ISSN 2395-8618

Filtering Compromised Environment Sensors Using Autoregressive Hidden Markov Model

Bushra Anjum and Chaman Lal Sabharwal

***Abstract*—We propose a method based on autoregressive hidden Markov models (AR-HMM) for filtering out compromised nodes from a sensor network. We assume that sensors are healthy, self-healing and corrupted whereas each node submits a number of readings. A different AR-HMM (A, B, π) is used to describe each of the three types of nodes. For each node, we train an AR-HMM based on the sensor's readings, and subsequently the B matrices of the trained AR-HMMs are clustered together into two groups: healthy and compromised (both self-healing and corrupted), which permits us to identify the group of healthy sensors. The existing algorithms are centralized and computation intensive. Our approach is a simple, decentralized model to identify compromised nodes at a low computational cost. Simulations using both synthetic and real datasets show greater than 90% accuracy in identifying healthy nodes with ten nodes datasets and as high as 97% accuracy with 500 or more nodes datasets.**

***Index Terms*—Autoregressive hidden Markov models, environment sensing, filtering corrupted nodes, sensor network, clustering, anomaly detection.**

I. Introduction

SENSOR systems have significant potential for aiding scientific discoveries by instrumenting the real world. For example, the sensor nodes in a wireless sensor network can be used collaboratively to collect data for the purpose of observing, detecting and tracking scientific phenomena. Sensor network deployment is becoming more commonplace in environmental, business and military applications. However, sensor networks are vulnerable to adversaries as they are frequently deployed in open and unattended environments. Anomaly detection is a key challenge in ensuring both the security and usefulness of the collected data. In this paper, we propose a method to filter the compromised nodes, be it self-healing or corrupted, using an autoregressive hidden Markov model (AR-HMM).

The paper is organized as follows. In the next Section II we cover the literature review for this work. In Section III, we give a brief overview of AR-HMMs. In Section IV, we describe the proposed algorithm, and in Section V we give numerical results. The conclusions are given in Section VI.

Manuscript received on September 30, 2016, accepted for publication on October 24, 2016, published on October 30, 2016.

The authors are with Missouri University of Sci & Tech, Rolla, MO 63128, USA (e-mail: bushra.anjum@gmail.com, chaman@mst.edu).

II. Related Work

Hidden Markov models and the Baum–Welch algorithm were first described in a series of articles by Leonard E. Baum and his peers at the Institute for Defense Analysis in the late 1960s [1]. However detecting compromised nodes using AR-HMM is a new area of investigation and we were unable to find any references that researched the same. Hence we provide the literature review in two parts, first, how compromised nodes are currently filtered and second, on the use of AR-HMM for solving diverse problems of identification, filtering, and prediction.

Wang & Bagrodia [2] have designed an intrusion detection system for identifying compromised nodes in wireless sensor networks using common application features (sensor readings, receive power, send rate, and receive rate). Hinds [3] has used Weighted Majority voting algorithm to create a concept of a node which could not be compromised, and to develop detection algorithms which relied on the trustworthiness of these nodes. Li, Song and Alam [4] have defined a data transmission quality function which keeps close to constant or change smoothly for legitimate nodes and decreases for suspicious nodes. The final decision of whether or not a suspicious node is compromised is determined by a group voting procedure. These designs take the en-network detection approach: misbehaved nodes are detected by their neighboring watchdog nodes. However en-network designs are insufficient to defend collaborative attacks when many compromised nodes collude together in the network. Zhang, Yu and Ning [5] present an alert reasoning algorithm for intelligent sensors using cryptographic keys. Zhanga et al. [6] exploit a centralized proven collision-resilient hashing scheme to sign the incoming, outgoing and locally generated/dropped message sets. These algorithms work on pinpointing exactly where the false information is introduced and who is responsible for it, but they are centralized and do so at a high computational cost. We propose a simple, decentralized model based approach to identify compromised nodes at a low computational cost using HMMs.

HMMs have been successfully used to filter unreliable agents, see Chang & Jiliu [7], Anjum et al. [8]. However the approach is unable to model correlation between observations. Autoregressive HMMs alleviate this limitation. Introduced in the 1980's, Juang and Rabiner published a series of papers [9], [10] regarding the application of Gaussian mixture

ISSN 2395-8618

autoregressive HMM to speech recognition. Switching autoregressive processes are well understood and have been applied in many areas. They have been particularly popular in *economics*, see Alexander [11] and Hamilton [12]. A brief summary of this subject and an extensive list of references can be found in [13]. The model is also successfully used by Park, Kwon and Lee [14], they have used AR-HMM by modeling the probabilistic dependency between sequential target appearances, presenting a highly accurate algorithm for robust visual tracking.

III. Proposed Model using Autoregressive Hidden Markov Models

A hidden Markov model is a Markov Chain with N states, which are hidden from the user. When the Markov chain is in a state, it produces an observation with a given state-dependent probability distribution. There are M different observation paths. It is this sequence of observations that the user sees, and from which it is possible to estimate the parameters of the HMM. An HMM is defined by the A matrix which is the one-step transition matrix of the Markov chain, the B matrix (referred to as the event matrix) which contains the probability distribution of the observations likely to occur when the Markov chain is in a given state, and π, the vector of probabilities of the initial state of the Markov chain. The (A, B, π) are together represented by λ. In this paper, we will make use of the following notation:

A: One-step transition Matrix ($N \times N$)
a_{ij}: One-step probability from state i to state j
B: Event Matrix ($N \times M$)
$b_i(k)$: Probability that the kth value will be observed when system shifts to state i
π: Initial Probability vector ($N \times 1$)
N: Number of hidden states
M: Number of observations
O: An observation sequence
T: Length of the observation sequence
Q: A sequence of hidden states traversed by the system
$\lambda = (A, B, \pi)$

A useful extension of HMM is *autoregressive* HMM (AR-HMM), which enhances the HMM architecture by introducing a direct stochastic dependence between observations. In AR(p)-HMM, the observation sequence is not only dependent on the HMM model parameters but also on a subset of p previous observations. Thus the model switches between sets of autoregressive parameters with probabilities determined by a state of transition probability similar to that of a standard HMM:

$$v_t = \sum_{r=1}^{P} a_r(s_t) v_{t-r} + \eta_t \quad with \quad \eta_t \sim N(0, \sigma^2)$$

where $a_r(s_t)$ is the r^{th} autoregressor when in state s ∈ {1 … N} at time t and each η_t is an i.i.d. normally distributed innovation with mean 0 and variance σ^2. Observations in the general AR-HMM can be continuous, but for our purposes we restrict the discussion to the discrete case. For example, a discrete HMM with AR(1) can be depicted by the Directed Acyclic Graph (DAG) as shown in Fig. 1. Three different, but related, problems have been defined for HMMs, briefly described below.

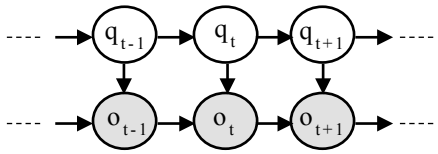

Fig. 1. DAG of the AR(1)-HMM where q_i and o_i represent the state and the observation generated at time slot i respectively.

Problem 1: Forward Probability Computation: Given an observation sequence, compute the probability that it came from a given λ.

Let the system be in state i at time t. The probability of the system shifting to state j at t+1 is given by the one-step transition probability a_{ij}. The probability that of the M possible observed states, the kth one is observed given that the system shifted from state i to j is $a_{ij}b_j(k)$. Since the state i can be any one of the states from 1 to N, the probability of observing the kth value, given that the system shifts to state j at time t+1 is the sum of all probable paths to state j, i.e., $\left[\sum_{i=1}^{N} a_{ij}\right] b_j(k)$.

Now, the only remaining unknown is the joint probability of having observed the sequence from O_1 through O_t and being in state i at time t, given λ. Representing this quantity by $\alpha_t(i)$ and representing the kth observed value at time t+1 by O_{t+1}, we have

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), t = 1.2, \ldots, T-1, \qquad j \in [1, N]$$

To start off this induction, the initialization step for calculating $\alpha_t(i)$ can be obtained as follows. The probability of the system being in state i at time 1 is given by π_i and the probability of observing O_1 is then given by $b_j(O_1)$. Thus,

$$\alpha_1(i) = \pi_i b_j(O_1), i \in [1, N]$$

$\alpha_T(i)$ obtained from the induction step represents the probability of observing the sequence from O_1 through O_T, and ending up in state i. Thus, the total probability of observing the sequence O_1 through O_T, given λ is

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

ISSN 2395-8618

Problem 2: Backward Probability Computation: Given an observation sequence, compute the probable states the system passed through.

The quantity $\beta_t(i)$ is defined as the probability that the sequence of observations from O_{t+1} through O_T are observed starting at state i at time t for a given λ. It is calculated in the same way as $\alpha_t(i)$, but in the backward direction. We have,

$$\beta_t(i) = \sum_{j=1}^{N} \beta_{t+1}(j) a_{ij}\, b_j(O_{t+1}), \qquad t = T-1, \ldots, 1$$

For $t = T$ we have $\beta_T(i)$=1, i = 1, 2, …, N. To calculate the probability that the system was in state i at time t given O and λ, we observe that $\alpha_t(i)$ accounts for the observation sequence from O_1 through O_t and $\beta_t(i)$ accounts for O_{t+1} through O_T, and both account for the state i at time t. So the required probability is given by $\alpha_t(i)\beta_t(i)$. Introducing the normalizing factor from problem 1, $P(O|\lambda)$, we have

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)}$$

At each time t, the state with the highest γ is the most probable state at time t. A better way to obtain the most probable path of states Q that give rise to an observation sequence O, is to use dynamic programming, as described in [13].

Problem 3: Matrix Estimation: Given an observation sequence O, compute the most probable λ.

The term a_{ij} can be calculated as the ration of the number of transitions made from state i to state j over the total number of transitions made out of state i. We have from problem 2 that $\gamma_t(i)$ is the probability of being in state i at time t. Extending this, the probability of being in state i at time t and in state j at time t+1 can be calculated as follows:

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j \beta_{t+1}(j)}{P(O|\lambda)}$$

IV. The Algorithm

We consider a set of sensors that consists of a mix of healthy, self-healing and corrupted sensors. We assume that time is slotted, let T be the total number of slots. During each slot, each sensor submits a reading about the environment. For simplicity, let all the nodes submit a reading at each time slot. (This constraint can be easily removed, by appropriately modifying the way the $\alpha_i(t)$ and $\beta_i(t)$ are calculated.)

For each node, we have a sequence of T readings (observations) to train an AR-HMM. Thus, we end up with as many AR-HMMs as the number of nodes. Using statistical clustering techniques, we cluster the B matrices of these AR-HMMs into two groups, one for healthy and the other for compromised nodes (self-healing and corrupted). This permits us to identify and filter out the compromised sensors.

To test the accuracy of the algorithm, we consider three types of sensors, healthy, self-healing and corrupted. To achieve this, we define three λ's, all having the same A and π, but different B. The B matrix of the healthy nodes (B_h) should be such that the readings generated echo the environmental phenomenon it is sensing. The B matrix of self-healing nodes (B_s) introduces spurs of invalid data before moving back to the valid state and the B matrix of the corrupted nodes (B_c) is set up to predominantly generate invalid data. The exact matrices are defined in the next section.

Next, this sequence of T readings is used to estimate the A, B, and π matrices of the sensor. Using the B matrices, we cluster the sensors into two groups, i.e., healthy and compromised nodes. For this, we take the mean squared error (MSE) of each estimated B matrix with the perfectly healthy matrix B_p (see Section IV), where

$$MSE = \frac{1}{N}\sum_{i=1}^{N}\sum_{k}^{M} (b_i(k) - b_{pi}(k))^2$$

and $b_i(k)$ and $b_{pi}(k)$ are the $(i,k)^{\text{th}}$ element of the B and B_p matrices respectively.

The resulting MSE values are then classified into two clusters using the k-means clustering algorithm. As B_p represents the truly healthy matrix, the cluster with a center closer to 0 represents the group of healthy nodes.

V. Experimental Setup and Results

A. Synthetic Datasets – Richardson's Model

For our simulation study, we use the classic Richardson's model for the temperature [14] to define the transition matrix A and the autoregressors v_t. Richardson's model uses two states S_t = *{Dry, Wet}*, hence N = 2, and second order autoregression, AR(2), to define the temperature readings. The model is given as follows:

$$A = \begin{bmatrix} 0.55 & 0.45 \\ 0.33 & 0.77 \end{bmatrix}$$

$$v_t = \begin{cases} 1.22 + 0.90Y_{t-1} - 0.13Y_{t-2} + 2.11\eta_t & (dry) \\ 2.14 + 0.70Y_{t-1} - 0.003Y_{t-2} + 1.87\eta_t & (wet) \end{cases}$$

The results reported in this section were based on 60% healthy nodes, 20% self-healing, and 20% corrupted nodes. In all experiments, we set the Markov chain to start with an equal probability of being in any of the two hidden states, i.e., π = [0.5, 0.5]. We define three B matrices (B_h, B_s and B_c), such that they model the behavior of the three nodes under consideration, healthy, self-healing and corrupted respectively. B is a 2 × 2 matrix where the first column corresponds to the probability of generating a value using autoregressive equations, and second column corresponds to the probability

ISSN 2395-8618

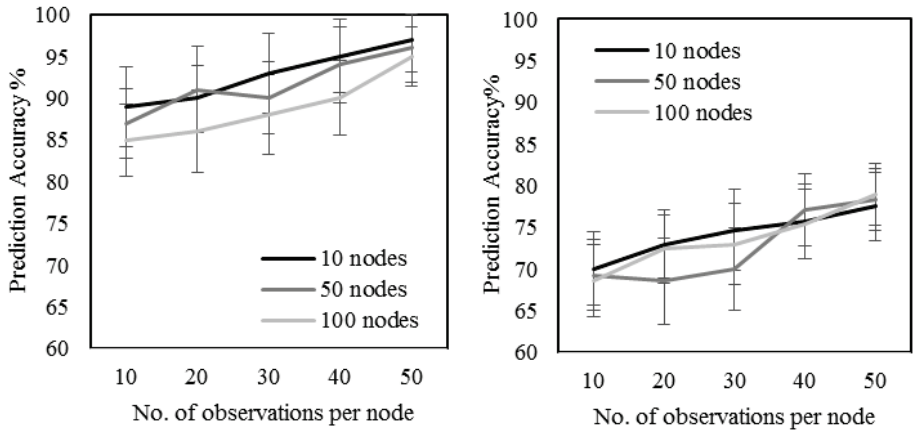

Fig. 2. Prediction accuracy for healthy nodes (left), corrupted nodes (right)

TABLE I
PREDICTION ACCURACY FOR SYNTHETIC SENSOR DATA MODEL

	Percentage of self-healing nodes					
	0%	10%	20%	30%	40%	50%
Healthy nodes	0.98	0.96	0.95	0.97	0.96	0.95
Corrupted nodes	0.95	0.88	0.84	0.78	0.75	0.68

of entering the corrupted state and generating an invalid value. (The selection of N = M = 2 is not significant, and any other values can be readily used).

$$B_h = \begin{bmatrix} 0.95 & 0.05 \\ 0.95 & 0.05 \end{bmatrix}, B_s = \begin{bmatrix} 0.65 & 0.35 \\ 0.65 & 0.35 \end{bmatrix}, \quad B_c = \begin{bmatrix} 0.05 & 0.95 \\ 0.05 & 0.95 \end{bmatrix}$$

The perfectly healthy matrix, B_p used for MSE calculations is defined as

$$B_p = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

For the above A matrix, we define three AR(2)-HMMs, namely, $\lambda_h = (A, B_h, \pi)$, $\lambda_s = (A, B_s, \pi)$ and $\lambda_c = (A, B_c, \pi)$.. Subsequently, we generate random samples of temperature readings for all the nodes, and then apply the algorithm described in the previous section to identify the group of healthy nodes. Then, we compare the number of identified healthy nodes to the original set of healthy sensors and report the result as a percentage of correctly identified healthy nodes. We used MS_Regress MATLAB package for Markov Regime Switching Models by Perlin [15] to implement the above algorithm and obtain numerical results.

The results are given in Fig. 2 which gives the percentage of correctly identified healthy (left) and corrupted (right) nodes along with the 95th confidence interval. For the figure, we assume 10, 50 and 100 sensors, and vary the number of observations per sensor from 10 to 50. The confidence intervals are obtained by replicating each result 100 times, and each time using a different seed for the pseudo-random number generator.

Interesting observations (Fig. 2): As the number of readings increases per node, the prediction accuracy the nodes (both healthy and corrupted) increases as well. For healthy nodes, the average prediction accuracy for 10 readings is around 88% and for 50 readings per node, it jumps to around 97%. Overall, the prediction accuracy of healthy nodes is much higher than that of corrupted nodes. This can be explained by the fact that the cluster distribution of self-healing nodes and corrupted nodes is quite similar whereas both differ significantly from the cluster distribution of healthy nodes.

Based on the results, we also observe that it is not necessary to have a large sample of nodes and readings per node. We see that with as little as 10 nodes and 10 readings per node, we obtain results which are similar to those obtained with larger number of nodes and readings per node. This leads us to conclude that our approach can be used in a decentralized manner, i.e., we do not need data from all (or a large number) of the sensors in order to prune out the compromised ones.

Finally, we turn towards sensitivity analysis of our approach, i.e., how does the percentage of self-healing nodes impact the filtering accuracy. For this we varied the percent of self-healing nodes from 0 to 50%. Keeping the number of nodes to 50 and the number of readings per node to also 50, we calculate the prediction accuracy of the healthy and corrupted nodes. The results are presented in Table I.

We make the following observations. As the number of self-healing nodes increases, the prediction accuracy of both healthy nodes and corrupted nodes decreases. However, where

ISSN 2395-8618

the change in healthy node prediction is minor (98% to 95%), corrupted node prediction suffers increasingly as the number of self-healing nodes increases (95% to 68%). This confirms our initial observation that the cluster distribution of self-healing nodes is similar in nature to corrupted nodes. Hence it becomes difficult for the algorithm to pick corrupted nodes exclusively from the set of corrupted and self-healing nodes. The success of our approach, however, lies with the identification of healthy nodes, with accuracy greater than 90%.

B. Real Sensor Datasets

In order to test our models on real world sensor datasets, we use datasets from two sources: Intel Berkley Research Laboratory [16] and Labeled Wireless Sensor Network Data Repository (LWSNDR) projects [17]. Both projects represent the state of the art in sensor systems and collect measurements in very different environments. Hence, these datasets allow us to evaluate the accuracy of AR-HMM classification on representative and diverse sensor system data.

First dataset is collected from 54 sensors deployed in the Intel Berkeley Research lab. The Mica2Dot sensors with weather boards collected time stamped topology information, along with humidity, temperature, light and voltage values once every 31 seconds. This dataset includes a log of about 2.3 million readings collected from the 54 motes (mote is a sensor that is capable of doing some processing in addition to collecting and transmitting data), where data from some motes may be missing or truncated.

Second dataset is collected from a simple single-hop and a multi-hop wireless sensor network deployment using TelosB motes. The data consists of humidity and temperature measurements collected during 6 hour period at intervals of 5 seconds. For this evaluation, we are using the single hop labeled readings which consist of approximately 15,000 entries.

The first dataset is unlabeled. We have tested our anomaly detection algorithms with by visually inspecting the sensor data time series. The second dataset, is a labelled wireless sensor network dataset where label '0' denotes normal data and label '1' denotes an introduced event (anomaly). More details can be obtained from [18].

We assume that the sensor readings collected over a reasonable duration capture the normal patterns in the sensor data series. As a general rule, we have used 20% of the data points to work out the auto regression equations for the system. The auto regressive coefficients are calculated using the least squares method. Also, the same data points are used to calculate $\lambda = (A, B, \pi)$ as defined under Problem 3 in Section 2.

TABLE II
OVERALL PREDICTION ACCURACY FOR REAL SENSOR DATA

	Algorithm	Accuracy
Intel Berkley Research Lab	Naïve Bayes	89.785 %
	ZeroR	85.543 %
	AR-HMM	**97.231 %**
Labelled WSN Data Repository	Naïve Bayes	90.754 %
	ZeroR	86.352 %
	AR-HMM	**98.413 %**

For comparison, we are using two popular classification algorithms ZeroR and Naïve Bayes along with our proposed AR-HMM model. Briefly, ZeroR is a useful predictor for determining a baseline performance, predicting mean for a numeric class and mode for a nominal class, and Naïve Bayes is a conditional probabilistic classifier based on Bayer's theorem. The filtering accuracy to identify healthy nodes is given in Table II.

Table II displays that as ZeroR provides a baseline accuracy, AR-HMM clearly surpasses Naïve Bayesian classification for correctly identifying the healthy sensors. We further describe our method's accuracy using (1) number of false positives (detecting non-exist compromised nodes) and (2) number of false negatives (not being able to detect a compromised node) as our metrics. Specifically, the results in Table III below are presented as follows – the x/y number indicates that x out of y compromised nodes were detected correctly (corresponding to y-x false negatives) plus we also indicate the number of corresponding false positives.

VI. CONCLUSION

In this paper, we propose a method based on autoregressive hidden Markov models (AR-HMMs) for filtering out

TABLE III
PREDICTION ACCURACY FOR HEALTHY AND COMPROMISED NODES INDIVIDUALLY

	Algorithm	Healthy nodes	Compromised nodes
Intel Berkley Research Lab	Naïve Bayes	33/40	9/14
	ZeroR	30/40	8/14
	AR-HMM	**37/40**	**11/14**
Labelled WSN Data Repository	Naïve Bayes	12/16	3/5
	ZeroR	11/16	2/5
	AR-HMM	**14/16**	**4/5**

ISSN 2395-8618

compromised nodes in a sensor network. We confirm through experimentation, based on revised Richardson's temperature model, that our filtering method is quite accurate and identifies the healthy sensors with an accuracy greater than 90%. We further used real sensor data from Intel Labs and WSN repository from UNC to calculate the prediction accuracy of AR-HMM approach as opposed to Naïve Bayes and ended up with encouraging results, with prediction accuracy as high as 97%

REFERENCES

[1] R. Lawrence, "First Hand: The Hidden Markov Model". *IEEE Global History Network*. Retrieved 2 October 2013.

[2] Y.T Wang and R. Bagrodia, "Com-Sen: A Detection System for Identifying Compromised Nodes in Wireless Sensor Networks", *SECURWARE 2012*.

[3] C.V. Hinds, *Efficient detection of compromised nodes in wireless sensor networks*, 2012.

[4] T. Li, M. Song, M. Alam, "Compromised Sensor Nodes Detection: A Quantitative Approach," *ICDCSW 2008*.

[5] Q. Zhang, T. Yu, and P. Ning, "A Framework for Identifying Compromised Nodes in Wireless Sensor Networks," *ACM Transactions on Information and Systems Security*, vol. 11, no. 3, Article 12, 2008.

[6] Y. Zhanga, J. Jun Yangb, W. Lia, L. Wangc, and L. Jind, "An authentication scheme for locating compromised sensor nodes in WSNs," *Journal of Network and Computer Applications*, vol. 33, no. 1, 2010.

[7] L. Chang, and Z. Jiliu. "The reputation evaluation based on optimized hidden Markov model in e-commerce," *Mathematical Problems in Engineering*, vol. 2013, Hindawi, 2013.

[8] B. Anjum, M. Rajangam, H. Perros, and W. Fan, "Filtering Unfair Users: A Hidden Markov Model Approach," *ICISSP*, Loire, France, 2015.

[9] B.H. Juang and L.R. Rabiner, "A Probabilistic Distance Measure for Hidden Markov Models," *AT&T Tech. J.*, vol. 64, no. 2, pp. 391–408, 1985.

[10] B.H. Juang and L.R. Rabiner, "Mixture Autoregressive Hidden Markov Models for Speech Signals," *IEEE Trans.*, vol. ASSP-33, no. 6 , pp. 1404–1413, 1985.

[11] C. Alexander, "Market Risk Analysis: Practical Financial Econometrics," Wiley Books, 2008.

[12] J.D. Hamilton, "Regime Switching Models," *Palgrave Dictionary of Economics*, available at http://dss.ucsd.edu/~jhamilto/palgrav1.pdf, 2005.

[13] Y. Ephraim and N. Merhav, "Hidden Markov processes," *IEEE Trans. Inf. Theory*, vol. 48, no. 6, pp. 1518–1569, 2002.

[14] C.W. Richardson, "Stochastic simulation of daily precipitation, temperature, and solar radiation," *Water Resour. Res.*, vol. 17, no. 1, 182–190, doi:10.1029/WR017i001p00182, 1981.

[15] M. Perlin, "MS Regress. The MATLAB Package for Markov Regime Switching Models." Available at http://ssrn.com/abstract=1714016, 2014.

[16] Intel Lab Data. http://db.csail.mit.edu/labdata/labdata.html

[17] UNC Greensboro, *Machine Learning Models and Algorithms for Big Data Classification*. http://www.uncg.edu/cmp/downloads.

[18] S. Suthaharan, M. Alzahrani, S. Rajasegarar, C. Leckie and M. Palaniswami, "La-belled Data Collection for Anomaly Detection in Wireless Sensor Networks", in *Proceedings of the Sixth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP 2010)*, Brisbane, Australia, 2010.