

Applying the Q_n Estimator Online

Robin Nunkesser¹ Karen Schettlinger² Roland Fried²

¹Department of Computer Science, University of Dortmund

²Department of Statistics, University of Dortmund

GfKI 2007

1 Introduction

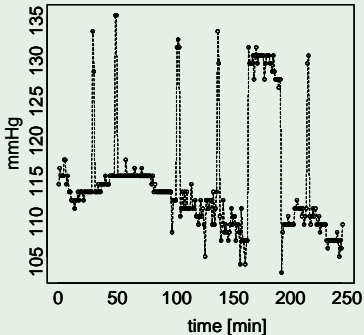
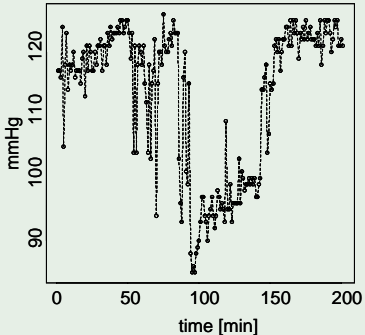
2 Computation

- Offline Computation
- Online Computation
- Running Time Simulations

3 Application in a Simulation Study

Motivation: Intensive Care Online Monitoring

Arterial Mean Pressure of Two Patients



Goal

Fast detection of relevant patterns (level/trend changes) resisting the outliers

Signal and Noise Model for (y_t)

Model

$$y_t = \mu_t + u_t + v_t$$

- μ_t (signal): smooth with a few sudden shifts
- u_t (observational noise): symmetric, mean zero
- v_t (spiky noise): measurement artifacts

Moving window $(y_{t-m+1}, \dots, y_t, \dots, y_{t+h})$ to approximate μ_t

Choice of Width $n = m + h$

- ↑ variance, robustness
- ↓ bias, computation time, delay

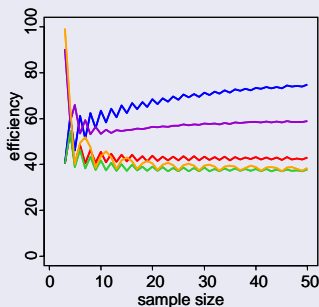
The Q_n Scale Estimator

Definition (Rousseeuw and Croux (1993))

For n points $x_i \in \mathbb{R}$ and $h = \lfloor n/2 \rfloor + 1$ the estimator Q_n is defined as

$$Q_n = d_n 2.2219 \{ |x_i - x_j|; i < j \}_{(h)}.$$

Gaussian Efficiencies of Different Robust Scale Estimators



Q_n

S_n

Length of the shortest half (LSH)

Interquartile range (IQR)

Median absolute deviation (MAD)

Problem Transformation

Transformation of the Q_n

$$Q_n = c \cdot \{|x_i - x_j|; i < j\}_{(k)} = c \cdot \{x_{(i)} - x_{(n-j+1)}; 1 \leq i, j \leq n\}_{(k+n+\binom{n}{2})}$$

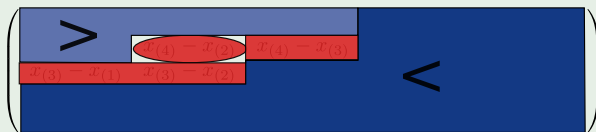
Q_n Computation by Selecting a Rank in a Matrix of Type $X + Y$

$$\begin{pmatrix} x_{(n)} + (-x_{(1)}) & \cdots & 0 \\ \vdots & & 0 & x_{(n-1)} + (-x_{(n)}) \\ \vdots & & & \vdots \\ x_{(2)} + (-x_{(1)}) & 0 & & \vdots \\ 0 & x_{(1)} + (-x_{(2)}) & \cdots & x_{(1)} + (-x_{(n)}) \end{pmatrix}$$

Definition (Hodges-Lehmann estimator)

$$\hat{\mu} = \text{median} \left\{ \frac{x_i + x_j}{2}, 1 \leq i \leq j \leq n \right\}$$

The Offline Algorithm of Johnson and Mizoguchi



- 1 Select one of the (remaining) matrix elements
- 2 Divide the remaining elements in $\mathcal{O}(n)$ time into three parts by
 - 1 Finding elements certainly smaller than the selected element
 - 2 Finding elements certainly greater than the selected element
- 3 Determine one or two of the parts to exclude from the search
- 4 If less than n elements remain determine the sought-after element, else go to step 1

If the selection in step 1 is done by means of the weighted median, the algorithm takes optimal $\mathcal{O}(n \log n)$ time

Concept of the Online Algorithm

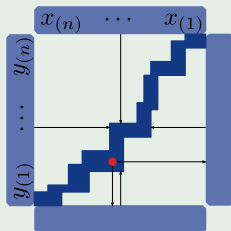
Use a buffer of size $\mathcal{O}(n)$ and data structures supporting fast update

Indexed AVL tree

Getting, inserting, removing and rank information available in $\mathcal{O}(\log n)$

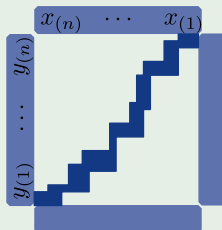
Each element in X , Y and the buffer is stored in an IAVL and supported by two pointers

The Data Structure Used



Updating needs $\mathcal{O}(|\text{inserted/deleted buffer elements}| \log n)$ per update

Example of a Column Deletion



Column Deletion

- 1 Search in X for the element that is to be deleted ($\mathcal{O}(\log n)$)
- 2 Follow pointers to the largest/smallest element in the column ($\mathcal{O}(1)$)
- 3 Determine the rows of these elements ($\mathcal{O}(1)$)
- 4 Compute and delete all elements in between from the buffer ($\mathcal{O}(|\text{deleted elements}| \log n)$)

Drawback

In a worst-case scenario we still need $\mathcal{O}(\sqrt{n + k'} \log n)$ accumulated time per update with $k' = \mathcal{O}(n^2)$

But...

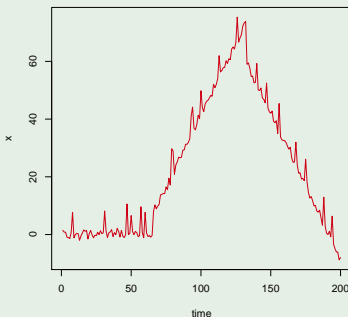
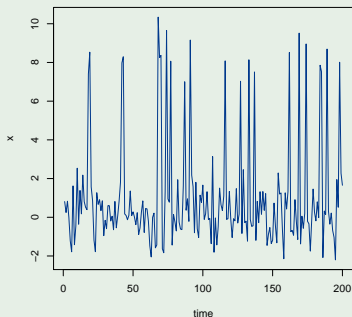
- If every update position has equal probability the expected accumulated time per update is $\mathcal{O}(\log n)$
- This case occurs e.g. for a constant signal with stationary noise
- General bound: $\mathcal{O}(\text{Prob}(\text{most probable update position})n \log n)$
- We expect good behaviour in practice

Running Time Simulations

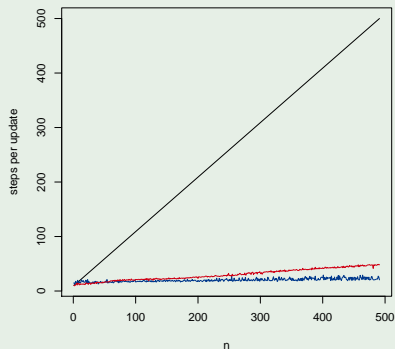
Simulated Datasets

- 1 Constant signal with standard normal noise and 10% outliers
- 2 Additional shifts and trends

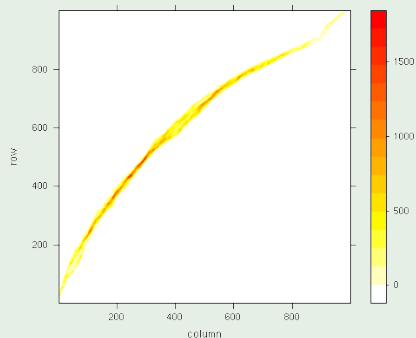
Simulated Datasets



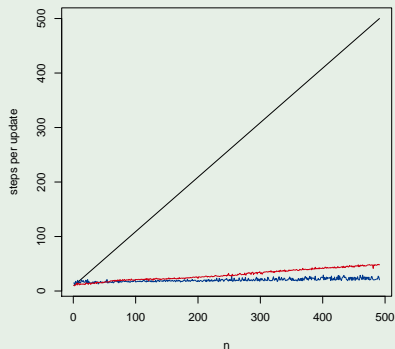
Steps Performed per Update



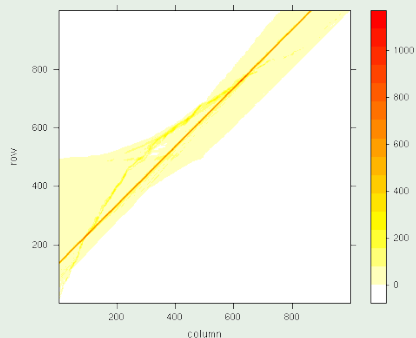
Measured Buffer Positions



Steps Performed per Update



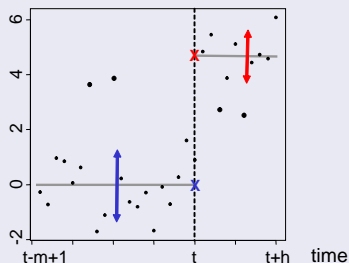
Measured Buffer Positions



Shift Detection

Comparison of Two Window Level Estimates Right and Left of Time t

Local constant level

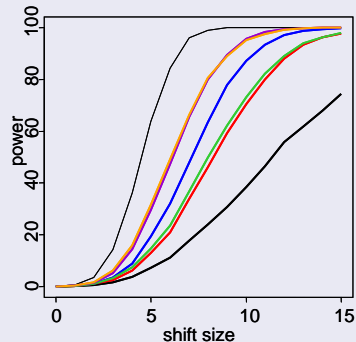


$$T(y_t) = \frac{\hat{\mu}_{t+} - \hat{\mu}_{t-}}{\sqrt{c \left(\frac{\hat{\sigma}_{t+}^2}{h} + \frac{\hat{\sigma}_{t-}^2}{m} \right)}}$$

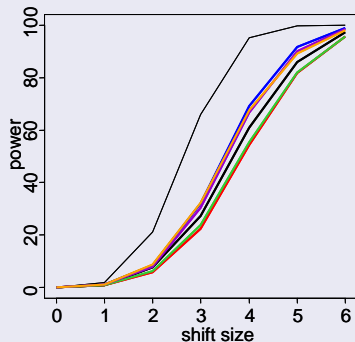
- Two-sample t -test (Stein & Follow, 1985)
- Trimmed two-sample t -test (Hou & Koh, 2003)
- Median comparison (Bovik & Munson, 1986, Hwang & Haddad, 1994)

Power in the Standard Normal Case

Widths $m = h = 7$



Widths $m = h = 15$



t-test

Qn

LSH

MAD

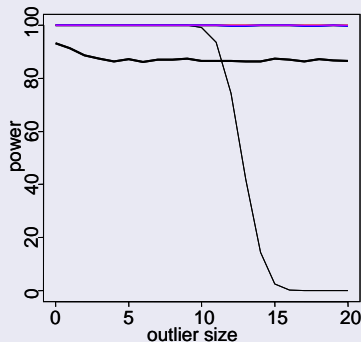
30%-trimmed t-test

Sn

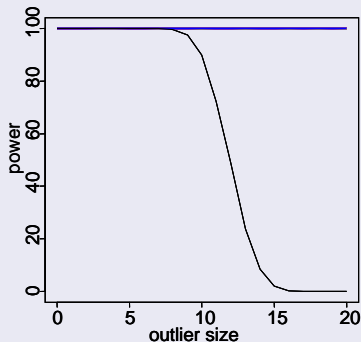
IQR

Power in Case of a Single Outlier of Increasing Size

Widths $m = h = 7$



Widths $m = h = 15$



t-test

Q_n

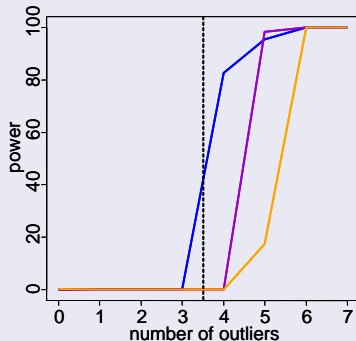
30%-trimmed t-test

S_n

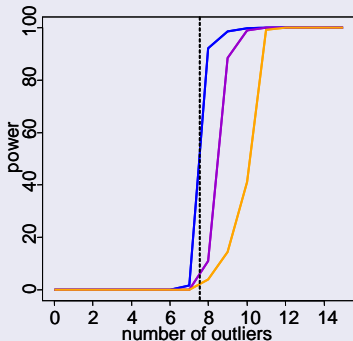
IQR

Increasing Number of Outliers

Widths $m = h = 7$



Widths $m = h = 15$



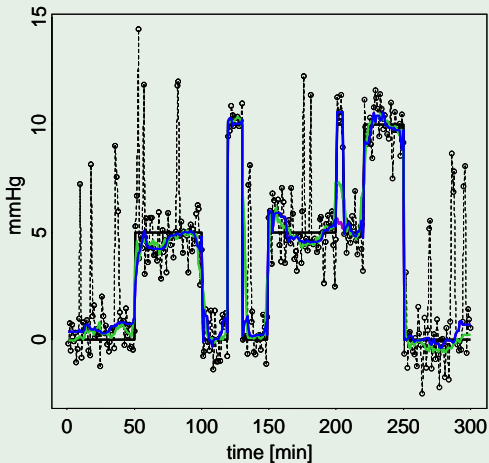
Qn

Sn

IQR

Example: Piecewise Constant Signal

Moving window filters (width $m = h = 9$)










Modified Trimmed Mean

Median with MAD

Median with Q_n

- New online algorithm which is substantially faster in practice
- Shift detection based on half-window medians and Q_n scale estimator
- Choose thresholds for shift detection based on estimated lag-1 autocorrelation derived using the Q_n once more

Thank you!

-  Bernholt, T., Fried, R., 2003. Computing the update of the repeated median regression line in linear time. *Inf. Process. Lett.* 88, 111–117.
-  Bernholt, T., Fried, R., Gather, U., Wegener, I., 2006. Modified repeated median filters. *Statistics and Computing* 16 (2), 177–192.
-  Davies, P. L., Fried, R., Gather, U., May 2004. Robust signal extraction for on-line monitoring data. *J. Stat. Plan. Infer.* 122 (1-2), 65–78.
-  Fried, R., Jun. 2004. Robust filtering of time series with trends. *Journal of Nonparametric Statistics* 16 (3 - 4), 313–328.
-  Fried, R., Bernholt, T., Gather, U., May 2006. Repeated median and hybrid filters. *Comput. Statist. Data. Anal.* 50 (9), 2313–2338.
-  Gather, U., Fried, R., 2003. Robust scale estimation for local linear temporal trends. *Tatra Mt. Math. Publ.* 26, 87–101.
-  Gather, U., Fried, R., 2004. Methods and algorithms for robust filtering. In: *Proceedings of COMPSTAT 2004*. Physica Verlag,