Chapter 12
Anomaly Detection with HTM

We model information and communications technology (ICT) systems as complex adaptive systems. Since we cannot hope to predict all future incidents in complex systems, real-time monitoring is needed to detect local failures before they propagate into global failures with an intolerable impact. In particular, monitoring is required to determine the consequences of injecting artificial errors into production systems and to learn how to avoid or limit the impact of future incidents.

In Part II we argued that anti-fragile ICT solutions in the cloud should have a service-oriented architecture with microservices, preferably created by development and operations (DevOps) teams. Since microservices depend on much fewer variables than a complete system, it is possible to monitor and diagnose microservice failures. However, the ability to monitor these services does not come for free [53]. DevOps teams need monitoring and logging setups for each type of microservice showing the up/down status, current throughput and latency, and details on circuit breaker status.

In this chapter, we discuss what an anomaly means and how the hierarchical temporal memory (HTM) learning algorithm detects anomalies in data streams. The HTM algorithm can be applied to many different types of data streams. Grok is an application that Numenta built on top of the Numenta Platform for Intelligent Computing (NuPIC) implementation of HTM (http://numenta.org/nupic.html) to detect anomalies in metric data provided by the Amazon Web Services (AWS) cloud. Here, we examine how Grok detects and displays anomalies in AWS streaming data. We then study how HTM detects rogue human behavior. The chapter is mostly based on information provided by Numenta [109, 110], including talks by Ahmad (http://youtube.com/watch?v=nVCKjZWYavM) and Purdy (http://youtube.com/watch?v=I5lSEHvngaI).

12.1 Anomalies

Complex ICT systems generate much data about their own operations. Cloud solutions are no exception. In fact, cloud providers offer services that allow solution owners to easily access operational data from their own cloud applications. The Internet of Things will likely lead to a huge increase in sensors generating continuous data streams about the status of both natural and man-made systems. The many data streams from current and future systems will make it impossible to analyze all the data in detail. One interesting alternative is to look for anomalies in the streams to detect the beginning of failures. There is evidence that it is possible to detect the beginning of large failures in different types of complex adaptive systems before the impact becomes intolerable [111, 112].

Anomalies are data patterns that do not conform to expected behavior [113]. A data stream of patterns can have several types of anomalies. A *spatial* (static) *anomaly* is a single pattern or set of relatively closely spaced patterns in the data stream that deviates from what is standard, normal, or expected. A *temporal anomaly* is a set of surprising transitions between patterns. Note that it is the temporal sequence that is surprising, not the individual patterns themselves. If the patterns in a stream are highly random, then it is hard or even impossible to detect spatial and temporal anomalies. However, it is possible to detect a change in the distribution of the random data, denoted a *distribution anomaly*. All three types of anomalies are temporary anomalies. When a surprising change first appears, then it is an anomaly. If it appears multiple times, then it is the "new normal" and ceases to be an anomaly.

12.2 HTM Anomaly Score

The Grok application built on top of HTM detects spatial, temporal, and distribution anomalies. Since HTM is an online continuous learning system, it will detect temporary anomalies and quickly learn when they are the new normal. HTM works for both numerical and categorical input data. The two data types can be mixed in an input stream to HTM because they are both converted to a sparse distributed representation (SDR).

HTM calculates an *anomaly score* for each new pattern it receives [109]. If a received pattern was predicted, then the anomaly score is zero. If the pattern was not predicted at all, then the score is one. A partially predicted pattern has a score between zero and one. The actual score depends on the "similarity" between the actual received pattern and the predicted pattern. The similarity is determined by the SDR. The larger the overlap between actual and predicted bits in column space, the smaller the anomaly score.

If none of the cells in a column were predicted, then all the cells are made active. This process is referred to as *bursting*. It occurs when there is no context, that is, when HTM is learning a new transition. At each time instance, the anomaly score is simply the fraction given by the number of bursting columns divided by the total number of active columns. In the beginning of the training, the anomaly score will be high because most patterns will be new. As HTM learns, the anomaly score will diminish until there is a change in the pattern stream.

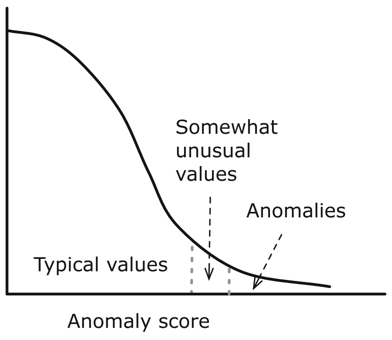

Fig. 12.1 Normal distribution of anomaly scores divided into typical values, somewhat unexpected values, and anomalies

12.3 HTM Anomaly Probabilities

There are cases where the anomaly score is all that is needed to detect anomalies, but there are also cases where the anomaly score produces too many false positives because the metric data are very noisy. To deal with noise, we compute *anomaly probabilities*. The anomaly probability values are calculated relative to historical metric data rather than being absolute measurements of anomalous behavior. In other words, the goal is to detect changes in the anomaly score itself.

To determine anomaly probabilities, we consider a window of previous calculated anomaly scores and compute estimates of the expectation and standard deviation of the values, assuming normally distributed scores. Figure 12.1 depicts the right half of a normal distribution of possible score values. When a new anomaly score arrives, we estimate how likely the value is using the normal distribution based on the window of previous values. A new value on the x-axis under the central area of the curve in Fig. 12.1 is a typical value that we should expect to see often. Typical values of the anomaly score indicate that the system is operating as desired.

To detect anomalies, we look for values associated with the right tail of the computed normal distribution. Values falling in the beginning of the tail in Fig. 12.1 are somewhat unusual, while values further out in the tail represent anomalous behavior. Because the distribution of the anomaly scores can change over time, the estimates of the expectation and standard deviation of the normal distribution are recalculated as the window slides over the previously received scores.

12.4 Grok the Cloud

The word *grok* was coined by Robert A. Heinlein in his 1961 science fiction novel *Stranger in a Strange Land*. To grok means to understand so thoroughly that the observer becomes a part of the observed. Numenta has built an application called Grok on top of the NuPIC implementation of HTM to detect anomalies in metric data from the AWS cloud. The application utilizes HTM to learn streaming metrics

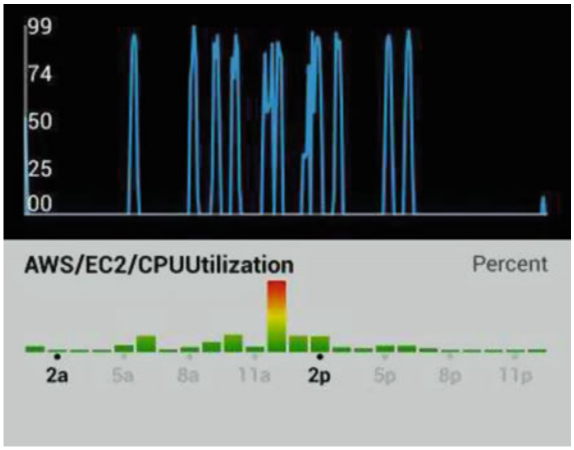

Fig. 12.2 While the blue curve showing CPU utilization looks normal to humans, Grok detected an anomaly (picture from [109])

from virtual machine clusters and to identify anomalies in these metrics. Grok builds a separate model for each monitored AWS metric. The metric values are combined with timestamps to allow Grok to learn patterns related to the time of day or the day of the week. To reduce the number of false positives, Grok calculates anomaly probabilities.

Figure 12.2 shows a part of the Grok user interface. The blue graph with the black background shows the CPU utilization of a virtual machine in the AWS cloud. The corresponding anomaly score is shown directly below. Grok uses color-coded bars to depict anomaly scores. The color and height of a bar have the same meaning, making it easier to see anomalies. The three types of anomaly probabilities, typical, somewhat unusual, and anomalies (see Fig. 12.1), are used to color the bars. Red represents an anomaly, a highly improbable score with a probability around 0.001 %. Yellow and green represent progressively more common scores.

The example in Fig. 12.2 illustrates that Grok can detect anomalies that are hard for a human to see in a raw metric stream. When it is not obvious why Grok flagged an anomaly, an operator can view the anomaly scores of other AWS metrics to gain more insight. Since Grok builds an independent model for each monitored metric stream, a system operator can obtain several independent confirmations that a virtual machine has unusual behavior.

In the next example, a load balancer distributes requests from many clients over a set of servers. The load balancer produces a fairly unpredictable or noisy metric stream showing the latency in serving web pages to clients. The blue curve in Fig. 12.3 represents the metric values fed into Grok, while the green, yellow, and red bars represent the anomaly scores colored according to the calculated anomaly probabilities. The example illustrates that Grok can find anomalies in noisy data.

In Chap. 5, we discussed how software engineers induced artificial failures into Netflix's media streaming system to discover vulnerabilities early, when their impacts are small. Early vulnerability detection allows engineers to improve systems and

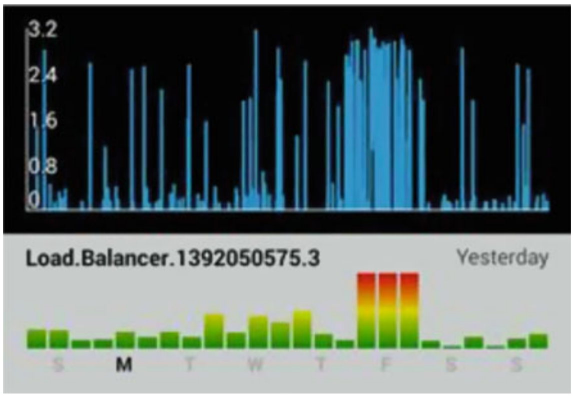

Fig. 12.3 An anomalous pattern detected within a noisy metric stream from a load balancer (picture from [109])

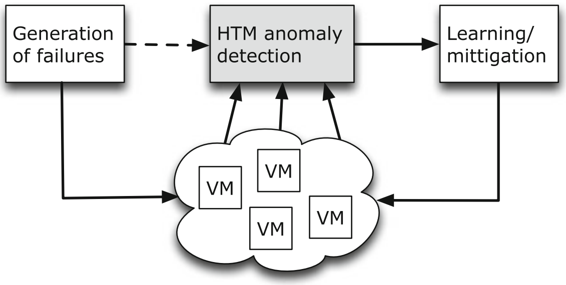

Fig. 12.4 A process to detect and mitigate the impact of induced (and natural) failures in virtual machines (VMs)

avoid failures with intolerable impacts. The flow diagram in Fig. 12.4 illustrates how Grok can be integrated into a process to detect and mitigate the impact of induced (and natural) failures in applications running in the AWS cloud. How the learning/mitigation step will be carried out depends on the application being monitored. Today, this step is carried out by humans. In the future, it may be possible to automate at least part of the step.

12.5 Rogue Behavior

Numenta has developed an application for rogue behavior detection (RBD) based on HTM [110]. Using human- and machine-generated data, the RBD application automatically models an individual's behavior and identifies irregular actions. This anomaly detection of irregular human behavior is useful for ICT security, device access control, and fraud detection.

The RBD application has several attractive properties due to HTM. First, it is not necessary to divide employees into classes and define what normal behavior is for each class. Furthermore, there is no need for a separate training period or retraining, since employee behavior changes over time. The application learns continuously in

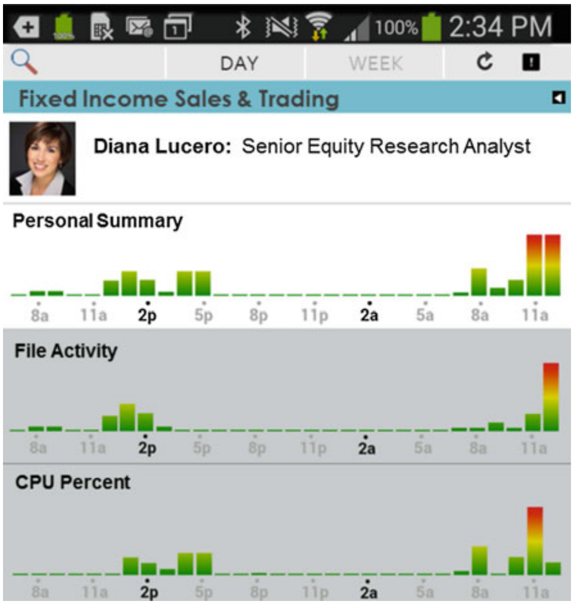
2:34 PM
DAY
WEEK
Fixed Income Sales & Trading
Diana Lucero: Senior Equity Research Analyst
Personal Summary
File Activity
CPU Percent
8a 11a 2p 5p 8p 11p 2a 5a 8a 11a

Fig. 12.5 An employee anomaly (picture from [110])

real time and builds a separate model for each monitored individual, thus achieving high-precision anomaly detection for all individuals. The same high-quality detection is generally not possible with class-based monitoring. Finally, real-time anomaly detection enables quick corrective actions to avoid or at least reduce the negative consequences of illegal actions.

Figure 12.5 shows part of the user interface for the RBD application. The senior analyst monitored, Diana Lucero, is part of an experiment to test the application. She exhibits unusual behavior at 11a.m. Drilling down to see the anomaly scores for the individual metric streams, we find spikes in both the file activity and CPU usage. Further investigation finds that the RBD application reacted because the analyst generated and stored a large .zip file containing intellectual property. The early detection of this activity made it possible to stop the analyst from transmitting the file to a third party.

12.6 Detecting the Beginning of Swans

In this book, we have assumed that there is no fundamental difference between frequent incidents with a tolerable impact and rare incidents with an intolerable impact, called swans. Most swans simply start out as local incidents that do not stop but propagate due to positive feedback loops. According to Sect. 2.3, to predict any future incident, we must describe the event, estimate its probability, and calculate the impact. In Chap. 2, we argued that humans have limited ability to predict swans.

It is unlikely that a group of stakeholders will predict all potential swans in a complex adaptive ICT system, even if they use significant resources in classical risk analysis.

At the time of this writing, the detection of catastrophic events in real time is an active area of research [111, 112]. Because global or emergent failures very often start out as local failures in complex ICT systems, it is possible to detect the beginning of a swan in real time, even though we may not immediately understand the underlying reasons for its occurrence. HTM detects unlikely behavior by observing the fraction of bursting cell columns. Because HTM can be applied to different data streams, it can detect the beginning of swans in different types of complex ICT systems. It is still essential to realize the four design principles in Chap. 4 to avoid positive feedback loops that quickly propagate local failures into global failures before countermeasures can be introduced.

12.7 Discussion and Summary

Government agencies regulate many complex adaptive ICT systems of national importance. Unfortunately, it is very hard for a regulator to gain an adequate understanding of a complex ICT system without being closely involved in its design and daily operation. A regulator can set all kinds of non-functional requirements but cannot discover system fragilities or request useful improvements from afar. Regulation and compliance really only make sense for relatively simple systems that have one best method of working [18]. There will always be a significant gap between a regulator's understanding of a complex ICT system and the way it really operates. This gap must be filled by other stakeholders. This is particularly true for complex ICT systems with microservice architectures.

For a system to achieve anti-fragility to a class of negative events, stakeholders must monitor the operation of the microservices, especially their outputs, and detect anomalies. While information technology (IT) departments know how to monitor monolithic applications with single executables, it is more challenging to monitor applications of microservices running in clouds and communicating over network connections. Since a solution may fail even though all its microservices work according to their specifications (see Sect. 4.6), it may be necessary to trace the communication between services to understand why a particular service received input values for which it was not designed. Furthermore, there are many network connections where latency could cause intermediate problems. Hence, sophisticated monitoring of a large number of microservices and their communications is needed to detect anomalies, determine failures, and create anti-fragile solutions.

A comprehensive comparison of different techniques to detect anomalies in streaming data is outside the scope of this book. We have only illustrated how HTM detects anomalies in two domains. However, the performance results of Price [97], Galetzka [98], and Numenta [109, 110] strongly indicate that HTM is a good choice for anomaly detection in streaming data. In 2015, Numenta published source code and test data to compare the performance of anomaly detection algorithms. The initial results show that the HTM algorithm detects anomalies earlier than other popular algorithms (http://github.com/Numenta/NAB). If we have good anomaly detectors connected to a complex adaptive ICT system, then we can detect anomalies before the whole system breaks down. We have seen that HTM is able to detect changes before it is obvious to a human that a new problem is brewing.

A reader interested in more information about anomaly detection with HTM, as well as more examples detecting sudden, slow, and subtle anomalies, should study Numenta's two white papers [109, 110]. At the time of this writing, is also possible to use Grock for IT analytics and Grok for stocks on the Web.

What to learn from Part IV

Part IV introduced a novel learning algorithm based on Hawkins' HTM theory. HTM explains how the neocortex learns by modeling and processing data from the body's sensory organs. We concentrated on understanding how the HTM learning algorithm can detect anomalies in complex adaptive ICT systems. While most anomaly detection techniques are created to determine anomalies in data stored in databases, HTM finds anomalies in real-time streaming data. There is no need to store huge amounts of data since HTM builds models representing the properties of the data.

The ability to process streaming data makes the HTM learning algorithm ideal for applications running on cloud platforms since leading cloud providers offer services that stream metrics about an application's state. HTM's ability to process streaming data from a huge number of sensors also makes the algorithm perfect for monitoring the Internet of Things. While the current version of HTM is implemented in software, a hardware implementation is needed to seriously scale the algorithm's operation.