Using Data Mining to Estimate Missing Sensor Data

Le Gruenwald
School of Computer Science
The University of Oklahoma
Norman, OK 73019, U.S.A.
ggruenwald@ou.edu

Hamed Chok
School of Computer Science
The University of Oklahoma
Norman, OK 73019, U.S.A.
Hamed.Chok-1@ou.edu

Mazen Aboukhamis
School of Computer Science
The University of Oklahoma
Norman, OK 73019, U.S.A.
mazen@ou.edu

Abstract

Estimating missing sensor values is an inherent problem in sensor network applications; however, existing data estimation approaches do not apply well to the context of datastreams, a major characteristic of sensornet applications. Additionally, they fail to account for relationships among sensors and simultaneously, incorporate the time factor making the estimation process computationally aware of the relative relevance of each data round in the datastream. To address this gap, we propose a data estimation technique, FARM, which uses association rule mining to discover intrinsic relationships among sensors and incorporate them into the data estimation while taking data freshness into consideration. FARM was tested with data from two real sensornet applications, namely climate sensing and traffic monitoring. Simulation shows that in terms of estimation accuracy, FARM outperformed existing techniques costing only marginally more space and time overheads while scaling well with the network size, thus assuring quality of service for real-time applications.

1. Introduction

In wireless sensor networks, sensors send their data to servers and other nodes. The continuous flow of data readings from a sensor farther into the network is called datastreams. It can be expected that transmitted sensor data are lost or corrupted due to many reasons, such as power outage at the sensor's node, random occurrences of local interferences, or a higher bit error rate of the wireless radio transmissions as compared with wired communications. Simply re-querying data is a naïve alternative as it may induce a long wait, quicken the power exhaustion of the node, and above all, it is still not guaranteed to make available the original reading. Hence, to be able to efficiently process queries that need to access missing sensor readings, it is crucial that missing stream values be estimated.

Data mining is employed to discover knowledge from the existing data. Suck knowledge can be utilized in computing estimates for missed values. In this paper, we present a data mining based technique, called Freshness Association Rule Mining (FARM) to estimate values for missing, corrupted, or late readings from one or more sensors in a sensornet at any given round. The technique aims to provide a good quality of service (QoS) for real-time applications, which can be defined as a function of both the time needed by the queries to produce results based on the data gathered by the sensors and the accuracy of these results. The need to be aware of the time factor in sensornet applications gives rise to the data *freshness* concept, which FARM integrates through a formal data representation scheme capable of compacting datastreams and assigning *freshness* value to each round.

To evaluate the performance of FARM, we conduct simulation experiments to compare it to the following estimation techniques: WARM [14], SPIRIT [8], TinyDB [9], and four statistical estimation methods: Simple Linear Regression (SLR), Multiple Linear Regression (MLR), Curve Regression (CE), and estimation by average [11]. Two real data sets were used in the experiments: the air temperature sensor data of the Huntington Botanical Gardens in San Marin, California collected from the NASA/JPL Webs Sensor Project in 2006 [12], and the traffic data collected by the Department of Transportation in Austin, Texas in 2000 [13]. Additional synthetic datasets were used to study the scalability of the algorithm.

2. Related work

The problem of estimating missing values has been extensively investigated in statistics. Examples include Mean Substitution, Imputation by Regression, Hot/Cold Deck Imputation, Expectation Maximization, Maximum Likelihood, Multiple Imputations, Bayesian Estimation, and Pairwise and Litwise Deletion [1-6]. None of these methods fits the wireless sensor networks and stream data environments for reasons of efficiency, disregards to

temporal factors, or priory requirements such as "Miss at Random (MAR)".

In the NASA/JPL Sensor Webs project [7], if one sensor fails, its neighboring sensors compensate for the lost data by increasing their sampling rates. This implies that there must be a tight collaboration among sensors for a sensor to know that its neighboring sensor has failed. This increases power consumption on every sensor even during its normal operation. In addition, the approach does not address how sampling rates should be adjusted in order to guarantee good QoS. It is also possible that when some neighboring sensors fail, no sampling adjustment could potentially compensate for missing values.

Spirit [8] uses auto-regression as its basic forecasting model to estimate missing values in datastreams. It spots correlations on numerical streams and extracts the hidden variables which summarize the key trends in the entire stream collection. To estimate, SPIRIT uses auto-regression on the extracted hidden variables, and uses the results to predict the values of the missing data. To estimate missing data of current round, SPIRIT bases the forecast on the values of the previous round. It is unclear whether SPIRIT can provide good estimation accuracy in datasets where a sizeable portion of the data is missed.

TinyDB [9] is a query processing system for extracting information from a network of special type of sensors. Given a query specifying the user data interests, TinyDB collects that data from motes in the environment, filters it, aggregates it together, and routes it to a PC. TinyDB does this via power-efficient in-network processing algorithms. It estimates the missing values by taking the average of all the values reported by the other sensors in the current round. This straightforward estimation heuristic is blind to any correlations between sensors, which might yield poor estimation accuracy.

WARM [14] uses association rule mining to estimate missing readings. It makes use of the sliding window concept where only the latest *w* rounds of data reports are stored and used for estimation. One deficiency of WARM is its sensitivity to the window size – a small *w* induces a risk of loosing data trends while large values require a considerable space overhead. An additional deficiency is WARM's disregards to temporal aspects since it views all round data as equally important.

3. Freshness Association Rule Mining (FARM)

3.1. Motivation and Contribution

FARM uses association rule mining to find related sensors. The estimation of a missing sensor reading is based on a weighted average of the current reading of the sensors related to the sensor with the missing reading. Each participating weight in the averaging is directly derived from the strength of the corresponding sensor association. Pioneer work in [14] used association rules to estimate lost sensor data. A major contribution of FARM is its use of a data *freshness* framework on top of an association rule method. The integration of this framework is non-trivial and translates the rationale of FARM. The fundamental idea behind using *freshness* is to incorporate the temporal factor that is inherent in most data stream applications. The contribution of FARM is; however, threefold:

- Incorporate the temporal aspect into association rules and estimation
- Compact data streams and allow a large history to appropriately influence sensor rules
- Guarantee retrievability of original data from its compact form

Association rules shall be constructed while paying a close consideration to the *freshness* of data. The benefit from this conception is self-evident in that the fresher the data is the more importance it shall be given. Typically, the current state of any sensed physical environment is more likely to be influenced by the set of its previous states that has manifested in the nearest time period. It is this rationale that gives rise to the notion of data *freshness*, which, if properly combined with association rules, can potentially uncover temporal trends across different sensors. Temporal patterns shall be obtained by combining information on the data of a particular sample together with the round order of the same sample in a way to allow the more recent (*fresh*) data to contribute more towards the estimation.

By assigning each round a different weight that grows in accordance with its order, it is possible to define a reversible mapping between an entire stream history from one sensor and the set of real numbers. This allows the data to be compact and yet sufficient for estimation. Our analysis shows that the reverse mapping procedure is nicely scalable as the number of samples increases and thus can be interfaced with a separate query processor module. All these advantages of our proposed technique are possible under the umbrella of the *freshness* framework.

To recapitulate, it is this duality between the data *freshness* and the data compaction scheme as well as their positive implications on the space and time complexities that derive the motives of the FARM method.

3.2. Considering freshness of data in estimation

To actualize the previous motives, FARM makes each round in the sensor stream participate with a different

level of contribution (round weight) that is based on its relative recency. Round weights satisfy the following recursive relation: $w(1) = 1$; $w(n) = p*w(n-1)$ where $p \geq 1$, input to w is the round order, and w is a function that returns the weight of a given round. The weight of an arbitrary round n is p times the weight of the previous round ($n-1$). p is referred to as the damping factor, which represents the relative importance of a round comparatively with the previous round. The weight of the 1st round is 1. The above recursive definition stems from the rather intuitive conception of relative round importance. An equivalent definition is $w(n)=p^{n-1}$.

Obviously, $w(n)$ grows in terms of n and, thus, the recency of a round is reflected in the value of its weight. The more recent the data is, the higher weight it will have and subsequently, the more it will contribute to the calculation of the estimated values. It is worth noting that the choice of the exponential scheme might eventually yield an overflow due to the buffer limit in finite arithmetic. Refer to Section 3.7.

Assuming $p = 3$, Table 1 presents an example of a centralized network with five sensors (S0, S1, S2, S3, and S4) that send the traffic state data to a central server at specific time intervals. Four traffic states can be reported as follows: Light = "L", Moderate = "M", Heavy = "H" and Congestion = "C". Table 1 shows the weight of the three rounds of data readings.

Table 1. RMSEs for Climate Sensing data set

Round Number	S0	S1	S2	S3	S4	Round Weight
1	M	L	M	M	M	3^0
2	M	M	L	L	M	3^1
3	C	M	M	C	C	3^2

As shown in Table 1, the weights of rounds 1, 2, and 3 are respectively 3^0, 3^1, and 3^2. This simple data example in Table 1 connotes two basic assumptions used in our technique. One is that the sensor data is categorical where data discretization is done offline. Secondly, it is assumed that the sensor network in question follows a centralized architecture where all sensors are one-hop from a base station. All sensors report their readings to the server where data is normally processed and analyzed and where also estimation is to be performed. Although an in-network data collection/estimation scheme might be more power-aware, an application of FARM to such non-centralized sensornets is possible though will only be addressed in future work. As we shall see in Section 4, this type of centralized net topology conforms to the two real-world sensornet deployments whose datasets are used to test our method.

3.3. Determining the relationships among sensors

FARM attempts to find association rules between sensors in an apriori-like fashion [10]. Apriori is a level-wise iterative algorithm that successively determines the frequent itemsets where in each new iteration the frequent itemsets of the next order are found.

One can quickly think of the different sensors as the entities to be substituted in for the generic concept of items in traditional data mining. However, few modifications need to be put in place. To address the issue of categorical data as apposed to basket data, we shall define the frequency of a sensor with respect to a certain state [14]. Therefore, one state is assimilated to *true* and all others to *false*. This will in turn complete the definition of the prior and posterior probabilities (respectively, the actual support and actual confidence of an association rule).

Datastreams are continuously generated, which makes it time consuming to re-generate frequent itemsets of all order given the new arriving data. Note that an incremental online update of the frequent itemsets of any order is not possible. This is because given only frequent itemsets of all orders up until the last round and the reports (transactions) of the new round, it is impossible to update the different frequent itemsets without having the details of the reports in all previous rounds. Furthermore, saving frequent itemsets of high orders require large storage capacity. For these two reasons, the maximum order of frequent itemsets is limited to two. In other words, FARM only seeks sensor rules where there is one antecedent sensor and one consequent sensor. A FARM sensor rule would then read: if Sensor A reports state e then "likely" Sensor B also reports state e. Experimentation shows good estimation accuracy despite this rule-load shedding approach.

To incorporate the *freshness* factor, reported states are weighted according to the round weight in which they were reported. To accommodate for this principle, actual support (*actSup*) and actual confidence (*actConf*) need to be reconfigured. We propose an actual weight support *actWeightSup* and an actual weight confidence *actWeightConf* for the pattern Sensor $A \rightarrow$ Sensor B w.r.t. e. The actual weight support is the sum of the weights of the rounds where both sensors A and B report state e divided by the sum of all round weights. The actual weight confidence is the sum of round weights where both A and B report the same state e divided by the sum of the round weights where e is reported by A. Clearly, these formulations of *actWeightSup* and *actWeightConf* reflect the temporal factor translated by the notion of round recency. To develop an intuition, consider the data rounds of Table 1. It can be verified that the *actWeightSup* for the

rule Sensor *S1*→ *S2* with respect to state "M" equals 9/13. The *actWeightConf* for the same rule is 9/10. This completes how round weights are used for the purpose of incorporating data *freshness* in association rules.

3.4. The proposed data structures

FARM uses two main structures: (1) buffer and (2) 2D-Ragged Array. The buffer is simply a one-dimensional array that is reset with new round data after each new sample with a special value for missing/corrupted values. The ragged array can be viewed as the upper triangular part of a matrix where each of the column and row sets consists of the entire set of sensors. An element of the 2d ragged array is an object that corresponds to a particular pair of sensors not necessarily distinct. An object holds the history of round information for the particular pair of sensors that it corresponds to. This information is useful in evaluating the association mining variables before potentially qualifying the sensor elements in the given pair as being related. Object $[S_i][S_j]$ $(i \geq j)$ contains a one dimensional array of s entries, where s is the number sensor states. Each array entry in $[S_i][S_j]$ stores the sum of all round weights in which both sensors reported the same particular state. The index of each particular array entry determines the state with respect to which the weight sum is computed. For instance, in Table 1, object $[S_4][S_0]$ contains the following array of weight sums: {0,4,0,9} with respective indices: "L", "M", "H", and "C". Note that it is the round weights that make it possible to store only one number (weight sum) and yet be able to summarize all information about the specific rounds in which a given pair of sensors had reported a common state. Given a weight sum and the weight sequence w, it is algorithmically possible to retrieve the orders of all respective rounds where the two sensors had a common reading. Section 3.6 annotates the data retrieval algorithm.

3.5. The proposed algorithms

FARM consists of three main algorithms, *checkBuffer*, *update*, and *estimateValue*. *checkBuffer* serves as the main routine, where it checks for any existing missing values and accordingly directs the *estimateValue* method, to perform estimation before finally calling the *update* procedure to update the 2d ragged array with new round data including any possibly predicted values. Although this update scheme makes no distinction between estimated and original values, we are still able to control the quality of the estimate under a basic sensor model assumption. In our theoretical analysis (omitted here), we formulate the RMSE as well as other metrics in terms of the mining variables. Then a constrained minimization setting assures the lowest possible RMSE and a bounded estimate error with the respect to the assumed data model. This initiative also sets the ground for a parameterless data mining framework.

The *update* method traverses the buffer to check if there are any two sensors reporting the same value in the current round. For any such pair, it sets the current common report and it increments the appropriate weight sum by the current round weight. It can be quickly verified that this algorithm requires $O(d^2)$ operations where d is the number of sensors as it iterates in the order of all possible sensor pairs.

The *estimateValue* finds all sensor associations with the missing sensor (hereon *MS*) then uses these associations to compute an average by weighing in each involved state in all rules. The average is rounded up or down to the nearest legal sensor state. The algorithm follows the below steps:

Step 1: Determine eligible states for estimation. A state is said to be eligible if its actual support is larger than the minimum support, *minSup*. In the context of a state, the actual support is simply its frequency divided by the total frequencies of all states.

Step 2: Create a temporary data structure called *StateSet* for each eligible state. The purpose of this set structure is to group sensors reporting the same state in the current round. Each *StateSet* is initialized to empty set and then sensors are distributed into the appropriate state sets depending on the values of their current readings. The individual *StateSet* will go through several updates before they exactly contain sensors that associate with *MS*.

Step 3: Test for potential sensor associations by identifying all sensors of each *StateSet* that individually with *MS* constitute a 2-frequent itemset. This test is done by comparing *actWeightSup* with the user-defined *minSup*. If the support, *actWeightSup*, between one sensor and *MS* fails to be larger than *minSup*, that sensor is deleted from the *StateSet*. If the test succeeds, the *actWeightConf* of the potential rule $S_i \rightarrow MS$ with respect to the eligible state reported by S_i is compared with the user-defined *minConf*. If the confidence test is satisfied, then S_i is considered as an eligible sensor and will contribute in estimating the missing sensor value; otherwise, S_i will be deleted from the *StateSet*.

Step 4: Compare the contribution weight of each eligible state towards the estimation in terms of the supports between *MS* and each of the eligible sensors reporting that particular eligible state. The contribution weight is distinct from the accumulated state weight computed in terms of round recency. Evidently the latter weights influence the former.

Step 5: Calculate the missing value and round it to the closest state value. The estimated reading of *MS* is

obtained by averaging the readings of its related sensors. Averaging coefficients are based on the strengths of the relations involved defined in terms of the supports of those relations. Our analysis shows that *estimateValue* requires no more than O(*ds*) operations while on average it requires only O(max(*d*,*s*)).

3.6. Retrieving data at any point in time

Our data compaction scheme that maps the stream history of one sensor to a real quantity is reversible by the aid of a procedure that scales well even as the number of rounds increases. Compacted report history of any sensor is located in one diagonal entry of the 2d ragged array. Given a weight sum *ws* relative to a particular sensor with respect to particular state, the algorithm determines the orders of rounds in which the sensor reported a certain state. The algorithm computes the partition of round weights that sum up to the given weight sum. However, when $p = 1$, the decomposition is completely ambiguous and thus retrievability is not possible. For values of p greater than or equal to 2, the algorithm iterates in the order of the cardinality of the state weight partition summing to *ws*. Complexity remains practical for the upper half range between 1 and 2 [15].

3.7 Implementation Issue

Evidently, from a practical perspective it is impossible to store an infinitely growing sum in a finite buffer limit since the 64-bit buffer will overflow at some point. This limits the value of the maximum weight sum that can be stored and, implicatively, the maximum number of rounds that can be allowed. Our analysis (omitted for brevity) shows that for a damping factor of 2, we would be able to store at most 1023 rounds [15]. This can be practical in the sense that neglecting much older rounds does not significantly change the support of the rules due to their exponentially lower weights. Few remedies that allow the storage of a much larger history are elaborated on in [15].

4. Simulation experiments

We compare FARM to WARM, SPIRIT, TinyDB, and four statistical estimation methods: (1) The Simple Linear Regression (SLR) approach, (2) The Multiple Linear Regression (MLR) approach, (3) The Curve Regression (CE) approach, and (4) estimation by average (Avg) [11]. Comparison was done with respect to the space, time, and accuracy metrics. Additionally, we evaluate FARM and WARM's ability to estimate by means of data mining without resort to a default procedure when no associations are found. We evaluate the programs on two different pre-collected datasets as noted in Section 1 with 15% of the data missing. We averaged the results for FARM for damping factors ranging from 1 to 10. We provide a summary of results below. Refer to [15] for a comprehensive discussion and analysis.

4.1. Evaluation of Time

By execution time, we mean the average time it takes to finish estimation per round. The execution time of FARM is longer than that of most other methods by less than one millisecond. Execution took 9.73×10^{-4} seconds in the climate dataset and 434.37×10^{-4} seconds in the traffic dataset. However, it is still manageable for the purpose of most sensor network applications since it took less than one millisecond to complete all estimations of one round. This time is negligible compared to the sampling rates (5 and 15 minutes) in our two test applications and most sensornet applications for that matter.

4.2. Evaluation of Estimation Ability

Each of FARM and WARM has estimation ability exceeding 80% on each of the datasets.

4.3. Evaluation of estimation accuracy

The accuracy of the estimation is evaluated using the normalized root mean square error (*RMSE*). The RMSE results on both data sets for all the approaches are presented from the best to the worst in Table 2. We tabularize both the RMSE and the relative difference in accuracy between the best method and the rest. This latter measure is a better assessment of the difference as apposed to directly basing the comparison on an evaluation of the difference in magnitudes of the individual RMSEs. We note that FARM's average RMSE is the best on both real datasets.

In particular, on the climate dataset, the relative estimation error is about 23% and 42% better than the second and third best methods. As for the traffic dataset, FARM is best with a margin of about 3% and 14% relative to the two next best methods. We can attribute this outcome to the nature of the dataset at hand, i.e. whether the data rounds have in reality different levels of relevance or not. This result can best interpreted by noting the average of all damping factors used. Precisely, the average p was about 3.9, which is considerably higher than 1. This basically translates the assumption that the given dataset naturally encloses data rounds of varying importance. This explains why the average error on the traffic dataset is only marginally better (about 3%) than

the second best method. It might therefore be suspected that the best results on that dataset would show a larger gap in the relative estimation accuracy if the real importance of round recency is far off the average decay rate. To verify this claim, we here state the best results. On the climate sensing dataset the lowest attained RMSE was 0.0043, which was obtained for the damping factor values of 4. This explains why the best RMSE is close to the average RMSE for this dataset. As for the traffic dataset, the lowest RMSE was 0.069 which is 25.96% better than SPIRIT, the second best approach. This lowest RMSE was in fact yielded by a damping factor of 1.25, which is noticeably far from the average value. Despite the compromise yielded by considering the average results, FARM still achieved the best estimation accuracy. We refer the reader to our technical report [15] on how to optimize the choice of the damping factor by considering a data model for the sensor readings.

Table 2. RMSEs for both datasets

Climate Sensing Data			Traffic Monitoring Data		
Method	**RMSE**	**Relative Accuracy %**	**Method**	**RMSE**	**Relative Accuracy %**
FARM	**0.00487**	**Best Method**	**FARM**	**0.090**	**Best Method**
WARM	0.00636	23.43%	**SPIRIT**	0.0932	3.43%
Tiny DB	0.0085	42.71%	**WARM**	0.105	14.28%
SPIRIT	0.0116	58.02%	**Tiny DB**	0.137	34.31%
Avg	0.015	67.53%	**Avg**	0.1445	37.72%
MLR	0.121	95.98%	**MLR**	0.493	81.74%
SLR	0.342	98.58%	**CE**	2.253	96.01%
CE	0.346	98.59%	**SLR**	2.286	96.06%

4.4. Evaluation of Sensitivity

For lack of space, we do not include any sensitivity figures (see [15]). We note that FARM enjoys a good scalability in terms of network and state space size on the time and space metrics as well as the estimation error. Total average time cost is $O(\max(nd^2,ms,md))$ where m is the number of estimations. Total space cost is $O(sd^2)$.

5. Conclusions

This paper presents a data estimation technique, called FARM, which uses association rule mining to estimate missing sensor data in data streams. FARM recognizes the difference in importance between recent and old rounds of sensor readings and achieves this by appropriately decaying data of old rounds. It does this via the virtue of the data *freshness* concept, which incorporates a temporal factor in the estimation. The data *freshness* framework not only serves to assimilate the temporal element, but also offers a way of compacting the data stream and thus enhancing the space complexity and, in turn, the access time since compacted data is sufficient to perform estimation. Moreover, the *freshness* framework allows the retrievability of raw data when requested by queries. FARM is compared to a pool of statistical and algorithmic methods, which it outperforms on the RMSE measure. FARM's time and space performances easily assure a quality of service for sensor network applications. FARM also enjoys good estimation accuracy on datasets that naturally contain a temporal element as well as datasets where data recency is irrelevant through the flexibility in choosing the decay rate or damping factor.

References

[1] A. Dempster, N. Laird, D. Rubin, Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society,* 1977, pp. 1-38.

[2] P. Allison, Multiple Imputation for Missing Data: A Cautionary Tale, *Sociological Methods and Research,* 2000, pp. 301-309.

[3] A. Gelman, J. Carlin, H. Stern., D. Rubin. *Bayesian Data Analysis*, Chapman & Hall, 1995.

[4] J. Shafer, Model-Based Imputations of Census Short-Form Items. *Proc of the Annual Research Conference*, 1995, pp. 267-299.

[5] F. Dellaert, *The Expectation Maximization Algorithm*. Technical Report, February 2002.

[6] L. Wilkinson, Statistical methods in psychology journals: guidelines and explanations, *American Psychologist,* 1999, pp. 594-604.

[7] S. Ramakrishnan, Sensing the World. *Jasubhai Digital Media*, October 2003, pp. 26-28.

[8] S. Papadimitriou, J. Sun, C. Faloutsos, Pattern Discovery in Multiple Time-Series**,** *VLDB, 2005, pp .697-708.*

[9] S. Madden, M. Franklin, J. Hellerstein, W. Hong, TinyDB: An Acquisitional Query Processing System for Sensor Networks, In *TODS*, 2005.

[10] R. Agrawal, T. Imielinski, A. Swami, Mining Association Rules between Sets of Items in Large Databases, *ACM SIGMOD*, 1993, pp. 207-216

[11] R. Little, D. Rubin, *Statistical analysis with missing data*. New York: John Wiley & Sons, 1987.

[12] NASA/JPL Sensor Webs Project, http://caupanga.huntington.org/swim/, accessed Jan 2006.

[13] Austin Freeway ITS Data Archive, http://asutindata.tamu.edu/default.asp accessed Dec 2005.

[14] Halatchev, M., Gruenwald, L. Estimating Missing Values in Related Data Streams. *COMAD*, 2005, 83-94

[15] Chok, H., Gruenwald, L. Using Association Rules to Estimate Missing Sensor Data, Technical Report, University of Oklahoma.