

# Online Outlier Detection in Sensor Data Using Non-Parametric Models

Themis Palpanas

*Univ of Trento*

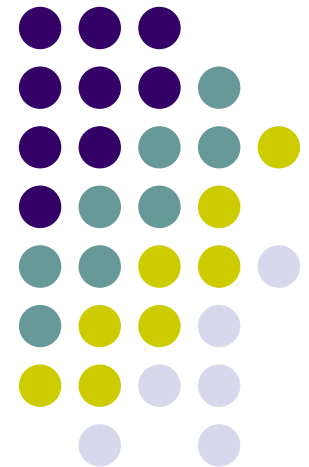
Sharmila Subramaniam

Dimitris Papadopoulos

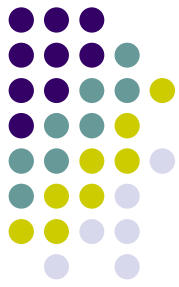
Vana Kalogeraki

Dimitrios Gunopulos

*Univ of California, Riverside*

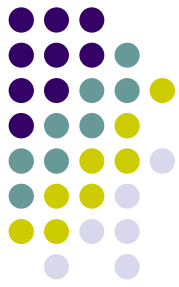


# Introduction

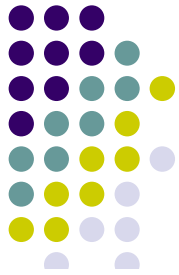


- several emerging applications across industries are event-driven
  - consume streaming data produced by a variety of data sources
  - process those data, reason about them, take corresponding actions
- streaming data management desiderata
  - process data in real time
  - be able to scale in number of sources, data rates
  - perform intelligent data analysis
- some applications are only interested in special events that constitute abnormal behavior
  - then, we can filter out of the streaming data the normal behavior
  - focus on the interesting (and infrequent) data values

# Applications: Monitoring Production Control Systems

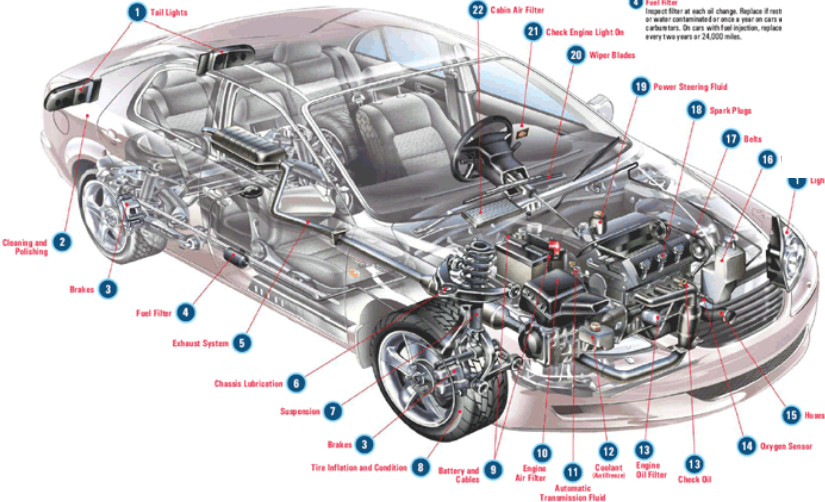


# Applications: Monitoring Vehicle Operation



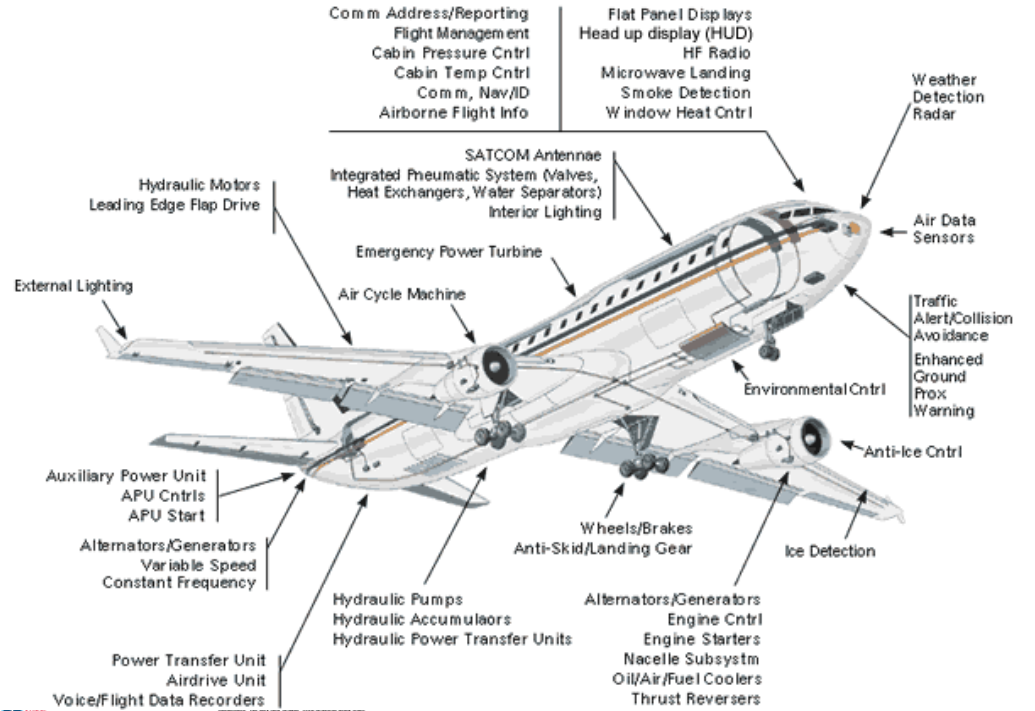
Be Car Care Aware

## Vehicle System/Component Service Notes



- 1 **Lights**  
Replace bulb immediately if light is out. Check fuses first.
- 2 **Cleaning and Polishing**  
To prevent stripping the vehicle's wax finish, use only automotive car wash products, not dishwashing liquids. Polish at least twice a year to maintain protection of the finish.
- 3 **Brakes**  
Check the entire brake system every year. Inspect brake linings, rotors and drums.
- 4 **Fuel Filter**  
Inspect filter at each oil change. Replace if rust or water contamination is present or use on cars with carburetors. On cars with fuel injection, replace every two years or 24,000 miles.
- 5 **Exhaust System**
- 6 **Chassis Lubrication**
- 7 **Suspension**
- 8 **Tire Inflation and Condition**
- 9 **Battery and Cables**
- 10 **Engine Air Filter**
- 11 **Engine Oil Filter**
- 12 **Clutch Adjustment**
- 13 **Engine Oil Filter**
- 13 **Check Oil**
- 14 **Oxygen Sensor**
- 15 **Hoses**
- 16 **Belts**
- 17 **Spark Plugs**
- 18 **Power Steering Fluid**
- 19 **Wiper Blades**
- 20 **Check Engine Light On**
- 21 **Cabin Air Filter**
- 22 **Tail Lights**

- 17 **Belts**  
Check V belts and serpentine belts for looseness and condition. Replace when cracked, frayed, glazed or showing signs of excessive wear. Replace timing belt per interval specified in owner's manual. Typically this is 60,000 to 90,000 miles. Not replacing the belt as required could cause a breakdown or serious engine damage.
- 18 **Spark Plugs**  
Typical replacement intervals range between 20,000 and 150,000 miles, depending on the vehicle and type of spark plug. Always consult your owner's manual for your specific vehicle.
- 19 **Power Steering Fluid**  
Check the fluid with the car warmed up. Add correct type of fluid if low. If frequent topping off is required, inspect for leaks and replace if contaminated.
- 20 **Wiper Blades**  
Replace every six months or when cracked, cut, torn, streaking or chattering.
- 21 **Check Engine Light On**  
If light comes on while driving or remains on, your vehicle may have an emissions or sensor problem and should be inspected. If light flashes, the condition is more severe and must be checked immediately to prevent catalytic converter damage.
- 22 **Cabin Air Filter**  
Replace annually, or more often in areas with heavy airborne contaminants or whenever heating or cooling efficiency is reduced.



# Problem Overview



- detect abnormal behavior (identify outliers)
- important for
  - situation detection
  - focusing on the interesting events in the data
  - react only to the important readings
- focus of this study:
  - streaming data
  - sliding window model
  - distributed processing (in network of sensors)

# Roadmap



- Outliers
  - Distance-Based Outliers
  - Density-Based Outliers
- Input Data Distribution Estimation
  - Kernel Density Estimators
- Proposed Solution for Online, Distributed Outlier Detection
- Experimental Evaluation
- Related Work
- Conclusions

# Abnormal Behavior

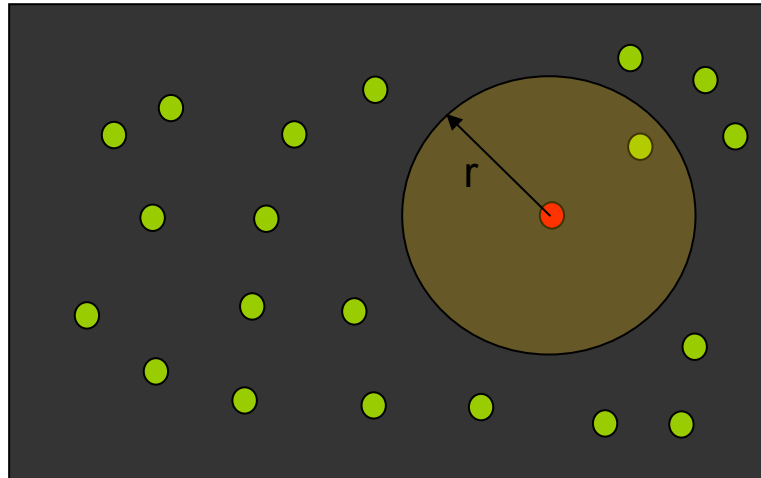


- deviations / outliers
  - a value that deviates significantly from the rest of the values in the dataset
  - several definitions
  - distance-based, density-based
- consider two definitions
  - $O(r, K)$  (distance-based)
  - $MDEF$  (density-based)
    - Multi-granularity Deviation Factor

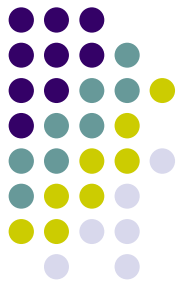


# $O(r, K)$ Outliers

- outlier
  - a value that has few near neighbors
  - set of outliers  $O = \{ p \in D \mid D_r, \forall q \in D_r : dist(p, q) < r \wedge |D_r| \leq K \}$
  - corresponds to statistical tests for outliers
    - for particular choices of  $(r, K)$ , gives the same result as statistical tests, for several probability distributions



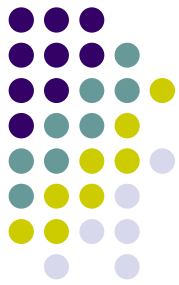




# Identifying $O(r, K)$ Outliers

- problem
  - for every data point in the stream:
    - count the number of near neighbors
    - if these neighbors are too few, declare the data point an outlier
- issues
  - how can we count the number of neighbors?
  - how can we do these computations in a distributed fashion?
  - how can we do that fast, with an online algorithm?

# MDEF Outliers

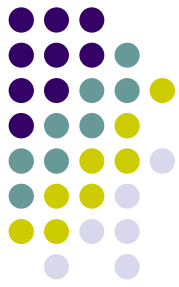


- outlier
  - a value whose near neighborhood is significantly less dense than its extended neighborhood

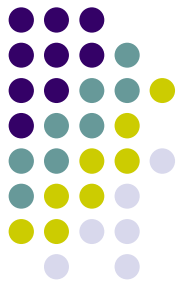


graph by S.Papadimitriou

# MDEF Outliers



- outlier
  - a value whose near neighborhood is significantly less dense than its extended neighborhood
  - set of outliers  $O = \{ p \in D \mid MDEF (p, r, a) > k_{\sigma} \sigma_{MDEF} (p, r, a) \}$ 
    - $MDEF$  at radius  $r$  for point  $p$  is relative deviation of its local neighborhood density from the average local neighborhood density in its  $r$ -neighborhood  
 $MDEF(p, r, \alpha) = 1 - n(p, \alpha r) / n'(p, \alpha, r)$
    - in uniformly distributed dataset (almost) all points have  $MDEF$  equal to 0
    - essentially parameter free:  $\alpha$  and  $k_{\sigma}$  predetermined constants with robust behavior across different datasets



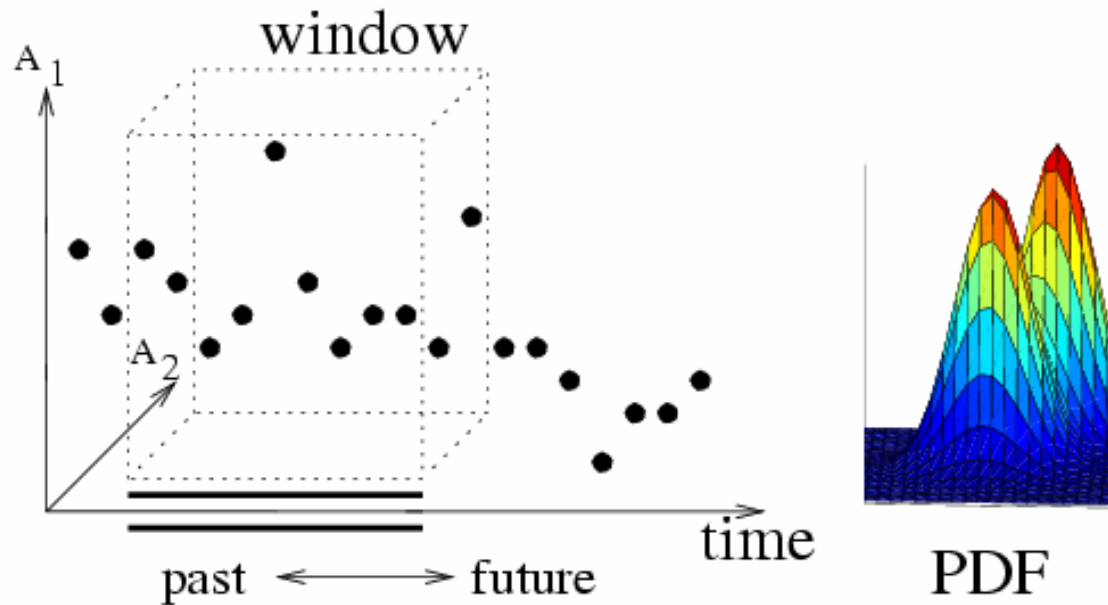
# Identifying *MDEF* Outliers

- problem
  - for every data point in the stream:
    - count the number of near neighbors
    - average the number of near neighbors for all the points in the extended neighborhood
    - sum of number of neighbors for a grid decomposition of the data space
- issues
  - how can we compute all these counts for the number of neighbors?
  - how can we do these computations in a distributed fashion?
  - how can we do that fast, with an online algorithm?



# Input Data Distribution Estimation

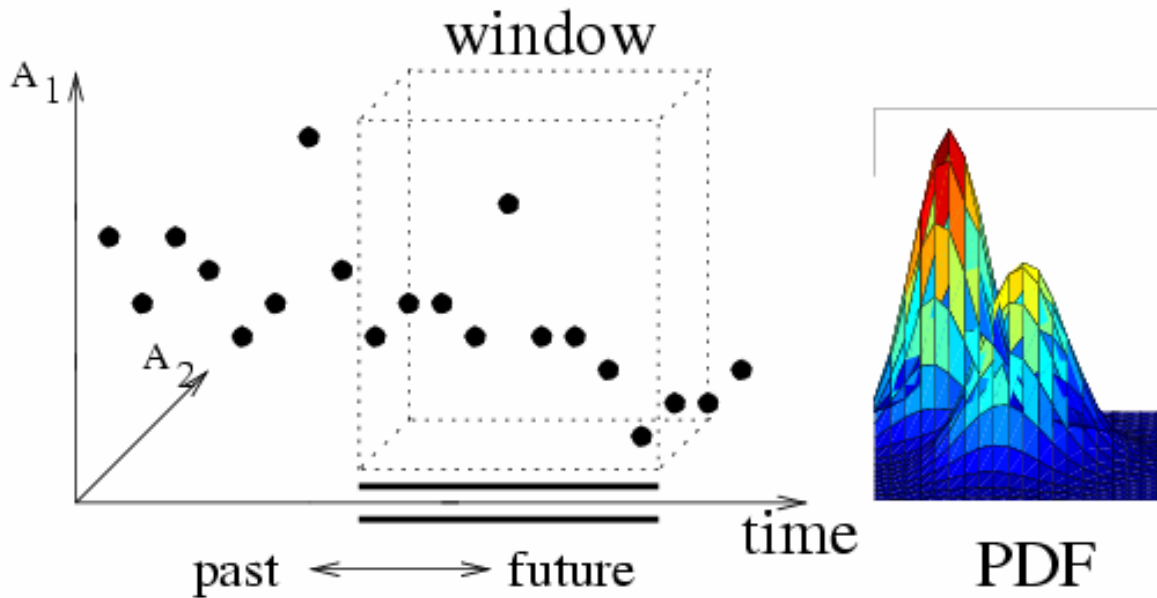
- time  $t_1$





# Input Data Distribution Estimation

- time  $t_2 > t_1$

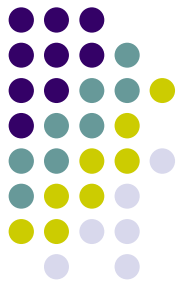


# Our Approach



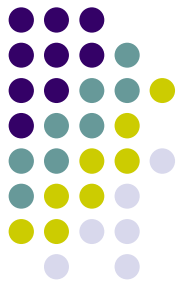
- kernel density estimation
  - model estimation technique
- benefits
  - effectively approximates an unknown data distribution
  - non-parametric
  - efficiently computed in streaming environment
  - adjusts to changes in the input
  - can operate in a distributed fashion

# Kernel Estimation



- kernel estimator
  - generalized form of random sampling
- works as follows
  - sample the data
  - assign a weight to each sample
  - distribute the weight of each sample in its neighborhood
    - according to a *kernel function*





# Kernel Function

- Epanechnikov kernel function
  - generalized form of random sampling

$$k(x) = 3/4B (1 - (x/B)^2), \text{ if } |x/B| < 1, 0 \text{ otherwise}$$

B is the kernel function bandwidth

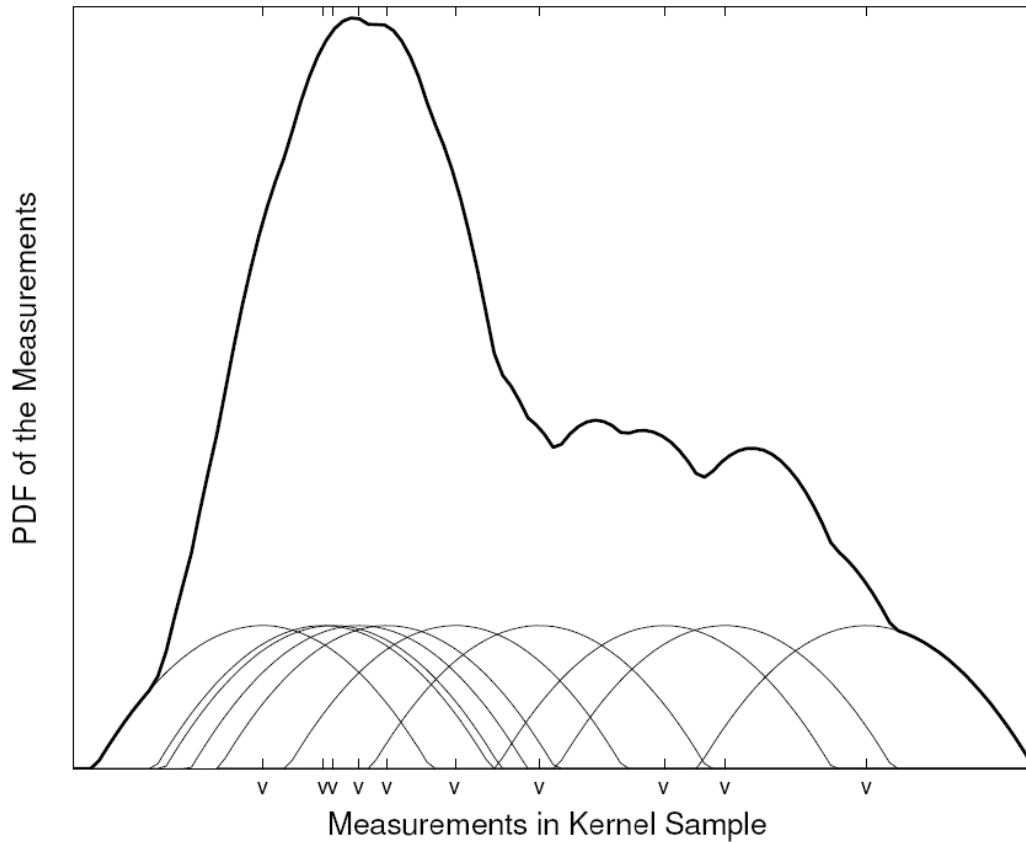
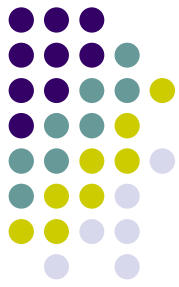
$$B = 5^{1/2}\sigma|R|^{-1/5} \quad (\text{Scott's rule})$$

$\sigma$  standard deviation of points in the dataset

$|R|$  sample size

- easy to integrate
- extends naturally to multiple dimensions

# Kernel Density Estimation: Example

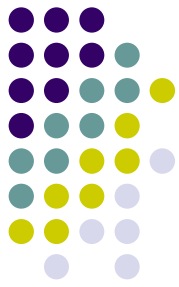




# Kernel Density Estimation

- kernel estimation in a streaming environment (assume sliding window model)
  - compute and maintain online
    - random sample of data
    - standard deviation of data
- random sample
  - chain-sample algorithm produces uniform random sample
- standard deviation
  - concise histogram technique
- both algorithms adapt to shifting input distributions
- both algorithms can operate in a distributed fashion
  - models can be combined

# Online Outlier Detection: Distance-Based Outliers



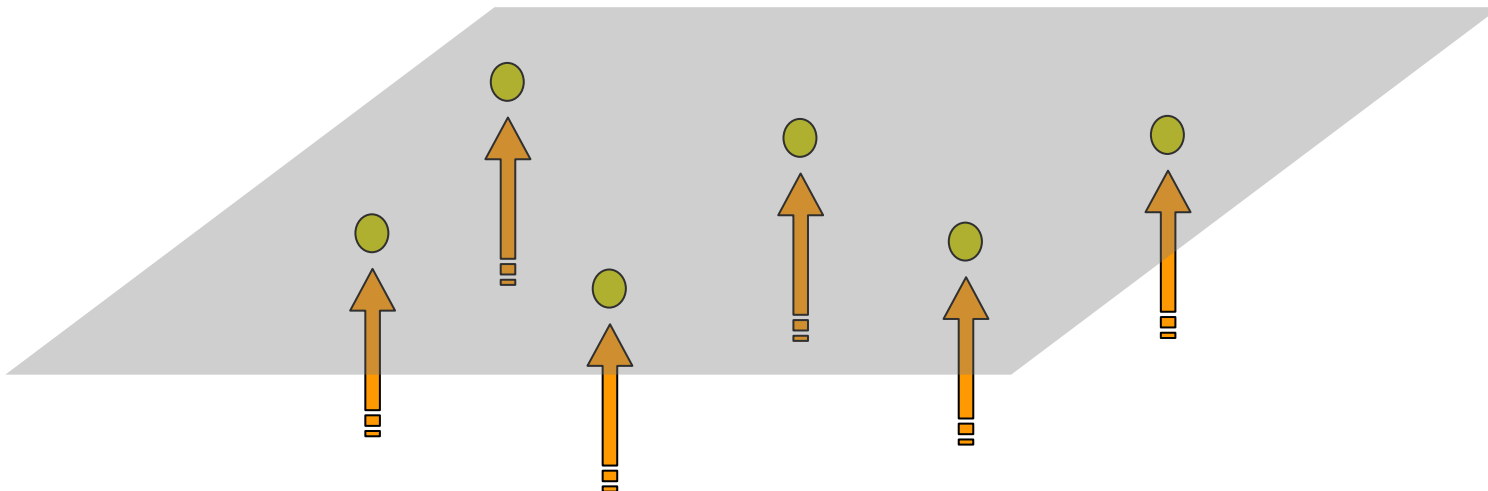
- $O(r, K)$  outliers
  - count the number of points within a circle of radius  $r$

- solution based on kernel density estimation

$$N(p, r) = \int_{[p-r, p+r]} \left( \frac{1}{|T|} \sum_{p_i \in D} \frac{3}{4B} \left( 1 - \left( \frac{x - p_i}{B} \right) \right) \right) dx$$

- estimates the number of neighboring points
- space and time efficient for each sensor  
(space:  $O(d(|R|+1/\epsilon^2 \log |W|))$ , time 1-d:  $O(\log |R| + |R'|)$ , time m-d:  $O(d|R|)$  )

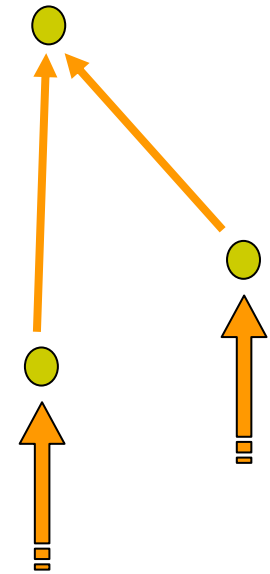
# Online Outlier Detection: Distance-Based Outliers





# Detection of Region Outliers

- identify outliers wrt multiple data streams
- parent has to build a model for the combined data distribution of its children
- possible solution: each sensor in hierarchy has to compute its own sample
- expensive solution!
  - even if sampling only happens at leaf level

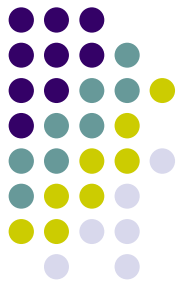


# Distributed Computation of Estimators



- kernel estimator model composition
  - combine random sample and kernel bandwidth of children nodes
    - new random sample is union, possibly followed by downsampling
    - kernel bandwidth estimation based on:  $V_{12} = V_1 + V_2 + N_1 N_2 / N_{12} (\mu_1 - \mu_2)^2$
  - single model describing the behavior of all children nodes
- adapting to shifting data distributions
  - children propagate estimator updates to parent nodes according to:
    - changes in input distribution
      - have to monitor changes, adapt update rate accordingly
        - monitor first moments of distribution, or apply specialized techniques
    - probability that depends on number of children and sample sizes
      - update probability  $f = |R_p| / c|R|$

# Online Distributed Outlier Detection: Distance-Based Outliers



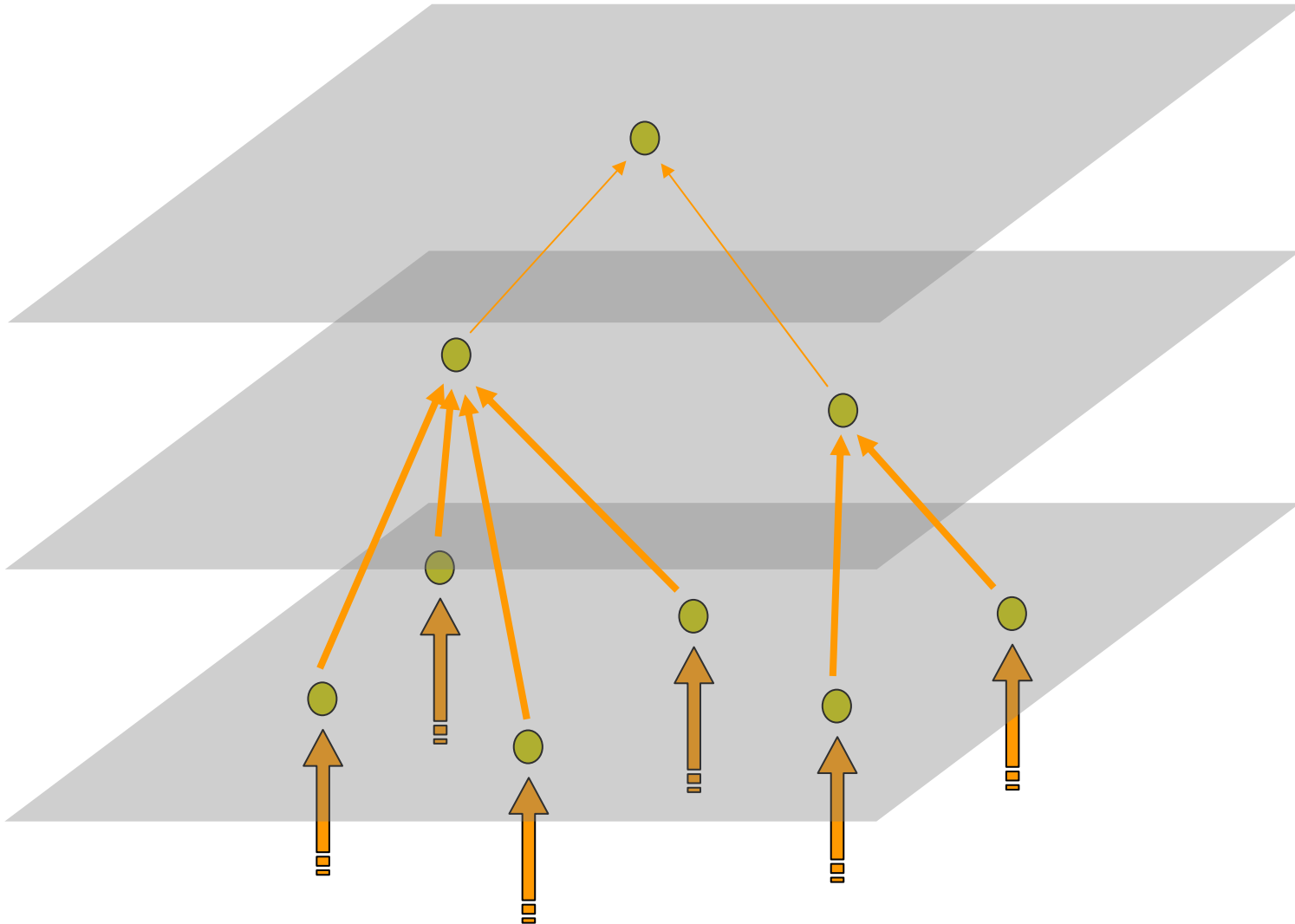
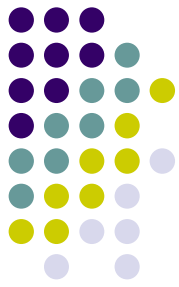
- theorem

*Assume nodes  $n_1, \dots, n_l$  children of node  $n_p$ . Assume data streams  $S_1, \dots, S_l$  referring to the  $l$  children nodes, and corresponding sliding windows  $W_1, \dots, W_l$ . The sliding window of node  $n_p$  is defined as  $W_p = \bigcup_{i=1}^l W_i$ . Let, at some point in time,  $O_1, \dots, O_l$  be the sets of distance based outliers corresponding to each one of the  $l$  sliding windows. Then, for the set  $O_p$  of outliers in  $W_p$  it holds that  $O_p$  subset of  $\bigcup_{i=1}^l O_i$ .*

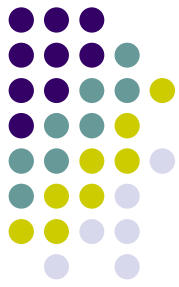
- if a value is an outlier in the combination of two or more streams, then it is an outlier in at least one of those streams
- as we combine streams we can ignore all points that are not outliers



# Online Distributed Outlier Detection: Distance-Based Outliers

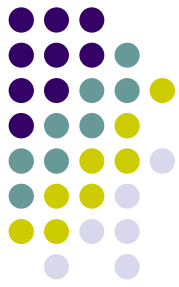


# Online Distributed Outlier Detection: Density-Based Outliers



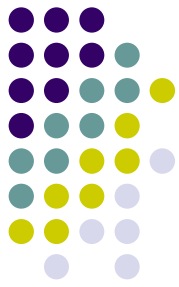
- *MDEF* outliers
  - count the number of near neighbors
  - compare to the average count across the extended neighborhood
    - an outlier at the parent node may not be an outlier at any child node!
  - leaf level nodes report outliers wrt to the values they observe, or wrt to the values of the entire region they belong in
- when combining streams, the children nodes have to know the global distribution
  - parents have to communicate their models to the children
- we apply the following scheme:
  - children update parent models about their changes with probability  $f$
  - when the global model changes, the changes are propagated to all the leaf nodes
    - may reduce communication by propagating only if change is significant ( by computing the distance of the models )

# Experimental Evaluation

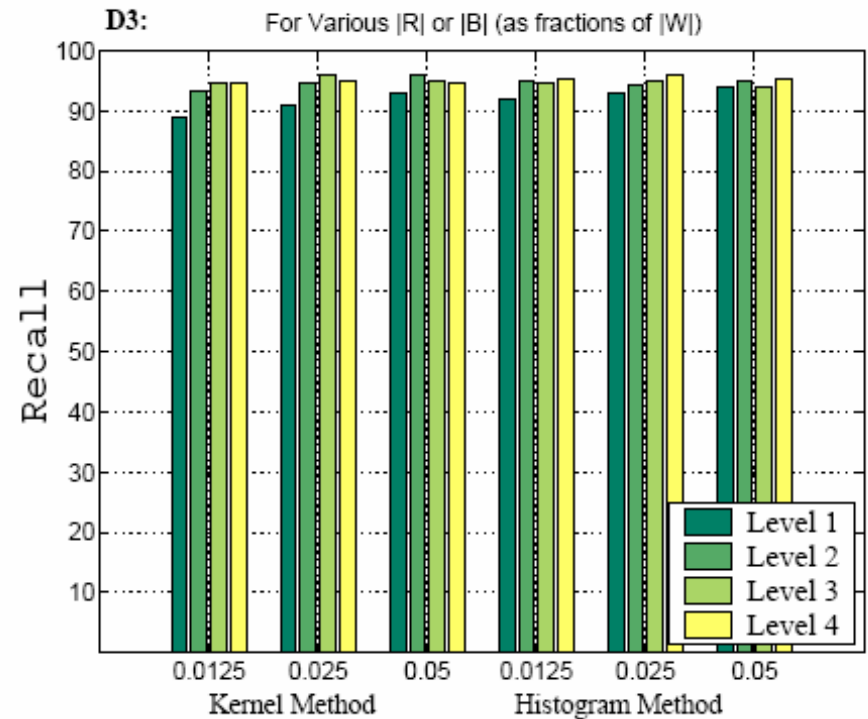
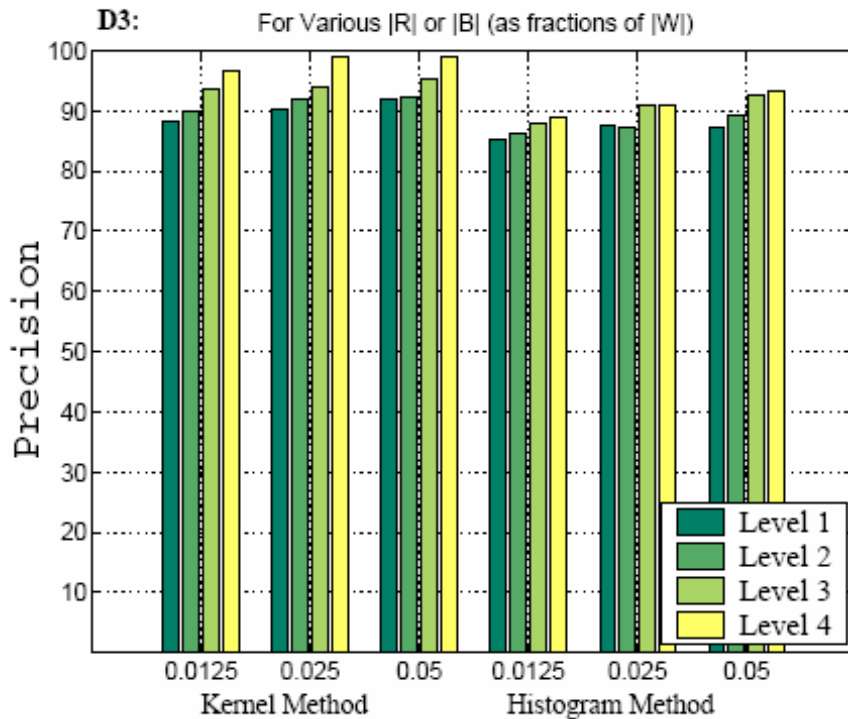


- technique implemented on top of TAG sensor network simulator
  - 5,000 lines of java code
- synthetic datasets
  - mixtures of Gaussians
  - 35,000 observations
  - values normalized to  $[0, 1]$
- real datasets
  - sensor readings from Pacific Northwest region (35,000 observations)
  - engine operation measurements (50,000 observations)
- measured precision and recall (compared to offline algorithm)

# Experimental Results: Accuracy – $O(r, K)$ Outliers



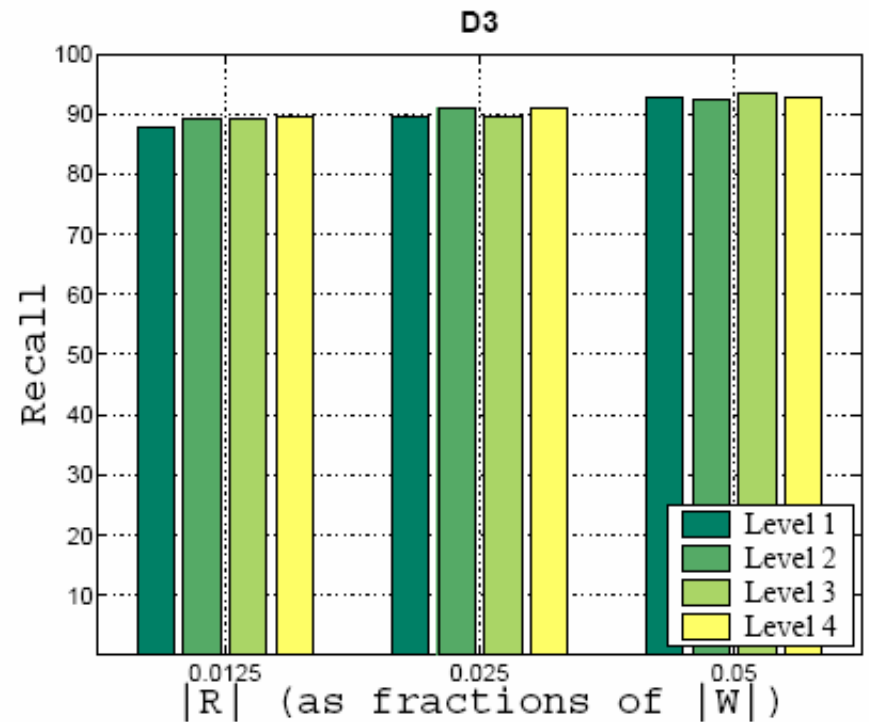
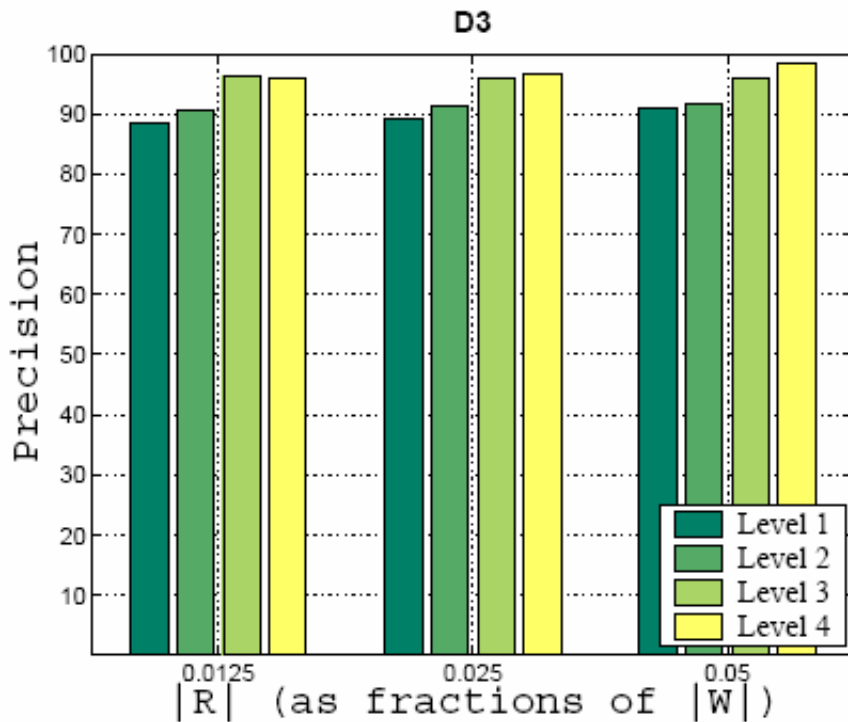
- varying the sample size (available memory), 1-d synthetic data



# Experimental Results: Accuracy – $O(r, K)$ Outliers



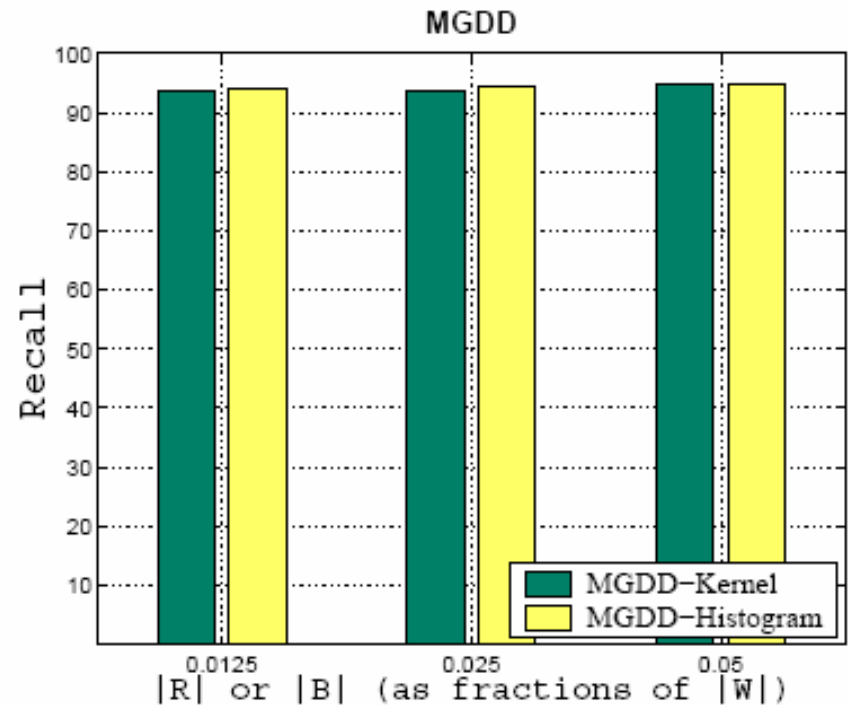
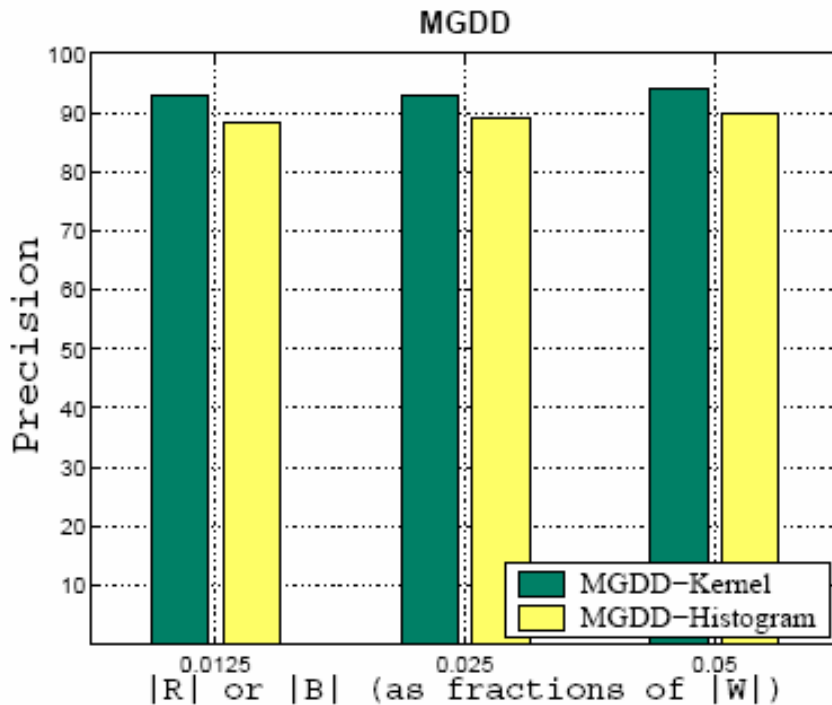
- varying the sample size, 2-d real data



# Experimental Results: Accuracy – *MDEF* Outliers



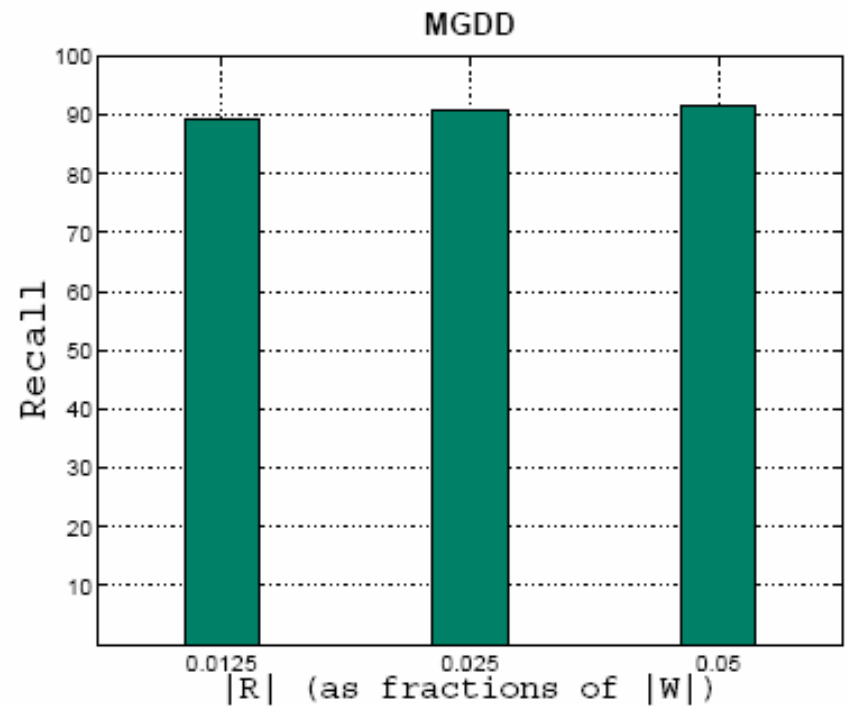
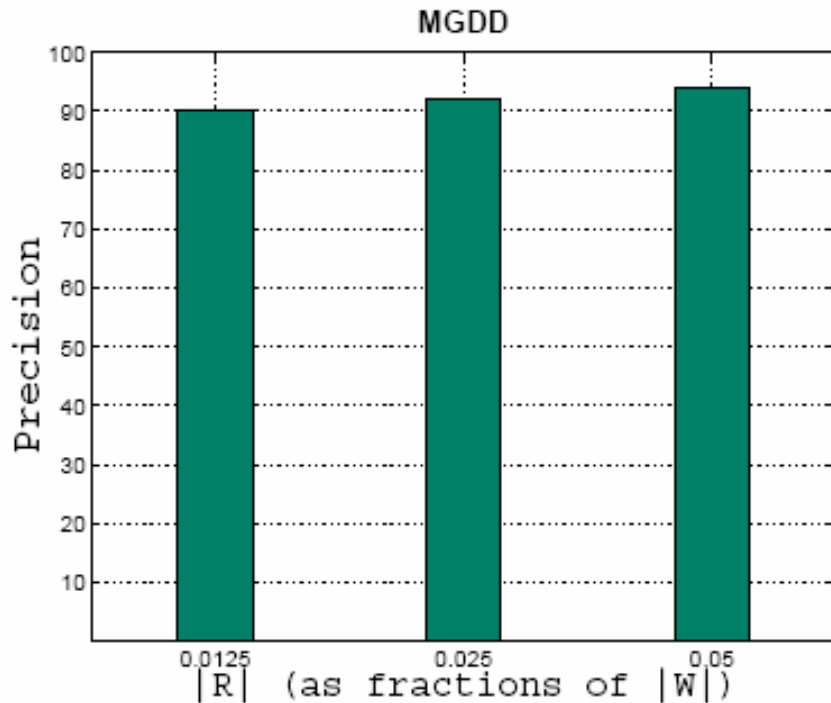
- varying the sample size (available memory), 1-d synthetic data



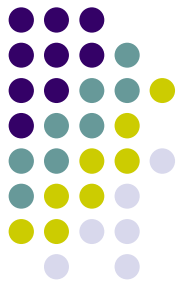
# Experimental Results: Accuracy – *MDEF* Outliers



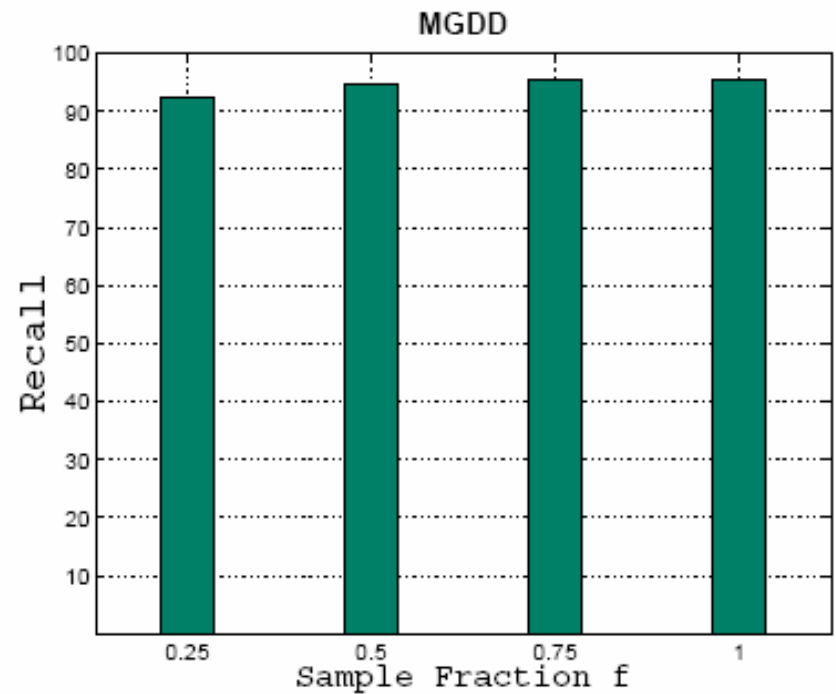
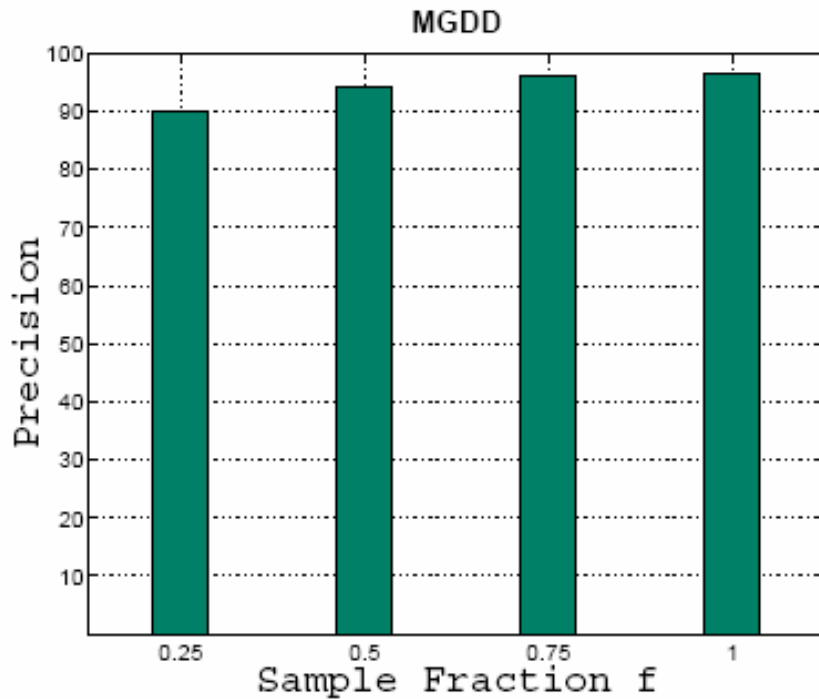
- varying the sample size, 2-d real data



# Experimental Results: Accuracy – *MDEF* Outliers

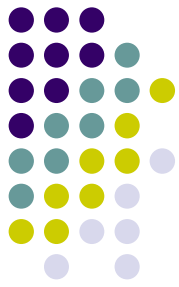


- varying the update probability  $f$ , 1-d synthetic data

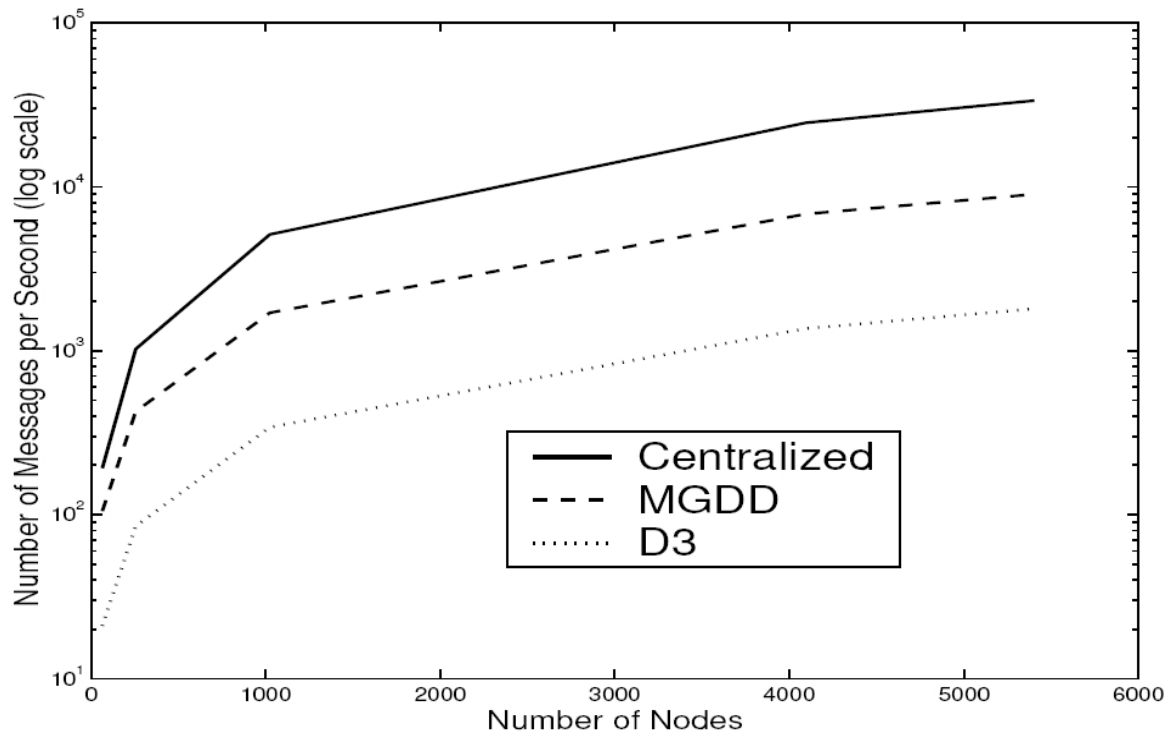


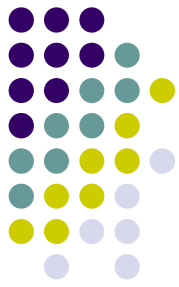


# Experimental Results: Communication Costs



- cost comparison of outlier detection algorithms
  - distance-based *D3*, density-based *MGDD*, centralized approach

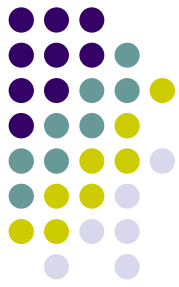




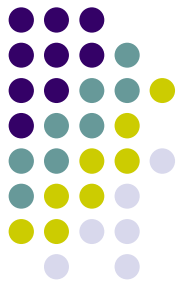
# Related Work

- statistical outliers
  - suppose knowledge of input distribution, offline  
*[Barnet, Lewis'94]*
- outliers in databases
  - offline algorithms  
*[Arning et al'96][Knorr, Ng'98][Papadimitriou et al'03][Breunig et al'00]*  
*[Ramaswamy et al'00]*
- outliers in time series
  - temporal ordering is key  
*[Puttagunta, Kalpakis'02][Muthukrishnan et al'04][Yamanishi et al'04]*
- sensor data processing systems
  - query processing  
*[Madden et al'02][Yao, Gehrke'03][Bonfils, Bonnet'03]*
  - approximate query answering  
*[Deshpande et al'05][Guestrin et al'04][Cormode, Garofalakis'05][Olston et al'03][Jain et al'04]*

# Conclusions



- studied the problem of online outlier detection in sensor networks
- proposed general and flexible data distribution approximation framework
  - does not require a priori knowledge of the input data distribution
  - based on non-parametric model
- described technique for efficient distributed deviation detection
  - focus on the interesting, unexpected events
- validated the proposed approach experimentally



thank you!

Themis Palpanas  
themis@dit.unitn.it