Chapter 9

Time Series and Multidimensional Streaming Outlier Detection

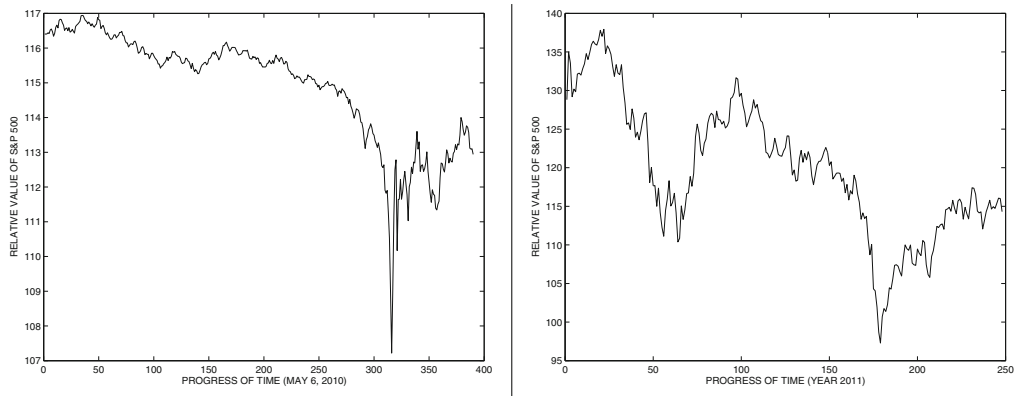
"To improve is to change; to be perfect is to change often."– Winston Churchill

9.1 Introduction

The temporal and streaming outlier-detection scenarios arise in the context of many applications such as sensor data, mechanical systems diagnosis, medical data, network intrusion data, newswire text posts, or financial posts. In such problem settings, the assumption of *temporal continuity* plays a critical role in identifying outliers. Temporal continuity refers to the fact that the patterns in the data are not expected to change abruptly, unless there are abnormal processes at work. It is worth noting that outlier analysis has diverse formulations in the context of temporal data, in some of which temporal continuity is more important than others. In *time-series data*, temporal continuity is immediate, and expected to be very strong. In multidimensional data with a temporal component (e.g., text streams), temporal continuity is much weaker, and is present only from the perspective of *aggregate trends*. Therefore, two different scenarios arise:

- **Abrupt change detection in time series:** These correspond to sudden changes in the trends in the underlying data stream. In such cases, the issues of temporal continuity are critical, and the outlier is defined as unusual because it exhibits a *lack of continuity* with its immediate [579] or long-term history. For example, sudden changes in time-series values (with respect to immediate history), or distinctive shapes of subsequences of the time series (with respect to long-term history) are identified as outliers. In cases where the entire time series is available offline, the advantage of hindsight may be leveraged to identify abnormal time-series values or shapes. In time-series analysis, *vertical* analysis is more important where each individual series (or dimension) is treated as a unit, and the analysis is primarily performed on this unit.

C.C. Aggarwal, *Outlier Analysis*, DOI 10.1007/978-3-319-47578-3_9



(a) Flash-crash (May 6, 2010) (b) Year 2001

Figure 9.1: Behavior of the S&P 500 on (a) the day of the flash crash (May 6, 2010), and (b) year 2001

In the event that multiple series are available, cross-correlations may be leveraged, although they typically play a secondary role to the analysis of each individual series. This is because time-series data is *contextual*, which imposes strong temporal locality on series values.

- **Novelty and change detection in multidimensional data:** In this case, the data contains individual multidimensional points that are independent of one another, and the issue of temporal continuity is much weaker as compared to time-series data. For example, in a time series derived from sensor data, two successive data values are often almost identical. On the other hand, an individual text document (multidimensional data point) in a stream of newswire articles may normally be quite different from its immediately preceding article. The data point needs to be compared to a much larger history of the documents in order to construct a robust outlier model. The anomalies in multidimensional streaming data could correspond to either *time-instants* at which *aggregate* trends have changed or novelties in *individual* data points. The latter requires the identification of incoming data points in the stream that are very different from previously seen points. Such cases are almost identical to offline outlier analysis. The temporal aspects of the stream are important only to the extent that the novelties are defined with respect to the *past* history rather than the entire data set. In such cases, all the dimensions of a record are treated as a unit, and the anomalies are identified by aggregating the temporal trends in these units. This type of analysis is *horizontal*.

  For example, a first story on a particular topic in a text stream is considered a novelty outlier, while a change in the aggregate trend of the topics in the text stream is a change point outlier. It is worth noting that novelties are often *trend-setters*. Therefore, at a later stage, similar data points may no longer be considered novelties, but may become a normal part of the data. Thus, while the temporal aspects of the stream are used, they are slightly less important from an *immediate* continuity perspective. However, significant changes from aggregate trends continue to be important.

Both the aforementioned definitions are consistent with Hawkins's notion of outliers, which was introduced at the very beginning of the book.

Outliers in a time series are either *contextual* or *collective* anomalies. Outliers are contextual when the values at specific time stamps suddenly change with respect to their temporally adjacent values, whereas outlier are collective when entire time series or large subsequences within a time series have unusual shapes. For example, consider the two cases illustrated in Figures 9.1(a) and (b). The first time series illustrates the relative behavior[1] of the S&P 500, on May 16, 2010 which was the date of the stock market flash crash. This is a very unusual event *both* from the perspective of the deviation during the stock value drop, *and* from the perspective of the time-series shape. On the other hand, Figure 9.1(b) illustrates the behavior of the $S\&P$ 500 during the year 2001. There are two significant drops over the course of the year both because of stock-market weakness and also because of the 9/11 terrorist attacks. Although the *specific time-stamps of drop* may be considered somewhat abnormal, the *shape* of this time series is not unusual because it is frequently encountered during stock-market drops. The detection of unusual points in time or unusual shapes arises in different types of applications.

It is evident from the aforementioned example that temporal data allows multiple ways of defining anomalies. This is consistent with the observation in the first chapter: "*The more complex the data is, the more the analyst has to make prior inferences of what is considered normal for modeling purposes.*" The appropriate way of defining anomalies is highly application-dependent, and therefore it is important for the analyst to have a good understanding of the application domain at hand. For example, in the case of sensor data, it may sometimes be helpful to use deviation detection methods for noise outlier filtering, whereas in other cases, unusual shapes in a medical stream can diagnose heart conditions such as arrythmia.

There are also some subtle differences between the *offline* and *online* (streaming) settings. In the former case, the entire history of the stream may be available for analysis, whereas only the stream up to the current time is available in the latter case. In the offline setting, the advantage of hindsight allows the discovery of better outliers with more sophisticated models.

Labels may be available to supervise the anomaly detection process in both the time-series or multidimensional outlier detection settings. In the time-series setting, the labels may be associated with time-instants, with time intervals, or they may be associated with the entire series. In the multidimensional setting, labels are associated with the individual data points. In such cases, the problem reduces to a special case of rare-class detection (cf. Chapter 7). In general, supervised methods almost always perform better than unsupervised methods because of their ability to discover *application-specific* abnormalities. The general recommendation is to always use supervision when it is available.

Even though temporal data may comprise either continuous data or discrete sequences, this chapter focuses on continuous data in order to preserve homogeneity in presentation. This is because the concept of temporal continuity is defined differently in discrete data than in continuous data. In the case of discrete data, a lack of ordering in the data values significantly influences the nature of the methods used for outlier analysis. It should be noted that a continuous series can always be discretized into symbolic data, albeit at the loss of some information. Thus, some of the methods for outlier detection are common to both types of data. The case of discrete data will be discussed separately in the next chapter. The material in the two chapters on temporal and sequence-based outlier detection is carefully organized, so as to point out the relationships among these scenarios.

[1]The tracking ETF SPY was used.

This chapter is organized as follows. In the next section, algorithms for detection of outlier *instants* in streaming time series will be presented. Typically, these methods are based on deviation-detection from predicted values at time instants. The detected anomalies are contextual outliers. In section 9.3, methods for detecting unusual shapes in time-series data will be presented. These are collective outliers. Methods for multidimensional streaming outlier detection are discussed in section 9.4. Section 9.5 presents the conclusions and summary.

9.2 Prediction-Based Outlier Detection in Streaming Time Series

The most common application of temporal outlier detection is that of detecting *deviation-based* outliers of specific time-instants with the use of regression-based forecasting models. These anomalies are contextual anomalies, because they define abnormalities at specific instants of the data, on the basis of relationships between data values at adjacent time instants. Such an approach can either be used to detect sudden changes in the underlying process, or to filter noise from the underlying streams. Deviation-based outliers in time series are very closely related to the problem of time-series forecasting, since outliers are declared on the basis of deviations from expected (or forecasted) values. This point of view is closely related to the discussion in section 7.7 of Chapter 7, which establishes a relationship between prediction and outlier detection.

In these methods, *temporal continuity* plays a significant role, since it is assumed that time-series data values are highly correlated over successive instants, and the temporal trends do not change abruptly. Deviation-based outliers use the predicted value at the next time-stamp through a variety of regression models. The correlations in a single time series, or across multiple series, may be used in order to perform the prediction. Thus, two types of correlations are used:

- **Correlations across time:** This is the same principle as temporal continuity, which is typically implemented using autoregressive modeling and forecasting. *Significant* deviations from the *expected* (i.e., forecasted) predictions are defined as outliers. Such *significant* deviations are therefore defined by violations of temporal continuity.
- **Correlations across series:** Many sensor applications result in time series that are often closely correlated with one another. For example, a bird call at one sensor will typically also be recorded by a nearby sensor. In such cases, one series can frequently be used in order to predict another. Deviations from such expected predictions can be reported as outliers.

Regression modeling techniques for *non-temporal data* are discussed in Chapter 3. This chapter will study their application in the context of temporal data, which has a number of unique characteristics in terms of data specification and modeling. While the core theory behind both domains is essentially the same, the way in which it is applied is different. The subsequent discussion assumes familiarity with the regression modeling techniques presented in section 3.2.1 of Chapter 3.

9.2.1 Autoregressive Models

Autoregressive models (AR) are particularly useful in the context of univariate time series. Let $X_1 \ldots X_t \ldots$ be the values in the univariate time series. In the autoregressive model,

the value of X_t is defined in terms of the values in the immediately preceding window of length p.

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + c + \epsilon_t \tag{9.1}$$

A model that uses the preceding window of length p is referred to as an $AR(p)$ model. The values of the regression coefficients $a_1 \ldots a_p, c$ need to be learned from the training data, which is the previous history of the time series. Here, the values of ϵ_t are assumed to be error terms, which are uncorrelated with one another. Since these error terms represent unexpected behavior, they are natural candidates for being considered outlier scores.

A set of linear equations between the coefficients can be created by using Equation 9.1 on each time stamp in the training data. Therefore, for a time series of length n, one can extract $(n-p)$ such linear equations. When this number is much larger than the number of variables $(p+1)$, this is a severely over-determined system of equations with no exact solution. The coefficients $a_1, \ldots a_p, c$ can be approximated with *least-squares regression* in which the squared-error of the over-determined system is minimized. The details of the solution are provided in section 3.2.1 of Chapter 3. From the point of view of the framework in section 3.2.1 of Chapter 3, one can generate an $(n-p) \times (p+1)$ independent-variable matrix D with $(p+1)$ dimensions (including the constant term) and $(n-p)$ points for regression modeling. The rows of matrix D contain the behavioral attribute values in windows of size p along with a value of 1 appended at the end in order to handle the constant term:

$$D = \begin{bmatrix} X_1 & X_2 & \ldots & X_p & 1 \\ X_2 & X_3 & \ldots & X_{p+1} & 1 \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ X_{n-p} & X_{n-p+1} & \ldots & X_{n-1} & 1 \end{bmatrix} \tag{9.2}$$

The $(n-p)$-dimensional column vector $\overline{y}$ contains the dependent variables, which are the behavioral attribute values at each of the $(n-p)$ modeled time-stamps at ticks $p+1 \ldots n$. In other words, the column-vector $\overline{y}$ is defined as follows:

$$\overline{y} = \begin{bmatrix} X_{p+1} \\ X_{p+2} \\ \ldots \\ X_n \end{bmatrix} \tag{9.3}$$

We would like to learn the values of $a_1 \ldots a_p, c$ that minimize the least-squares error of the following system of equations, which is a matrix representation of Equation 9.1 over the $(n-p)$ modeled time-stamps:

$$\overline{y} \approx D \begin{bmatrix} a_p \\ a_{p-1} \\ \ldots \\ a_1 \\ c \end{bmatrix} \tag{9.4}$$

Note the use of approximate equality "$\approx$" because we have omitted the errors ϵ_t^j in Equation 9.1. These errors need to be minimized by linear-regression modeling. As discussed in section 3.2.1 of Chapter 3, a matrix $D^T D$ of size $(p+1) \times (p+1)$ needs to be inverted in order to determine[2] the coefficients $a_1 \ldots a_p, c$ so that the least-squares error of approximation is

[2]Many alternatives and derivatives, such as the Yule-Walker equations [309], can also be used. Most off-the-shelf forecasting packages contain highly refined versions of these algorithms.

minimized. The corresponding solution is as follows:

$$[a_p, a_{p-1} \ldots a_1, c]^T = (D^T D)^{-1} D^T \overline{y} \tag{9.5}$$

Regularization is helpful in cases where the number of time stamps is not significantly larger than p. In such cases, the matrix $D^T D$ might not be invertible. If regularization is used, then the matrix $(D^T D + \alpha I)$ is inverted instead of $D^T D$ for some small value of the regularization parameter $\alpha > 0$. It is noteworthy that the matrices $D^T D$ and $D^T \overline{y}$ can be maintained incrementally as new rows are added to D and $\overline{y}$ with the arrival of new time-stamps. However, one must still perform the inversion of the matrix $D^T D$ in each iteration. This can be performed more efficiently in an incremental manner with the use of the *matrix-inversion lemma* [592].

In Equation 9.1, the value of ϵ_t represents the *deviation* from the expected value, which corresponds to the outlier score of each time stamp. One can use the learned coefficients to estimate these deviations, and large absolute values correspond to anomalous time stamps. For simplified analysis, these values are assumed to be independent and identically distributed (i.i.d.) random variables from a normal distribution. This assumption can be used for hypothesis testing in converting outlier scores to outlier labels on time-stamps.

The autoregressive model can be made more robust by combining it with a moving-average model (MA Model). This model predicts subsequent values in the time series as a function of the past history of deviations. The moving-average model is defined as follows:

$$X_t = \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + \mu + \epsilon_t \tag{9.6}$$

In other words, a value is predicted using the past history of *shocks* in the time series. The aforementioned model is also referred to as $MA(q)$. The value of μ is the mean of the time series. The values of $b_1 \ldots b_t$ are the coefficients, which need to be learned from the data. The moving-average model is quite different from the autoregressive model, in that it relates the current value to the mean of the series and the previous history of deviations rather than the previous history of values. Note that the term $\sum_{i=1}^{q} b_i \cdot \epsilon_{t-i}$ represents a linear combination of historical *shocks* or outlier scores. In this sense, the moving average model is very interesting because it expresses the current value of the series as a function of the level of unexpected behavior in the past. It is also noteworthy that the values of ϵ_{t-i} are not a part of the observed data, but they represent the deviations from the expected values (which are themselves computed from historical deviations). Therefore, this particular system of equations is inherently nonlinear. In general, closed-form solutions cannot be found for such systems, and iterative non-linear fitting procedures are used [467].

In practice, both the previous history of deviations and values may be important for calculating expected values. The two models can then be combined with p autoregressive terms and q moving-average terms to create the following *Autoregressive Moving Average (ARMA)* model:

$$X_t = \sum_{i=1}^{p} a_i \cdot X_{t-i} + \sum_{i=1}^{q} b_i \cdot \epsilon_{t-i} + c + \epsilon_t \tag{9.7}$$

The aforementioned model is the $ARMA(p, q)$ model. A key question here is about the choice of the parameters p and q in these models. If the values of p and q are selected to be too small, then the model will not fit the data well, and the absolute values of all noise terms will be too large to provide information about true anomalies. On the other hand, if the values of p and q are selected to be too large, then the model is likely to overfit the data. In

such cases, all noise terms will be close to 0. In general, it is good to select the values of p and q as small as possible, so that the model fits the data reasonably well. If one chooses to use larger values of p and q, then it becomes increasingly important to use regularization within the linear regression. The optimal choices of the model and parameters can be identified by minimizing the forecasting error on the observed data using leave-one-out cross-validation.

In some cases, the time series may have some persistent trends, as a result of which it may drift away from the mean. A random-walk time series would be an example of such a situation. This is referred to as the *non-stationarity* of the time series. Non-stationarity is a problem from the perspective of forecasting because the older data becomes stale over time, and it may no longer remain relevant for regression-based modeling. For example, one cannot expect to predict prices today based on the prices from a hundred years back, using a window-based regression model. This is because the prices show a clear trend, and the statistics of the price-based time series today may be very different from that a hundred years back. In such cases, the series can be de-trended by first differencing the time series before ARMA modeling. Such a modeling approach is referred to as *Autoregressive Integrated Moving Average Model (ARIMA)*. In other cases, a function such as the logarithm is applied to the series before the differencing operation. Specific details of the solution techniques for these models are beyond the scope of this book, and are discussed in detail in [467].

9.2.2 Multiple Time Series Regression Models

In many applications, multiple time series are available which can be used in order to perform the prediction of time-series values in a more robust way. The idea is that different time series may often contain the same information, and may sometimes contain *lag correlations*. For example, a bird-call at one sensor will also be heard at a nearby sensor, albeit with a small lag. Such lag correlations can be used in order to make more robust predictions. The standard regression-based model can be generalized to this case, by defining the variable X_t^j in terms of its past history, as well as the history of other time series. A number of common methods are discussed in the following.

9.2.2.1 Direct Generalization of Autoregressive Models

The basic idea in multivariate autoregressive models is to predict the values at each time-stamp with the past window of length p. The main difference from univariate regressive models is that the value at each time-stamp (for any particular series) is predicted as a linear function of all the $d \cdot p$ values in *all* the streams in the previous window of length p. Therefore, by sliding the window over the series, one can generate a system of linear equations with $d \cdot p$ coefficients. The coefficient a_i^{kj} represents the predictive power of the ith previous time-stamp (from current time-stamp) of series k on the current time-stamp of the jth series. Let t be the current time-stamp. Therefore, if $X_t^1 \ldots X_t^d$ represent the tth values of all the d different series, the simple autoregressive model expresses X_t^j as follows:

$$X_t^j = \left[\sum_{k=1}^{d} \sum_{i=1}^{p} a_i^{kj} \cdot X_{t-i}^k \right] + c^j + \epsilon_t^j \tag{9.8}$$

Note that the main difference between Equations 9.1 and 9.8 is that the former expresses a time series as a linear function of its own immediate history, whereas the latter expresses a time series as a linear function of not only its own recent history but also that of the

other time series. Therefore, for n time-stamps, one can now construct a matrix D of size $(n-p) \times [(d \cdot p)+1]$, where the ith row of D is the following $(d \cdot p+1)$-dimensional vector:

$$[X_i^1 \dots X_{i+p-1}^1, X_i^2 \dots X_{i+p-1}^2, \dots, X_i^d \dots X_{i+p-1}^d, 1]$$

Therefore, D is an $(n-p) \times (d \cdot p+1)$-dimensional matrix. Similarly, we define Y to be an $(n-p) \times d$ dimensional matrix in which the jth column contains the values of the time-stamp for the jth time series at ticks $p+1, \dots n$. In other words, the jth column of Y contains the $(n-p)$ entries corresponding to $X_{p+1}^j \dots X_n^j$. Therefore, we have:

$$Y = \begin{bmatrix} X_{p+1}^1 & \dots & X_{p+1}^d \\ X_{p+2}^1 & \dots & X_{p+2}^d \\ \dots & \dots & \dots \\ X_n^1 & \dots & X_n^d \end{bmatrix} \tag{9.9}$$

One can write Equation 9.8 in matrix form as follows:

$$Y \approx D \begin{bmatrix} a_p^{11} & \dots & a_p^{1d} \\ a_{p-1}^{11} & \dots & a_{p-1}^{1d} \\ \dots & \dots & \dots \\ a_1^{11} & \dots & a_1^{1d} \\ a_p^{21} & \dots & a_p^{2d} \\ a_{p-1}^{21} & \dots & a_{p-1}^{2d} \\ \dots & \dots & \dots \\ a_1^{21} & \dots & a_1^{2d} \\ \dots & \dots & \dots \\ a_p^{d1} & \dots & a_p^{dd} \\ a_{p-1}^{d1} & \dots & a_{p-1}^{dd} \\ \dots & \dots & \dots \\ a_1^{d1} & \dots & a_1^{dd} \\ c^1 & \dots & c^d \end{bmatrix} \tag{9.10}$$

It is helpful to compare this equation with the univariate case of Equation 9.4. Note the use of approximate equality "≈" because we have omitted the errors ϵ_t^j in Equation 9.8. These errors need to be minimized with least-squares optimization.

We can denote the matrix of coefficients in Equation 9.10 by A. Therefore, one needs to learn a $(d \cdot p+1) \times d$ matrix of coefficients A, which minimizes the least-squares error of the following relationship:

$$Y \approx DA \tag{9.11}$$

As in the previous case, the least-squares solution for A can be expressed as follows:

$$A = (D^T D)^{-1} D^T Y \tag{9.12}$$

One can perform regularization by adding αI to $D^T D$ for some small value of $\alpha > 0$. It is noteworthy that the matrices $D^T D$ and $D^T Y$ can be maintained incrementally as new rows are added to D and Y with each time-stamp, although one still needs to invert a matrix of size $(d \cdot p+1) \times (d \cdot p+1)$ in each iteration. However, it is also possible to perform the inversion incrementally with the use of the matrix-inversion lemma [592].

At each time instant, a total of d different residuals denoted by ϵ_t^j (for different values of j), which correspond to the outlier scores of the d different series. Large absolute values

of ϵ_t^j are reported as anomalies. Note that the scores within a series are comparable to one another but the scores across different series might not be comparable to one another if the series are not normalized to unit variance as a preprocessing step. This is because the values of ϵ_t^j across different values of j (which might represent different physical quantities such as temperature and pressure) are likely to have different means and standard deviations. Normalization is often not possible in online settings. Therefore, the normal distribution assumption (or the t-distribution) can be used in order to determine the level of significance of the different anomalies. In general, the different values of ϵ_t^j for a *fixed* value of j are assumed to be drawn from a normal or t-distribution.

This broad formulation can also be extended to the autoregressive moving average (ARMA) and autoregressive integrated moving-average models (ARIMA). An exponential forgetting mechanism can also be incorporated in order to give more importance to the recent history of the stream in learning the cross-stream and autoregressive coefficients. This is because the stream auto-correlations and cross-correlations may also change significantly over time.

One problem with the approach is that of increased computational complexity because of the inversion of a matrix of size $(d \cdot p+1) \times (d \cdot p+1)$. How can one use the basic principle of multivariate regression for forecasting, while keeping the complexity to a manageable level? Two such methods have been proposed in the literature:

1. One can select a subset of streams in order to perform the regression with respect to a smaller set of variables.

2. A fundamentally different approach is to use the notion of *hidden variables* to decompose the multivariate forecasting problem into a (more easily solvable) set of univariate forecasting problems.

In the following, we will discuss each of these mechanisms.

9.2.2.2 Time-Series Selection Methods

The *Muscles* and *Selective Muscles* techniques [592] can speed up the regression by using recursive and selection-based tricks. The *Muscles* approach is a relatively straightforward application of the least-squares model for multivariate regression. A variation known as *Recursive Least Squares* is employed to solve the regression more efficiently. The basic idea is that the solution to linear regression requires the inversion of a $(p \cdot d + 1) \times (p \cdot d + 1)$ matrix. In a real-time application one would need to perform the inversion at each time-stamp. The work in [592] shows how to do the inversion incrementally in an efficient way with the use of the matrix-inversion lemma [592]. Refer to [592] for details of this lemma. Another difference from the standard multivariate autoregressive model is that the current values of the other series are also used for the purposes of prediction. This provides a more accurate estimation of the time stamps, and is also therefore likely to improve the accuracy of anomaly detection. It is noteworthy, however, that the current values of the other series might not always be available in a given application.

In spite of its leveraging of the matrix inversion lemma, the *Muscles* technique is slow because too many series are used. In practice, most of the series do not have predictive correlations towards one another and the presence of irrelevant series in the model can cause overfitting. The *Selective Muscles* technique therefore uses only a small subset of the series for predictive modeling. For each time series $\overline{X^j}$, which needs to be predicted, a subset S_j of predictor streams (satisfying $|S_j| \ll d$) needs to be identified.

A greedy algorithm forms the basis of the *Selective Muscles* technique [592]. The first series to be added to S_j is the one with the highest correlation coefficient to $\overline{X^j}$. Subsequently, the next series to be selected minimizes the expected prediction error of the values in $\overline{X^j}$, when added to the current set of series in S_j. This process is continued until k series have been selected or the prediction error cannot be reduced any further. These series are then used in order to make the predictions for the jth time series (in addition to an autoregressive model on the jth series). These predictions are used for anomaly detection by finding values at time stamps which deviate significantly from expected values. The subset selection process can be performed periodically in the context of an evolving stream, in order to minimize the overhead resulting from the selection algorithm itself.

9.2.2.3 Principal Component Analysis and Hidden Variable-Based Models

The aforementioned models predict the values in each stream both from its own history and those of other streams. Even with the use of *Selective Muscles*, the approach can be slow. Furthermore, such an approach is sometimes not effective because of the increased number of regression coefficients, which makes the regression modeling more susceptible to noise and outliers. A specific example is presented in Figure 3.3 of Chapter 3, where a complete breakdown of regression analysis is caused by a single outlier.

As discussed in Chapter 3, PCA-based techniques are generally more robust to the presence of noise and outliers. Such PCA-based methods are able to express a large number of correlated data streams into a small number of uncorrelated data streams, which facilitates compact and noise-resistant autoregressive analysis. The most well-known technique among these methods is SPIRIT [427]. The basic idea here is that one can construct a $d \times d$ covariance matrix between the various streams. The projections of the d-dimensional values at each time-stamp on the top-k eigenvectors of this matrix provide a new set of k uncorrelated time series. These time series are also referred to as the *hidden variables*. The remaining $(d-k)$ time series have very little variance and can be treated as constant time series, which do not need to be explicitly forecasted with an autoregressive model (since their constant values provide a trivial forecast). Furthermore, one only needs to perform the analysis on the larger eigenvectors, with *a single auto-correlation model*. Once the k values of the (hidden) time series have been predicted at a particular time-stamp, one can combine them with the remaining $(d-k)$ (constant) values and transform the predictions to the original space. The use of auto-correlation on hidden variables provides a more robust prediction than multivariate regression. One can report the deviations from forecasted values (in each stream) as the outlier scores for the corresponding stream/time-stamp pair. This deviation can also be fitted to the normal distribution or t-distribution in order to convert the scores to binary labels. Another possibility is to create a composite deviation by computing the sum of the squares of the k components after scaling each deviation to unit variance. Since this is the sum of the squares of k normal distributions, a χ^2-distribution can also be used to measure the significance of the composite level of deviation.

In the time-series streaming setting, the incremental implementation of PCA-based methods also requires the incremental maintenance of a covariance matrix. This can be easily achieved with the observation that all covariances can be maintained as functions of additive stream statistics. Note that the covariance between the time-series $\overline{X^j}$ and $\overline{X^k}$ at time t can be expressed as follows:

$$Cov(\overline{X^j}, \overline{X^k}) = \frac{\sum_{i=1}^{t} X_i^j \cdot X_i^k}{t} - \frac{\sum_{i=1}^{t} X_i^j}{t} \cdot \frac{\sum_{i=1}^{t} X_i^k}{t} \tag{9.13}$$

The computation above requires the maintenance of (i) the d additive sums of the values of each variable in the stream, and (ii) the d^2 additive sums of the pairwise products of the time series values in the stream. This can easily be achieved in an incremental setting. Furthermore, an exponential forgetting mechanism can be incorporated in order to ensure that the co-variance matrix is based on the recent history of the stream, rather than the entire stream of data points. Therefore, the overall approach may be summarized as follows:

1. Transform the d time series into $k \ll d$ uncorrelated hidden time series with PCA. The remaining $(d-k)$ (hidden) time series have little variance and, therefore, their constant values[3] can be noted.

   Let t be the current time-stamp. If P_k is the $d \times k$ matrix with columns containing the top-k eigenvectors, then the values in the k hidden time series at the rth time-stamp (for each $r < t$) are contained in the k-dimensional row-vector $[Y_r^1 \ldots Y_r^k]$ using the transformation discussed in Equation 3.15 of Chapter 3:

   $$[Y_r^1 \ldots Y_r^k] = [X_r^1 \ldots X_r^d] P_k \quad \forall r < t \tag{9.14}$$

2. For each of the k time series, use univariate auto-correlation forecasting methods to predict the value at time-stamp t. For the remaining $(d-k)$ hidden time series, their (roughly) constant values at time t are already known. This results in the d-dimensional hidden representation of the time series in the transformed space. Note that to create the forecasts in real-world applications, one can only use values up to time $(t-1)$. However, if the entire time series is available for offline analysis, it is possible to use future values as well, although the predicted value would not be technically referred to as a forecast.

3. The new d-dimensional representations are transformed back to the original space to create *forecasts* at time t. The forecasted values at time t can be expressed as follows:

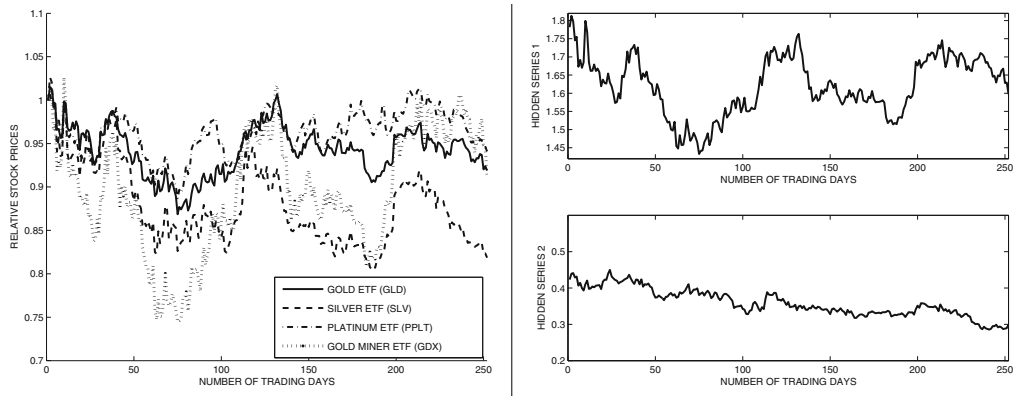
   $$[\hat{X}_t^1 \ldots \hat{X}_t^d] = [Y_t^1 \ldots Y_t^d] P_d^T \tag{9.15}$$

   Note that P_d^T is the reverse transformation of P_d because we have $P_d P_d^T = I$. The "hat" (circumflex) symbol on top of $\hat{X}_t^j$ denotes that it is a forecasted value.

4. Compute the absolute deviations $|\hat{X}_t^j - X_t^j|$ as the outlier scores. One can standardize these deviations to zero mean and unit variance over a particular time series and use a t-value distribution to report specific time-stamps at which a time series shows abnormal behavior. Note that the outlier score is specific to the combination of time-stamp and time series.

One can also create a composite score across all the time series by summing up the squares of the scores across all the time series. This score provides outlier scores for time-instants, rather than time-stamp/time-series pairs. Examples of the hidden series for four precious metal exchange-traded funds (ETFs) and their hidden series are illustrated in Figure 9.2. Each of the four series can be approximately expressed as a linear combination of the top-2 hidden series. It is noteworthy that the main trends in the four series can be summarized with only two series using PCA. Furthermore, these series are uncorrelated with one another and each of them can be forecasted efficiently with the use of univariate methods. These forecasts can then be transformed to the original space in order to discover series that are behaving anomalously.

[3]For mean-centered time series these values are zero. For non-mean centered time series, one can transform the d-dimensional mean of the all time series to the new space using Equation 9.14 with all $k = d$ eigenvectors included.



(a) Precious metal ETFs (b) Hidden variable series

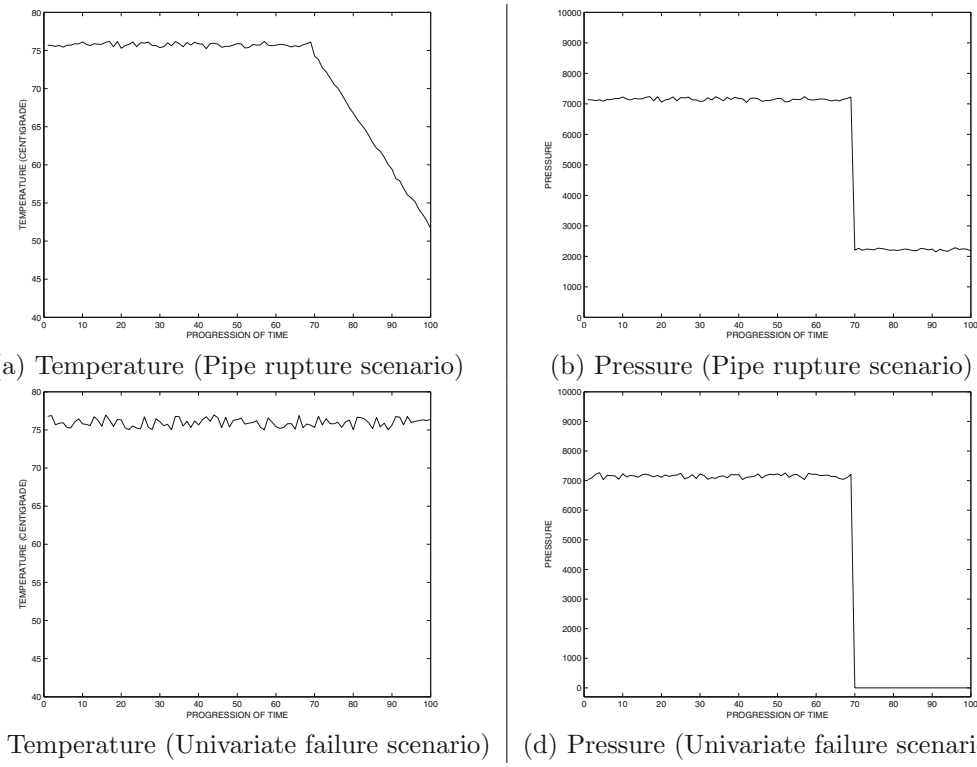
Figure 9.2: Example of four precious metal exchange traded funds (ETFs) and their top-2 hidden series [33]. Note the greater variability in the top hidden series and its ability to summarize the broader/comon trends in all four series. Each of the four series can be approximately expressed as a different linear combination of the two hidden series.

9.2.3 Relationship between Unsupervised Outlier Detection and Prediction

The methods in this section use supervised prediction and forecasting methods for unsupervised outlier detection. The close relationship between prediction and outlier detection is not restricted to the time-series domain; the method in section 7.7 of Chapter 7 shows how one can transform a generic instance of multidimensional outlier detection to a set of supervised prediction problems [429]. Outliers are, after all, violations of the "normal" model of data dependencies. A prediction model, therefore, helps in modeling these dependencies as they apply to a specific data point. Violations of these dependencies represent violation of the model of normal data and therefore correspond to outliers.

9.2.4 Supervised Point Outlier Detection in Time Series

The aforementioned methods determine the significant deviations from the expected values and report them as outliers. In many cases, such deviations could have many causes that are not necessarily indicative of events of interest. For example, in the context of an environmental monitoring applications, many deviations may be result of the failure of the sensor equipment or another spurious event that causes deviations in sensor values. This may not necessarily reflect an anomaly of interest. Although anomalous events often correspond to extreme deviations in sensor-stream values, the precise causality of different kinds of deviations may be quite different. Therefore, in the context of noisy time-series data, the anomalies of interest may be embedded among a number of spurious abnormalities, which may not be of any interest to an analyst. For example, consider the case illustrated in Figure 9.3, in which we have illustrated the temperature and pressure values inside pressurized pipes containing heating fluids. Figures 9.3(a) and (b) illustrate values on two sensors in a pipe-rupture scenario. Figures 9.3(c) and (d) illustrate the values of the two sensors in a scenario where the pressure sensor malfunctions, and this results in a value of 0 at each tick. In the first case, the readings of both pressure and temperature sensors are affected by the



(a) Temperature (Pipe rupture scenario)

(b) Pressure (Pipe rupture scenario)

(c) Temperature (Univariate failure scenario)

(d) Pressure (Univariate failure scenario)

Figure 9.3: Readings of temperature and pressure sensors in various scenarios

malfunction, although the final pressure values are not zero; rather, they reflect the pressure in the external surroundings. The readings on the temperature sensor are not affected at all in the second scenario, since the malfunction is specific to the pressure sensor.

So how does one distinguish between noise and true anomalies of interest to the analyst? The time-tested method for making the approach more sensitive to analyst interest is to use *supervision* from previous examples. In the multivariate scenario, the truly anomalous events of interest may be detected only from the *differential* behavior of the deviations across different time-series data streams. In such scenarios, supervision can be very helpful in distinguishing the true anomalies from the spurious abnormalities in the time-series data stream. The approach discussed in [9] proposes a method for abnormality detection in spuriously populated data streams. It is assumed that the true events of interest are available as the *ground-truth time stamps*, $T_1 \ldots T_r$, which are used for supervision. These are referred to as *primary abnormal events*. Note that $T_1 \ldots T_r$ might include consecutive lag periods because time-series abnormalities often show up over small lag windows.

In addition, it is possible that some other causative factors might cause secondary abnormal events. Although the training data for such spurious events can also be continuously collected in some settings [9], this may not always be possible. Nevertheless, one can still discriminate between the primary and secondary events by using the normal time stamps as baseline periods. The following description is a simplified and generalized version of the approach in [9], which does not use specific knowledge of the secondary events.

The overall process of event prediction is to create a composite alarm level from the error terms in the time-series prediction. The first step is to use a univariate time-series

prediction model in order to compute the error terms for each series/time-stamp pair. This step is identical to the methods discussed earlier in this section, and any of these methods can be used, with or without exponential forgetting factors. These error terms are then normalized to Z-values for the d streams, and their *absolute* values at the tth time-stamp are denoted by $z_t^1 \ldots z_t^d$. Then, one can use a set of coefficients, $\alpha_1 \ldots \alpha_d$ to create the composite alarm level Z_t at time t:

$$Z_t = \sum_{i=1}^{d} \alpha_i \cdot z_t^i \tag{9.16}$$

This vector of *discrimination coefficients* $\alpha_1 \ldots \alpha_s$ should be learned with the objective of maximizing the differences in the alarm level between the primary events and normal time periods. The alarm level $Q^p(\alpha_1 \ldots \alpha_d)$ at the time of the primary events is as follows:

$$Q^p(\alpha_1 \ldots \alpha_d) = \frac{\sum_{i=1}^{r} Z_{T_i}}{r} \tag{9.17}$$

We can also compute the normal alarm level $Q^n(\alpha_1 \ldots \alpha_d)$ for the entire time series of length n is as follows:

$$Q^n(\alpha_1 \ldots \alpha_d) = \frac{\sum_{i=1}^{n} Z_i}{n} \tag{9.18}$$

Then, we would like to learn the coefficients to maximize the discrimination between the alarm at the primary events and normal periods:

$$\text{Maximize } Q^p(\alpha_1 \ldots \alpha_d) - Q^n(\alpha_1 \ldots \alpha_d)$$
$$\text{subject to:}$$
$$\sum_{i=1}^{d} \alpha_i^2 = 1$$

The normalization constraint on the coefficients is necessary to prevent unbounded solutions. This objective function essentially provides the maximum discrimination between the primary events and normal periods. One can use any off-the-shelf optimization solver in order to learn $\alpha_1 \ldots \alpha_d$. The learned coefficients are used to construct the outlier scores of each time-stamp according to Equation 9.16. In practice, the anomaly detection and learning processes are executed simultaneously, as new events are encountered. As the result, the vector of discrimination coefficients can be learned more accurately over time.

9.3 Time-Series of Unusual Shapes

Much of the work on time-series outlier detection is to determine *unusual changes or very large deviations* from the underlying series. However, certain types of deviations are based not only on the individual deviations of the data points, but also on the *shapes* of specific portions of the time series with respect to the other extracted portions. For example, consider the case of the flash-crash illustrated in Figure 9.1(a). In this case, the stock market showed very unusual behavior over *multiple time stamps* by first dropping precipitously over a very short period, and then recovering very quickly to almost the original level. This unusual behavior had a different causality from most other drops in the stock market. Therefore, the *shape* of the series is different from other large deviations. It is clear that

determining large deviations from previous time-stamps cannot *differentially* discover such anomalies, because the entire series (or subsequence) needs to be viewed from the perspective of other normal series in the database. On other words, the set of time-stamps must be viewed *collectively* in order to learn whether or not they should be considered anomalies.

The goal in such methods is to determine windows of the data (or subsequences) in which a given series behaves differently from a database of multiple sequences. Unlike the cases discussed earlier in this chapter where outliers are defined by *a single position*, such outliers correspond to multiple consecutive time stamps; the unusual co-occurrence of specific patterns at these time stamps represent outliers. Thus, the previous cases correspond to *contextual* outliers, whereas this case corresponds to *collective* outliers. Two possibilities may exist within this case of anomaly detection:

- **Full-series anomaly:** In this case, the shape of the entire series is treated as an anomaly. This shape is compared against a database of similar time series. However, in most cases, unless the database of sequences corresponds to a relatively short segment of time-stamps, the noise variations within the series will mask the anomalous shape. This is analogous to the problems of noise encountered in detecting outliers in high-dimensional data.

- **Subsequence-based anomaly:** If the time series is collected over long periods of time, then we might have a single time series that shows typical trends over shorter time-periods. Therefore, the anomalous shape is detected over small windows of the time series as deviations from these typical trends.

The distinction between these two types of problems is somewhat artificial because most of the methods for subsequence anomaly detection can also be used for full-series anomaly detection. This is because the first step in subsequence anomaly detection is to extract windows from the time series and then treat the extracted subsequences as whole series for the purpose of anomaly detection. Subsequently, various types of distance-based, probabilistic, or linear methods can be applied on the extracted windows.

There are, however, some subtle implementation differences between subsequence anomalies and whole-series anomalies. This is because one must always account for the overlap between adjacent windows when addressing subsequence anomalies. For full-series anomalies the longer length of the series is a greater concern. In all cases, it is assumed that the series has been normalized to zero mean and unit standard deviation because it is not meaningful to compare series of different means and amplitudes. Furthermore, in many models, it is assumed that all the underlying series are of the same length n.

9.3.1 Transformation to Other Representations

There are two types of transformations that are used to compare the various segments of a time series:

- **Numeric multidimensional transformation:** In one case, the series (or each subsequence of the series) is transformed into a multidimensional vector. The representation of each dimension corresponds to numeric coefficients in standard vector space. Proximity can be computed on this representation with the Euclidean distance. Therefore, many of the methods discussed in Chapter 4 of this book can be used in order to determine proximity-based anomalies.

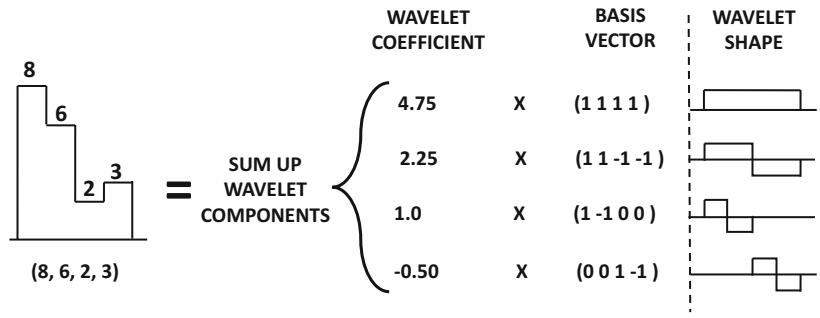


Figure 9.4: Wavelet decomposition of series of length 4. The wavelet coefficients implicitly capture long-term and short-term contextual dependencies.

- **Discrete sequence transformation:** In the second case, the series can be transformed to symbolic representations by using discretization methods. In such cases, the methods discussed in Chapter 10 can be used in order to determine unusual shapes in the time series.

This section will discuss transformations of various types.

9.3.1.1 Numeric Multidimensional Transformations

The simplest possible transformation would be to consider each window of length n in the time series as a multidimensional vector of length n. Other methods such as *Discrete Wavelet Transform (DWT)* and *Discrete Fourier Transform* are available for compressing the series with the multi-scale approach into numeric coefficients [439].The most common type of wavelet is the *Haar wavelet*. The global average of the series is stored as the first coefficient. Then, for a time series of length n, one can generate the remaining $(n-1)$ wavelet coefficients recursively as follows:

1. Report half the difference between the averages of the first-half and second-half of the series as the first wavelet coefficient. This provides the most basic global trend between the first-half and second-half of the series.

2. Recursively compute the $(n/2-1)$ wavelet coefficients for *each* of the first half and second half of the series in order to determine the remaining $(n-2)$ coefficients. These provide local trends in each of the two halves of the series.

Note that the number of wavelet coefficients is exactly equal to the length of the time series. It can be shown [33] that each point in the series can be reconstructed *exactly* with this set of wavelet coefficients. The time series can be exactly reconstructed as a coefficient-weighted sum of n primitive, wavelet-like, time series, each of which looks like a step function drawn on $\{-1, 0, 1\}$. Note that each coefficient represents the difference between the first half and second half of a particular segment of a series. The wavelet-like primitive series for a particular coefficient is defined by (i) setting the first-half of the relevant segment of the series to $+1$, (ii) setting the second half of the relevant segment of the series to -1, and (iii) setting all other portions of the time series to 0.

For example, consider the time series $(8, 6, 2, 3)$. The global average of the series is 4.75. The top-level coefficient is $(7 - 2.5)/2 = 2.25$. The two second-order coefficients for the

first-half and second-half of the series are $(8-6)/2 = 1$ and $(2-3)/2 = -0.5$. Therefore, the wavelet representation is $(4.75, 2.25, 1, -0.5)$. One can reconstruct the original series as an additive sum of primitive wavelet series as follows:

$$[8,6,2,3] = \underbrace{4.75 * [1,1,1,1] + 2.25 * [1,1,-1,-1]}_{\text{Long-term trends}} + \underbrace{1 * [1,-1,0,0] + (-0.5) * [0,0,1,-1]}_{\text{Short-term trends}}$$

A pictorial illustration of the decomposition of the time series into its wavelet coefficients is shown in Figure 9.4. Note that the basis vectors are mutually orthogonal and the coefficients of higher granularity capture detailed (local) trends, whereas coefficients of lower granularity capture long-term (global) trends. In this particular example, we only have two levels of granularity. Wavelet decomposition is inherently a *multiresolution* decomposition, in which the number of levels of resolution is $O(\log_2(n))$ for a series of length n.

For a database containing N time series of length n, one can use the wavelet transform to create N new multidimensional data points with dimensionality n. Although it is common to drop small coefficients while using *single series*, it is much harder to prune small wavelet coefficients in a database of *multiple* series because a particular coefficient might be small in *most* series but a large value of that coefficient in even a single series might be relevant from the point of view of outlier detection. Therefore, all coefficients are retained.

Since the transformed representation is of the same size (i.e., $N \times n$) as the original data, what is the advantage of the wavelet representation? The key point of the wavelet transform is that one can treat the new representation *as a multidimensional data set rather than as a dependency-oriented data set because the short-term and long-term dependencies in the data are already encoded inside the wavelet coefficients.* Therefore, one can use traditional models for outlier direction of multidimensional data (like one-class SVMs or subspace methods) on the wavelet representation in a straightforward way. Other than a few distance-based models, it is generally not possible to (effectively) use off-the-shelf models with the original time-series representation (because they ignore the underlying dependencies among data values). Distance functions such as the Euclidean function are preserved, when using[4] the wavelet coefficients rather than the time series. This is because the new set of basis vectors are mutually orthonormal, and rotating the axis system does not affect Euclidean distances.

The discrete Fourier transform is an alternative way to perform dimensionality reduction and transformation into a new representation in which implicit dependencies are encoded in coefficients. However, in the case of the Fourier transform, the focus is on capturing *periodic* dependencies rather than local variations. Interested readers may refer to [33] for details of the Fourier transform. Which of these transformations is more effective and when should one use each? In general, a transformation is more effective when only a small number of coefficients are dominant in the representation. This tends to magnify outliers when (typically) non-dominant coefficients show large values. There are a few general guidelines in selecting the correct transformation:

1. Time series that have a significant amount of seasonality or periodicity built into them generally work well with the DFT.

2. Time series that tend to show local continuity (i.e., values that do not vary much over small ranges) generally work well with the DWT.

[4]The coefficients need to be normalized so that the basis vectors have unit norm. The basis vectors in Figure 9.4 are not normalized. The normalized coefficients in this case would be $(4.75\sqrt{4}, 2.25\sqrt{4}, 1\sqrt{2}, -0.5\sqrt{2})$.

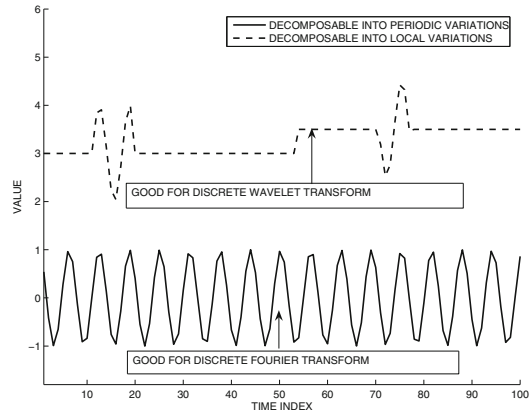


Figure 9.5: Effectiveness of different transformations in different settings

Therefore, it is often useful to have a deeper semantic understanding of the problem domain at hand before choosing the specific transformation to be used. Examples of two types of series in which the two transformations will work well are shown in Figure 9.5. A precise explanation of why such types of transformations are more relevant in various scenarios is provided in [33]. It is also possible to apply some of these transformations to multivariate series and spatial data with appropriate modifications.

Euclidean distances can be utilized on these representations to compute outlier scores. In particular, given two subsequences (or transformed representations) of length n denoted by $A = (a_1 \ldots a_n)$ and $B = (b_1 \ldots b_n)$, the Euclidean distance between them can be computed as follows:

$$Dist(A, B) = \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \tag{9.19}$$

The kth nearest-neighbor distance to a series can be used as its outlier score. A major challenge is that all these methods require $O(N^2)$ time to compute pairwise distances, where N is the total number of subsequences. As discussed in Chapter 4, this is a problem with all such distance-based models. As in the case of multidimensional data, section 9.3.2 will discuss analogous pruning methods for improving computational efficiency.

9.3.1.2 Discrete Sequence Transformations

In theory, it is possible to convert continuous time series to discrete data, and then use the methods discussed in the next chapter for anomalous shape discovery. This transformation is typically performed on windows of the data, and it leads to a compressed and approximate representation of the underlying time series. Such methods can either be used for *stand-alone* anomaly detection of the discrete sequences with the methods discussed in Chapter 10, or for improving the *efficiency* of the nearest-neighbor detectors discussed above by quick approximate representations and pruning. The latter case will be discussed in the next subsection. A variety of discrete transformations are possible such as (symbolically discretized representations of) the means over specific windows, slopes over specific windows, discrete wavelet coefficients, and Fourier transform coefficients. The specific representation which is used should depend upon the application domain at hand.

A commonly used discretization technique is the *Symbolic Aggregate Approximation (SAX)* [361]. In this method, *Piecewise Aggregate Approximations (PAA)* are used in order to represent the time series. This method comprises two steps:

- **Window-based averaging:** The series is divided into windows of length w, and the average time-series value over each window is computed.
- **Value-based discretization:** The (already averaged) time-series values are discretized into a smaller number of approximately *equi-depth* intervals. The idea is to ensure that each symbol has an approximately equal frequency in the time series. The actual interval boundaries are constructed by assuming that the time-series values are distributed with a Gaussian assumption. It is to be noted that the mean and standard deviation of the (windowed) time-series values need to be computed in order to construct the Gaussian distribution. The quantiles of the Gaussian distribution are used to determine the boundaries of the intervals. Typically, the values are discretized into 3 or 4 intervals for the best results [311]. Each such equi-depth interval is mapped to a symbolic value. This creates a symbolic representation of the time series.

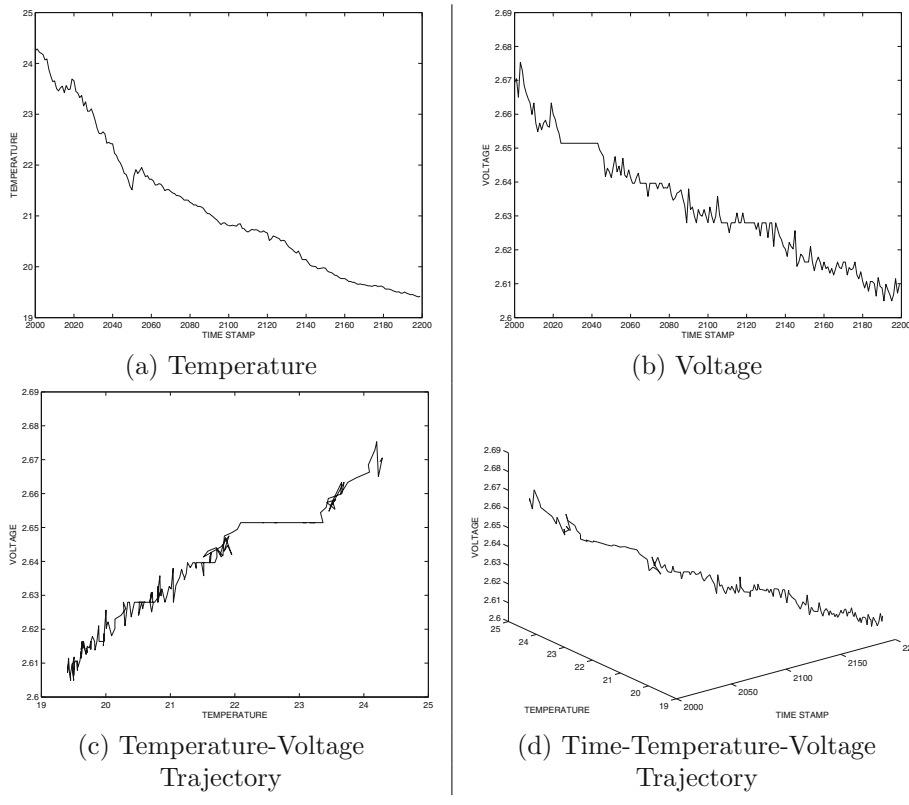
It is important to note that symbolic discretization does lose some information about the underlying time series. For example, the symbols provide no information about how close or far the different intervals are from one another. Nevertheless, such approximations are useful in streaming scenarios because of their simplicity, ease in construction, and their ability to facilitate the use of various pattern- and rule-based models. Such models are discussed in the next chapter.

9.3.1.3 Leveraging Trajectory Representations of Time Series

The case of finding unusual shapes from multivariate series is much more challenging. Here, different behavioral attributes such as temperature, pressure, or the same behaviorial attribute such as the temperature may be measured *at the same instant* by different sensors. The problem of finding unusual shapes therefore needs to be very carefully defined in this case. As will be evident from the subsequent discussion, this problem maps directly to that of trajectory outlier detection, which will be discussed in detail in Chapter 11.

In multivariate temporal data, the different behavioral attributes are typically measured with the use of multiple sensors simultaneously. An example is the *Intel Research Berkeley Sensor data* described in [427], which measures different behavioral attributes over time. For example, the behavior of one of the temperature and pressure sensors at the same segment of time is illustrated in Figures 9.6(a) and (b), respectively.

Existing work on trajectory outlier detection can be leveraged to detect outlier shapes in this scenario. Even though the existing work [347] has been designed for spatial trajectories, it can also be extended to the non-spatial case with arbitrary values on the X-coordinates and Y-coordinates. In this case, it is possible to visualize the variation of the two behaviorial attributes by eliminating the common time attribute, or by creating a 3-dimensional trajectory containing the time and the other two behaviorial attributes. Examples of such trajectories are illustrated in Figures 9.6(c) and (d), respectively. The most generic of these trajectories is illustrated in Figure 9.6(d), which shows the simultaneous variation between all three attributes. In general, a multivariate time series with n behavioral attributes can be mapped to a $(n+1)$-dimensional trajectory. One issue is that when the number of behaviorial attributes increases, the dimensionality of the corresponding trajectory also increases. This leads to challenges arising from the curse of dimensionality. In such cases, it may be



(a) Temperature

(b) Voltage

(c) Temperature-Voltage Trajectory

(d) Time-Temperature-Voltage Trajectory

Figure 9.6: Multivariate time series can be mapped to trajectory data

better to explore subsets of behavioral attributes in order to determine outliers. This corresponds to the subspace methods discussed in Chapter 5. This is an extremely difficult problem because it combines trajectory analysis with multivariate time-series analysis, and is still an open problem in the literature.

The TROAD method [347] can be used in order to determine unusual shaped trajectories in such cases. This approach is described in detail in section 11.4 of Chapter 11. While this method was designed for 2-dimensional spatial data, a generalized version of the method can be used for higher dimensional trajectory data containing any kind of attributes. Care needs to be takes to normalize the data along each attribute to unit variance. This is particularly important in the non-spatial scenario, since the different axes of the trajectory may be drawn on very different scales.

9.3.2 Distance-Based Methods

The *Hotsax* approach [311] uses the standard numeric Euclidean distances introduced earlier in this section in order to define distance-based outliers. Furthermore, it uses the discrete approximations in order to perform pruning. The Euclidean distance to the k-nearest neighbor is used in order to determine the outlier score of each subsequence. In order to facilitate pruning, it is assumed that only the scores of the top-r outliers need to be reported.

The outlier analysis is performed over windows of length p. These subsequences are extracted from the time series and the nearest Euclidean distances are determined. The Euclidean distance between two series is computed using Equation 9.19. Care must be taken to compare a window of values with non-overlapping windows, in order to minimize the self-similarity bias of overlapping windows [311]. Therefore, an overlapping window is not included in the computation of the k-nearest neighbor. Therefore, the basic version of the scheme (without pruning) may be described as follows:

1. Extract overlapping subsequences of length p from the time series. For a time series of length n, a total of $(n - p + 1)$ subsequences can be extracted.

2. Report outlier scores as the k-nearest neighbor distance of the subsequences, while excluding overlapping windows in the computation.

In principle, a variety of other similarity or distance functions exist for time-series data, the most prominent of which is *Dynamic Time Warping*. Different types of similarity functions may be qualitatively more effective in different application settings, although the Euclidean distance function has the advantage that it can be efficiently computed and lends itself to an effective pruning methodology. Therefore, more general distance functions may require the use of sophisticated indexes for efficient retrieval. The reader is referred to [33, 229] for reviews of time-series similarity measures and time-series indexing.

Next, we discuss the pruning methodology that is used in the context of Euclidean distance functions. In this case, the assumption is that we only wish to discover the top-r outliers (rather than report all the outlier scores). The goal of pruning is to improve the *efficiency* of the algorithm, but the same outliers will be reported as one would obtain without pruning.

A nested loop approach is used to implement the method. The algorithm examines the candidate subsequences iteratively in an outer loop. For each such candidate subsequence, the k-nearest neighbors are computed progressively in an inner loop with distance computations to other subsequences. Each candidate subsequence is either included in the current set of best r outlier estimates at the end of an outer loop iteration, or discarded via *early*

abandonment of the inner loop without computing the *exact* value of the k-nearest neighbor. This inner loop can be terminated early, when the currently approximated k-nearest neighbor distance for that candidate subsequence is less than the score for the rth best outlier found so far. Clearly, such a subsequence cannot be an outlier. In order to obtain the best pruning results, the subsequences need to be heuristically ordered, so that the earliest candidate subsequences examined in the outer loop have the greatest tendency to be outliers. Furthermore, the pruning performance is also most effective when the subsequences are ordered in the inner loop such that the k-nearest neighbors of the candidate subsequence are found early. It remains to explain how the heuristic orderings required for good pruning are achieved.

Pruning is facilitated by an approach that can measure the clustering behavior of the underlying subsequences. Clustering has a well-known relationship of complementarity with outlier analysis, and therefore it is useful to examine those subsequences first in the outer loop, which are members of clusters containing very few (or one) members. The SAX representation is used in order to create a simple mapping of the subsequences into clusters. The piecewise aggregate approximations of SAX are performed over windows of length $w < p$. Then, each window for outlier analysis examines a sequence of symbols (or words) of length p/w. Subsequences which map to unique words are more likely to be discordants than those which map to the same word. This is because the latter case corresponds to subsequences that are members of the same cluster. This observation is leveraged in [311] in order to design a pruning mechanism for outlier analysis. In the discrete case, since the number of distinct words is often limited, a trie data structure can be used in order to maintain the counts of the distinct words in the subsequences, and identify the more unusual ones. Furthermore, the identities of the set of actual subsequences to which these series map to are also maintained in this data structure. Subsequences that are anomalous can be identified by words with low frequency count. This provides an *ordering* in which to examine the different candidate subsequences. For each candidate subsequence, those subsequences which map to the same word may be considered first for computing the nearest neighbor distances in order to provide quick and tight upper bounds on the nearest neighbor distances. As these distances are computed one by one, a tighter and tighter upper bound on the nearest neighbor distance is computed. *A candidate can be pruned, when an upper bound on its nearest neighbor distance is guaranteed to be smaller (i.e., more similar) than the rth best outlier distance found so far.* Therefore, for any given series, it is not necessary to determine its exact nearest neighbor by comparing to all subsequences. Rather, early termination of the inner loop is often possible during the computation of the nearest neighbor distance. This forms the core of the pruning methodology used in [311], and is similar in principle to one of the pruning methodologies used in multidimensional distance-based methods [456].

The SAX representation is used in order to provide a good *ordering* of candidates for outliers, as well the ordering in which to iteratively compute the nearest neighbor distance for a candidate. A good ordering of the candidates ensures that strong outliers are found early, and therefore a tight *lower bound* on the outlier score is obtained early, as the rth best outlier found *so far*. A good ordering for nearest neighbor computation of a candidate ensures that the iterative nearest neighbor computation process reaches a tight *upper bound* on the outlier score of a particular candidate early. The processing of the candidate can be terminated early, when its upper bound distances are less than the lower bounds on the rth best outlier score. The effectiveness of the pruning is dependent on the tightness of the approximations, which in turn is dependent on the quality of the ordering created by SAX-based clustering. It is relatively easy to extend this approach to multivariate time series by using a multivariate distance function.

9.3.2.1 Single Series versus Multiple Series

The beginning of this section introduced the problem scenario where the subsequence-based time series-based anomalies needed to be detected over a database of multiple time series of the same type. For example, one may need to determine anomalies from a database of a few thousand electrocardiogram (ECG) time series. However, in many cases, a single very long series may be available, and it may be desirable to determine unusual segments of the series from this large series. The case of a single large series is not very different from that of multiple series because one can always extract subsequences of length p in both cases. The anomalous series are discovered from this set of extracted series subsequences. Although some methods such as *Hotsax* assume a single long time series in the original problem definition, others assume multiple series. The (approximate) equivalence of the two definitions ensures that methods for one can be used for the other and vice versa.

The presence of a larger number of series makes the modeling process more robust. Furthermore, if some of these series are known to be normal instances (non-outliers), the data modeling process should use only the normal series because it provides a cleaner model. For example, in a nearest-neighbor anomaly detector, a candidate sequence should be compared only to subsequences from the normal series. It is noteworthy that the case of multiple series of the same type is different from the *multivariate* scenario, in which the different *types* of time series are available, which are synchronized by their time stamps.

9.3.3 Probabilistic Models

Probabilistic models use a generative process to model the generation of a set of time series. For example, one can extract a set of subsequences from a given time series and then model these time series to be generated from one of the components of this mixture. The most common generative model for clustering time series is that of a hidden Markov model (HMM). A HMM can be viewed as an analog of the mixture model discussed in section 2.4 of Chapter 2 except it allows temporal dependencies among different components of the mixture in order to generate the values on successive time-stamps. As in Chapter 2, one can compute the probabilistic fit of each subsequence with respect to the probabilistic model. Data points with low levels of fit are declared outliers. Hidden Markov models are generally more suitable for discrete sequences, although they are sometimes also used for continuous data [500]. Because of the preponderance of these models in discrete data (as opposed to continuous data), a detailed discussion of HMMs will be deferred to the next chapter on outlier detection in discrete sequences.

9.3.4 Linear Models

It is also possible use the linear models discussed in Chapter 3 to discover unusual shapes in the underlying time series. One can use this approach both for the univariate and the multivariate case.

9.3.4.1 Univariate Series

One can repeatedly extract overlapping windows of length p from the time series of length n to create $(n - p + 1)$ objects. Each such window of length p can be treated as a p-dimensional data point. Therefore, one can now extract a data matrix of size $(n-p+1) \times p$. Once the data matrix has been extracted, a $p \times p$ covariance matrix can be constructed on this representation in order to execute PCA. Note that the covariance matrix effectively

contains information about the auto-correlations in the series at maximum lag value of p. Deviations from the normal pattern of auto-correlations in a particular window should be viewed as a window-outlier.

In order to achieve this goal, one can determine the eigenvectors of this covariance matrix to determine a new p-dimensional representation in the transformed space. Each of these p directions is standardized to zero mean and unit variance. The squared distance of each point to the p-dimensional mean of the transformed data provides the outlier score. Note that this approach is a direct application of the Mahalanobis method discussed in section 2.3.4 of Chapter 2 (with a PCA-based interpretation provided in section 3.3.1 of Chapter 3).

As discussed in section 3.5 of Chapter 3, PCA methods are special cases of matrix factorization. One can, in fact, use any type of matrix factorization and not just PCA. In such a case, the approach can also be used for anomaly detection in incompletely specified time series. This is because the method in section 3.5 can be used for anomaly detection in an incompletely specified multidimensional data set. An incompletely specified time series maps to an incompletely specified multidimensional data set using the aforementioned transform. This type of approach can be very useful because time series are often incompletely specified in many settings. It is also noteworthy that matrix factorization provides an anomaly score for each *entry* in the matrix; this can be used in order to flag specific *positions* in the time series as outliers. Therefore, the approach can also be used to determine point outliers. However, such point outliers cannot be used in an *online* setting because anomaly scores at specific positions might be using the values of the time series at future positions. Nevertheless, the approach can still be used to flag outliers in the context of retrospective and offline analysis.

9.3.4.2 Multivariate Series

The approach can also be applied to multivariate time series. The main difference is in terms of *feature engineering*; in other words, the multidimensional representation is created differently. Instead of extracting p-dimensional points from a univariate time series, we now extract $p \cdot d$-dimensional points from the set of d different time series. For the d different time series there are $d \cdot p$ different values in the most recent window of length p. One can treat these values as a single $(d \cdot p)$-dimensional data point, and slide a window across the data stream to generate $(n - p + 1)$ such points.

These points can be used in order to create a $d \cdot p \times d \cdot p$ covariance matrix. Each entry of this covariance matrix provides the relationship both across streams and time over the past window of length p. As a result, the covariance matrix computes the covariances not only between different time-points in the same series (auto-correlations), but also between across different time series. The value of p represents the maximum lag at which such covariances are tracked. Window-based anomalies are defined as those windows in which the correlations across different time series and time-points are different from the "usual" trends. These usual trends are summarized in the covariance matrix, and PCA can be used to extract the key hidden directions of correlation.

The remaining steps of generating the outlier scores are similar to the univariate case. The eigenvectors of this covariance matrix can be used to transform the $(d \cdot p)$-dimensional data points into a new space. One can standardize the $d \cdot p$ transformed coordinates in this new space. The distance of each of these $d \cdot p$ transformed points from the mean of all the points provides the Mahalanobis score for each window of length p. Note that this approach provides an outlier score for a window of length p *across all different series*. In other words,

if the entire set of series are *collectively* behaving unusually over a window of length p, then that window should be considered an outlier. This approach requires the computation of the eigenvectors of a covariance matrix of size $(d \cdot p) \times (d \cdot p)$. This can sometimes be computationally expensive.

One can also use matrix factorization as a generalization of PCA (cf. section 3.5 of Chapter 3) on the extracted multidimensional data set with dimensionality $d \cdot p$. As in the univariate case, the matrix factorization methodology has the advantage that it can be used for incompletely specified time series. Furthermore, specific positions in the time series can be flagged as outliers because matrix factorization associates each *entry* in the matrix with an outlier score.

9.3.4.3 Incorporating Arbitrary Similarity Functions

It is often overlooked that PCA makes the implicit assumption that the distances between various series are represented by the Euclidean distance function. This may not always be the case in real applications. For example, in the univariate case, one might want to use distance functions like edit distance or dynamic time-warping (DTW). Such distance functions can be addressed by using kernel PCA methods discussed in section 3.3.8 of Chapter 3. In this case, one generates the embedding *directly*[5] from the similarity matrices, rather than creating a basis system from the covariance matrices. Consider a setting in which $(n - p + 1)$ multidimensional points have been extracted from the time series. The steps are as follows:

1. Construct an $(n - p + 1) \times (n - p + 1)$ similarity matrix on the data points. In cases in which distances are available[6] instead of similarities, one might apply a Gaussian kernel on these distances to convert the distance values δ_{ij} into similarity values s_{ij}.

$$s_{ij} = \exp(-\delta_{ij}^2 / t^2) \tag{9.20}$$

   Here, t is a parameter defining the kernel width. This results in an $(n-p+1) \times (n-p+1)$-dimensional similarity matrix S.

2. Extract the all strictly positive eigenvectors of this similarity matrix and standardize each of the eigenvectors to zero mean and unit variance. This provides a k-dimensional representation of each of the $(n - p + 1)$ objects. While extracting strictly positive eigenvectors, care must be taken to exclude zero eigenvectors that appear positive because of numerical errors in the eigenvector computation. A very small threshold such as 10^{-6} on the eigenvalues is usually sufficient to exclude such eigenvectors.

3. The squared distance of each point from the origin in this mean-centered representation is reported as the Mahalanobis outlier score.

For the case of multivariate series, the only difference is that the distance functions now need to computed using multivariate dynamic time-warping. Such distance functions are discussed in [33].

[5] A linear basis system no longer exists in the original space because a kernel transformation is used.

[6] Examples include dynamic time-warping and edit distance. Note that the specific choice of the distance function (e.g., DTW) depends on the application at hand. However, not all distance functions will result in a positive semi-definite kernel. In such a case, some of the tricks discussed in section 3.3.8.3 of Chapter 3 may need to be used.

9.3.4.4 Leveraging Kernel Methods with Linear Models

It is also possible to apply methods such as one-class support vector machines, as long as an appropriate kernel function can be defined to measure similarity between objects [379]. One challenge with the use of this approach is that kernels need to be positive semi-definite in order for a valid embedding to exist in the transformed space. Nevertheless, at a *heuristic* level, it is often possible to use similarity functions that do not satisfy the positive-semidefinite property to obtain reasonably good results (see section 3.3.8.3).

Kernel methods are often combined with linear models because linear models in the transformed space correspond to complex boundaries in the original space. The one-class support-vector machine (cf. section 3.4 of Chapter 3) is one example of such a linear model. Another example is the use of kernel PCA methods for arbitrary data types as discussed in section 3.3.8.3. Although such an approach has not been used widely in the literature, it remains a viable option because of the simplicity in implementation of kernel PCA methods.

9.3.5 Supervised Methods for Finding Unusual Time-Series Shapes

In many medical applications, characteristic pathological properties may be captured by unusual shapes of the underlying time series. In such cases, training data is often available about either the normal or the pathological behavior or both. Thus, such an approach requires the detection of unusual shapes in time series in a supervised manner. The labels may either be associated with the entire time series, with portions of the time series (subsequences). The simplest possible approach in such a scenario is develop subsequence profiles for both the normal class and the anomalous class. For a given test subsequence, a k-nearest neighbor approach may be used for classification purposes. Methods for querying and classifying data streams may be found in [171].

Feature extraction and transformation forms the core of all supervised methods for time-series classification. If the time series is represented in the form of discriminative features, it becomes much easier to classify it. One possibility is to transform the series to a discrete representation, and then use Hidden Markov Models (HMM) of Chapter 10 for the purposes of classification. A much larger number of models are available for sequence classification because of the natural ease in developing pattern-based classification models in the context of discrete data. A second possibility proposed in [408, 590], is to mine *shapelets* from the data, which are highly discriminative features for classification purposes. While many of these methods have been proposed for generic time-series classification, they can usually be generalized to the case where the classes are imbalanced. Detailed surveys on time-series classification may be found in [11, 575].

9.4 Multidimensional Streaming Outlier Detection

The previous sections discussed the case, where outliers were determined from time series based on either deviations from expected values or unusual shapes. Even when multiple correlated time series are available, each time series is processed as a unit for analysis. In traditional time-series analysis, *autocorrelation* is extremely important in the prediction and outlier computation process because of a strong assumption of temporal continuity.

On the other hand, in the case of multidimensional data, each record contains d-dimensions which form an indivisible unit. Furthermore, in the case of time series, a very high level of temporal continuity is observed in the individual series. This is not necessarily the case for multidimensional data streams, in which the temporal continuity is much

weaker. For example, in a stream of multidimensional text records, the individual frequency of an attribute in a text record cannot be reliably predicted from its immediately preceding records. On the other hand, the words present in the document can be compared at an *aggregate* level with the history of the stream in order to predict outliers. Thus, outlier analysis in multidimensional data steams is very different from outlier analysis in (possibly multivariate) time-series data because of the differences in the expected level of temporal continuity in these scenarios. The multidimensional streaming scenario is much closer to traditional methods for multidimensional outlier analysis. The only difference is the addition of a temporal component to the analysis, although this temporal component is much weaker than in the case of time-series data. In the context of multidimensional data streams, efficiency is a core concern, because the outliers need to be discovered quickly. There are two types of outliers that may arise in multidimensional data streams:

- One is based on outlier detection of individual records. For example, a first news story on a specific topic represents an outlier of this type. Such an outlier is also referred to as a *novelty* because it represents a point that was not seen before.
- The second is based on changes in the *aggregate trends* of the multidimensional data. For example, an unusual event such as a terrorist attack may lead to a burst of news stories on a specific topic. This essentially represents a higher level and aggregated outlier based on a specific time window. The second type of change point almost always begins with an individual outlier of the first type. However, an individual outlier of the first type may not always develop into an aggregate change point.

Both types of outliers will be discussed in this section. Furthermore, supervised methods for detecting rare and novel classes from multidimensional data streams will also be discussed.

9.4.1 Individual Data Points as Outliers

The problem of detecting individual data points as outliers is closely related to the problem of *unsupervised novelty detection*, especially when the entire history of the data stream is used. This problem is studied extensively in the text domain in the context of the problem of *first story detection* [622]. Such novelties are often trend-setters, and may eventually become a part of the normal data. However, when an individual record is declared an outlier in the context of a *window* of data points, this may not necessarily correspond to a novelty. In this context, proximity-based algorithms are particularly easy to generalize to the incremental scenario, by an almost direct applications of the algorithm to the window of data points. Numerous variations of proximity-based algorithms have been generalized to the temporal scenario in the literature.

9.4.1.1 Proximity-Based Algorithms

A distance-based method to detect such outliers was proposed in [60]. The original distance-based definition of outliers is modified in the following way:

> The outlier score of a data point is defined in terms of its k-nearest neighbor distance to data points in a time window of length W.

Note that this is a relatively straightforward modification of the original distance-based definition. The work in [60] proposes the STORM algorithm for distance-based outlier detection. When the entire window of data points can be maintained in main memory, it is

fairly easy to determine the outliers. On the other hand, in many interesting cases, it may not be possible to hold the entire window of data points in main memory. In such cases, the streaming scenario is more challenging because it is difficult to create efficient indexes for distance-based pruning. For this case, approximate outliers are returned because the exact determination of outliers is too expensive. Accuracy guarantees are provided for the outlier detection process. It is noteworthy that the averaged ensemble score from small subsamples of the previous window can often be more effective than an exact outlier score because of variance reduction effects. A detailed discussion of this effect is discussed in Chapter 6 on ensemble methods.

The LOF algorithm has also been extended to the incremental scenario [443]. Furthermore, the approach can handle both insertion and deletion of data points. Two steps are performed in the insertion process:

- The statistics of the newly inserted data point are computed with respect to the existing LOF model.
- The existing LOF model is updated in terms of densities, reachability distances, and the outlierness of the underlying data points. In other words, the parameters of many of the existing data points need to be updated, because they are affected by the addition of a new data point. However, not all points need to be updated, because only the locality of the new data point is affected. The work in [443] performs these updates in a judicious way, so as to efficiently determine the outliers.

Since distance-based methods are well known to be computationally expensive, many of the aforementioned methods are still quite expensive in the context of the data stream. Therefore, the complexity of the outlier detection process can be greatly improved by using a clustering-based approach.

Clustering-based methods are particularly effective when a large number of data points are available. In the context of a data stream, a sufficient number of data points are typically available in order to maintain the clusters at a very high level of granularity. *In the context of a streaming clustering algorithms, the formation of new clusters is often associated with unsupervised novelties.* Of course, this may not always be the case, when the entire history of the stream is not reflected in the limited-space summary provided by the current set of clusters. For example, the work in [28] explicitly regulates the creation of new clusters in the data stream, when an incoming data point does not lie within a specified statistical radius of the existing clusters in the data. Such data points may be considered outliers. In many cases, this is the beginning of a new trend, as more data points are added to the cluster at later stages of the algorithm. In some cases, such data points may correspond to novelties. In other cases, such points may correspond to trends that were seen a long time back, but are no longer reflected in the clusters. In either case, such data points are interesting outliers. However, it is not possible to distinguish between these different types of outliers, unless one is willing to allow the number of clusters in the stream to increase over time. The work in [322] leverages the clustering process discussed in [28] to improve the efficiency of the outlier analysis process.

This approach has also been generalized to the case of text data [29]. In these cases, it can be used in order to detect the first story [622] from a possibly new trend of more stories on the topic. Other distance-based algorithms for first story detection from text streams are discussed in section 8.6.3.1 of Chapter 8. Therefore, the reader is referred to Chapter 8 for a discussion on how such proximity-based methods for applied to first-story detection in text streams. Such methods can also be combined with window-based strategies in order to detect outliers by performing the clustering on specific chunks of the data stream [181].

9.4.1.2 Probabilistic Algorithms

The clustering approach discussed above can also be used in the context of probabilistic learning algorithms. As discussed in Chapter 2, probabilistic learning algorithms are generally not advisable in the context of limited data. However, in the context of data streams, a much larger amount of data is available for learning, and this is less of an issue, as long as the learning algorithm can be implemented *efficiently*. A method in [578] proposes methods for creating mixture models from mixed attribute data sets. The idea is to compute a fit of the incoming data point to the model both before and after the data point is added to the model. This is achieved with the use of an expectation-maximization algorithm.

Expectation-maximization algorithms are naturally suited to the streaming setting because of their iterative approach to the optimization process. As new data points arrive, it is relatively easy to include the adjust the parameters with these points. In addition, such methods have also been extended to first-story detection in text streams [608]. These approaches are discussed in detail in Chapter 8 of this book, and the reader is referred to that chapter.

9.4.1.3 High-Dimensional Scenario

The combination of the streaming scenario and high-dimensional data is particularly challenging because of the complexity of high dimensional projected clustering algorithms. Clearly, computational efficiency is a primary concern in the context of high-dimensional algorithms.

Many of the high-dimensional projected stream clustering algorithms can also be used in order to determine anomalies in the data stream. This is because many of these algorithms report outliers as side products of the clustering algorithm-[6, 139]. Data points which start new clusters in the stream can typically be reported as outliers.

A method called SPOT [604] is designed for detecting outliers in high-dimensional data streams. Unlike clustering methods, this method is designed to directly find sparse subspaces of the data. This approach uses a decaying cell based summary of the underlying data stream. This is then used in order to determine projected cell based summaries. The sparse subspaces in the underlying data stream. The data points which lie in the cells corresponding to the sparse subspaces are reported as outliers.

An approach that has rarely been explored, but holds significant promise is that of ensemble methods. For example, the variable subsampling methods and rotated bagging methods [32] are particularly designed for efficiency in large data sets and high-dimensional data sets, respectively. For a streaming data set, one can create multiple subsamples of variable size and combine with rotated bagging to provide outlier scores of data points in real time. The resulting data sets are often extremely compact, and it is possible to maintain a small history of multiple data sets. The scores across different components can be combined using any of the techniques discussed in section 6.6 of Chapter 6. Note that this approach is a meta-algorithm and it can be used in combination with virtually any base outlier detector. Furthermore, the ensemble approach is likely to provide it with numerous accuracy advantages.

9.4.2 Aggregate Change Points as Outliers

Numerous methods have been proposed in the literature in order to determine significant change points in a multidimensional data stream. Many change point detection techniques

have been studied independently in the database literature and the outlier detection literature. The reality is that these two areas are too closely related to be treated separately. The sudden changes in aggregate local and global trends in the underlying data are often indicative of anomalous events in the data. Many methods also provide statistical ways of quantifying the level of the changes in the underlying data stream. Therefore, we will discuss some of the significant methods for change detection, which can also be used for outlier detection. Some of these methods have also been used in the anomaly detection literature for finding significant changes in stock order data streams [372], or for finding anomalies in network data streams [334, 335, 377].

9.4.2.1 Velocity Density Estimation Method

The idea in velocity density estimation [19] is to construct a density-based velocity profile of the data. This is analogous to the concept of kernel density estimation in static data sets. In kernel-density estimation [496], we provide a continuous estimate of the density of the data at a given point. The value of the density at a given point is estimated as the sum of the smoothed values of kernel functions $K'_h(\cdot)$ associated with each point in the data set. Each kernel function is associated with a kernel width h which determines the level of smoothing created by the function. The kernel estimation $\hat{f}(\overline{X})$ based on N data points and kernel function $K'_h(\cdot)$ is defined as follows:

$$\hat{f}(\overline{X}) = \frac{1}{N} \cdot \sum_{i=1}^{N} K'_h(\overline{X} - \overline{X_i}) \tag{9.21}$$

Thus, each discrete point $\overline{X_i}$ in the data set is replaced by a continuous function $K'_h(\cdot)$ that peaks at $\overline{X_i}$ and has a variance determined by the smoothing parameter h. An example of such a distribution is the Gaussian kernel with width h.

$$K'_h(\overline{X} - \overline{X_i}) = \left(\frac{1}{\sqrt{2\pi} \cdot h}\right)^d \cdot \exp\left(-\frac{||\overline{X} - \overline{X_i}||^2}{2h^2}\right) \tag{9.22}$$

The estimation error is defined by the kernel width h which is chosen in a data driven manner. It has been shown [496] that for most smooth functions $K'_h(\cdot)$, when the number of data points goes to infinity, the estimator $\hat{f}(\overline{X})$ asymptotically converges to the true density function $f(\overline{X})$, provided that the width h is chosen appropriately.

In order to compute the velocity density, a temporal window h_t was used in order to perform the calculations. Intuitively, the temporal window h_t is associated with the time horizon over which the rate of change is measured. Thus, if h_t is chosen to be large, then the velocity density estimation technique provides long term trends, whereas if h_t is chosen to be small then the trends are relatively short term. This provides the user flexibility in analyzing the changes in the data over different kinds of time horizons. In addition, a spatial smoothing vector h_s is used, whose function is quite similar to the standard spatial smoothing vector which is used in kernel density estimation.

Let t be the current instant and S be the set of data points that have arrived in the time window $(t - h_t, t)$. We intend to estimate the rate of increase in density at spatial location X and time t by using two sets of estimates: the *forward time-slice density estimate* and the *reverse time-slice density estimate*. Intuitively, the forward time-slice estimate measures the density function for all spatial locations at a given time t based on the set of data points that have arrived in the *past* time window $(t - h_t, t)$. Similarly, the reverse time-slice

estimate measures the density function at a given time t based on the set of data points which will arrive in the *future* time window $(t, t + h_t)$. Let us assume that the ith data point in S is denoted by $(\overline{X_i}, t_i)$, where i varies from 1 to $|S|$. Then, the forward time-slice estimate $F_{(h_s,h_t)}(X, t)$ of the set S at the spatial location $\overline{X}$ and time t is given by:

$$F_{(h_s,h_t)}(\overline{X}, t) = C_f \cdot \sum_{i=1}^{|S|} K_{(h_s,h_t)}(\overline{X} - \overline{X_i}, t - t_i) \tag{9.23}$$

Here, $K_{(h_s,h_t)}(\cdot, \cdot)$ is a spatiotemporal kernel smoothing function, h_s is the spatial kernel vector, and h_t is temporal kernel width. The kernel function $K_{(h_s,h_t)}(\overline{X} - \overline{X_i}, t - t_i)$ is a smooth distribution which decreases with increasing value of $t - t_i$. The value of C_f is a suitably chosen normalization constant, so that the entire density over the spatial plane is one unit. Thus, C_f is defined as follows:

$$\int_{\text{All } \overline{X}} F_{(h_s,h_t)}(\overline{X}, t) \delta \overline{X} = 1 \tag{9.24}$$

The reverse time-slice density estimate is also calculated in a somewhat different way to the forward time-slice density estimate. We assume that the set of points that have arrived in the time interval $(t, t+h_t)$ is given by U. As before, the value of C_r is chosen as a normalization constant. Correspondingly, the value of the reverse time-slice density estimate $R_{(h_s,h_t)}(\overline{X}, t)$ is defined as follows:

$$R_{(h_s,h_t)}(\overline{X}, t) = C_r \cdot \sum_{i=1}^{|U|} K_{(h_s,h_t)}(\overline{X} - \overline{X_i}, t_i - t) \tag{9.25}$$

In this case, $t_i - t$ is being used in the argument instead of $t - t_i$. Thus, the reverse time-slice density in the interval $(t, t+h_t)$ would be exactly the same as the forward time-slice density, if time was reversed, and the data stream arrived in reverse order, starting at $t + h_t$ and ending at t.

The velocity density $V_{(h_s,h_t)}(\overline{X}, T)$ at spatial location $\overline{X}$ and time T is defined as follows:

$$V_{(h_s,h_t)}(\overline{X}, T) = \frac{F_{(h_s,h_t)}(X, T) - R_{(h_s,h_t)}(\overline{X}, T - h_t)}{h_t} \tag{9.26}$$

A positive value of the velocity density corresponds to a increase in the data density of a given point. A negative value of the velocity density corresponds to a reduction in the data density a given point. It has been shown [19] that the velocity density is directly proportional to a rate of density change at a given point with the following choice of the spatiotemporal kernel function:

$$K_{(h_s,h_t)}(X, t) = (1 - t/h_t) \cdot K'_{h_s}(X) \tag{9.27}$$

This kernel function is only defined for values of t in the range $(0, h_t)$. The Gaussian spatial kernel $K'_{h_s}(\cdot)$ was used because of its well-known effectiveness [496]. Specifically, $K'_{h_s}(\cdot)$ is the product of d identical Gaussian kernel functions, and $h_s = (h_s^1, \ldots h_s^d)$, where h_s^i is the smoothing parameter for dimension i.

The velocity density is associated with a data point as well a time-instant, and therefore this definition allows the labeling of both data points and time-instants as outliers. However, the interpretation of a data point as an outlier in the context of aggregate change analysis

is slightly different from the previous definitions in this section. An outlier is defined on an aggregate basis, rather than in a specific way for that point. Since outliers are data points in regions where abrupt change has occurred, *outliers are defined as data points $\overline{X}$ at time-instants t with unusually large absolute values of the* **local** *velocity density.* If desired, a normal distribution or Student's t-distribution could be used to determine the extreme values among the absolute velocity density values. Thus, the velocity density approach is able to convert the multidimensional data distributions into a quantification, which can be used in conjunction with extreme-value analysis.

It is important to note that the data point $\overline{X}$ is an outlier only in the context of *aggregate* changes occurring in its locality, rather than its own properties as an outlier. In the context of the news-story example, this corresponds to a news story belonging to a particular burst of related articles. Thus, such an approach could detect the sudden emergence of local clusters in the data, and report the corresponding data points in a timely fashion. Furthermore, it is also possible to compute the aggregate absolute level of change (over all regions) occurring in the underlying data stream, by computing the average *absolute* velocity density over the entire data space by summing the changes at sample points in the space [19]. Time instants with large values of the aggregate velocity density may be declared outliers.

9.4.2.2 Statistically Significant Changes in Aggregate Distributions

A different way to characterizing aggregate changes in multidimensional data streams would be to estimate the aggregate distributions in these time windows. Significant changes in these time windows can be reported as the unusual changes in the data stream. We note that the velocity-density method also estimates the aggregate distributions with the use of kernel-density estimation. However, in the context of change detection, it is also sometimes useful to be able to perform statistical tests directly on the underlying distributions in order to determine significant change points.

The work in [315] proposes a non-parametric framework for change detection in data streams. The key contribution in the work is a definition of the distance between two probability distributions. Generalizations of the Wilcoxon and Kolmogorov-Smirnoff tests are used in order to determine the significant change points. The work is, however, proposed for the case of 1-dimensional data streams, and the generalization to higher dimensions is not discussed.

The work in [159] is a bit more general, in that it can address multidimensional data streams effectively. Furthermore, it can be applied to different data types, because the computational aspects of the problem are different from the type of the underlying data. Since change detection is essentially relevant to the concept of finding distances between distributions, a very general way of representing this distance is the relative entropy from information theory. This is also known as the Kullback-Leibler (or KL) distance.

The broad principle is as follows. Given a set of data that should be fit to a family of distributions, the maximum likelihood estimator is the one that minimizes the KL-distance to the true distribution. This is a very general form of many other kinds of change analysis. For example, the t-test is equivalent to the KL-distance between two normal distributions. The KL-distance also allows for better intuitive interpretability, and is therefore particularly appealing for outlier detection. The definition of the KL-distance is also independent of the specific dimensionality or the type of the data representation itself. Therefore, this abstracts out the modeling of the change from the computational process. Details of this approach are presented in [159].

9.4.3 Rare and Novel Class Detection in Multidimensional Data Streams

An interesting case is when supervision is available in order to guide the temporal outlier-detection process. In such cases, class labels are attached to the individual data points. Many different types of supervised outliers may be of interest in such scenarios. The supervised scenario is a very rich one in the temporal domain, because *different kinds of temporal and frequency-based aspects of the classes could correspond to outliers*. These could correspond to novel class outliers, rare-class outliers, or infrequently recurring class outliers. Thus, a combination of methods for concept drift analysis, outlier detection, and rare-class detection may need to be used to determine interesting outliers in the streaming scenario. Such scenarios arise often in applications such as intrusion detection in which some known intrusions may be labeled, but new intrusions may also arise over time. Therefore, it is critical for the anomaly detection algorithm to use a combination of supervised and unsupervised methods to detect outliers.

The different types of outliers in such scenarios are as follows:

- **Rare-class outliers:** The identification of such outliers is similar to that in the static supervised scenario, except that it needs to be done more efficiently in the streaming setting. In such cases, a small fraction of the records may belong to a rare class, but they may not necessarily be distributed in a non-homogenous way from a temporal perspective. Although some amount of concept drift may also need to be accounted for, rare-class variations of stream classification algorithms may be used.

- **Novel-class outliers:** These classes were not encountered earlier in the data stream. Therefore, they may not be reflected in the training model at all. Eventually, such classes may become a normal part of the data over time. This scenario is somewhat similar to semi-supervised outlier detection in the static scenario, although the addition of the temporal component brings a number of challenges associated with it.

- **Infrequently recurring class outliers:** These are classes that have not been encountered for a while, but may reappear in the stream. Such classes are different from the first type of outliers, because they arrive in *temporally rare bursts*. Since most data stream classification algorithms use some form of discounting in order to address concept drift, they may sometimes completely age out information about old classes. Such classes cannot be distinguished from novel classes, if the infrequently recurring classes are not reflected in the training model. Therefore, issues of model update and discounting are important in the detection of such classes. This type of outlier was first proposed in [391].

We discuss each of the aforementioned cases below.

9.4.3.1 Detecting Rare Classes

Numerous classifiers are available for the streaming scenario [10], especially in the presence of concept drift. For detecting *rare classes*, the only change to be made to these classifiers is to add methods that are intended to handle the *class imbalance*. Such changes are not very different from those of addressing class imbalance in the static context. The reader is referred to Chapter 7 for a detailed description of these methods and also to the bibliography section of the current chapter for references to specific methods in the streaming context. Since the broad principles of detecting rare classes do not change very much between the

static and dynamic scenario, the discussion in this section will focus on the other two types of outliers.

9.4.3.2 Detecting Novel Classes

Detecting novel classes is also a form of *semi-supervision*, because models are available about many of the other classes, but not the class that is being detected. Furthermore, the problem is often encountered in an online setting. Important special cases include the first story detection setting [29, 608, 622] discussed in Chapter 8, and the streaming outlier-detection setting [28] in which no labels are available at all. In the latter case, the models are created in an unsupervised way with the use of clustering or other unsupervised methods. The work in [391, 392] addresses the problem of novel class detection in the streaming scenario with the use of labels on a subset of the classes.

Much of the traditional work on novel class detection [388] is focused only on finding novel classes that are different from the current models. However, this approach does not distinguish between the different novel classes that may be encountered over time. A more general way of understanding the novel class detection problem is to view it as a combination of supervised (classification) and unsupervised (clustering) models. Thus, as in unsupervised novel class detection models such as first story detection [29, 622], the degree of cohesion between the test instances of a novel class is important in determining whether they belong to the same novel class or not. The work in [391, 392] combines both supervised and semi-supervised models by the following methods:

- Maintaining a supervised model of the classes available in the training data as an ensemble of classification models.
- Maintaining an unsupervised model of the (unlabeled) novel classes received so far as cohesive groups of tightly knit clusters.

When a new test instance is received, the classification model is first applied to it to test whether it belongs to a currently existing (labeled) class. If this is not the case, it is tested whether it naturally belongs to one of the novel classes. The relationship of the test instance to a statistical boundary of the clusters representing the novel classes is used for this purpose. If neither of these conditions hold, it is assumed that the new data point should be in a novel class of its own. Thus, this approach combines supervised and unsupervised methods for novel class detection in a flexible way.

9.4.3.3 Detecting Infrequently Recurring Classes

Many outliers in real applications often arrive in *infrequent temporal bursts*. Many classifiers cannot distinguish between novel classes and rare classes, especially if the old models have aged out in the data stream. Therefore, one solution is to simply report a recurring outlier as a novel outlier.

This is, however, not a very desirable solution because novel-class detection is a semi-supervised problem, whereas the detection of recurring classes is a fully supervised problem. Therefore, by remembering the distribution of the recurring class over time, it is possible to improve the classification accuracy. The second issue is related to computational and resource efficiency. Since novel class detection is computationally and memory intensive, it is inefficient to treat recurring classes as novel classes. The work in [391] is able to identify the recurring classes in the data, and is also able to distinguish between the different types of recurring classes, when they are distinct from one another by examining their relationships

in the feature space. For example, two types of recurring intrusions in a network intrusion detection application may form distinct clusters in the stream. The work in [391] is able to distinguish between the different types of recurring classes as well.

9.5 Conclusions and Summary

This chapter studies the problem of outlier detection in streaming time-series data and multidimensional data streams. The nature of the outliers is very different in the two cases because time-series data requires the analysis of each series as a unit, whereas multidimensional data requires the analysis of each multidimensional point as a unit. Different types of outliers can also be defined in time-series data, depending on whether it is desirable to identify deviating points in the series, or whether it is desirable to identify unusual shape subsequences. For the case of multidimensional data, individual data points can be classified as novelties, or aggregate change points in the data may be defined as outliers. In cases where some of the labels are available, supervised and unsupervised methods can be combined for effective novel and rare class detection. Thus, the area of temporal outlier detection provides a wide variety of different problem definitions.

9.6 Bibliographic Survey

Outlier detection has been studied extensively in the context of traditional time-series data analysis [231, 232], especially from the perspective of removing noise from the underlying data [101, 129, 136, 202, 467]. Much of this work (such as Kalman Filtering [101]) has focused on the removal of noise from and smoothing of temporal time series in order to facilitate more accurate regression and prediction. Outliers are defined as deviations from predicted values in such cases. Detailed surveys on outlier detection in temporal data may be found in [231, 232].

Significant changes in the time-series trends can be detected as changes and outliers in the data [579]. This includes many traditional methods for regression modeling such as Autoregressive Modeling (AR), Autoregressive Moving Average (ARMA), and Autoregressive Integrated Moving Average (ARIMA). Methods such as principal component analysis [296] have also commonly been used in order to track correlations among multiple streams. The robustness of the prediction process is helpful for accurate outlier detection. Many methods have also been designed to speed up the regression modeling in the context of large number of data streams and real-time data [27, 278, 292, 427, 592]. An information-theoretic method for determining anomalous points in a time series was proposed in [284], where an outlier is defined as a point in a time series, whose removal results in a better histogram representation in the same storage. The storage in the two cases is compared after explicitly accounting for the space required by the outlier point. This is essentially an information-theoretic approach. Supervised methods can be used to perform discriminative analysis between different types of deviations and outliers in the time-series setting [9].

Whereas time-series outlier detection is often understood in terms of real-time change detection and deviations, the problem of finding unusual shapes in time series is an entirely different scenario. Whereas the former is related to extreme value analysis, the determination of the latter is more subtle, since it requires a careful analysis of the regularities in the series over different windows of the data. One of the earliest methods for anomaly detection in time series using ideas from immunology was proposed in [157]. In such cases, the overall

magnitude of the deviations matters less than the shape of the overall time series. The discovery of anomalies in dynamic product ratings is discussed in [228].

A variety of methods can be used in order to compute unusual shapes. The use of the Haar transform for anomaly detection in time series has been explored in [205]. The most common method [311] is the use of the Euclidean distance on the fixed windows of the time series. It has been shown in [311, 360] that this problem can be further sped up with the use of symbolic aggregate approximation. These methods have been scaled up to be disk-aware and work with terabyte-scale data sets in [587]. Methods for anomaly detection in multivariate time series are discussed in [73, 140]. Another interesting approach proposed in [76] was to determine anomalous *regimes* in time-series data, where the correlations and dependencies among the different time series have changed over time. Discrete transformations can also be used on a stand-alone basis [310] in order to perform the anomaly detection directly on the discrete representation, although this is rarely done because of the information loss associated with discretization.

Methods for finding anomalies in discrete sequences are discussed in Chapter 10. A detailed survey for the discrete case may be found in [126] and integrated perspectives in the context of both discrete and continuous data may be found in [231, 232]. Some semi-supervised and supervised methods have also been proposed for unusual shape detection in the literature. In the semi-supervised scenario, it is assumed that examples of normal time series are available. In the fully supervised scenario, examples of both normal and anomalous shapes are available [171, 289, 408, 574, 590]. The transformation to the discrete case also enables the use of supervised string-based models such as hidden Markov models (HMM) [126]. Other transformation methods include feature transformation methods, which construct relevant *shapelets* on the underlying time series [408, 590]. These are patterns that can discriminate between different classes of instances. Another well-known method is the use of distance function such as dynamic time-warping [289]. The design of effective distance functions is enables the development of proximity-based classification techniques. While most of the aforementioned techniques have been developed in the literature for the *generic* version of the classification problem, most of these methods can be easily adapted to the rare class scenario by using methods discussed in Chapter 7.

Multivariate temporal data can also be represented as trajectories. Methods for trajectory outlier detection are discussed in the next chapter. Significant *changes* in trajectory directions are useful for many applications, such as hurricane tracking [117]. In such cases, the trajectory can be treated as bivariate temporal data, and prediction-based deviation detection techniques can be applied to this representation. The works in [102, 215] determine anomalies in moving object streams in real time by examining patterns of evolution. On the other hand, the detection of anomalous trajectory *shapes* is a very different problem. The earliest methods for trajectory shape outlier detection were proposed in [319]. However, this method transforms the trajectories into point data by using a set of features describing meta-information about the trajectories. Unsupervised methods for trajectory outlier detection, which actually use the sequence information explicitly, were first investigated in [409, 347]. The work in [409] uses the Fourier transform in order to represent the trajectories in terms of the leading coefficients and find anomalies. In the second method [347], trajectories are divided into different line segments and anomalous patterns are identified in order to determine outliers. Supervised methods for anomaly detection in trajectory data may be found in [355]. These methods transform the data into discrete sequences, and a classifier is learned in order to relate the trajectories to the class labels.

Outlier detection has also been studied extensively in the context of sensor data [2, 108, 94, 204, 517, 614]. Sensor streams are one of the most common applications of anomaly

detection in temporal data [614]. Sensor data is also noisy because of errors and deviations in the data collection and transmission process. Therefore, the outlier detection problem has dual applicability to this scenario, both in terms of removing the underlying noise, and in terms of detecting unusual events from the sensor stream. The same techniques are typically used for both cases. In this context, supervision [9] can sometimes be useful in distinguishing between noise and anomalies. A number of unique technological issues arise in the context of outlier detection in sensor data, because of the large amounts of data involved. Consequently, *real-time* and *in-network* processing is important in order to minimize delays and communication costs. A detailed survey on outlier detection in sensor networks may be found in [614].

The problem of outlier detection in multidimensional data streams has also been studied extensively in the literature [28, 29, 476, 60, 63, 414, 443, 532, 608] in the context of different types of multidimensional and text data. The works in [532, 227, 571] are particularly notable for their extensions of the ideas in isolation forests to the case of streaming data. Recently, a subspace histogram technique, referred to as *RS-Hash*, has been proposed in [476] (cf. section 5.2.5 of Chapter 5). The extension to the streaming setting is referred to as *RS-Stream*. This technique averages the log-density in grid regions of varying sizes and shapes in order to provide the final score. The approach uses randomized hashing for efficiency and requires linear time. This approach is suitable for subspace outlier detection in high-dimensional data and therefore its efficiency is notable. Efficiency is particularly important in these cases because of the stream scenario. The problem of novelty detection is also closely related to clustering in the stream scenario, since the formation of new clusters corresponds to novelties in the data. Many of the aforementioned techniques use either clustering or distance-based methods to detect novelties. For example, the work in [322] uses stream-clustering algorithms [28] in order to improve the efficiency of the outlier analysis process. A method in [414] uses data editing techniques that progressively remove data points with the smallest nearest neighbor distance. This is referred to as *numerosity reduction*, which reduces the size of the data while retaining certain desirable properties for outlier detection. Recently, streaming methods for outlier detection have also been extended to the case of probabilistic data [560].

The determination of aggregate changes in data streams is another type of multidimensional outlier detection [193]. This requires the determination of whether the *aggregate distribution* of the multidimensional data has changed enough to be considered significant. Methods for characterizing the change in the form of velocity density contours may be found in [19]. These methods can also be generalized to the high-dimensional case. Other methods which uses different forms of statistical testing for change detection such as the KL-distance, Wilcoxon, and the Kolmogorov-Smirnoff test, may be found in [159, 315]. A number of other statistical methods [333, 504] use log-likelihood criteria in order to quantify the change. A martingale framework for change detection in data streams is proposed in [267]. Such change points can be very useful in determining anomalies in stock market order distributions [372] or temporal network traffic data streams [334, 335, 377]. The latter class of methods can sometimes also be used to diagnose network intrusions. Methods for aggregate change point detection in the form of correlated bursty topic patterns in coordinated text streams are proposed in [562].

A detailed survey of traditional methods for semi-supervised novelty detection may be found in [388, 389]. Methods for using support vector regression models for online novelty detection are discussed in [379]. The effectiveness of novelty detection can be enhanced by using class labels. A combination of streaming classification and clustering models are proposed in [46, 391, 392] for distinguishing between normal classes, novel classes, rare

classes, and rarely recurring classes.

9.7 Exercises

1. Develop a closed-form solution to the autoregressive model (AR) in this chapter using the relationship discussed in Chapter 3. What is the size of the matrix which needs to be inverted in order to solve this model?

2. Develop a solution to the ARMA model for deviation detection. How does the complexity compare to the simple AR model?

3. Apply each of the algorithms in Exercises 1 and 2 to detect point (time-stamp) outliers in the *Ozone Level Detection* data set of the UCI Machine Learning Repository [203], for each attribute (series) separately. How do the outliers compare for different attributes in the two cases?

4. Apply the unusual shape detection algorithm discussed in this chapter in order to detect unusual shape subsequences from the *Ozone level detection* data set of the UCI Machine Learning Repository [203]. Do the unusual shapes occur at time stamps that are in any way related to the time-stamps of unusual point deviations found in Exercises 2 and 3?

5. Apply a multivariate PCA technique across the different attributes of the *Ozone Level Data Set*, where each time-stamp is treated as a multi-attribute data point corresponding to the values over different series. Note that the time-stamps are treated independently of one another and temporal continuity is not used. Determine points of significant deviations from the regression model. Which time-stamps are the outliers?

6. Repeat Exercise 5 using windows of length p in order to generate each data point. Thus, a data point of dimensionality $p * d$ is generated at each time-stamp for multivariate series with d attributes.

7. Use the methodology of Exercise 5, in order to create a multidimensional data set from the time-series data set. Determine outliers with:

   - A k-nearest neighbor algorithm over the entire data set.
   - A k-nearest neighbor algorithm over the segment of the data set containing only earlier time stamps.

   How do the outliers found relate to each other, and to those found in Exercise 5?

8. Repeat Exercise 7 using windows of length p according to the approach in Exercise 6 rather than Exercise 5.