An Introduction into Anomaly Detection Using CUSUM

Dominik Dahlem

2015-11-24 Tue

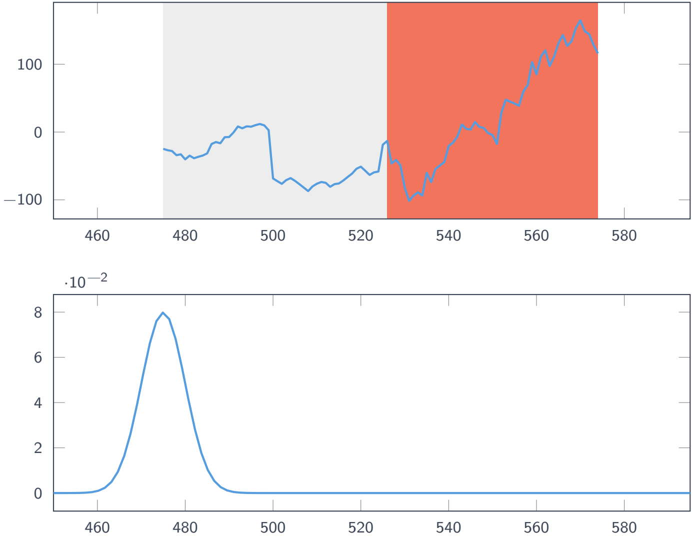
Outline

Who am I?

- Dominik Dahlem, Lead Data Scientist, Boxever

dominik.dahlem@boxever.com
http://ie.linkedin.com/in/ddahlem
http://github.com/dahlem
@dahlemd

Anomaly Creation



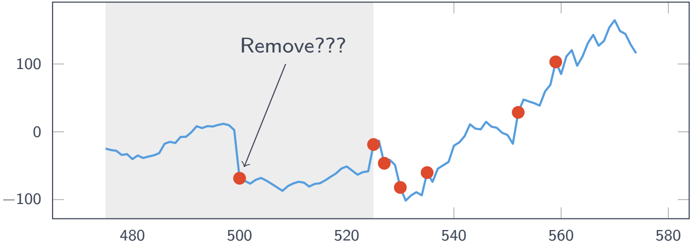
What is Anomaly Detection

- Find the problem before it affects businesses (especially SaaS)
- But what is an anomaly?
  - rare class mining
  - chance discovery
  - novelty detection
  - exception mining
  - noise removal

Challenges

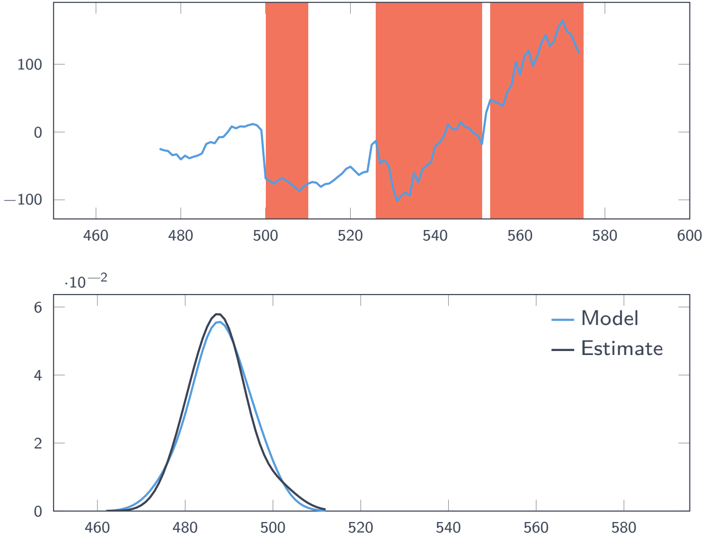
- What is normal behaviour...
  - ... if normal behaviour keeps evolving?
  - ... if the boundary between normal and anomolous behaviour is imprecise?
- What are we doing with noisy data...
  - ... if the notion of outliers differs across application domains?
- Availability of training data

Outlier Detection




```
outliers <- function(x, s) {
    q <- quantile(x, probs=c(.25, .75), na.rm=T)
    iqr <- q[2] - q[1]
    idx <- which((x < (q[1]-s*iqr)) | (x > (q[2]+s*iqr)),
                 arr.ind=T)
    return(idx)
}
```

Model-based Anomaly Detection



Some Definitions

- Let $M_t(s_t \mid s, c)$ denote the probability of event s_t given a context c of the current regime and the timeseries s
- Let $T_w(s_t)$ denote the probability of event s_t of the current regime over the past w observations
    - the context c is implicit and could be day of the week, hour of the day, etc.

The CUSUM Method[2] (I)

- Let's use the Gaussian kernel to estimate the densities[1]

$$\hat{T}_w(s_t) = \frac{1}{w} \sum_{\ell=t-w}^{t-1} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(s_\ell - s_t)^2}{2\sigma^2}}. \tag{1}$$

$$\hat{M}(s_t \mid s, c) = \frac{1}{k} \sum_{\ell \in \Phi(s,c)} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(s_\ell - s_t)^2}{2\sigma^2}} \tag{2}$$

```
gauss <- function(st, s, sigmaSq) {
    Z <- 1/(sqrt(2 * pi * sigmaSq))
    bw <- 1/(2 * sigmaSq)
    n <- length(s)
    return(1/n * Z * sum(exp(-bw * (s - st)^2)))
}
```

[1] E. Gine (2002). "Rates of strong uniform consistency for multivariate kernel density estimators". In: *Annales de l?Institut Henri Poincare (B) Probability and Statistics* 38.6, pp. 907–921.

[2] E. S. Page (1954). "Continuous Inspection Schemes". In: *Biometrika* 41.1/2, pp. 100–115.

The CUSUM Method (II)

- Cumulative sum of the log-likelihood ratios

$$R_t = \log\left[\frac{\hat{T}_w(s_t)}{\hat{M}(s_t \mid s, c)}\right] \tag{3}$$

$$S_t = S_{t-1} + R_t. \tag{4}$$

- Raise alarm if

$$\tau = \inf\{t \mid S_t - \min_{0 \leqslant k \leqslant t}(S_k) > \delta\}. \tag{5}$$

- The cumulative sum is zero if there is no anomaly

CUSUM in R (I)

```
1   cusum <- function(params, training, s) {
2       window <- params[1]
3       slope <- params[2]
4       df <- data.frame(t=seq(1,nrow(s)), g=rep(0, nrow(s)),
5                        R=rep(0, nrow(s)), Q=rep(0, nrow(s)),
6                        P=rep(0, nrow(s)), Sn=rep(0, nrow(s)),
7                        SnPrime=rep(0, nrow(s)), anomaly=rep(0, nrow(s)))
8
9       for (t in (2+window):nrow(s)) {
10          s.subset <- s[(t-window):t,]
11          cs <- unique(s.subset$c)
12          training.subset <- subset(training, c %in% cs)
```

- Lines 2 and 3 are the tuning parameters
- Line 9 steps over each time t
- Line 10 looks only at the last `window` observations
- Line 11 selects the contexts within the chosen observations
- Line 12: training set given the context(s)

CUSUM in R (II)

```
1   S <- var(training.subset$s)
2   q.s <- training.subset$s[sample(length(training.subset$s),
3            size=window, replace=T, prob=NULL)]
4
5   df$P[t] <- gauss(s$s[t], q.s, S)
6   df$Q[t] <- gauss(s$s[t], s.subset$s, S)
7
8   if (df$P[t] & df$Q[t]) {
9       df$R[t] <- -log(df$P[t]) + log(df$Q[t])
10  } else {
11      df$R[t] <- 0.0
12  }
13  df$Sn[t] <- df$Sn[t-1] + df$R[t]
14  df$g[t] <- max(0, df$Sn[t])
```

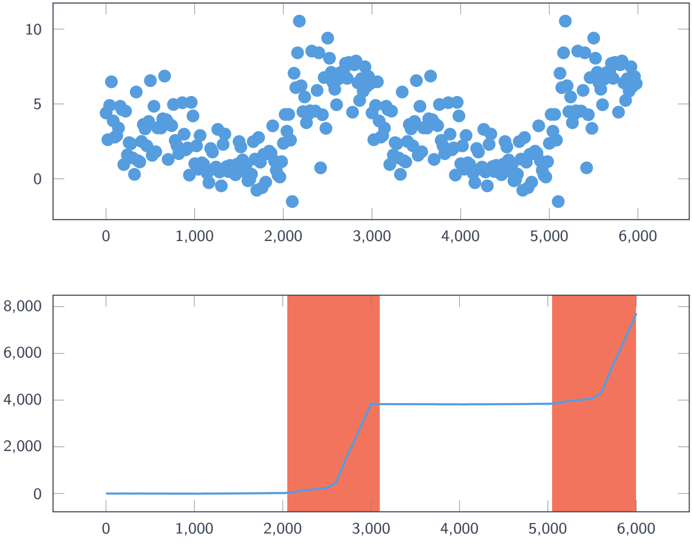
- Line 1: variance used for the kernels
- Line 2: sample `window` observations from the training set
- Lines 5 and 6: Compute probabilities of observing event s_t (see equations 2 and 1)
- Lines 9-14: CUSUM (see equations 3 and 4)

CUSUM in R (III)

```
1           df$SnPrime[t] <- (df$Sn[t]-df$Sn[t-window])/window
2           if (df$SnPrime[t] < slope) {
3               df$anomaly[t] <- 0
4           } else {
5               df$anomaly[t] <- 1
6           }
7       }
8       return(df)
9   }
```

- Line 1: compute the first derivative of the cumalative sum
  - The cumulative sum keeps growing
  - We would like to detect multiple anomalies
- Line 2: test whether we should raise an alarm

CUSUM in Action



Also look at...

- https://github.com/twitter/AnomalyDetection
- https://github.com/twitter/BreakoutDetection
- https://github.com/robjhyndman/anomalous-acm
- These slides and the R package is available at
    - https://github.com/dahlem/cusum

Concluding...

- We can detect anomalies if we can assume stationarity
  - We modelled $s_t - s_{t-1} \approx P(s_t - s_{t-1} \mid c)$
- CUSUM is sequential, so we can do anomaly detection easily in real-time
- Types of anomalies
  - unusual noise
  - more noise
  - break down
  - sudden grow
  - peaks
  - no noise
- We cannot detect all kinds of anomalies with a single algorithm
  - use ensembles
  - handle outliers first?

Boxever

Thank You!

dominik.dahlem@boxever.com

Misclassification Rate

- Use this function to compute the classification error between training and test set

```
classification.error <- function(x, training, s, ...) {
    df <- cusum(x, training, s, ...)
    return(1.0 - mean(df$anomaly == s$l))
}
```

Parameter Tuning

- We use the CRAN package `optimx` to tune the parameters for anomaly detection

```
library(cusum)

args <- commandArgs(trailingOnly = TRUE)

training <- read.csv(args[1],
                     header=F,
                     col.names=c("c", "t", "s", "tot", "l"),
                     stringsAsFactors=F)
training$s <- as.numeric(training$s)

s <- read.csv(args[2],
              header=F,
              col.names=c("c", "t", "s", "tot", "l"),
              stringsAsFactors=F)
s$s <- as.numeric(s$s)

window <- as.integer(args[4])
slope <- as.integer(args[5])

error <- optimx(c(window, slope), classification.error, training=training, s=s)
print(error)
```

R Script

- Use this script to run CUSUM on the command-line

```r
library(cusum)

args <- commandArgs(trailingOnly = TRUE)

training <- read.csv(args[1],
                     header=F,
                     col.names=c("c", "t", "s", "tot", "l"),
                     stringsAsFactors=F)
training$s <- as.numeric(training$s)

s <- read.csv(args[2],
              header=F,
              col.names=c("c", "t", "s", "tot", "l"),
              stringsAsFactors=F)
s$s <- as.numeric(s$s)

window <- as.integer(args[4])
slope <- as.integer(args[5])

df <- cusum(c(window, slope), training, s)
write.table(df, args[3], row.names=F, sep=",", quote=F)
```