Proc. 26th International Conference on Distributed Computing Systems, July, 2006

In-Network Outlier Detection in Wireless Sensor Networks

Joel Branch and Boleslaw Szymanski
Computer Science
Rensselaer Polytechnic Institute
110 8th Street, Troy, New York 12180
{brancj, szymansk}@cs.rpi.edu

Chris Giannella, Ran Wolff,
and Hillol Kargupta
Computer Science & Electrical Engineering
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
{cgiannel, ranw, hillol}@cs.umbc.edu

Abstract

To address the problem of unsupervised outlier detection in wireless sensor networks, we develop an algorithm that (1) is flexible with respect to the outlier definition, (2) works in-network with a communication load proportional to the outcome, and (3) reveals its outcome to all sensors. We examine the algorithm's performance using simulation with real sensor data streams. Our results demonstrate that the algorithm is accurate and imposes a reasonable communication load and level of power consumption.

1. Introduction

Outlier detection, an essential step preceding most any data analysis routine, is used either to suppress or amplify outliers. The first usage (also known as data cleansing) improves robustness of data analysis. The second usage helps in searching for rare patterns in such domains as fraud analysis, intrusion detection, and web purchase analysis (among others).

Several factors make wireless sensor networks (WSNs) especially prone to outliers. First, they collect their data from the real world using imperfect sensing devices. Second, they are battery powered and thus their performance tends to deteriorate as power is exhausted. Third, since these networks may include a large number of sensors, the chance of error accumulates. Finally, in their usage for security and military purposes, sensors are especially prone to manipulation by adversaries. Hence, it is clear that outlier detection should be an inseparable part of any data processing routine that takes place in WSNs.

Simply put, outliers are events with extremely small probabilities of occurrence. Since the actual generating distribution of the data is usually unknown, direct computation of probabilities is difficult. Hence, outlier detection methods are, by and large, heuristics. Because the problem is fundamental, a huge variety of outlier detection methods have been developed. In this paper we focus on non-parametric, unsupervised methods.

We develop a technique for the computation of outliers in WSNs. The typical WSN environment poses several restrictions on computation: (1) it must be done "in-network" to reduce both bandwidth and energy usage [18], (2) it must be resilient to sensor failure, (3) it must accommodate streaming, or dynamically updated, data. In addition to the above requirements, the algorithm presented here has also the following properties: (1) it is generic – suitable for many outliers detection heuristics, (2) it works in-network with a communication load proportional to the outcome (*i.e.* the number of outliers reported), (3) it is robust with respect to data and network change, (4) the outcome is revealed to all of the sensors.

We exemplify the benefits of our algorithm by implementing it using two different outlier detection heuristics and simulating 53 sensors using the SENSE sensor network simulator [13] with real sensor data streams. Our results show that the algorithm converges to an accurate result with reasonable communication load and power consumption. In most tested cases, our algorithm's performance bests that of a centralized approach.

2. Related work

2.1. Outlier detection

Outlier detection is a long studied problem in data analysis; hence, we provide only a brief sampling of the field.

Hodge and Austin [19] present a survey focusing on outlier detection methodologies based on machine learning and data mining. These include distance and density-based unsupervised methods, feed-forward neural networks and decision tree-based supervised methods, and auto-associative neural network and Hopfield network-based methods. Barnett and Lewis [6] provide a survey of outlier detection methodologies in the statistics community.

Our algorithm is flexible in that it accommodates a whole class of unsupervised outlier detection techniques such as (1) distance to k^{th} nearest neighbor [26], (2) average distance to the k nearest neighbors [4], and (3) the inverse of the number of neighbors within a distance α [22] (see Section 3 for details).

2.2. Wireless sensor networks

WSNs combine the capability to sense, compute, and coordinate their activities with the ability to communicate results to the outside world. They are revolutionizing data collection in all kinds of environments. At the same time, the design and deployment of these networks creates unique research and engineering challenges due to their expected massive size (up to thousands of sensor nodes), their often random and hazardous deployments, obstacles to their communication, their limited power supply, and their high failure rate.

The software for WSNs needs to be aware of their limitations and features. The most important among these are limited power, high communication cost, and limited direct communication range. In [17], Estrin *et al.* introduce scalable coordination as an important component of the needed software. A survey of the state-of-the-art in WSNs is given in in [3]. Another survey [2] focuses on challenges arising from specific applications such as military, health care, ecology, and security.

Energy-efficiency, a cardinal WSN requirement, is often achieved by minimizing communication using topology-control algorithms that dictate the active/sleep cycles of sensor nodes. Examples include Geographic Adaptive Fidelity (GAF) [31], ASCENT [11], STEM [27], and ESCORT [9]. While the focus of this paper is on WSN outlier detection, the challenge is the same as in the above mentioned works. Hence, while we do not propose a topology-control algorithm, we aim to design an energy-efficient algorithm by minimizing the required communication overhead.

In [25], Palpanas *et al.* also propose a distributed WSN outlier detection framework. However, in addition to normal sensor nodes, their algorithm assumes the availability of high capacity nodes necessary for finding all global outliers. Furthermore, only the high capacity nodes will know all global outliers at the algoritm's completion. Our algorthm is non-hierarchical and all sensors know all global outliers when the current epoch converges.

2.3. Data mining in large-scale dynamic networks

Very recently, researchers have started to consider data analysis in large-scale dynamic networks. The goal is to develop techniques that are highly asynchronous, scalable, and robust to network changes. Efficient data analysis algorithms often rely on efficient primitives, so researchers have developed several different approaches to computing basic operations (*e.g.* average, sum, max, or random sampling) on dynamic networks. Kempe *et al.* [21] and Boyd *et al.* [8] investigate gossip based randomized algorithms. Jelasity and Eiben [23] develop the "newscast model" as part of the DREAM project [28]. Both of the above approaches use an epidemic model of computation. Bawa *et al.* [7] have developed an approach in which similar primitives are evaluated to within an error margin. Wolff *et al.* [30] develop a local algorithm for majority voting. Finally, some work has gone into more complex data mining tasks: association rule mining [30], facility location [24] (both based on local majority voting), genetic algorithms [14], and k-means clustering [16, 29].

3. Preliminaries

In this section, we provide necessary background definitions and notations.

A distributed system architecture is a system of peers, p_i, each holding a set S_i composed of m_i points from $\mathbb{D}$. Each peer knows $\mathcal{A}$, an outlier detection algorithm, and R, an outlier ranking function. Peers communicate by exchanging messages over a connected graph. We assume the graph is undirected, messages are reliable, and each peer p_i can accurately maintain the list of its immediate neighbors, N_i, in the graph. In other words, our algorithm works as long as there exists, possibly unknown, a reliable path from each peer to every other peer.

An outlier detection algorithm $\mathcal{A}$ takes a finite set of points $P \subseteq \mathbb{D}$ and an outlier ranking function $R : \mathbb{D} \times 2^{\mathbb{D}} \rightarrow \mathbb{R}^+$ and returns the top n outliers, denoted $\mathcal{A}[P]$ (n is a user-defined parameter). If $n > |P|$, then $\mathcal{A}[P]$ returns P. We make no assumptions about R except that it satisfies the following two axioms. Given $x \in \mathbb{D}$, for all finite $P_1 \subseteq P_2 \subseteq \mathbb{D}$:

- (Anti-monotonicity) $R(x, P_1) \geq R(x, P_2)$;
- (Smoothness) if $R(x, P_1) > R(x, P_2)$, then there exists $z \in P_2 \setminus P_1$, such that $R(x, P_1) > R(x, P_1 \cup \{z\})$.

The first axiom is similar to the *Apriori rule* in frequent itemset mining [1]. The second axiom, intuitively, states that R changes gradually. As more points are added to P_1, the rating function changes gradually to $R(x, P_2)$. Some example outlier rating functions which satisfy these axioms include: the distance to the k^{th} nearest neighbor, the average distance to the k nearest neighbors, and the inverse of the population of an α neighborhood of x. However, some previously proposed rating functions do not satisfy these axioms (*e.g.* LOF [10]).

To break ties, we assume there exists a fixed but arbitrary total ordering, $\prec$, on $\mathbb{D}$. Hence $\mathbb{D}$ is totally ordered with respect to R and P as follows, $x \prec_{R,P} y$ if (i) $R(x,P) < R(y,P)$ or (ii) $R(x,P) = R(y,P)$ and $x \prec y$. Formally, $\mathcal{A}$, given P, returns

$$\begin{aligned}\mathcal{A}[P] &= \{x_1, \ldots, x_n \in P : \forall 1 \leq i \leq n \\ &\quad \text{and } y \in P \setminus \{x_1, \ldots, x_n\}, y \prec_{R,P} x_i\}.\end{aligned}$$

Given R, a set $P_0 \subseteq P$ is called a *support set of* $x \in \mathbb{D}$ *over* P if $R(x,P) = R(x,P_0)$. Note, a unique smallest support set need not exist. To break ties, we use $\prec$ to define a total ordering on the finite subsets of $\mathbb{D}$ as follows. Given P_1, P_2 finite subsets of $\mathbb{D}$, we define $P_1 \prec_{fin} P_2$ if (i) $|P_1| < |P_2|$ or (ii) $|P_1| = |P_2|$ and P_1 is strictly lexicographically smaller than P_2 with respect to $\prec$ (denoted $P_1 \prec P_2$). Since P is finite, then there exists a unique $\prec_{fin}$-smallest support set of x over P – let $[P|x]$ denote this set. Finally, given $Q \subseteq P$, we write $[P|Q]$ to denote $\bigcup_{x \in Q} [P|x]$.

4. Distributed outlier detection

In this section, we describe a distributed algorithm by which peers compute $\mathcal{A}\left[\bigcup_i S_i\right]$. The algorithm finds the outliers among the global dataset (the union of all peers' local datasets).

4.1 The algorithm

The peers will communicate by sending messages which include a set of data points describing sensor samplings. Each peer p_i will maintain for every neighbor $p_j \in N_i$ the set of points it has sent to p_j, $S_{i,j}$, and the set of points it received from p_j, $S_{j,i}$. We define the *knowledge* of p_i as $\bar{S}_i = S_i \cup \bigcup_{p_j \in N_i} S_{j,i}$. The algorithm is event based and employs the same logic once upon initialization and then again whenever $\bar{S}_i$ changes as a result of receiving a message, of a change to S_i, or of changes in N_i.

Whenever the algorithm is called, p_i invokes $\mathcal{A}$ and computes $A = \mathcal{A}\left[\bar{S}_i\right]$, $SA = [\bar{S}_i|\mathcal{A}[\bar{S}_i]]$. Now, for each neighbor $p_j \in N_i$, p_i must check if it has new information that p_j may not have but need. First of all, any of p_i's current outliers and their supports (A, SA) may be needed by p_j

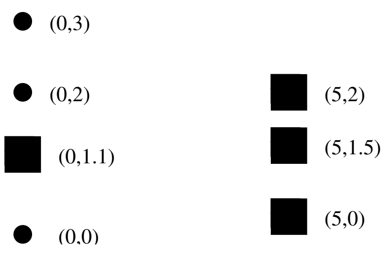


Figure 1. Two peers' datasets. p_1 holds the circles and p_2 holds the squares and each data item defines Cartesian coordinate of the center of the object.

since they could cause p_j to update its own outliers. If, for any of these points x, p_i cannot be certain that p_j has x (*i.e.* $x \notin S_{j,i}$), then x must be added to $S_{i,j}$.

Second, p_i may have points which would effect outliers previously sent by p_j, but these may not be accounted for in the first part (*i.e.* may not be in A or SA). It suffices for p_i to send the support of all of the outliers in $S_{i,j} \cup S_{j,i}$. Any of these points not in $S_{j,i}$ must be added to $S_{i,j}$. Therefore $S_{i,j}$ must be a minimal fixed-point of the following equation with S initially containing $(A \cup SA \cup S_{i,j}) \setminus S_{j,i}$:

$$S = S \cup ([\bar{S}_i|\mathcal{A}[S \cup S_{j,i}]] \setminus S_{j,i}). \tag{1}$$

If the fixed-point is not contained in $S_{i,j}$ (*i.e.* there are potentially points p_j has not yet seen), then these extra points are sent to p_j via a packet broadcast.

Example: Assume $R(x,S)$ is defined as the distance to $x's$ nearest neighbor in S ($k = 1$) and $\mathcal{A}[S]$ is the top rated outlier in S ($n = 1$). Consider the two peer datasets in Figure 1 (p_1 has circles, p_2 has boxes). Observe that the global outlier is $(5,0)$ since the distance to its nearest neighbor is larger than that of every other point. In this example, we assume the peers carry out the algorithm in alternating order (of course, in real use, the peers operate asynchronously). Initially $S_{1,2}$ and $S_{2,1}$ are empty.

p_1 will compute $A = \mathcal{A}[\bar{S_1}] = \{(0,0)\}$ and $SA = [\bar{S_1}|A] = \{(0,2)\}$. Then it computes the fixed-point. S is set to $A \cup SA$. Observe that $[\bar{S_1}|\mathcal{A}[S \cup S_{2,1}]] = [\bar{S_1}|\mathcal{A}[S]] = [\bar{S_1}|(0,0)] = \{(0,2)\}$. Since this is already in S, then the fixed-point computation is complete, $S = \{(0,0),(0,2)\}$. $S_{1,2}$ is set to $S \setminus S_{2,1} = S$ and sent to p_2.

Observe, at this point, p_1 mistakenly assumes the global outlier to be $A = \{(0,0)\}$.

p_2 receives $S_{1,2}$, thus, $\bar{S_2}$={(0,0),(0,1.1),(0,2),(5,0),(5,1.5),(5,2)}. It computes A=$\mathcal{A}[\bar{S_2}]$={(5,0)} and SA=$[\bar{S_2}|A]$=$\{(5,1.5)\}$. Note, if p_2 were to send only these points, p_1 would not change its mistaken belief that the global outlier is $(0,0)$. The fixed-point computation is needed.

So, S is set to $(A \cup SA) \setminus S_{1,2} = \{(5,0),(5,1.5)\}$. Observe that $[\bar{S}_2|\mathcal{A}[S \cup S_{1,2}]] = [\bar{S}_2|(0,0)] = \{(0,1.1)\}$. Thus, S becomes $\{(0,1.1),(5,0),(5,1.5)\}$. It can be seen that this is the fixed-point, so, $S_{2,1}$ is set to $S \setminus S_{1,2} = S$ which is sent to p_1.

p_1 receives $S_{2,1}$, thus $\bar{S}_1$ becomes {(0,0),(0,1.1),(0,2),(0,3),(5,0),(5,1.5)}. Now p_1 will change its global outlier belief (because of the presence of point $(0,1.1)$) to $A = \{(5,0)\}$. It can be seen that the fixed-point will be contained in $S_{1,2}$, so, p_1 sends nothing to p_2.

Both p_1 and p_2 have the same (correct) global outlier belief, $(5,0)$. This example illustrates the role of both types of information described above.

□

It is easy to modify the algorithm to work in a streaming setting: when a new point is sampled, S_i, and consequently, $\bar{S}_i$ change. This requires that the same calculation is made as in the case of a change in $\bar{S}_i$ due to receiving a message. If the algorithm needs to only consider points which were sampled recently (*i.e.* employ a sliding window), this can be implemented by adding a time-stamp to each point when it is sampled. Under the assumption that the clocks of different nodes are synchronized to a degree satisfying the needs of the application, each node can retire old points regardless of where they were sampled and at no communication cost at all.

The pseudo-code of the algorithm is given in Algorithm 1 – the "do-until" loop is responsible for computing the fixed-point of Equation (1). The algorithm assumes a sliding window mode of work. The algorithm also assumes that the addition of sensors during system operation is possible. However, if sensors are removed (*e.g.* when their battery is depleted) then their contribution to the computation is not explicitly annulled until those points are retired with time. It is easy to bypass the sliding window mechanism by setting τ to infinity. Yet, in that case, it is reasonable to dictate that points contributed by nodes which were removed should be explicitly removed, at a messaging cost. Note that in the "for" loop, the algorithm stores a list of all neighbors that should receive an update, appends the list to the packet M, and then broadcasts M afterwards. All of p_i's immediate neighbors will receive M and only those whose IDs are in the included neighbor list will process the packet. This drastically reduces the number of packet transmissions, making the algorithm more energy-efficient.

Algorithm 1 Global Outliers Detection

```
Input of $p_i$: $S_i$, $N_i$, $\mathcal{A}$, $\tau$
Output of $p_i$: $\mathcal{A}[\bar{S}_i]$ and $[\bar{S}_i|\mathcal{A}[\bar{S}_i]]$
Upon receiving $ADD\ M$ such that $M = \{(k_1, Q_{k_1}), \ldots\}$ from $p_j$:
if some $k_\ell = i$ set $S_{j,i} \leftarrow S_{j,i} \cup Q_{k_\ell}$
Upon addition of $p_j$ to $N_i$:
set $S_{i,j}$ and $S_{j,i}$ to $\emptyset$
Upon any change in $\bar{S}_i$, $N_i$:
retire points older than $\tau$ from $\bar{S}_i$ and $S_{i,j}$ and $S_{j,i}$ for all $p_j \in N_i$
set $A \leftarrow \mathcal{A}[\bar{S}_i]$ and $SA \leftarrow [\bar{S}_i|\mathcal{A}[\bar{S}_i]]$
let $M$ be an empty message.
for all $p_j \in N_i$
– set $S \leftarrow (A \cup SA \cup S_{i,j}) \setminus S_{j,i}$
– do
– – set $S \leftarrow S \cup ([\bar{S}_i|\mathcal{A}[S \cup S_{j,i}]] \setminus S_{j,i})$
– until no change in $S$
– if $S \nsubseteq S_{i,j}$
– – append $(p_j, S \setminus S_{i,j})$ to $M$
– – set $S_{i,j} \leftarrow S_{i,j} \cup S$
if $M$ is not empty broadcast $ADD\ M$
```

4.2. Correctness

The correctness of the algorithm can be proven in the following sense: if the data and network remain static, then communication will eventually stop at which point all peers' outlier belief will equal $\mathcal{A}[\bigcup_i S_i]$ (the correct global set of outliers). Note that the algorithm does not require that the data be static. It can handle dynamic or streaming data. Naturally, the correctness proof only holds if the data remains static long enough for convergence to occur.

The proof proceeds in two steps. First, barring data or network change, it can be shown that the algorithm does terminate, and, at this point, all nodes have the same outlier beliefs and support (Theorem 4.1). Next, it can be proven that the consistent outlier belief shared by all peers is indeed the correct one (Theorem 4.2).

Theorem 4.1. *If for all sites p_i, S_i and N_i do not change, then the algorithm will terminate and all sites will agree on their outliers and supports in the sense that: for all p_i, p_j, $\mathcal{A}[\bar{S}_i] = \mathcal{A}[\bar{S}_j]$ and $[\bar{S}_i|\mathcal{A}[\bar{S}_i]] = [\bar{S}_j|\mathcal{A}[\bar{S}_j]]$.*

The proof, omitted here for lack of space, first shows that $\mathcal{A}[\bar{S}_i] = \mathcal{A}[S_{i,j} \cup S_{j,i}] = \mathcal{A}[\bar{S}_j]$. Then, it demonstrates that $[\bar{S}_i|\mathcal{A}[\bar{S}_i]] = [\bar{S}_j|\mathcal{A}[\bar{S}_j]]$ from which the theorem follows.

Theorem 4.2. *If for all sites p_i, S_i and N_i does not change, then the algorithm will terminate and all sites will produce the globally correct outliers, i.e. for all p_i, $\mathcal{A}[\bar{S}_i] = \mathcal{A}[\bigcup_k S_k]$.*

The proof, again omitted for the lack of space, shows by contradiction that $\mathcal{A}[\bar{S}_1] = \mathcal{A}[\bigcup_k S_k]$.

Comments: (1) The proof of Theorem 4.1 does not use the smoothness axiom. Hence, for any anti-monotonic R, Theorem 4.1 holds, *i.e.* the algorithm converges and, at that point, all peers will agree on their outlier belief and their support. However, without the smoothness axiom, Theorem 4.2 does not hold, *i.e.* the consistent outlier belief might not be the correct one. There are counter-examples which show how an anti-monotonic, but not smooth R cause the algorithm to terminate with all peers agreeing upon an incorrect set of outliers.

(2) In general, it is not clear how to efficiently compute the minimum support set of a point x over a set P. We do not address the issue in this paper. However, efficient computation is straight-forward for the following rating functions that we consider in experiments, *distance to nearest neighbor* and *average distance to the* k^{th} *nearest neighbor*.

5. Evaluation

5.1 Experimentation setup

We used simulation to measure the average total values of the following metrics: (1) energy consumed per node for transmitting and receiving network packets (measured in Joules), and (2) number of data points transmitted by the algorithm per node. We also collected data on the number of $packets$ transmitted, but did not report it because the number of transmitted data points is a more dominant factor affecting energy consumption. Recall that using our scheme, one packet broadcast may replace that of many.

We compared the algorithm's results against two separate performance baselines. One, we implemented a purely centralized global outlier detection algorithm, in which all nodes periodically sent their sliding window contents to a designated *fusion* node, which then calculated the global outliers and flooded the results out to all nodes in the network. This occurred at the same frequency at which the distributed algorithm was executed. Two, we measured the energy consumption of the network in a strictly idle state. The comparisons (where applicable) are shown in the graphs in this section.

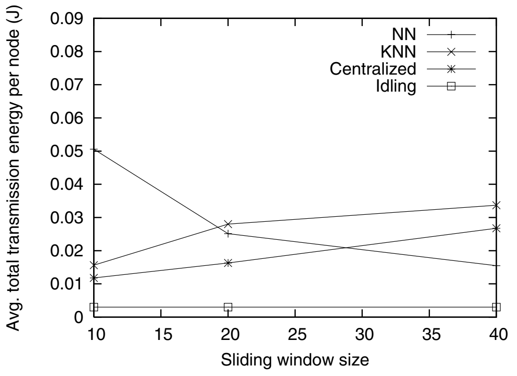
For experimentation, we used real-world sensor data streams from [20], in which distributed data points share spatial and temporal properties. The data we used was comprised of sensor readings (*e.g.* heat, light, temperature) from 53 sensors which were periodically transmitted to a base station. Missing data points were filled by the average values of the data points within a sliding window before the missing point as we believe that the majority of these points resulted from packets dropped in transit to the base station and not by faulty sensor components. The data points include the following features: (1) ID of the sensor that produced the point, (2) epoch (sequential number denoting the data points position in the entire stream), (3) data value (temperature), (4) location coordinates of the sensor. The energy model was based on the Crossbow mote specifications [15] and used a transmit/receive/idle power setting of 0.0159mW/0.021mW/3e-6mW, respectively (assuming a 3V power source).

We tested our algorithm using outliers defined by both *distance to nearest neighbor* and *average distance to k nearest neighbors* using the SENSE wireless sensor network simulator [13]. We simulated a 53-node network with sensor node placed according to specification in [20]. This resulted in a network testbed size of about 50m by 50m. We used the free-space signal propagation model and the fault-tolerant Self-Selective Routing (SSR) protocol [12] in the networking layer. The nodes were configured to have a transmission radius of 6.77m, to evaluate the algorithm in a true distributed setting. However, for the centralized version of the algorithm, we used radius of 16.23m for the following reason. Wireless routing protocols, including SSR, experience extensive collisions and delays in a multi-hop setting when the network density is high. In our application, the distributed version uses one-hop communications (*i.e.* nodes communicate only with neighbors) whereas the centralized version relies on multi-hop communication for nodes to reach the fusion node. Hence, with equal transmission ranges, routing difficulties prevented the centralized algorithm from matching the accuracy of the distributed algorithm. Therefore, we measured the results for centralized algorithm using larger transmission range that required at most a few hops from every node to the fusion node.

All experiments were run for 1000 seconds of simulated time. As shown in the following graphs, we collected performance results for different algorithm parameter values of (1) the length of the node's sliding window, *w*, (2) the number of outliers to be reported, *n*, and (3) the number of neighbors used in the distance-based outlier detection routines, *k*. The labeling of the data in the graphs is as follows: (1) NN for results using *distance to nearest neighbor* outlier detection with the distributed algorithm, (2) KNN for results using *average distance to k nearest neighbors* outlier detection with the distributed algorithm, (3) Centralized for results with the centralized algorithm, (4) Idling for energy use with the network idling.

The energy consumption at reception was by far the dominant term in energy use, so we did not include *total energy* graphs as they are nearly identical to receiving energy graphs.

Only one set of the centralized results is presented in each graph, as *distance to nearest neighbor* and *average distance to k nearest neighbors* outlier detection yielded the same results for the centralized approach. It should also be noted that the results generated by our algorithm were highly accurate. Node's reported the correct outliers 99% of the time. We believe that packet losses were the cause of any incorrect results.



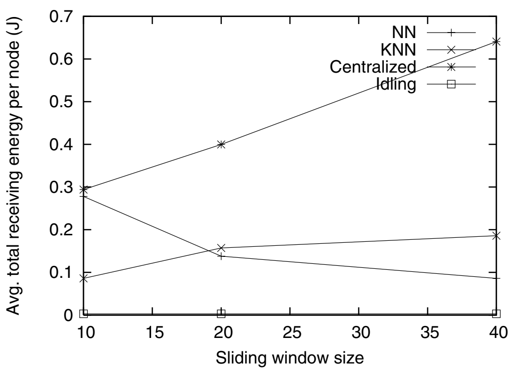


Figure 2. Transmitting and receiving energy consumed per node vs. w (n=4, k=4)

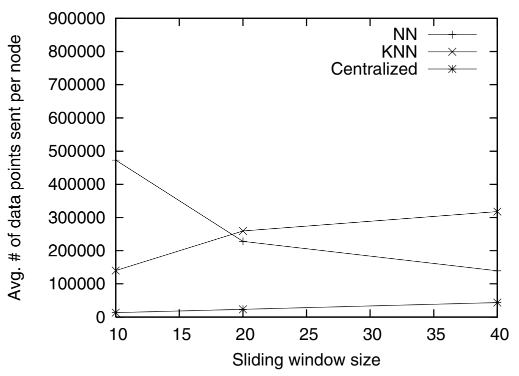


Figure 3. Average number of points sent per node vs. w (n=4, k=4)

5.2 Experimentation results

The sliding window size: As Figure 2 shows, NN is the most energy-efficient for large window sizes. When the window size grows, the number of new outliers communicated from from round to round decreases in NN because of larger number of redundant values amongst the data points. The opposite is true for KNN because multiple supporting points *per reported outlier* are transmitted by the algorithm. Under the centralized version of the algorithm, as w grows, nodes must send the entire contents of their sliding windows to a fusion node for outlier detection, so the energy use grows.

Our algorithm's decreasing energy use with larger window sizes promotes running the outlier detection with large sliding window. Such runs uncover the level of "outlierness" of a data point within a varying scope of other data points in the network. A centralized approach clearly does not support such runs.

It is interesting to see in Figure 3 that the centralized version performs better than the distributed versions in terms of transmitted points, even though the distributed versions conserve more energy. This is because the difference in transmission radii between the two algorithms. With the larger transmission radius required by the centralized version, over-listening by each node to messages addressed to other nodes also increased. We also note that the receiving energy is directly proportional to the number of points sent by each node (with different proportionality factor for each algorithm), so we omit the graphs showing the average number of points per node from further discussion.

The number of reported outliers: Network performance under our algorithm is largely affected by the number of outliers to be reported which define the number of points transmitted per node. This is true for both NN and KNN. In Figure 4, both NN and KNN yield better results than the centralized algorithm up to certain thresholds, above which NN starts to drain the most energy from the network. At this point, the effect of the degree of data point transmissions in NN is greater than that of over-listening in centralized algorithm.

What is interesting is that as n increases, KNN starts to yield better network performance than NN. There are no clear explanations for this particular behavior. One might expect that since NN uses only one supporting point per outlier, while KNN uses four supporting points, NN should be more efficient. However, we must remember that it is possible for NN and KNN to yield different sets of outliers. For the examples illustrated in Figure 4, it is highly likely that KNN calculated groups of outliers such that a significant number of the supporting points for those outliers (within a given round) overlapped. The effect of this behavior, regarding data point transmission overhead, was probably much softer than the behavior that occurred in NN, where a significant number of redundant points were most likely not encountered.

From this test, we conclude that KNN yielded the most efficient results for the given range of n so the performance may not strictly rely on the values of the algorithmic param-

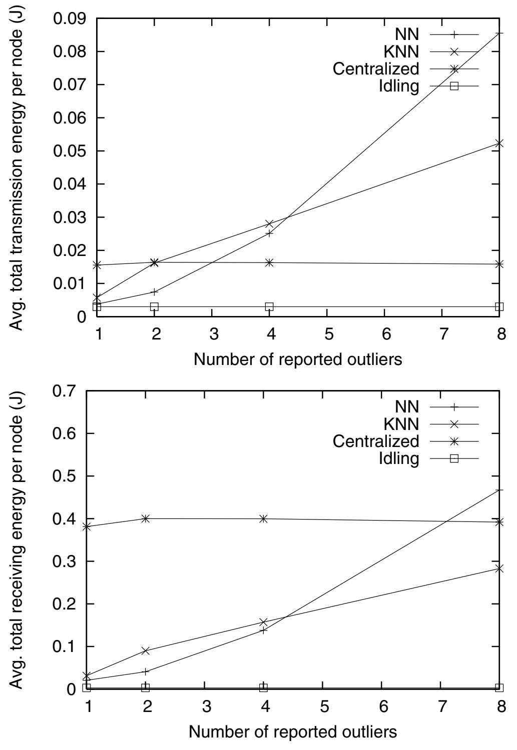


Figure 4. Transmission and receiving energy consumed per node vs. n (w=20, k=4)

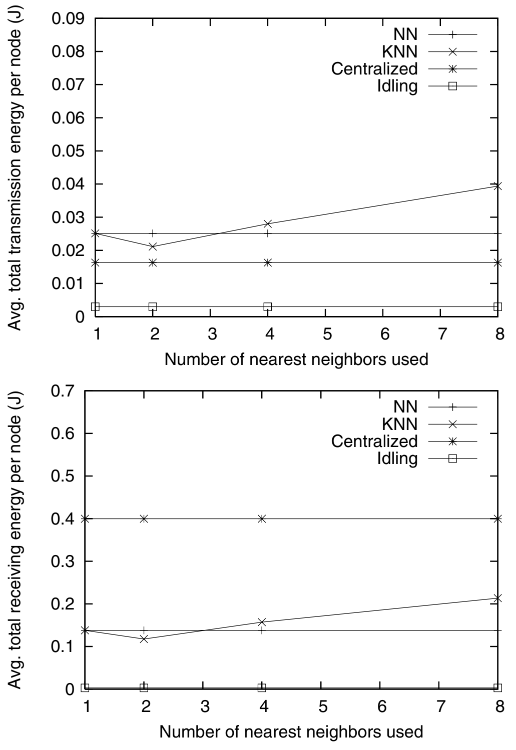


Figure 5. Transmission and receiving energy consumed per node vs. k (w=20, n=4)

eters, but on the nature of the data itself as well.

The number of nearest neighbors used for outlier detection: Amongst all of the parameters discussed in these experiments, k impacts the average node's behavior the least (all other parameters being equal). This is expected for NN and centralized versions of the algorithm, since k does not affect the number of transmitted points for these versions. As previously mentioned, for NN, only one supporting point per outlier is used at all times and for the centralized algorithm, supporting points are not transmitted at all. Hence, the network's energy use is practically unefected by changes in k for NN and centralized versions. Over-listening in the centralized versions still results in the largest energy use among all three versions, as shown in Figure 5. While KNN is more efficient than the centralized versions of the algorithm, it falls behind NN as k increases.

To further qualify these results, using KNN is beneficial because it allows flexibility in determining the confidence of an outlier by using more points to determine an outlier. For the range of k values shown in the graphs, our algorithm bests the performance of the centralized version, especially for higher k values. Depending on the application and available hardware resources, the small reduction in performance of KNN over NN might be worth the burden.

6 Conclusions

We addressed the problem of unsupervised outlier detection in WSNs. We developed a solution that (1) allows flexibility in the heuristic used to define outliers, (2) works in-network with communication load proportional to the outcome, (3) is robust with respect to data and network change, and (4) reveals its output to all of the sensors.

We evaluated the outlier detection algorithm's behavior on real-world sensor data using a simulated wireless sensor network. These initial results show promise for our algorithm in that it outperforms a strictly centralized approach under some very important circumstances. Our algorithm is well suited for applications in which the confidence of an outlier rating may be calculated by either an adjustment of sliding window size or the number of neighbors used in a distance-based outlier detection technique. We assert that these applications are critical for resource-constrained sensor networks for various reasons. One reason is that communication is a costly activity motivating the need for only the most accurate data to be transmitted to a client application. Another reason is that emerging safety-critical applications that utilize wireless sensor networks will require the most accurate data, including outliers. This work represents our contribution towards enabling efficient data cleaning solutions for these types of applications.

Acknowledgments

The authors thank the U.S. National Science Foundation for support of Wolff, Giannella, and Kargupta through award IIS-0329143 and CAREER award IIS-0093353 and of Szymanski through award OISE-0334667. The content of this paper does not necessarily reflect the position or policy of the U.S. Government—no official endorsement should be inferred or implied. The authors also thank Samuel Madden at Massachusetts Institute of Technology and the team at the Intel Berkeley Research Lab for generating the sensor data used in this paper and assisting in its use. Kargupta is also affiliated with Agnik, LLC., Columbia, Maryland.

References

[1] Agrawal R., Mannila H., Srikant R., Toivonen H., and Verkamo A. Fast Discovery of Association Rules. *Advances in Knowledge Discovery and Data Mining*, pp.307–328, 1996.

[2] Akyildiz I. F., Su W., Sankarasubramaniam Y., and Cayirci E. A Survey on Sensor Networks. *IEEE Communication Magazine*, pp. 102–114, 2002.

[3] Akyildiz I. F., Su W., Sankarasubramaniam Y., and Cayirci E. Wireless Sensor Networks: a Survey. *IEEE Trans. Systems, Man and Cybernetics* (B) **38**:393–422, 2002.

[4] Angiulli F. and Pizzuti C. Fast Outlier Detection in High Dimensional Spaces. *European Conf. Principles of Data Mining and Knowledge Discovery*, 2002.

[5] Bandyopadhyay S., Giannella C., Maulik U., Kargupta H., Liu K., and Datta S. Clustering Distributed Data Streams in Peer-to-Peer Environments. *Information Sciences*, 2005.

[6] Barnett V. and Lewis T. *Outliers in Statistical Data*. John Wiley & Sons, 1994.

[7] Bawa M., Gionis A., Garcia-Molina H., and Motwani R. The Price of Validity in Dynamic Networks. *ACM SIGMOD Conf. Management of Data*, pp. 515–526, 2004.

[8] Boyd S., Ghosh A., Prabhakar B., and Shah D. Gossip Algorithms: Design, Analysis, and Applications. *IEEE Infocom*, **3**:1653–1664, 2005.

[9] Branch J., Chen G., and Szymanski B. ESCORT: Energy-Efficient Sensor Network Communal Routing Topology Using Signal Quality Metrics. *IEEE Int. Conf. Networking*, pp. 438–448, 2005.

[10] Breunig M., Kriegel H.-P., Ng R., and Sander J. LOF: Identifying Density-Based Local Outliers. *ACM-SIGMOD Conf. Management of Data*, pp. 93–104, 2000.

[11] Cerpa A. and Estrin D. Adaptive Self-Configuring Sensor Network Topologies. *IEEE Infocom*, pp. 1278–1287, 2002.

[12] Chen G., Branch J., and Szymanski B. Self-Selective Routing for Wireless Ad Hoc Networks. *IEEE WiMob*, pp. 55–74, 2005.

[13] Chen G., Branch J., Pflug M., Zhu L., and Szymanski B. In *Advances in Pervasive Computing and Networking*, ch. 13 SENSE: A Wireless Sensor Network Simulator. pp. 249–267. Springer, New York, NY, 2004.

[14] Clemente J., Defago X., and Satou K. Asynchronous Peer-to-Peer Communication for Failure Resilient Distributed Genetic Algorithms. *IASTED Conf. PDCS*, pp. 769–773, 2003.

[15] Crossbow Technology. MPR, MIB User's Manual, see http://www.xbow.com.

[16] Datta S., Giannella C., and Kargupta H. K-Means Clustering over a Large, Dynamic Network. *SIAM Conf. Data Mining*, 2006.

[17] Estrin D., Govindan R., Heidemann J., and Kumar S. Next Century Challenges: Scalable Coordination in Sensor Networks. *ACM MobiCom*, pp. 263–270, 1999.

[18] Gupta P. and Kumar P. R. The Capacity of Wireless Networks. *IEEE Trans. Information Theory*, **46**(2):388–404, 2000.

[19] Hodge V. and Austin J. A Survey of Outlier Detection Methodologies. *Artificial Intelligence Review*, **22**:85–126, 2004.

[20] Intel Berkeley Research Lab. Wireless Sensor Data, see http://db.lcs.mit.edu/labdata/labdata.html.

[21] Kempe D., Dobra A., and Gehrke J. Computing Aggregate Information using Gossip. *IEEE FoCS*, pp. 482–491, 2003.

[22] Knorr E. and Ng R. Algorithms for Mining Distance-Based Outliers in Large Datasets. *VLDB*, pp. 24-27 1998.

[23] Kowalczyk W., Jelasity M., and Eiben A. Towards Data Mining in Large and Fully Distributed Peer-To-Peer Overlay Networks. *BNAIC*, pp. 203–210, 2003.

[24] Krivitski D., Schuster A., and Wolff R. A Local Facility Location Algorithm for Sensor Networks. *DCOSS*, 2005.

[25] Palpanas T., Papadopoulos D., Kalogeraki V., and Gunopulos D. Distributed Deviation Detection in Sensor Networks. *ACM SIGMOD Review* **32**(4):77-82, 2003.

[26] Ramaswamy S., Rastogi R., and Shim K. Efficient Algorithms for Mining Outliers from Large Datasets. *ACM SIGMOD Conf.*, 2000.

[27] Schurgers C., Tsiatsis V., Srivastava M. STEM: Topology Management for Energy Efficient Sensor Networks. *IEEE Aerospace Conf.*, pp. 78–89, 2002.

[28] The DREAM Project, see http://www.dcs.napier.ac.uk/ benp/dream/private.htm.

[29] Wolff R., Bhaduri K., and Kargupta H. Local L2 Thresholding Based Data Mining in Peer-to-Peer Systems. *SIAM Conf. Data Mining*, 2006.

[30] Wolff R. and Schuster A. Association Rule Mining in Peer-to-Peer Systems. *IEEE Trans. Systems, Man and Cybernetics* (B) **34**(6):2426–2438, 2004.

[31] Xu Y., Heidemann J., and Estrin D. Geography-informed Energy Conservation for Ad Hoc Routing. *ACM/IEEE Conf. Mobile Computing and Networking*, pp. 70–84, 2001.