Statistical Anomaly Detection for Train Fleets

Anders Holst, Markus Bohlin, Jan Ekman, Ola Sellin, Björn Lindström, Stefan Larsen

■ *We have developed a method for statistical anomaly detection that has been deployed in a tool for condition monitoring of train fleets. The tool is currently used by several railway operators across the world to inspect and visualize the occurrence of "event messages" generated on the trains. The anomaly detection component helps the operators quickly to find significant deviations from normal behavior and to detect early indications for possible problems. The method used is based on Bayesian principal anomaly, which is a framework for parametric anomaly detection using Bayesian statistics. The savings in maintenance costs of using the tool comes mainly from avoiding costly breakdowns and have been estimated to be several million Euros per year for the tool. In the long run, it is expected that maintenance costs can be reduced by between 5 and 10 percent with the help of the tool.*

Anomaly detection is a growing area with more and more practical applications every day. It has been used for fraud detection and intrusion detection for a long time, but in later years the usage has exploded to all kind of domains, like surveillance, industrial system monitoring, epidemiology, and so on. For an overview of different anomaly-detection methods and applications, see, for example, Chandola, Banerjee, and Kumar (2009).

The approach taken in statistical anomaly detection is to use data from (predominantly normal) previous situations to build a statistical model of what is normal. New situations are compared against that model and are considered anomalous if they are too improbable to occur in that model. Various statistical anomaly-detection methods have previously been applied to a wide variety of problems including intrusion detection (García-Teodoro et al. 2009; Cemerlic, Yang, and Kizza 2008; Chebrolu, Abraham, and Thomas 2005; Puttini, Marrakchi, and Mé 2003), fault detection and diagnosis (Lerner et al. 2000), spam filtering (Su and Xu 2009), environmental anomaly detection (Hill, Minsker, and Amir 2009), and energy expenditure estimation (Shahabdeen, Baxi, and Nachman 2010).

The Swedish Institute of Computer Science (SICS) has for several years developed methods for statistical anomaly detection based on a framework called Bayesian principal anomaly (Holst and Ekman 2011). The framework has already been successfully used and evaluated in several real application domains, such as alarm filtering in telecommunication networks, intrusion detection (Dey 2009), alarm call monitoring for crisis management, and maritime domain awareness (Bjurling et al. 2010). In this article we describe a novel application domain for the anomaly-detection method: condition monitoring of trains (Holst, Ekman, and Larsen 2006).

The method presented here is based on parametric statistical models. There are currently many popular anomaly-detection methods based on nonparametric models (see, for example, Ahmed, Oreshkin, and Coates [2007]; Sotiris, Tse, and Pecht [2010]; Tian, Yu, and Yin [2004]). A nonparametric model is very general since the parametric forms of the distributions need not be known. However, when a parametric form is known, the correct parametric model will require magnitudes less data before being useful and has a much higher detection precision. Especially in the case studied here, where the data are event based and the number of events in an interval is approximately Poisson distributed, a parametric model is definitely best suited. Furthermore, a very large class of data in the real world comes as events or event counts, which makes the presented method very widely applicable.

Addtrack is a tool for condition monitoring of trains developed in collaboration between Bombardier Transportation AB and Addiva AB. Addtrack is currently used by Bombardier and several railway operators to inspect and visualize the occurrence of "event messages" generated on train fleets. The data sets analyzed often consist of several thousand data points, and a common complaint from analysts was that it was, as a consequence, difficult to observe relevant patterns in the fleet of trains. Of particular interest was the ability to point out more directly the most anomalous cases hidden among all the normal ones, so that further investigative actions could be taken.

The anomaly-detection methods developed at SICS have been deployed as a component of the Addtrack system and are in use helping analysts to quickly find significant deviations from normal behavior in several train fleets and to detect early indications of possible problems.

The remainder of this article is organized as follows. First, the Addtrack tool is described together with the original requirements for an anomaly-detection module and an outline of how the application is used. Next, the principal anomaly is presented as the basis for the module, followed by a description of the Bayesian approach used to estimate the normality model and the particular application to event data anomaly detection. The anomaly-detection module is then described, and the development and deployment process follows. The article ends with a brief evaluation and a discussion of the results and future development.

The Addtrack Tool

Addtrack is a tool developed originally by Bombardier Transportation for general analysis, monitoring, and visualization of train conditions and event data, which are periodically or continuously uploaded from the train units. It is "intelligent" in the sense that analysis modules, such as the one described in this article, can be used to preprocess and visualize data sets. Today, Addtrack is developed by the independent company Addiva Consulting AB. Addtrack, including the anomaly-detection module described in this article, is currently deployed in Sweden, India, China, and Germany and has approximately 300 active users in total. Out of these, most use the anomaly-detection functionality. The main purpose of Addtrack is both to let the maintenance personnel get an overview of the condition of the fleet to catch problems early and also to find out the circumstances and causes when a failure has occurred.

The primary user of Addtrack is Bombardier Transportation, one of the world's largest rail-equipment manufacturing and servicing companies with more than 100,000 installed rail cars and locomotives worldwide. Here, the main use of Addtrack is to detect faulty units during the development of new railway vehicles and the subsequent commissioning phase. Following this, Addtrack is used during the warranty period to perform a simple form of root-cause analysis. Finally, the tool is used during the maintenance phase for acute fault localization. To a lesser extent, it is also used for predictive maintenance, but this mode of use is growing since it allows the operators to catch faults early, thereby avoiding much more severe stopping failures. Today, it is estimated that customers using Addtrack prevent on average one or two stopping faults annually for each train unit. Each stopping fault might cost on the order of several ten thousand Euros.

A key functionality of Addtrack is to visualize the number of events of different types that have occurred on a set of trains during some period. Figure 1 contains such a view, where the diagram shows the number of events per event code (on the *x*-axis) and train (on the *z*-axis) that have occurred during the last month. Only event codes that have occurred at all during the month are shown. In the menu to the left it is possible to select which set of trains to show. It is also possible to click a bar in the diagram to see a time series of the days on which those events occurred (not shown here).

As can be seen, there are rather many event codes with a high number of events for most trains, indicating that this is the normal state.

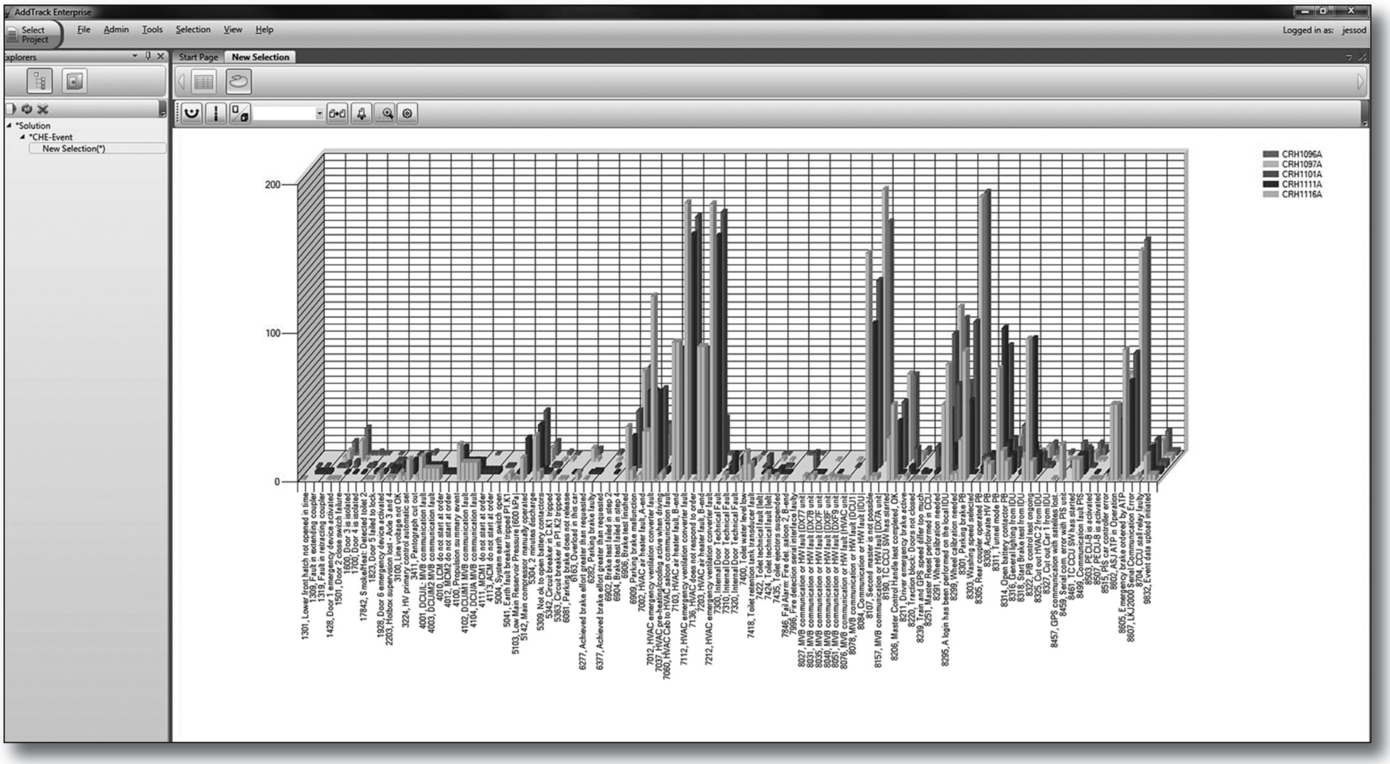


Figure 1. Addtrack Visualization of Selected Event Occurrences for Five Units in the Chinese Fleet.

There is also a high variability between trains in the number of events of some event codes, but it is not obvious from the diagram when a bar is (statistically) significantly higher than the others. Before the integration of the anomaly-detection module, analysts were analyzing the raw fault rates exclusively by choosing a period and a subset of trains and codes for events or condition data to visualize. It is however time-consuming and tedious work for an analyst to go through all high bars to see whether they indicate something out of the ordinary. There may even be quite low bars that nevertheless are significantly higher than for other trains but may go unnoticed among all the normally high bars. Indeed, sometimes it may even be remarkable with a too low number of events, which is almost impossible to spot in the diagram.

As an example of the size of the data set, for the Swedish fleet of Regina trains operated by SJ, which consists of 57 different train sets, there are more than 1000 different event and condition types. The data is typically collected weekly, and for each such interval the analysts therefore have a set of more than 57,000 data points to watch in total. On the more modern train sets, more than 12,000 data points are collected per train set. The main complaint of analysts was therefore that it was very difficult to observe patterns in the huge amounts of data that the analysts have to handle. A particular problem was the analysis of rare events, for which obvious patterns were hard to detect even for experienced analysts.

In summary, there is a need for a tool to indicate which bars are significantly different from expected and should thus be focused on. The following two sections describe the anomaly-detection module designed for this purpose.

Principal Anomaly Detection

The general idea in statistical anomaly detection is to build a statistical model over normal cases and then compare new samples with this model when they arrive. Samples that would have a too small probability of being generated by the statistical model are considered anomalous, that is, they are very unlikely to belong to the set of normal cases.

To be useful in practice, an anomaly detector must fulfill several conditions: (1) it must be able to handle a large number of input features; (2) it must be able to handle several different normal situations; (3) it must allow for training data to include realistic amounts of anomalous cases; ; (4) it must be fast when dealing with large amounts of data; (5) it must be robust in the light of very small amounts of training data; and (6) it must have a sufficiently small false alarm rate, while still being maximally sensitive to real anomalies.

The Bayesian principal anomaly framework was

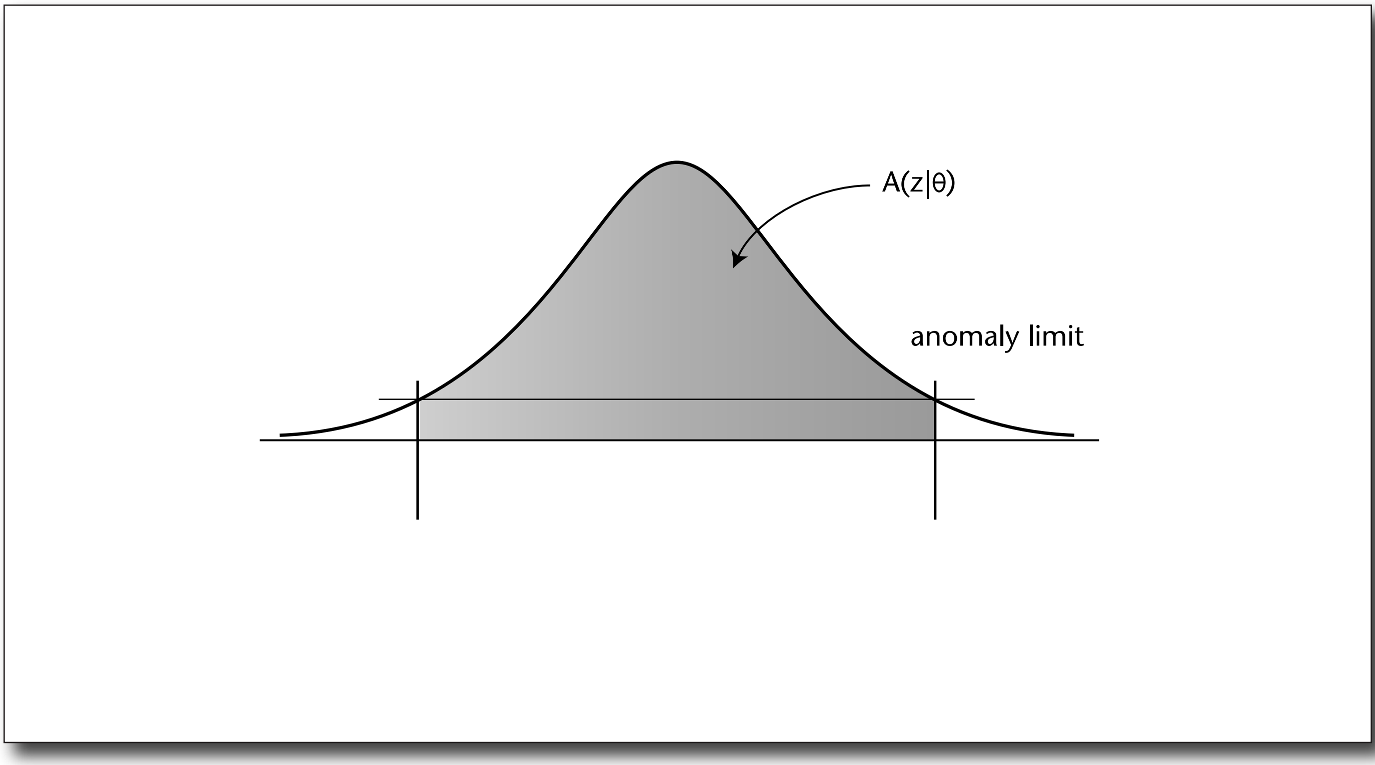


Figure 2. Illustration of the Principal Anomaly A(z|θ).

designed to fulfill these requirements. Robustness is achieved by using Bayesian statistics when estimating the parameters of the normality model. A classical maximum likelihood point estimate of the parameters can make the estimated distribution sensitive to random fluctuations in the data, especially when there are quite few data samples, which tends to result in too many false alarms. Using a Bayesian approach, we can remedy this by taking appropriate consideration to the uncertainties in the actually observed data, and thereby avoiding some of the false alarms.

Let us initially assume that the normal situation samples are generated by a known probability density $P(x|\theta)$ for a set of parameters θ. To define what is meant by an "anomalous" observation, we note that, intuitively, the smaller the probability of generating a new observation *z* from the distribution, the more anomalous is it. However, just looking at the probability $P(z|\theta)$ of generating the observation *z* from the distribution will not work, since the magnitude of $P(z|\theta)$ depends on the variance of the distribution (for continuous distributions). Therefore, it is not possible to specify a fixed probability threshold that is the same for all distributions, and below which a sample should be considered anomalous. Instead, we should look at the same entity as in hypothesis testing, that is, the probability of generating an observation at least as unusual as *z*. This also gives a natural way of controlling the rate of false alarms, which may otherwise be high in many anomaly-detection applications.

Thus, let us then define the principal anomaly of a new observation *z* as the probability of generating a more common sample than *z* from the distribution:

$$A(z \mid \theta) = \int_{x\in\Omega} P(x \mid \theta)$$
$$\text{where } \Omega = \{x : P(x \mid \theta) > P(z \mid \theta)\} \qquad (1)$$

This measure is illustrated in figure 2 and has a number of desirable properties. First, $A(z|\theta)$ increases when the sample *z* gets more unusual, which is intuitive. Second, it is directly comparable to the principal anomalies of other features, either with other parameters or completely different distributions. Third, it is directly connected to the rate of false alarms. If we set a threshold on the principal anomaly of $1 - \varepsilon$ over which an observation is judged anomalous, the probability that a normal sample is wrongly detected as anomalous is then simply ε.

In the context of statistical anomaly detection, we claim that all sound scores of anomaly should be related to this principal anomaly, since it defines how unusual a sample is. Conversely, any score that is a monotonic function of the principal anomaly will rank the samples identically with respect to anomaly.

However, although suitable in the formal sense, the principal anomaly itself is not as suitable for manual work and inspection, since its anomaly values are most of the time very close to 1. Its complement may therefore be more useful:

$$\bar{A}(z \mid \theta) = 1 - A(z \mid \theta) = \int_{x \in \bar{\Omega}} P(x \mid \theta)$$
$$\text{where } \bar{\Omega} = \{x : P(x \mid \theta) \leq P(z \mid \theta)\} \quad (2)$$

This is the probability of getting an equally or less probable sample than *z* from the distribution, that is, the probability of the tails beyond *z* (and beyond other samples with the same probability as *z*), and therefore equal to ε. For anomalous samples it will be very close to 0, and therefore easier to compute with high precision than the principal anomaly itself. Also useful, and more intuitive to work with, is the negative logarithm of the complementary principal anomaly:

$$\Lambda(z \mid \theta) = -\log(\bar{A}(z \mid \theta)) \quad (3)$$

It ranges from 0 to ∞, and increases with higher anomaly, such that each constant step higher represents a factor lower probability. But as mentioned above, exactly which transform is used for presentation purposes is irrelevant, as long as it is strictly monotonically increasing in the principal anomaly *A*.

Bayesian Estimation

Above it was assumed that the parameters θ of the statistical model are known. In practice we usually do not know the parameter values of the model in advance, but all we have is a number of observations that are assumed to be generated by the model.

Let us therefore now consider the case when the parameters θ are unknown, and we instead have to estimate the distribution from a number of samples. To be useful, the anomaly detector should be able to provide an anomaly score already after a very small number of training samples. With maximum likelihood estimation of the parameters, the estimates will be far too sensitive to chance occurrences, and the resulting detector will tend strongly to underestimate the probability of new samples, resulting in too many false alarms. With a Bayesian approach, all parameter values that may have given rise to the observed samples are considered (appropriately weighted). The result is a detector that is much more robust to random fluctuations in the training data and doesn't indicate an anomaly unless it is sufficiently certain. The side effect is that early on a Bayesian anomaly detector will accept more samples as normal. However, as more training data are collected, the parameter estimation will become more accurate, which in turn will make the anomaly detector more precise.

More formally, the Bayesian approach is thus to find the posterior distribution over the parameters θ given the set of training samples *X:*

$$P(\theta \mid X) \propto P(X \mid \theta)P(\theta)$$
$$= \prod_i P(x_i \mid \theta)P(\theta) \quad (4)$$

Here, P(q) is the prior distribution over the parameters. In this article we use a standard noninformative prior, which has proven adequate in practice. We can now obtain the principal anomaly by integration over all possible parameter values:

$$A(z \mid X) = \int_{\theta} A(z \mid \theta)P(\theta \mid X) \quad (5)$$

We define this as the Bayesian principal anomaly.

Anomalies in Event Data

In the train condition monitoring application, the data used to check for anomalies are the rates of different event messages generated on the trains. As on most technical systems, there is a large amount of log messages generated when things happen, representing events ranging from harmless to serious. The serious events are straightforward to handle, since they typically require service more or less immediately. More interesting from an anomaly-detection perspective are changes in the rates of less serious or seemingly harmless events. Such often subtle changes in the rates nevertheless indicate that something has changed on the train, which may merit action in the form of an extra inspection of the corresponding subsystem.

To model normal behavior, we assume that when the train is in a normal state, each event type has a certain normal rate with which it occurs. Under this assumption, we can model the number of events in an interval of length *T* with a Poisson distribution:

$$P(x \mid \lambda T) = (\lambda T)^x e^{-\lambda T} / x! \quad (6)$$

where λ is the rate of events per time unit. For each event type x_i, we get the Bayesian principal anomaly from equation (5) by inserting the above expression with = λ*T* in equation 4. The resulting expression requires numerical evaluation of an indefinite sum, but this is not more complicated than what can be computed rather fast on a normal computer.

It is now possible to test each train against the others by counting the number of events of each type for each train during a certain time period of interest and computing the Bayesian principal anomaly for the counts of each train, basing the normality model on the other counts. This will find trains that behave differently from the others with respect to some event type. Alternatively it is possible to find a train that has changed behavior recently by testing its counts from a recent time period against those from a longer historical time period.

Practical Considerations

If the model assumptions are correct for a parametric anomaly detector, the detector will be very sensitive and able to detect even small deviations from normal behavior. However, if the model assumptions are not correct, the precision will be much worse. There are a number of ways in which the real-world data may violate the Poisson assumption that is used here and that have to be considered when using the anomaly detector in a practical application.

First, it is not always obvious in what unit to measure the length of the period during which events are counted. It should be measured in effective time rather than real time. In the case of trains it is easy to see that those that are utilized more are likely to have more alarms, but whether effective time is the time the train is in operation or instead the number of traveled kilometers is not as clear. The unit that is closest to being proportional to the number of events should be used. (It may even be that this differs between events, that is, some event types are proportional to kilometers and others to operation time.) In this application though it turns out that the operation time works best to use as effective time.

Then, events may not occur independently in time. For example, events of some types may occur in bursts or be repeated until handled, so that a single external event gives rise to a sequence of generated events. This multiplication of events will inflate the significance of a random fluctuation and may result in a false anomaly indication. Therefore such runs of events must be removed (and counted as just one event), for example by using a small latency time after each event before an event of the same type is counted again on the same train.

Further, all trains, or cars in a train, may not be exactly the same. For example, different cars in the train may be differently equipped, and thus produce different types of events, or just different frequencies of the same event types. Different statistical models should therefore be used for different categories of trains or cars.

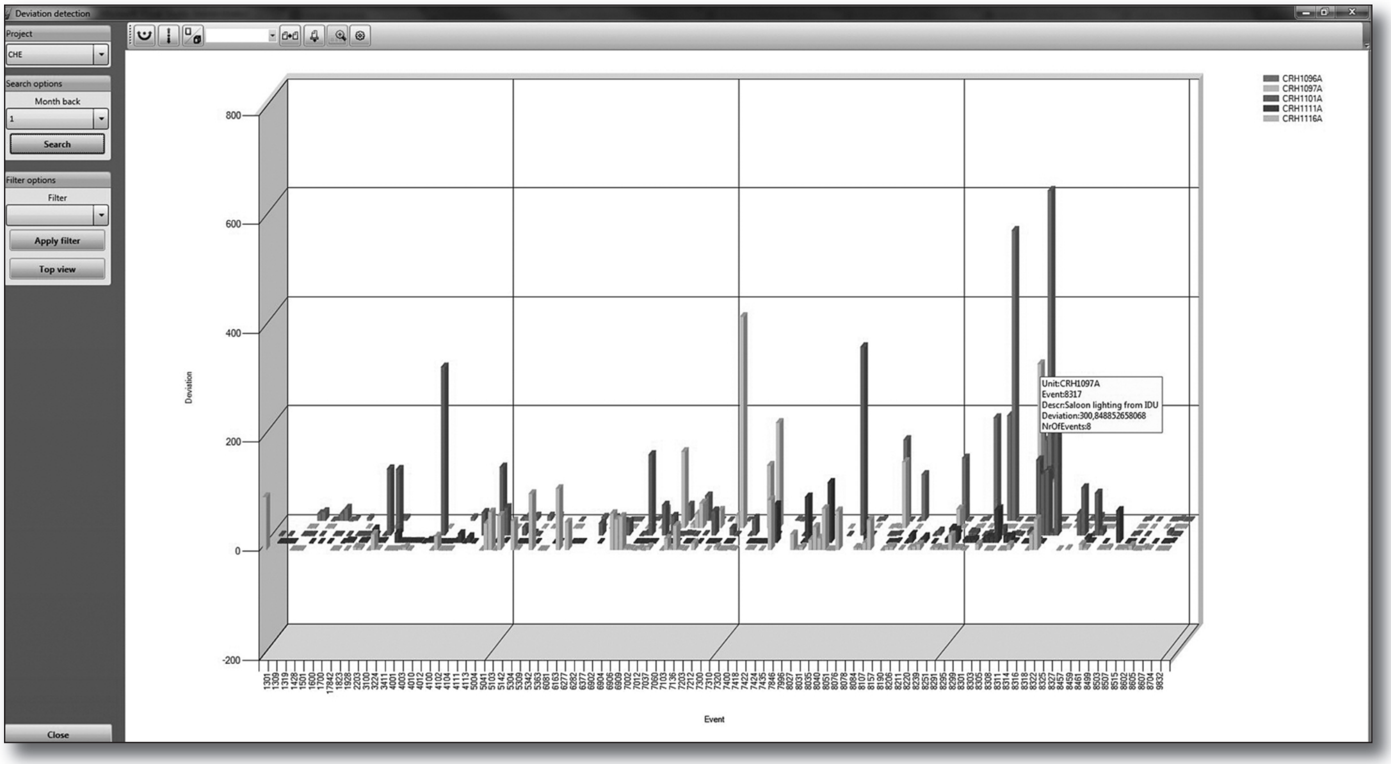
Finally, real-world data are never clean. When estimating the Poisson distribution, there will be anomalous cases in the historical data. If they are few and with varied parameters they will have only marginal effect on the estimation, but if they constitute a substantial part of the data or have extremely different parameter values than the rest, they may disturb the estimation such that otherwise normal cases are considered anomalous. The way to solve this is to use an iterative training method, where after each training pass each training sample is tested for anomaly, and the most anomalous samples are removed from training in the next pass. This is continued until no anomalous samples remain in the training set. In this way the anomaly detector filters its own training data. (This will work of course only as long as the normal samples are in a majority, but this is the very definition of "normal" in all of anomaly detection.)

The Anomaly-Detection Module

Figure 3 shows the anomaly-detection view in Addtrack, which is powered by the anomaly-detection module described in this article. Here, the bars indicate how much each event code deviates from normal for each train during the last month, as expressed by Λ. The high bars are much more sparse in this view, and thus easier to go through, compared to those in figure 1. Also, there are indeed some high bars that correspond to rather low bars in the count diagram, and that consequentially might have been missed without the anomaly detection. Using the module, analysis can therefore be made much more efficiently by allowing the analyst to focus on a particular subset of trains, event codes, and time periods.

Savings from using the anomaly-detection module in Addtrack comes mainly from being able to faster diagnose and correct faults on train units in all phases during their life cycle, which has a number of benefits including a higher ratio of trains delivered on time. The annual savings from using Addtrack are in the order of ten million Euros. Out of these, the monetary savings from using the anomaly-detection part are substantial but harder to quantify. Bombardier estimates however that in the long run, maintenance costs can be reduced by between 5 and 10 percent by using Addtrack with anomaly detection.

To simplify the application of anomaly detection for different kinds of log and alarm data in different domains, the anomaly-detection algorithm for event data is placed in a separate program module with a very simple API designed to be independent of the specific domain. Since Addtrack runs on the Windows platform, the module was wrapped in a DLL compiled from the C source code, which was then integrated in Addtrack. The API has only four functions: CreateAnomalyDetector, which creates a new anomaly detector and trains it with a set of provided event counts and intervals; ApplyAnomalyDetector, which tests it on some also provided event counts and intervals; SetAnomalyThreshold to set a threshold used when filtering out anomalous samples from the training data; and DeleteAnomalyDetector to call when the detector is no longer needed.



Unit:CRH1097A
Event:8317
Descr:Saloon lighting from IDU
Deviation:300,848852658068
NrOfEvents:8

Figure 3: Addtrack Deviation Detection Mode for Selected Event Occurrences for Five Units in the Chinese Fleet.

Same units as in figure 1.

This simple design with a minimum of free parameters makes it easy to provide anomaly-detection capacity to other programs. To use the anomaly detector, the host program will have to take care of any user interaction to, for example, select a time period of interest and a set of entities to check for anomalies (like trains in the case of this article); look up from appropriate databases a list of event counts for the different entities during the selected interval; create and apply an anomaly detector to get anomaly values for the entities; and visualize the results in a suitable way to the user. If the surrounding program is a commercial product aimed at analysis of log or event data, it is likely to have functionality for user interaction, database access, and result visualization already, and it will therefore be relatively simple to add anomaly-detection functionality as well.

Development and Deployment Process

Addtrack has been continuously developed for approximately 10 years by a small team of developers, first at Bombardier Transportation, and then at Addiva. The need for anomaly detection was first uncovered in a collaborative research project that was initiated in 2003. At this time, Addtrack was used internally at Bombardier under the name of Edgar (the graphical user interface part) and T-Rex (the database part). Early experiments with much simpler anomaly detectors resulted in a high number of false alarms, which is why a Bayesian approach was chosen. The anomaly-detection techniques had also been previously developed at SICS since 2001 during several basic and applied research projects. Isolation of the anomaly-detection functionality into its own module started in 2007, and it has since then required only a few person months of work, spread out over time. The addition of the anomaly-detection module to Addtrack was, in comparison to the development effort of the main program, quite small, and took in the order of 3 to 4 person weeks of interface design for SICS and approximately the same amount of integration work for Addiva. The rollout of the module was done through an automatic update of Addtrack over the Internet, and distribution and deployment were therefore relatively easy.

The anomaly-detection module was officially released by Addiva at the Addtrack user group conference in 2009, which was used as a marketing channel. From there, the users have spread the information further within their respective companies. Manuals and other supporting material have been developed by Addiva, and maintenance

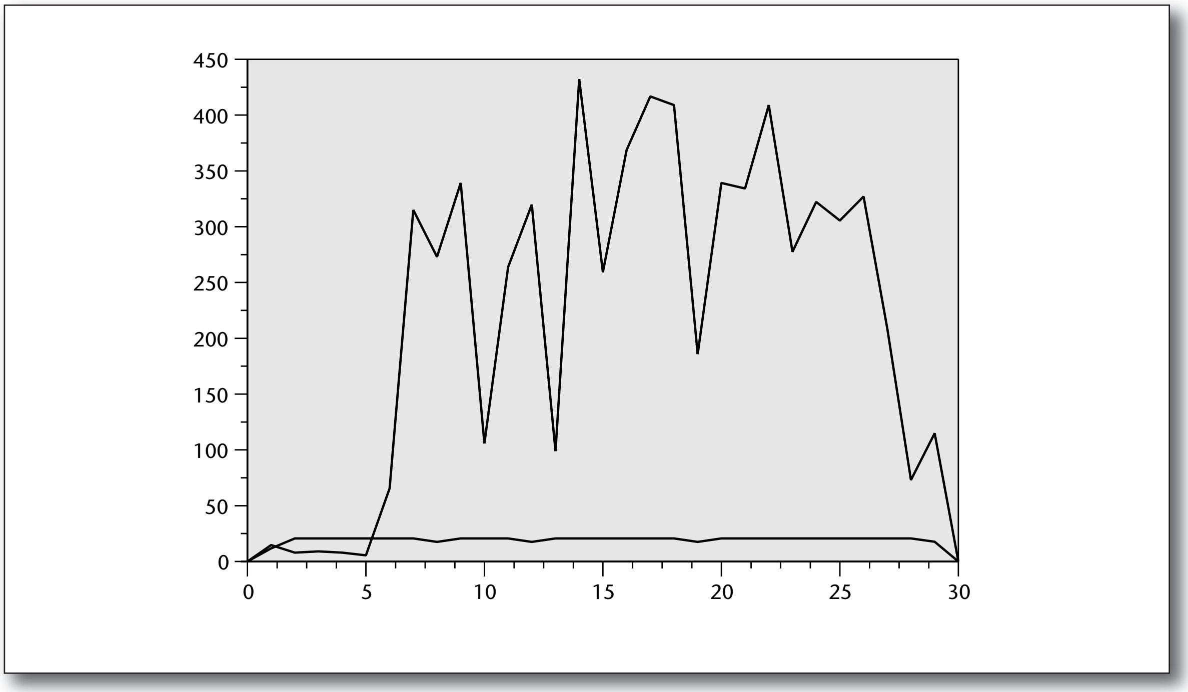


Figure 4. Anomaly per Week for the Event "HVAC Communication Failure."

The detected car 3029 (upper curve), and the "normal" car 3030 for comparison (lower curve).

and further development of the module have been done as part of two consecutive collaborative research projects. In the end, Addiva has the responsibility to integrate new versions of the anomaly-detection module into Addtrack.

Evaluation

To illustrate how the Bayesian anomaly-detection approach performs we have used event data from Regina train sets, which are operated by SJ (the largest passenger railway operator in Sweden) and manufactured by Bombardier Transportation. A Regina train set may consist of two or three cars: either a type DMA car coupled with a type DMB car, or a type DMA car followed by a type T0 car and a type DMB car. These different car types have somewhat different profiles in what events are generated. Therefore each car is compared only to other cars of the same type when looking for anomalies. The data were collected from 57 Regina train sets running in the Mälar region in Sweden in the period between October 1, 2003, and April 19, 2004. The number of different possible event codes is 1013, of which 695 different codes were registered during the period. The number of events in each sample was between 1 and 1849 with a median of 5 and interval lengths varied between 16,440 and 40,845 kilometers. The number of samples per event type varied between 1 and 21, with a median of 6 samples per type.

For each of the occurring event codes, each Regina car was compared to all other cars of the same type, over the time period as a whole. The cars displaying the largest anomalies are shown in table 1. It is then possible to take a closer look at the anomalies of interest by comparing the data for a single week at a time of a car and event code in question against all other cars and weeks. This is shown for the two first entries of table 1 in figures 4 and 5.

As can be seen from the figures, many of the detected anomalies above lasted for a long time before they were eventually eliminated. At the same time it is clear that they are detectable already from the first week or even first few days. (Both examples in the figures are confirmed as corresponding to real problems that were subsequently fixed.) Using this kind of anomaly detection therefore has the potential of giving important information to the service organization, with the possibility to rectify the problem earlier than today, often requiring less extensive repairs, causing less wear on related components, and reducing the time of operating with reduced functionality.

We have previously mentioned the importance of using Bayesian statistics, rather than classical point estimates of parameters, to limit the false alarm rate. For comparison between the approaches, a simple anomaly detector was implemented based on a point estimate $\hat{\lambda}$ of the Poisson rate per time unit λ together with the principal anomaly defined in equation 1, using $\theta = \hat{\lambda}T$ for the interval length T. The simple detector uses the maximum

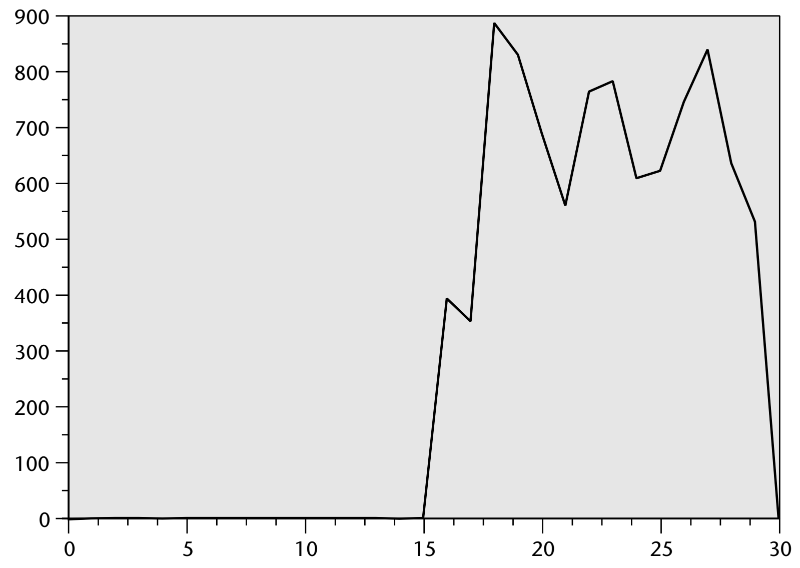


Figure 5. Anomaly Per Week for the Event "Train Heating, Fuse Failure" for Car 9024.

Car	Event type	Anomaly (Λ)
3029	HVAC communication failure	9316.7
9024	Train heating, fuse failure	8900.6
9004	Incorrect speed, axle 4	5605.0
1002	Large speed difference, axle 3	3213.6
9046	Low cooling water level in converter	3185.0
3044	Incorrect speed, axle 3	3046.3
9052	Large speed difference, axle 2	3024.3
9049	Large speed difference, axle 1	2834.8
1054	Carbon strip supervision disconnected	2371.1
1053	Carbon strip supervision disconnected	2304.7

Table 1. The Ten Most Abnormally Occurring Event Codes.

likelihood estimate for the Poisson rate, which is equal to the total number of events divided by the total interval length:

$$\hat{\lambda} = \sum_{i=1}^{n} x_i \Big/ \sum_{i=1}^{n} T_i \qquad (7)$$

where T_i is the length of the interval for sample *i*.

An anomaly threshold of $\varepsilon = 10^{-6}$ was used in both cases. When comparing the two anomaly detectors, out of 2841 samples, 2483 were classified as normal and 323 as anomalous by both. In addition, 35 samples were classified as anomalous by the point estimate anomaly detector but not by the Bayesian anomaly detector, while no samples were classified as anomalous by the Bayesian anomaly detector but not by the point estimate anomaly detector. This illustrates how the Bayesian anomaly detector is more conservative when there is not sufficient statistical significance to assume an anomaly. This makes the Bayesian anomaly detector more robust against false alarms than the non-Bayesian approach.

Discussion

There are many challenges when trying to use anomaly detection in a real-world application. The Bayesian principal anomaly, however, has many suitable properties for this: the false alarm rate, which is a major problem for many anomaly-detection algorithms when used in practice, can be controlled directly by adjusting the anomaly threshold; the Bayesian approach makes the system work also when there are limited amounts of training data, as is often the case; and the training data used may itself contain anomalies, that is, it need not be absolutely clean as for some other anomaly-detection methods, since the method will itself test each sample and learn only those that are judged nonanomalous.

Anomaly detection of event data is a rather general task that is applicable in a large number of situations. Many systems today generate log messages or alarms, and typically in such volumes that manual inspection is problematic. The risk that an operator misses an important alarm among the large amounts of less important alarms is significant. The chance of manually noticing subtle changes in rates of different alarms is small. The anomaly-detection method presented here is not limited to the train domain, but will function on log data from almost any system.

The design of the anomaly-detection module, with only a small number of access routines and a minimal amount of free parameters, makes it easy to plug into existing analysis tools for various domains and application areas. This allows the module to take advantage of existing analysis products, thereby avoiding the overhead of producing, marketing, and supporting a stand-alone anomaly-detection product.

The anomaly-detection method described here has been successfully deployed in the case of condition monitoring of trains. Deployment in furher domains will certainly follow.

References

Ahmed, T.; Oreshkin, B.; and Coates, M. 2007. Machine Learning Approaches to Network Anomaly Detection. Paper presented at the Second Workshop on Tackling Computer Systems Problems with Machine Learning. Cambridge, MA, 10 April.

Bjurling, B.; Holst, A.; Stahl, O.; and Wallgren, A. 2010. Statistical Anomaly Detection and Visualization. Paper presented at the 1st National Symposium on Technology and Methodology for Security and Crisis Management. Linköping, Sweden, October 27–28.

Cemerlic, A.; Yang, L.; and Kizza, J. 2008. Network Intrusion Detection Based on Bayesian Networks. Paper presented at the 20th International Conference on Software Engineering and Knowledge Engineering. San Francisco, 1–3 July.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly Detection: A Survey. *ACM Computing Surveys* 41(3): 1–58.

Chebrolu, S.; Abraham, A.; and Thomas, J. 2005. Feature Deduction and Ensemble Design of Intrusion Detection Systems. *Computers and Security* 24(4): 295–307.

Dey, C. 2009. Reducing IDS False Positives Using Incremental Stream Clustering (ISC) Algorithm. Master's thesis, Department of Computer and Systems Sciences, Royal Institute of Technology, Stockholm, Sweden.

García-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G.; and Vázquez, E. 2009. Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges. *Computers and Security* 28(1–2): 18–28.

Hill, D.; Minsker, B.; and Amir, E. 2009. Real-Time Bayesian Anomaly Detection in Streaming Environmental Data. *Water Resources Research* 46(4).

Holst, A., and Ekman, J. 2011. Incremental Stream Clustering for Anomaly Detection and Classification. In *Proceedings of the 11th Scandinavian Conference on Artificial Intelligence,* 100–107. Amsterdam, The Netherlands: IOS Press.

Holst, A.; Ekman, J.; and Larsen, S. 2006. Abnormality Detection in Event Data and Condition Counters on Regina Trains. Paper presented at the IET International Conference on Railway Condition Monitoring, Birmingham, UK, 29–30 November.

Lerner, U.; Parr, R.; Koller, D.; and Biswas, G. 2000. Bayesian Fault Detection and Diagnosis in Dynamic Systems. In *Proceedings of the 12th Innovative Applications of Artificial Intelligence Conference*, 531–537. Menlo Park, CA: AAAI Press.

Puttini, R.; Marrakchi, Z.; and Mé, L. 2003. A Bayesian Classification Model for Real-Time Intrusion Detection. Paper presented at the 22nd International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering, Moscow, ID, 3–7 August.

Shahabdeen, J. A.; Baxi, A.; and Nachman, L. 2010. Ambulatory Energy Expenditure Estimation: A Machine Learning Approach. In *Proceeding of the 22nd Innovative Applications of Artificial Intelligence Conference*. Menlo Park, CA: AAAI Press.

Sotiris, V.; Tse, P.; and Pecht, M. 2010. Anomaly Detection Through a Bayesian Support Vector Machine. *IEEE Transactions on Reliability* 59(2): 277–286.

Su, B., and Xu, C. 2009. Not So Naive Online Bayesian Spam Filter. In *Proceedings of the 21st Innovative Applications of Artificial Intelligence Conference*. Menlo Park, CA: AAAI Press.

Tian, S.; Yu, J.; and Yin, C. 2004. Anomaly Detection Using Support Vector Machines. In *Advances in Neural Networks,* volume 3173, Lecture Notes in Computer Science, 592–597. Berlin: Springer.

Anders Holst is a senior research scientist at Swedish Institute of Computer Science and associate professor at the Royal Institute of Technology in Stockholm, Sweden. His research focus is on machine learning, statistical models, anomaly detection, diagnosis, and prediction.

Markus Bohlin is a senior research scientist at Swedish Institute of Computer Science. He has a Ph.D. in computer science, and more than 10 years of experience with industrially motivated research. His research is focused on combinatorial optimization with applications to industrial maintenance, railways, and software analysis.

Jan Ekman is a senior research scientist at Swedish Institute of Computer Science. He has a Ph.D. in computer science in the field of proof theory as a branch of mathematical logic. A substantial part of his work at present concerns statistical modeling and anomaly detection in industrial projects.

Ola Sellin is working as a manager at Bombardier Transportation Sweden. His technical focus is mainly in the test tools and diagnostic data collection field. The objective is to find and, when possible in advance, predict errors in train systems by using state-of-the-art diagnostic techniques and tools.

Björn Lindström is a senior manager and engineer. He is one of the founders of the Addiva group. His focus is on business development for technical solutions and products.

Stefan Larsen is a senior diagnostic intelligence specialist and software architect at Eduro AB. He has more than 20 years of experience with the commissioning of — and after market for — complex equipment and systems in industry, and more than 10 years of experience in developing IT solutions for diagnostic intelligence. Since 2003, he has been involved in various research projects within anomaly detection, diagnosis, prediction, and optimization.