as $K(x - \mu, h)$.

Given m training data points $\bar{x}_i$, the density of any point x can be computed as the kernel average:

$$f(x, \bar{x}, h) = \frac{1}{m} \sum_{i=0}^{m} K(x - \bar{x}_i, h)$$

Therefore, the train data themselves are used as the parameters of the model while the bandwidth h has to be estimated.

Data split Time series are usually split chronologically:

Train Should ideally contain only data representing the normal pattern. A small amount of anomalies might be tolerated as they have low probabilities.

Validation Used to find the threshold ε.

Test Used to evaluate the model.

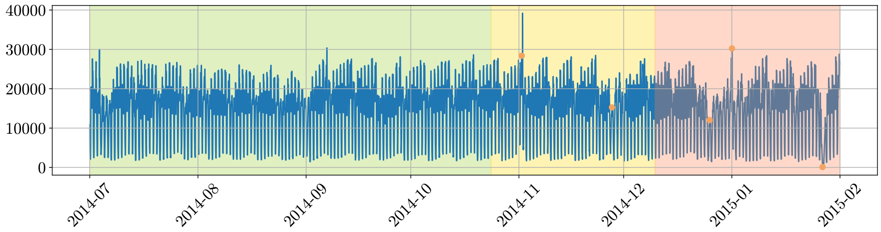


Figure 2.2: Train, validation, and test splits

Metrics It is not straightforward to define a metric for anomaly detection. A cost model to measure the benefits of a prediction is more suited. A simple cost model can be based on:

True positives (TP) Windows for which at least an anomaly is detected;

False positives (FP) Detections that are not actually anomalies;

False negatives (FN) Undetected anomalies;

Advance (adv) Time between an anomaly and when it is first detected;

and is computed as:

$$(c_{\text{false}} \cdot \texttt{FP}) + (c_{\text{miss}} \cdot \texttt{FN}) + (c_{\text{late}} \cdot \texttt{adv}_{\leq 0})$$

where c_{false}, c_{miss}, and c_{late} are hyperparameters.

Bandwidth estimation According to some statistical arguments, a rule-of-thumb to estimate h in the univariate case is the following:

$$h = 0.9 \cdot \min \left\{ \hat{\sigma}, \frac{\texttt{IQR}}{1.34} \right\} \cdot m^{-\frac{1}{5}}$$

where:

- $\texttt{IQR}$ is the inter-quartile range.
- $\hat{\sigma}$ is the standard deviation computed over the whole dataset.

Threshold optimization Using the train and validation set, it is possible to find the best threshold ε that minimizes the cost model through linear search.

> **Remark.** The train set can be used alongside the validation set to estimate ε as this operation is not used to prevent overfitting.

> **Remark.** The evaluation data should be representative of the real world distribution. Therefore, in this case, to evaluate the model the whole dataset can be used.

> **Remark.** KDE assumes that the Markov property holds. Therefore, each data point is considered independent to the others.

2.2.2.2 Multivariate kernel density estimation

Remark. In this dataset, nearby points tend to have similar values.

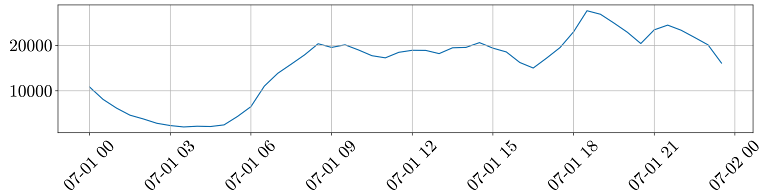


Figure 2.3: Subset of the dataset

Autocorrelation plot Plot to visualize the correlation between nearby points of a series. Given the original series, it is duplicated, shifted by a lag l, and the Pearson correlation coefficient is then computed between the two series. This operation is repeated over different values of l.

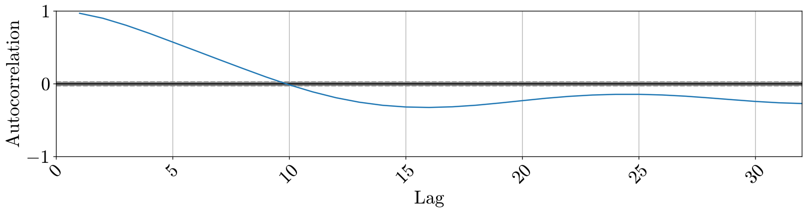


Figure 2.4: Autocorrelation plot of the subset of the dataset. There is strong correlation up to 4-5 lags.

Sliding window Given a window size w and a stride s, the dataset is split into sequences of w continuous elements.

Remark. Incomplete sequences at the start and end of the dataset are ignored.

Remark. In `pandas`, the `rolling` method of `Dataframe` allows to create a slicing window iterator. This approach creates the windows row-wise and also considers incomplete windows. However, a usually more efficient approach is to construct the sequences column-wise by hand.

Multivariate KDE Extension of KDE to vector variables.

Window size estimation By analyzing the autocorrelation plot, an ideal window size can be picked as the lag with a low correlation (e.g., 10 according to Figure 2.4).

Bandwidth estimation Differently from the univariate case, the bandwidth has to be estimated by maximizing the log-likelihood on the validation set. Given:

- The validation set $\tilde{x}$,
- The observations x,
- The bandwidth h,
- The density estimator $\hat{f}$

the likelihood is computed, by assuming independent observations, as:

$$L(h, x, \tilde{x}) = \prod_{i=1}^{m} \hat{f}(x_i, \tilde{x}_i, h)$$

Maximum likelihood estimation is defined as:

$$\arg\max \mathbb{E}_{x\sim f(x),\tilde{x}\sim f(x)}[L(h,x,\tilde{x})]$$

where $f(x)$ is the true distribution.

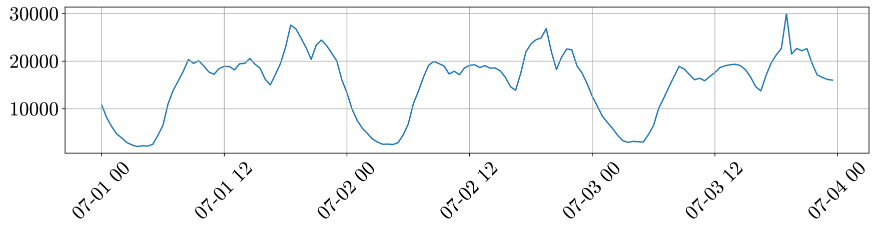
In practice, the expected value is sampled from multiple validation and training sets through cross-validation and grid-search.

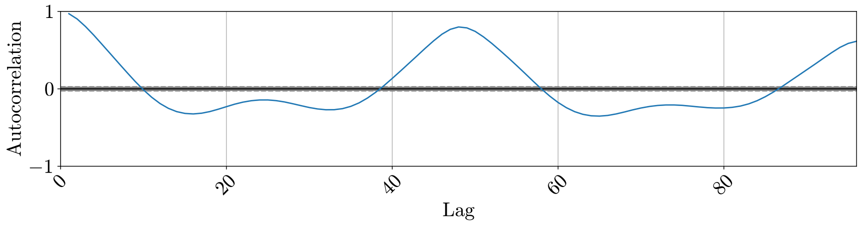
> **Remark.** As different folds for cross-validation must be tried, grid-search is an expensive operation.

Threshold optimization The threshold can be determined as in Section 2.2.2.1.

2.2.2.3 Time-dependent estimator

The approach taken in Sections 2.2.2.1 and 2.2.2.2 is based on the complete timestamp of the dataset. Therefore, the same times on different days are treated differently. However, it can be seen that this time series is approximately periodic, making the approach used so far more complicated than necessary.





Time input It is possible to take time into consideration by adding it as a parameter of the density function:

$$f(t,x)$$

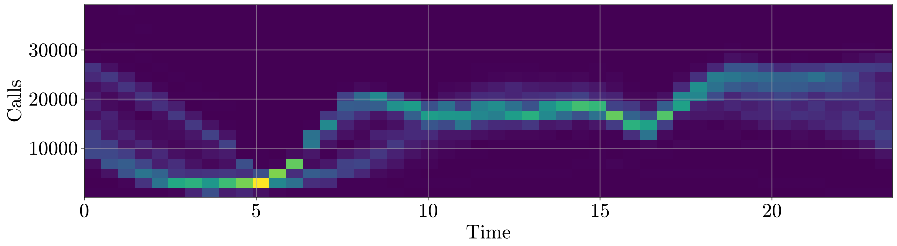


Figure 2.5: 2D histogram of the distribution. Lighter colors indicate a higher frequency of occurrence.

However, t is a controlled variable and is completely predictable (i.e., if times are samples with different frequencies, they should not affect the overall likelihood). Therefore, a conditional density should be used:

$$f(x \mid t) = \frac{f(t, x)}{f(t)}$$

where $f(t)$ can be easily computed using KDE on the time data.

> **Remark.** For this dataset, the time distribution is uniform. Therefore, $f(t)$ is a constant and it is correct to use $f(t, x)$ as the estimator.

Bandwidth estimation The bandwidth can be estimated as in Section 2.2.2.2.

> **Remark.** When using the `scikit-learn`, the dataset should be normalized as the implementation of KDE use the same bandwidth for all features.

Threshold optimization The threshold can be determined as in Section 2.2.2.1.

2.2.2.4 Time-indexed model

Ensemble model Model defined as:

Ensemble model

$$f_{g(t)}(x)$$

where:

- f_i are estimators, each working on subsets of the dataset and solving a smaller problem.
- $g(t)$ determines which particular f_i should solve the input.

Time-indexed model Consider both time and sequence inputs by using an ensemble model. Each estimator is specialized on a single time value (i.e., an estimator for `00:00`, one for `00:30`, ...).

Bandwidth estimation The bandwidth can be estimated as in Section 2.2.2.2.

Threshold optimization The threshold can be determined as in Section 2.2.2.1.

3 High dimensional anomaly detection: HPC centers

3.1 Data

The dataset is a time series with the following fields:

`timestamp` with a 5 minutes granularity.

HPC data technical data related to the cluster.

`anomaly` indicates if there is an anomaly.

In practice, the cluster has three operational modes:

Normal the frequency is proportional to the workload.

Power-saving the frequency is always at the minimum.

Performance the frequency is always at the maximum.

For this dataset, both power-saving and performance are considered anomalies.

3.1.1 High-dimensional data visualization

Individual plots Plot individual columns.

Statistics Show overall statistics (e.g., `pandas describe` method)

Heatmap Heatmap with standardized data.

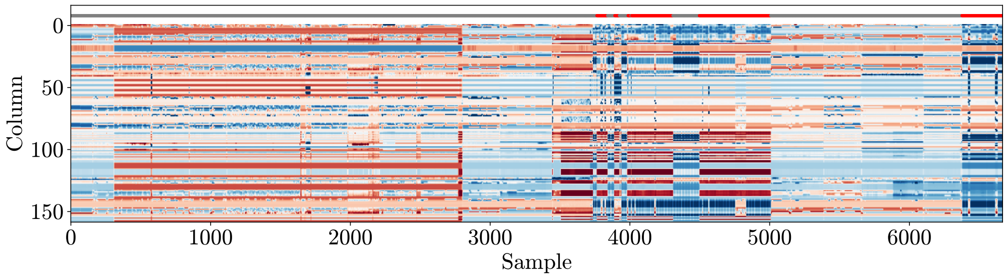


Figure 3.1: Heatmap of the dataset. On the top line, gray points represent normal behavior while red points are anomalies. In the heatmap, white tiles represent the mean, red tiles represent values below average, and blue tiles represent values above average.

3.2 Approaches

3.2.1 Multivariate KDE

Given the train, validation, and test splits, the dataset is standardized using the training set alone. This avoids leaks from the test set.
The KDE model, bandwidth, and threshold are fitted as in Chapter 2.

Cost model A simple cost model can be based on:

- False positives,
- False negatives,
- A time tolerance.

Problems KDE on this dataset has the following problems:

- It is highly subject to the curse of dimensionality and requires more data to be reliable.

  > **Remark.** KDE does not compress the input features and grows with the training set. In other words, it has low bias and high variance.

- As the dataset is used during inference, a large dataset is computationally expensive (with time complexity $O(mn)$, where m is the number of dimensions and n the number of samples).
- It only provides an alarm signal and not an explanation of the anomaly. In low-dimensionality, this is still acceptable, but with high-dimensional data it is harder to find an explanation.

3.2.2 Gaussian mixture model

Gaussian mixture model (GMM) Selection-based ensemble that estimates a distribution as a weighted sum of Gaussians. It is assumed that data can be generated by the following probabilistic model:

Gaussian mixture model (GMM)

$$X_Z$$

where:

- X_k are random variables following a multivariate Gaussian distribution. The number of components is a hyperparameter that can be tuned to balance bias-variance.
- Z is a random variable representing the index k of the variable X to use to generate data.

More specifically, the PDF of a GMM g is defined as:

$$g(x, \mu, \Sigma, \tau) = \sum_{k=1}^{n} \tau_k f(x, \mu_k, \Sigma_k)$$

where:

- f is the PDF of a multivariate normal distribution.
- μ_k and Σ_k are the mean and covariance matrix for the k-th component, respectively.

- τ_k is the weight for the k-th component and corresponds to $\mathcal{P}(Z = k)$.

Training Likelihood maximization can be used to train a GMM to approximate another distribution:

$$\arg\max_{\mu,\Sigma,\tau} \mathbb{E}_{x\sim X}\left[L(x, \mu, \Sigma, \tau)\right] \qquad \text{subject to } \sum_{k=1}^{n} \tau_k = 1$$

where the expectation can be approximated using the training set (i.e., empirical risk minimization):

$$\mathbb{E}_{x\sim X}\left[L(x, \mu, \Sigma, \tau)\right] \approx \prod_{i=1}^{m} g(x_i, \mu, \Sigma, \tau)$$

> **Remark.** Empirical risk minimization can be solved in two ways:
>
> - Use a single large sample (traditional approach).
> - Use many smaller samples (as in cross-validation).

By putting the definitions together, we obtain the following problem:

$$\arg\max_{\mu,\Sigma,\tau} \prod_{i=1}^{m} \sum_{k=1}^{n} \tau_k f(x, \mu_k, \Sigma_k) \qquad \text{subject to } \sum_{k=1}^{n} \tau_k = 1$$

which:

- Cannot be solved using gradient descent as it is an unconstrained method.
- Cannot be solved using mixed-integer linear programming as the problem is non-linear.
- Cannot be decomposed as the variables μ, Σ, and τ appear in every term.

It is possible to simplify the formulation of the problem by introducing new variables:

- A new latent random variable Z_i is added for each example. $Z_i = k$ iff the i-th example is drawn from the k-th component. The PDF of the GMM can be reformulated to use Z_i and without the summation as:

  $$\tilde{g}_i(x_i, z_i, \mu, \Sigma, \tau) = \tau_{z_i} f(x, \mu_k, \Sigma_k)$$

  The expectation becomes:

  $$\mathbb{E}_{x\sim X, \{z_i\}\sim\{Z_i\}}\left[L(x, z, \mu, \Sigma, \tau)\right] \approx \mathbb{E}_{\{z_i\}\sim\{Z_i\}}\left[\prod_{i=1}^{m} \tilde{g}_i(x_i, z_i, \mu, \Sigma, \tau)\right]$$

  Note that, as the distributions to sample z_i are unknown, Z_i cannot be approximated in the same way as X.

- New variables $\tilde{\tau}_{i,k}$ are introduced to represent the distribution of the Z_i variables. In other words, $\tilde{\tau}_{i,k}$ corresponds to $\mathcal{P}(Z_i = k)$. This allows to approximate the expectation as:

  $$\mathbb{E}_{\hat{x}\sim X, \hat{z}\sim Z}\left[L(\hat{x}, \hat{z}, \mu, \Sigma, \tau)\right] \approx \prod_{i=1}^{m} \prod_{k=1}^{n} \tilde{g}_i(x_i, z_i, \mu, \Sigma, \tau)^{\tilde{\tau}_{i,k}}$$

The intuitive idea is that, if Z_i is sampled, $\tilde{\tau}_{i,k}$ of the samples would be the k-th component. Therefore, the corresponding density should be multiplied by itself for $\tilde{\tau}_{i,k}$ times.

Finally, the GMM training can be formulated as follows:

$$\arg\max_{\mu,\Sigma,\tau,\tilde{\tau}} \prod_{i=1}^{m} \prod_{k=1}^{n} \tilde{g}_i(x_i, z_i, \mu, \Sigma, \tau)^{\tilde{\tau}_{i,k}} \quad \text{s.t.} \quad \sum_{k=1}^{n} \tau_k = 1, \forall i = 1 \dots m : \sum_{k=1}^{n} \tilde{\tau}_{i,k} = 1$$

which can be simplified by applying a logarithm and solved using expectation-maximization.

Remark. Differently from KDE, GMM allows making various types of prediction:

- Evaluate the (log) density of a sample.
- Generate a sample.
- Estimate the probability that a sample belongs to a component.
- Assign samples to a component (i.e., clustering).

Remark. GMM can be seen as a generalization of k-means.

Remark. Differently from KDE, most of the computation is done at training time.

Number of components estimation The number of Gaussians to use in GMM can be determined through grid-search and cross-validation.

Remark. Other method such as the elbow method can also be applied. Some variants of GMM are able to infer the number of components.

Threshold optimization The threshold can be determined in the same way as in Section 2.2.2.1.

3.2.3 Autoencoder

Autoencoder Neural network trained to reconstruct its input. It is composed of two components:

Autoencoder

Encoder $e(x, \theta_e)$ that maps an input x into a latent vector z.

Decoder $d(z, \theta_d)$ that maps z into a reconstruction of x.

Training Training aims to minimize the reconstruction MSE:

$$\arg\min_{\theta_e,\theta_d} \| d\left(e(x_i, \theta_e), \theta_d\right) - x_i \|_2^2$$

To avoid trivial embeddings, the following can be done:

- Use a small-dimensional latent space.
- Use L1 regularization to prefer sparse encodings.

Autoencoder for anomaly detection By evaluating the quality of the reconstruction, an autoencoder can be used for anomaly detection:

$$\| x - d(e(x, \theta_e), \theta_d) \|_2^2 \geq \varepsilon$$

The advantages of this approach are the following:

- The size of the neural network does not scale with the training data.
- Neural networks have good performances in high-dimensional spaces.
- Inference is fast.

However, the task of reconstruction can be harder that density estimation.

Remark. It is always a good idea to normalize the input of a neural network to have a more stable gradient descent. Moreover, with normalized data, common weight initialization techniques make the output approximately normalized too.

Remark. Counterintuitively, neural networks have a higher bias compared to traditional machine learning techniques for mainly two reasons:

- A change in a single parameter affects the whole network.
- Training using SGD inherently prevents overfitting.

Theorem 3.2.1. Under the following assumptions:

- Normally distributed noise (i.e., distance between prediction and ground truth),
- Independent noise among each output component (i.e., columns),
- Same variance (i.e., homoscedasticity) in the noise of each output component,

autoencoders are trained as density estimators.

Remark. When minimizing MSE, these assumptions hold.

Proof. When training an autoencoder h using MSE on m examples with n features, the following problem is solved:

$$\begin{aligned}
\arg\min_\theta \|h(\mathbf{x},\theta)-\mathbf{x}\|_2^2 &= \arg\min_\theta \sum_{i=1}^m \sum_{j=1}^n (h_j(\mathbf{x}_i,\theta)-\mathbf{x}_{i,j})^2 \\
&= \arg\min_\theta \log\exp\left(\sum_{i=1}^m \sum_{j=1}^n (h_j(\mathbf{x}_i,\theta)-\mathbf{x}_{i,j})^2\right) \\
&= \arg\min_\theta \log\prod_{i=1}^m \exp\left(\sum_{j=1}^n (h_j(\mathbf{x}_i,\theta)-\mathbf{x}_{i,j})^2\right) \\
&= \arg\min_\theta \log\prod_{i=1}^m \exp\big((h(\mathbf{x}_i,\theta)-\mathbf{x}_i)^T \boldsymbol{I} (h(\mathbf{x}_i,\theta)-\mathbf{x}_i)\big)
\end{aligned}$$

The following adjustments can be done without altering the problem:

- Negate the argument of exp and solve a maximization problem,
- Multiply the argument of exp by $\frac{1}{2}\sigma$, for some constant σ,
- Multiply exp by $\frac{1}{\sqrt{2\pi\sigma}}$.

The problem becomes:

$$\arg\max_{\theta} \log \prod_{i=1}^{m} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{1}{2}(h(\mathbf{x}_i, \theta) - \mathbf{x}_i)^T (\sigma \boldsymbol{I})(h(\mathbf{x}_i, \theta) - \mathbf{x}_i) \right)$$

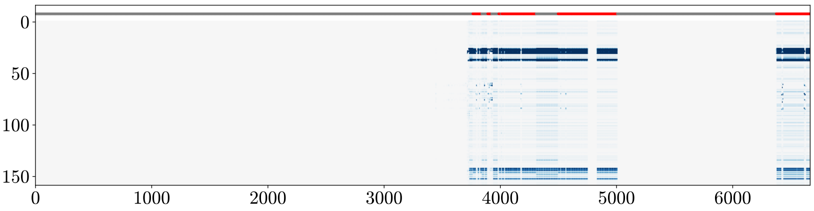
which is the PDF of a multivariate normal distribution $f(\mathbf{x}_i, h(\mathbf{x}_i), \sigma\boldsymbol{I})$ with a diagonal covariance matrix. More specifically, this is a distribution:

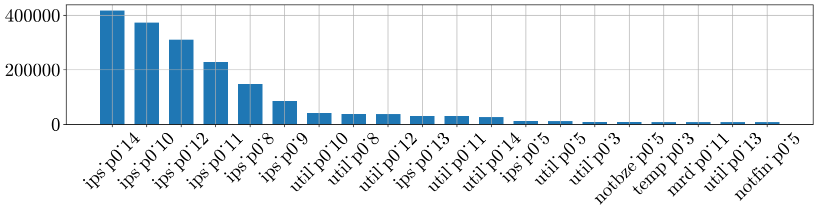
- Centered on $h(\mathbf{x}_i)$,
- With independent normal components,
- With components with uniform variance.

Therefore, when using MSE as loss, training is a likelihood maximization problem. □

Threshold optimization The threshold can be determined in the same way as in Section 2.2.2.1.

Multiple signal analysis With autoencoders, it is possible to compare the reconstruction error of single components.

> **Remark.** In most cases, reconstruction errors are often concentrated on a few features.
>
> 
>
>
> Figure 3.2: Reconstruction error of each feature

> **Remark.** It is possible to rank the feature with the highest reconstruction error to provide more insights.
>
> 
>
>
> Figure 3.3: Top-20 features with the largest error

Remark. In industrial use-cases, a usually well working approach consists of building an estimator for the sensors. More specifically, observed variables can be split into:

- Controlled variables x_c.
- Measured variables x_s.

During anomaly detection, controlled variables should not be used to determine anomalous

behaviors. In addition, there is a causal relation between the two groups as measured variables are caused by the controlled variables ($x_c \rightarrow x_s$).
A well working method is to use a regressor f as a causal model, such that:

$$x_s \approx f(x_c; \theta)$$

4 Missing data: Traffic data

4.1 Data

Time series on traffic anomalies with missing values.

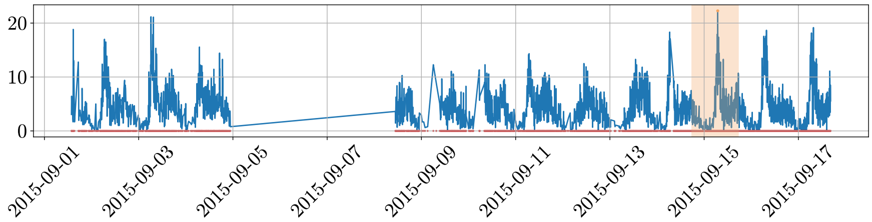


Figure 4.1: Plot of the data. Straight lines are artifacts of missing values. Red dots below represent the actual data points.

4.2 Preliminaries

The dataset has sparse indexes (i.e., indexes are non-contiguous) and missing values are represented by gaps. It is necessary to use dense indexes where missing values are explicitly marked as `NaN`.

4.2.1 Resampling / Binning

Resampling / binning Resample the indexes of the dataset so that they have a regular step (e.g., 5 minutes).

Resampling / binning

> **Remark.** Values that end up in the same bin need to be aggregated (e.g., mean).

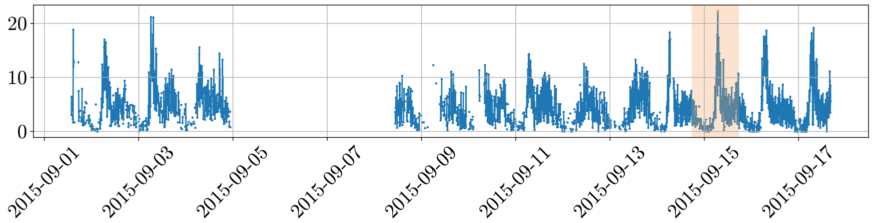


Figure 4.2: Plot of the resampled data without artifacts

4.3 Approaches

Benchmark dataset A portion of known data where some values are artificially removed can be used to evaluate a filling method. As accuracy metric, RMSE can be used.

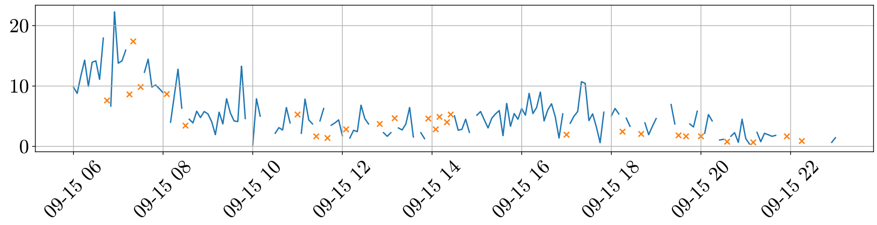


Figure 4.3: Benchmark dataset. Marked points have been artificially removed.

4.3.1 Forward/Backward filling

Forward filling Set the missing value to the last valid observation.

Backward filling Set the missing value to the next valid observation.

Remark. The idea of this approach is that time series usually have strong local correlation (i.e., some sort of inertia).

Remark. Forward/backward filling tend to work well on low variance portions of the data.

4.3.2 Geometric interpolation

Interpolate a function to determine missing points. Possible methods are:

- Linear,
- Nearest value,
- Polynomial,
- Spline.

Remark. (R)MSE assumes that the data is normally distributed, independent, and with the same variability at all points. This is not usually true with time series.

4.3.3 Density estimator

A density estimator can be used to determine a distribution from all the available data. Given an estimator $f(x, \theta)$ for $\mathcal{P}(x)$, predictions can be obtained through maximum a posteriori (MAP):

$$\arg\max_x f(x, \theta)$$

However, MAP with density estimators is computationally expensive.

Remark. Almost all inference approaches in machine learning can be reduced to a maximum a posteriori computation.

4.3.4 Regressor

A regressor can be used to fill missing values autoregressively through a rolling forecast by repeatedly making a prediction and including the new point as a training sample. However, regression only relies on the data on one side (past or future) and each autoregressive iteration accumulates compound errors.

4.3.5 Gaussian processes

> **Remark.** The ideal estimator is the one that is:
>
> - At least as powerful as interpolation (i.e., considers both past and future data).
> - Able to detect the expected variability (i.e., a measure of confidence).

Gaussian process

Gaussian process Stochastic process (i.e., collection of indexed random variables) such that:

- The index variables x are continuous and represents an input of arbitrary dimensionality.
- The variable y_x represents the output for x.

The random variables y_x respect the following assumptions:

- They follow a Gaussian distribution.
- The standard deviation depends on the distance between a point and the given observations (i.e., a confidence measure).
- y_xs are correlated. Therefore, every finite subset of y_x variables follows a multivariate normal distribution.

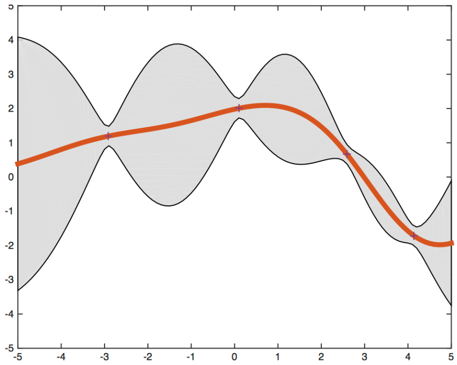


Figure 4.4: Example of Gaussian process. The red line represents the mean and the gray area the confidence interval.

> **Remark.** The PDF of a multivariate normal distribution is defined by the mean vector $\boldsymbol{\mu}$ and the covariance matrix $\boldsymbol{\Sigma}$. By recentering ($\boldsymbol{\mu} = \bar{\mathbf{0}}$), knowing $\boldsymbol{\Sigma}$ is enough to compute the joint or conditional density.

> **Remark.** To fill missing values, the conditional density $f(y_x | \bar{y}_{\bar{x}})$ is used to infer a new observation y_x given a set of known observations $\bar{y}_x$.

Naive implementation

Training Given the training observations $\bar{y}_x$, the covariance matrix can be defined as a parametrized function $\boldsymbol{\Sigma}(\theta)$ and optimized for maximum likelihood:

$$\arg\max_{\theta} f(\bar{y}_{\bar{x}}, \theta)$$

where f is the joint PDF of a multivariate normal distribution.

Inference To infer the variable y_x associated to an input x, the joint distribution $f(y_x|\bar{y}_{\bar{x}})$ has to be computed:

$$f(y_x|\bar{y}_{\bar{x}}) = \frac{f(y_x, \bar{y}_{\bar{x}})}{f(\bar{y}_{\bar{x}})}$$

- $f(\bar{y}_{\bar{x}})$ can be computed using the $\mathbf{\Sigma}$ determined during training, which is an $n \times n$ matrix assuming n training observations.
- $f(y_x, \bar{y}_{\bar{x}})$ introduces an additional variable and would require an $(n+1) \times (n+1)$ covariance matrix which cannot be determined without further assumptions.

Kernel implementation It is assumed that the covariance of two variables can be determined through parametrized kernel functions $K_\theta(x_i, x_j)$.

| **Remark.** Typically, the kernel is a distance measure.

Training Given the training observations $\bar{y}_x$ and a parametrized kernel function $K_\theta(x_i, x_j)$, training is done by optimizing the kernel for maximum likelihood (e.g., gradient descent).

The trained model is represented by both the kernel parameters θ and the training samples $\bar{y}_x$.

Inference Given a new input x, the joint distribution $f(y_x|\bar{y}_{\bar{x}})$ can be computed by obtaining $\mathbf{\Sigma}_{\bar{x}}$ and $\mathbf{\Sigma}_{x,\bar{x}}$ using the kernel.

Common kernels

Radial basis function Based on the Euclidean distance $d(x_i, x_j)$ between the input points:

$$K(x_i, x_j) = e^{-\frac{d(x_i,x_j)^2}{2l}}$$

where l is a parameter and represents the scale.

White kernel Captures the noise in the data:

$$K(x_i, x_j) = \begin{cases} \sigma^2 & \text{iff } x_i = x_j \\ 0 & \text{otherwise} \end{cases}$$

where σ is a parameter and represents the noise level.

Constant kernel Represents a learnable constant factor, useful to tune the magnitude of other kernels.

Exp-Sine-Squared Captures a period:

$$K(x_i, x_j) = e^{-2\frac{\sin^2\left(\pi\frac{d(x_i,x_j)}{p}\right)}{l^2}}$$

where l and p are parameters representing periodicity and scale, respectively.

Dot product Is more or less able to capture a trend:

$$K(x_i, x_j) = \sigma^2 + x_i x_j$$

where σ is a parameter and represents the base level of correlation.

Remark. This kernel is not translation-invariant

Remark. Bounding the domain of the parameters of a kernel can help control training.

Remark. With Gaussian processes, as we have both the prediction and the confidence interval, likelihood can be used as evaluation metric.

Remark. With Gaussian processes, by predicting points far away from the training observations (i.e., extrapolation), the mean starts to fall to 0. Only when there is a period, predictions outside the reference observations can be reasonably made.

Remark. As the reference observations and the trained kernel are detached, it is possible to change reference observations without retraining.

Inference As changing reference observations can be done without retraining the kernel, the whole series can be used when doing inference to obtain more accurate results.

To fill missing values, there are two main strategies:

Prediction Use the mean as filling value.

Sampling Use mean and variance to sample a point to fill the missing value. Clipping might be needed to make the sampled point valid (e.g., prevent negative values for traffic).

Remark. Using the mean results in a smoother filling, while sampling produces more realistic data.

4.3.6 Multiplicative ensemble

Remark. Gaussian process alone only accounts for covariance and do not consider input-dependent variance (i.e., variance of the traffic at the same week day and time on different days).

Remark. Variance scales via multiplication but not summation:

$$Var(x + \alpha) = Var(x) \qquad Var(\alpha x) = \alpha^2 Var(x)$$

for a constant α.

Multiplicative ensemble Product of the outputs of two models f and g:

$$g(x, \lambda) f(x, \theta)$$

More specifically, the training process aims to obtain:

$$g(x_i, \lambda) f(x_i, \theta) \approx y_i \Rightarrow f(x_i, \theta) \approx \frac{y_i}{g(x_i, \lambda)}$$

In other words, f is trained on a series with variance altered by g.

For this specific problem, f is a Gaussian process and g a standard deviation model.

Standard deviation model A simple standard deviation model consists of mapping time intervals to their standard deviations. This approach is sensitive to the choice of the granularity of the interval (time unit in this problem):

- If it has too many missing values or too little samples, it is not enough to compute a reliable standard deviation.
- If it is too coarse, the computed standard deviation is not useful.

> **Remark.** From empirical considerations, the central limit theorem is observable starting from 30 samples. Therefore, 30 data points are enough to make a reasonably stable prediction of the standard deviation.

As the final model might be too coarse, the following can be done:

Upsampling Use a finer grain unit (i.e., x-axis, time unit in this problem) and fill missing values through linear interpolation.

Smoothing Smooth the upsampled data through a low-pass filter.

For this problem, an exponentially weighted moving average works best as recent data are more relevant.

5 Remaining useful life: Turbofan engines

Maintenance can be of three types:

Reactive maintenance Repair when something is broken.

Preventive maintenance Periodically change something, in a conservative way, before it breaks.

Predictive maintenance Change when something is close to break.

Remaining useful life (RUL) is a metric useful for predictive maintenance.

5.1 Data

The dataset contains run-to-failure experiments on NASA turbofan engines. Excluding domain specific features, the main columns are:

`machine` Index of the experiment.

`cycle` Time step of the experiment.

`rul` Remaining useful life.

From the dataset heatmap, the following can be observed:

- Rows with a uniform blue or red color represent features that contain frequent short-lived variations (i.e., peeks) that skew the standard deviation.
- Some features show a trend synced with the experiments (see rows around 15 at the y-axis with blue peeks at the end of each experiment).

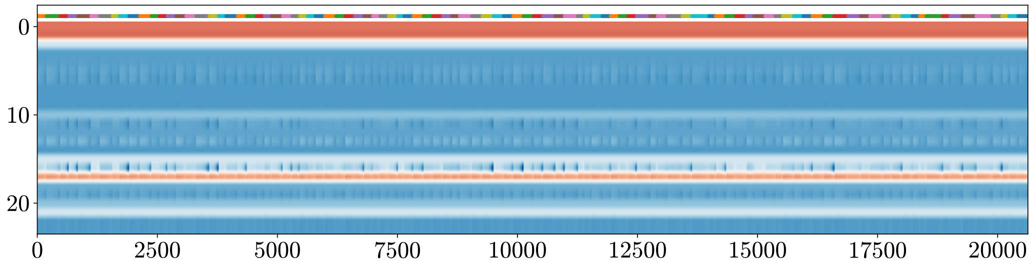


Figure 5.1: Heatmap of the dataset. On the top line, each section represents an experiment.

5.1.1 Data splitting

As the dataset is composed of experiments, standard random sampling will mix experiments and leak information. Therefore, sampling is done on the experiments in chronological order (train first).

Remark. When splitting, the train data should be representative of the test data. Moreover, the test set should be representative of the real world.

5.2 Approaches

5.2.1 Regressor

Predict RUL with a regressor f and set a threshold to trigger maintenance:

$$f(x, \theta) \leq \varepsilon$$

Linear regression A linear regressor can be used as a simple baseline for further experiments.

> **Remark.** For convenience, a neural network without hidden layers can be used as the regressor.

> **Remark.** Training linear models with gradient descent tend to be slower to converge.

Multi-layer perceptron Use a neural network to solve the regression problem.

> **Remark.** A more complex model causes more training instability as it is more prone to overfitting (i.e., more variance).

> **Remark.** Being more expressive, deeper models tend to converge faster.

> **Remark** (Self-similarity curves)**.** Exponential curves (e.g., the loss decrease plot) are self-similar. By considering a portion of the full plot, the subplot will look like the whole original plot.

> **Remark** (Common batch size reason)**.** A batch size of 32 follows the empirical statistical rule of 30 samples. This allows to obtain more stable gradients as irrelevant noise is reduced.

Convolutional neural network Instead of feeding the network a single state, a sequence can be packed as the input using a sliding window as in Section 2.2.2.2. 1D convolutions can then be used to process the input so that time proximity can be exploited.

> **Remark.** Using sequence inputs might expose the model to more noise.

> **Remark.** Even if the dataset is a time series, it is not always the case (as in this problem) that a sequence input is useful.

Remark. The accuracy of the trained regressors are particularly poor at the beginning of the experiments due to the fact that faulty effects are noticeable only after a while. However, we are interested in predicting when a component is reaching its end-of-life. Therefore, wrong predictions when the RUL is high are acceptable.

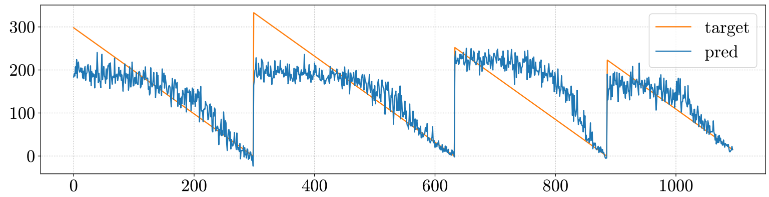


Figure 5.2: Predictions on a portion of the test set

Cost model Intuitively, a cost model can be defined as follows:

- The steps of the experiments are considered as the atomic value units.
- The cost for a failure (i.e., untriggered maintenance) is high.
- Triggering a maintenance too early also has a cost.

Threshold estimation To determine the threshold ε that determines when to trigger a maintenance, a grid-search to minimize the cost model can be performed.

Formally, given a set of training experiments K, the overall problem to solve is the following:

$$\begin{aligned} &\arg\min_{\varepsilon} \sum_{k=1}^{|K|} \texttt{cost}(f(\mathbf{x}_k, \boldsymbol{\theta}^*), \varepsilon) \\ &\text{subject to } \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(f(\mathbf{x}_k, \boldsymbol{\theta}), y_k) \end{aligned}$$

Note that ε do not appear in the regression problem. Therefore, this problem can be solved as two sequential subproblems (i.e., regression and then threshold optimization).

5.2.2 Classifier

Predict RUL with a classifier f_ε (for a chosen ε) that determines whether a failure will occur in ε steps:

$$f_\varepsilon(x, \theta) = \begin{cases} 1 & \text{failure in } \varepsilon \text{ steps} \\ 0 & \text{otherwise} \end{cases}$$

Remark. Training a classifier might be easier than a regressor.

Logistic regression A logistic regressor can be uses as baseline.

Remark. For convenience, a neural network without hidden layers can be used as the classifier.

Multi-layer perceptron Use a neural network to solve the classification problem.

Remark. As classifiers output a probability, they can be interpreted as the probability of not failing.

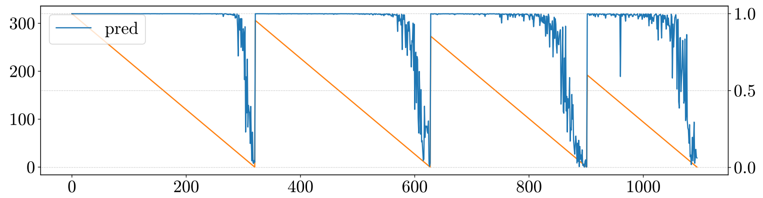
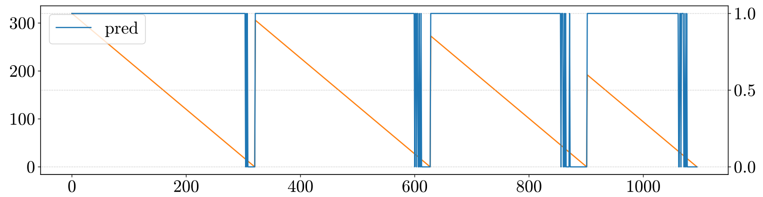


Figure 5.3: Output probabilities of a classifier

By rounding, it is easier to visualize a threshold:



Cost model As defined in Section 5.2.1.

Threshold estimation

> **Remark.** The possibility of using a user-defined threshold ε allows choosing how close to a failure maintenance has to be done. However, early signs of a failure might not be evident at some thresholds so automatically optimizing it might be better.

Formally, given a set of training experiments K, the overall problem to solve is the following:

$$\arg\min_{\varepsilon} \sum_{k=1}^{|K|} \texttt{cost}\left(f(\mathbf{x}_k, \boldsymbol{\theta}^*), \frac{1}{2} \right)$$
$$\text{subject to } \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(f(\mathbf{x}_k, \boldsymbol{\theta}), \mathbb{1}_{y_k \geq \varepsilon})$$

Differently from regression, the threshold ε appears in both the classification and threshold optimization problems, so they cannot be decomposed.

Black-box optimization Optimization approach based on brute-force.

Black-box optimization

For this problem, black-box optimization can be done as follows:

1. Over the possible values of ε:
   a) Determine the ground truth based on ε.
   b) Train the classifier.
   c) Compute the cost.

> **Remark.** Black-box optimization with grid-search is very costly.

Bayesian surrogate-based optimization Class of approaches to optimize a black-box function f under trivial constraints. It is assumed that f is expensive to evaluate and a surrogate model (i.e., a proxy function) is instead used to optimize it.

Bayesian surrogate-based optimization

Formally, Gaussian surrogate-based optimization solves problems in the form:

$$\min_{x \in B} f(x)$$

where B is a box (i.e., hypercube). f is optimized through a surrogate model $\hat{f}$ that is trained on some observations from f (prior). At each iteration, an acquisition function that uses $\hat{f}$ is used to determine a new point to explore and evaluate with f. The newly discovered sample (posterior) is used to improve $\hat{f}$ and the process is repeated.

> **Remark.** Under the correct assumptions, the result is optimal.

Gaussian process surrogate A good surrogate model should be able to accurately approximate all available training samples and provide a prediction confidence.

A Gaussian process has these properties and can be used as surrogate model.

Acquisition function Function to determine which point to explore next. It should account for both predictions and confidence.

> **Remark.** Acquisition functions should balance exploration (i.e., area with low predictions) and exploitation (i.e., area with high confidence).

Lower confidence bound Acquisition function defined as:

$$\texttt{LCB}(x) = \mu(x) - Z_\alpha \sigma(x)$$

where $\mu(x)$ and $\sigma(x)$ are the predicted mean and standard deviation, respectively. Z_α is a multiplier for the confidence interval.

Given a set of training samples $\{x_i, y_i\}$ and a black-box function f, surrogate-based optimization does the following:

1. Until a termination condition is not met:
    a) Train a surrogate model $\hat{f}$ on $\{x_i, y_i\}$ to approximate f.
    b) Optimize an acquisition function $a_{\hat{f}}(x)$ to find the next data point x' to explore.
    c) Evaluate x' with the black-box function $y' = f(x')$.
    d) Store y' if it is the current best optimum for f.
    e) Add (x', y') to the training set.