Artificial Intelligence in Industry

Matteo Donati

October 17, 2022

Contents

1 Anomaly Detection via Simple Methods

1.1 Problem and Data

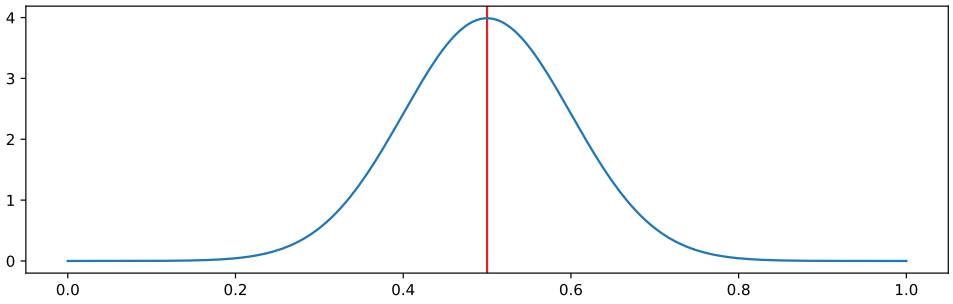
The goal of anomaly detection is to detect, analyze and anticipate abnormal situations (i.e. **anomalies**). Usually, anomaly detection is based on time-series analysis, where a time-series is a sequence whose index represents time. Time-series have one difference with respect to classical table datasets: their row index is meaningful, since it represents the position of the example in the sequence. The labels associated with such a dataset usually indicate an instant of time at which an anomaly occurs. Considering a specific anomaly, one could also compute the window of time inside which the specific anomaly occurs.

1.2 Anomaly Detection and Kernel Density Estimation

A possible approach to detect anomalies is based on the fact that anomalies are often unlikely. If one can estimate the probability of every occurring observation x, then one can spot anomalies based on their low probability. Formally, a detection condition can be states as $f(x) \leq \theta$, where $f(x)$ is a **probability density function** (**PDF**), and θ is a scalar threshold. Given some training data $\hat{\boldsymbol{x}}$, the true density function $f^*(x) : \mathbb{R}^n \to \mathbb{R}^+$, and a second function $f(x, \omega)$, a supervised learning approach to estimate the probability densities considers a suitable loss function, $L(y, y^*)$, that has to be optimized so to find the best set of parameters ω that minimizes the considered loss:

$$\text{argmin}_\omega \, L(f(\hat{\boldsymbol{x}}, \omega), f^*(\hat{\boldsymbol{x}})) \tag{1}$$

However, this approach cannot work, because usually one does not have access to the true density f^*. Thus, density estimation is an unsupervised learning problem. Such problem can be solves via a number of techniques (e.g. via Kernel Density Estimation). In **Kernel Density Estimation** (**KDE**) the main idea is that wherever, in the input space, there is a sample, then it is likely that there are more samples, so one can assume that each training sample is the center for a density kernel. Formally, the kernel $K(x, h)$ is just a valid PDF, where x is the input variable (scalar or vector), and h is a parameter (scalar or matrix respectively) called *bandwidth*. For example, given a single sample $x = 0.5$, then a Gaussian estimator with $h = 0.1$ will produce the following:



Indeed, in `sklearn`, a Gaussian kernel is given by:

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

which is similar to the PDF of the Normal distribution, where the mean can be interpreted as zero, and h controls the standard deviation of the distribution. However, since the mean is zero, the kernel will be centered on zero. To solve this, one can use an affine transformation, $K(x - \mu, h)$, which gives the value of a kernel computed for the value x and centered on μ. Moreover, the estimated density of any point is obtained as a kernel average:

$$f(x, \hat{\boldsymbol{x}}, h) = \frac{1}{m} \sum_{i=0}^{m} K(x - \hat{x}_i, h) \tag{3}$$

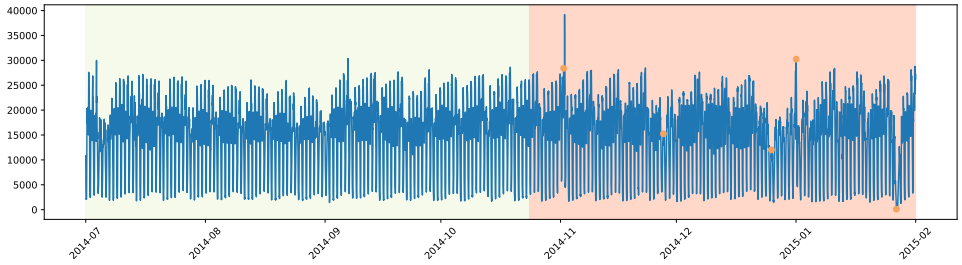
where x is the input for which to compute the estimate, $\hat{\boldsymbol{x}}$ is the matrix containing the training samples, $x - \hat{x}_i$ is the difference between x and the i-th training sample. Thus, KDE models are not trained in the usual sense: the training set is part of the model parameters. The only thing that one needs to train is h. For the univariate case, one can apply the following rule of thumb:

$$h = 0.9 \min\left(\hat{\sigma}, \frac{IQR}{1.34}\right) m^{-\frac{1}{5}} \tag{4}$$

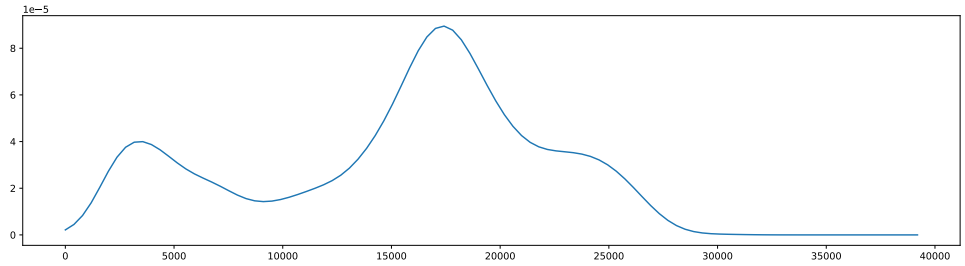
where $\hat{\sigma}$ is the standard deviation computed using the training data, and IQR is the inter-quartile range. Lastly, to avoid taking products, one can work with negated log probabilities, so that the anomaly detection condition becomes $-\log f(x, \omega) \geq \theta$.

1.3 KDE for Anomaly Detection

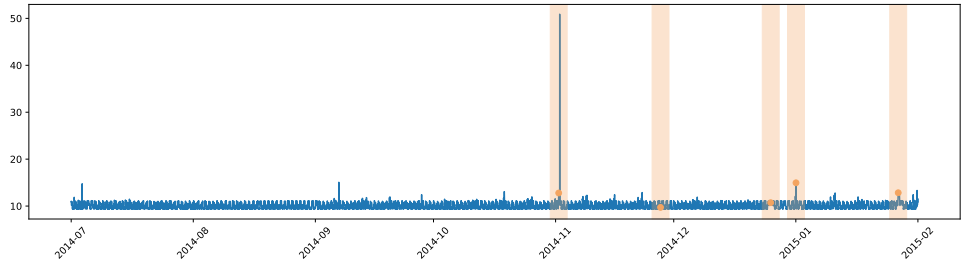
Considering a time-series, one can split it into training set and test set, where the training set only includes data about normal behavior and it will be used to fit a KDE model, while the test is used to asses how well the approach can generalize. If the training set contains some anomalies, then it is fine, as long as these are very infrequent.



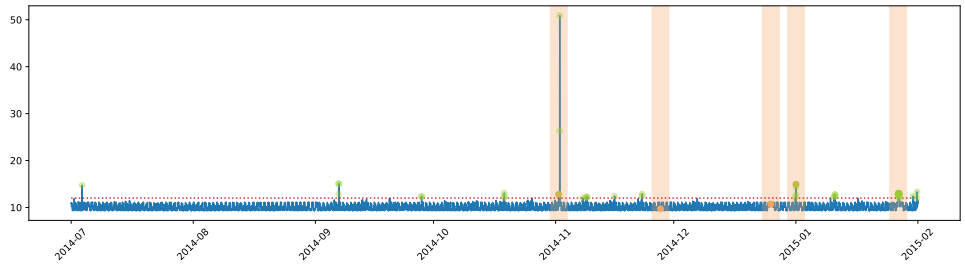
At this point, a univariate KDE is fit to the training data, obtaining the following estimated distribution:



The alarm signal can be then computed from such estimated distribution:



One could also pick a threshold and try to detect some anomalies:



However, the result contains many false positives, which are usually common in anomaly detection.

1.4 Metrics for Anomaly Detection

In order to evaluate costs and benefits of one's predictions, one can rely on a cost model. A simple cost model is based on the concepts of true positives (i.e. windows for which one detects at least one anomaly), false positives (i.e. detected anomalies that do not fall in any window), false negatives (i.e. anomalies that go undetected), and advance (i.e. time between an anomaly and when first it was detected). Then, one can introduce: a cost c_{alarm} for loosing time in analyzing false positives, a cost c_{missed} for missing anomalies, and a cost c_{late} for a late detection. This simple cost model can be used for choosing the threshold to detect anomalies. Indeed, the best threshold is the one that minimizes

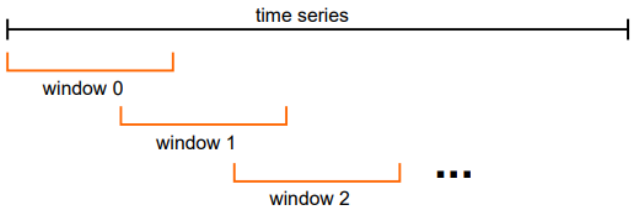
$c_{alarm} \times |FP| + c_{missed} \times |FN| + c_{late} \times |\text{late detections}|$. To do so, one can define a validation set, and apply a linear search, to tune the θ threshold.

1.5 Sliding Windows

Given a time-series, nearby points tend to have similar values, meaning that they are correlated. A useful tool to study such correlation are the autocorrelation plots:

1. Consider a range of possible **lags**.
2. For each lag value l:
   (a) Make a copy of the series and shift it by l time-steps.
   (b) Compute the Pearson correlation coefficient (i.e. linear correlation coefficients) with the original series.
3. Plot the correlation coefficients over the lag values.

Where the curve is far from zero, there is a significant correlation, and where it gets close to zero, no significant correlation exists. These correlations are a source of information and can be exploited to improve the estimated probabilities. Indeed, to take advantage of such information, one could feed one's model with sequences of observations, instead of individual observations. A common approach consist in using a **sliding window**:



In general, let m be the number of examples and w be the window length, the result of this approach is the table

	$\boldsymbol{s_0}$	$\boldsymbol{s_1}$	$\cdots$	$\boldsymbol{s_{w-1}}$
$\boldsymbol{t_{w-1}}$	x_0	x_1	$\cdots$	x_{w-1}
$\boldsymbol{t_w}$	x_1	x_2	$\cdots$	x_w
$\boldsymbol{t_{w+1}}$	x_2	x_3	$\cdots$	x_{w+1}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\boldsymbol{t_{m-1}}$	x_{m-w}	x_{m-w+1}	$\cdots$	x_{m-1}

where $\boldsymbol{t_i}$ is the time window index, and $\boldsymbol{s_j}$ is the position of an observation within a window. In general, one can pick the window length to be equal to the number of lags for which the aforementioned correlation is still high.

1.6 Sequence Input in KDE

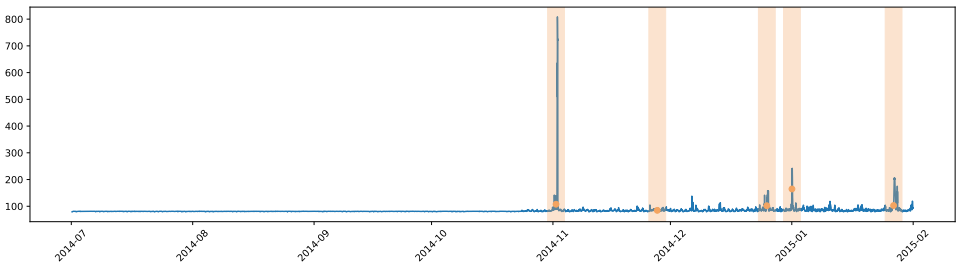
When dealing with KDE, there exists a straightforward approach to take sequences input into account. In particular, this can be done by using multivariate KDE: each sequence is treated as a vector variable, and individual sequences are treated as independent (Markov property). Then, a multivariate KDE estimator is learned. First of all, a suitable bandwidth has to be chosen: i. pick a validation set, ii. tune th bandwidth for maximum likelihood. Formally, let $\tilde{\boldsymbol{x}}$ be a validation set of m samples. Assuming independent observations, its likelihood is:

$$L(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{x}}, h) = \prod_{i=1}^{m} f(\tilde{x}_i, \hat{\boldsymbol{x}}, h) \tag{5}$$

Then, h is chosen to maximize such likelihood:

$$\text{argmax}_h \, \mathbb{E}_{\tilde{\boldsymbol{x}} \sim \mathcal{D}}[L(\tilde{\boldsymbol{x}}, \hat{\boldsymbol{x}}, h)] \tag{6}$$

where $\mathcal{D}$ is the true distribution of samples. However, as many training problems, this cannot be solves in an exact fashion. Instead, one can approximate $\mathbb{E}$ by sampling multiple $\tilde{\boldsymbol{x}}$ and picking the bandwidth h^* leading to the maximum average likelihood. This approximation can be implemented by applying a grid search and searching for h^*. For example, the alarm signal produced when considering sequences input instead of single observations is the following:



This signal shows much less noise with respect to the one computed when considering single observations.