Artificial Intelligence in Industry

Last update: 27 September 2024

Academic Year 2024 – 2025
Alma Mater Studiorum · University of Bologna

Contents

1 Preliminaries 1

2 Low dimensional anomaly detection: Taxi calls 2

2.1 Data . . . 2

2.2 Approaches . . . 2

2.2.1 Gaussian assumption . . . 2

2.2.2 Characterize data distribution . . . 2

2.2.2.1 Univariate kernel density estimation . . . 3

2.2.2.2 Multivariate kernel density estimation . . . 4

2.2.2.3 Time-dependent estimator . . . 6

2.2.2.4 Time-indexed model . . . 7

3 High dimensional anomaly detection: HPC centers 8

3.1 Data . . . 8

3.1.1 High-dimensional data visualization . . . 8

3.2 Approaches . . . 9

3.2.1 Multivariate KDE . . . 9

3.2.2 Gaussian mixture model . . . 9

3.2.3 Autoencoder . . . 11

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.

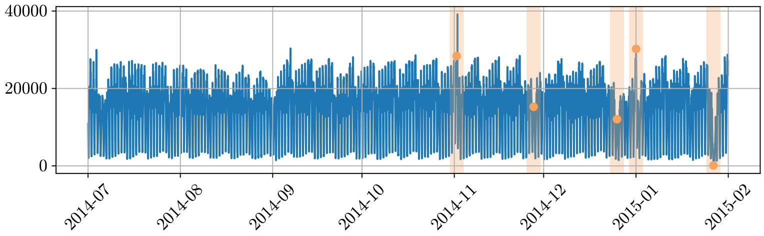


Figure 2.1: Plot of the time series, anomalies, and windows

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.

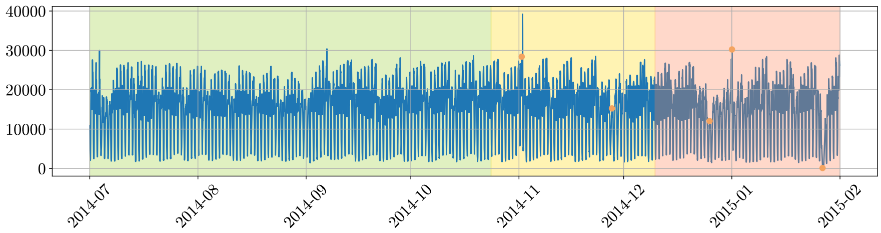


Figure 2.2: Train, validation, and test splits

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.

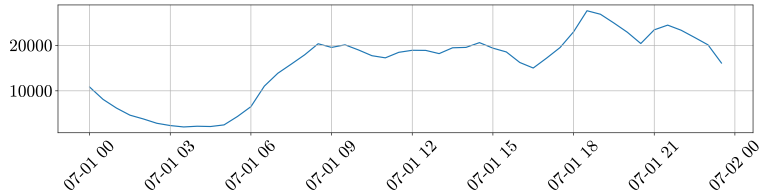


Figure 2.3: Subset of the dataset

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.

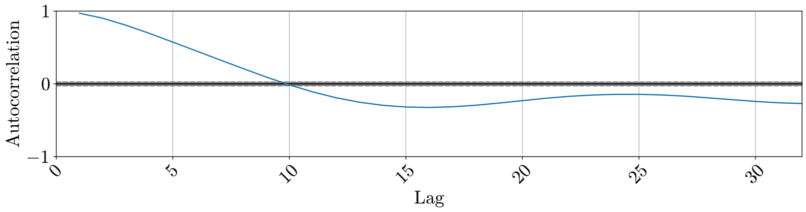


Figure 2.4: Autocorrelation plot of the subset of the dataset. There is strong correlation up to 4-5 lags.

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.

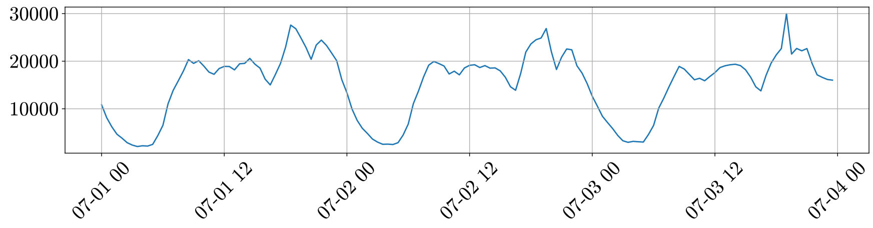
In practice, the expected value is sampled from multiple validation and training sets through cross-validation and grid-search.

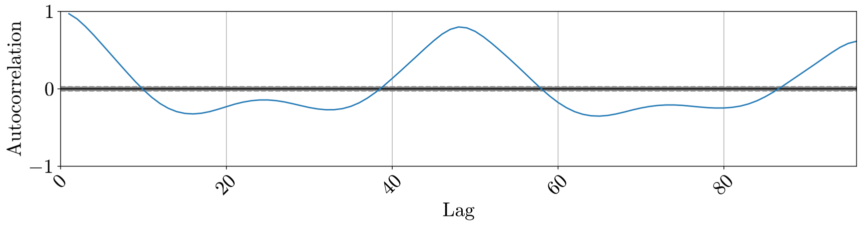
> **Remark.** As different folds for cross-validation must be tried, grid-search is an expensive operation.

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:

$$f(t,x)$$

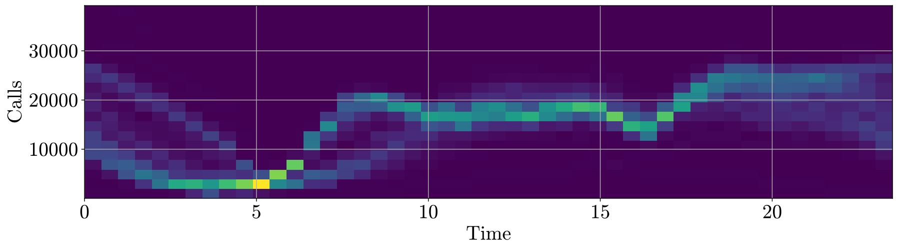


Figure 2.5: 2D histogram of the distribution. Lighter colors indicate a higher frequency of occurrence.

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.

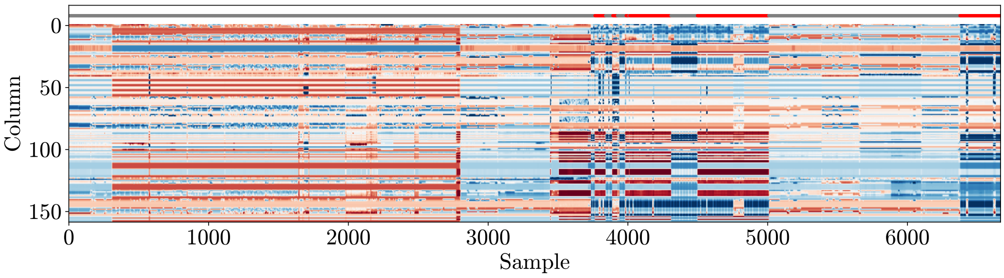


Figure 3.1: Heatmap of the dataset. On the top line, gray points represent normal behavior while red points are anomalies. In the heatmap, white tiles represent the mean, red tiles represent values below average, and blue tiles represent values above average.

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)

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:

$$X_Z$$

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.