Sensor validation with machine learning

Matt Smith[1] and Charles Castello[2]
[1] *The University of Alabama*
[2] *Oak Ridge National Laboratory*

(Dated: 26 April 2013)

Abstract: Since commercial and residential buildings account for nearly half of the United States energy consumption, making them more energy-efficient is a vital part of the nations overall energy strategy. Sensors play an important role in this research by collecting data needed to analyze building performance. Given this reliance on sensors, ensuring that sensor data is valid is a crucial problem. The solution we are researching is machine learning techniques. We have looked at two such techniques: artificial neural networks and fuzzy clustering. Artificial neural networks have been able to predict data, and thus correct data, for three of the five sensors we are investigating. Our implementation of fuzzy clustering as a validation tool was not as successful. Our method was able to cluster data into "correct" and "errant" clusters reliably, but only when the points in the "errant" cluster were three to seven standard deviations away from their correct value.

I. INTRODUCTION

Commercial and residential buildings are the largest consumers of energy in the United States, accounting for 41% of the nation's total energy consumption.[1] Clearly, improving building energy efficiency is one of the most important energy challenges we face, and better sensor technologies can help accomplish this goal. Sensors are used in buildings to analyze variables affecting energy use and efficiency. Given our reliance on sensors, ensuring that sensor data are valid emerges as a crucial problem.

Previous research into this problem by the Energy and Transportation Science Division at Oak Ridge National Laboratory has looked at statistical and filtering techniques.[2] These both use a sensor's past data to try to predict its future data. What we would like is a solution that can incorporate data from other sensors into the predictions. The solution we investigate here is a set of three machine learning techniques. Machine learning is a branch of artificial intelligence that studies software that can learn. We examine two such techniques in this work: (1) artificial neural networks and (2) fuzzy clustering.

In the remainder of this work, we first describe our data acquisition process and explain each of the machine learning techniques in detail. We then examine the results of the application of each technique, and finally we discuss what conclusion can be drawn from the work.

II. METHODS

????

A. Data acquisition

The sensors we used for this research came from ZEBRAlliance houses. ZEBRAlliance is a collaboration by Oak Ridge National Laboratory, Department of Energy, Tennessee Valley Authority, and industry partners to develop four of the most energy-efficient houses on the market. The houses were unoccupied during the study, but they were made to simulate national average energy expenditure. Each of the houses was equipped with over 250 sensors.[2]

We chose to study the data from five of the sensors of one house: (1) outside temperature, (2) outside humidity, (3) refrigerator energy, (4) heat pump water flow, (5) and heat pump liquid line pressure.These are called our target sensors. We did some of our initial testing with 1-min resolution data, but switched to 15-min resolution data to reduce the number of data points to consider.

B. Artificial neural networks

Artificial neural networks (ANNs) are programs meant to mimic the structure and function of the human brain. Both are made of layers of neurons. Real neurons, as seen on the left-hand side of fig. 1 on the following page, receive input from each neuron in the previous layer through their dendrites. Depending on the state of the inputs received, the neuron may then fire its own output to the next layer through its axon.[3] Similarly, the artificial neuron, seen in the right-hand side of fig. 1 on the next page, receives input from the neurons in the previous layer, which it uses to calculate a weighted sum f, as in equation (1).

$$f = \sum_{i=0}^{N} w_i x_i \tag{1}$$

The final value of the output y_i is the value of an *activation function*, g, with f as an argument. This is typically taken to be either a purely linear function or the logistic function, as seen in equation (2).

$$g = \frac{1}{1 + e^{-f}} \tag{2}$$

The range of the logistic function is [0,1]. It is similar to the unit step function, but with a "softer" transition from 0 to 1. It is meant to mimic the firing of a real neuron.

The unit step function is a better approximation of this, but the logistic function is used because it is easier to work with computationally.[4]

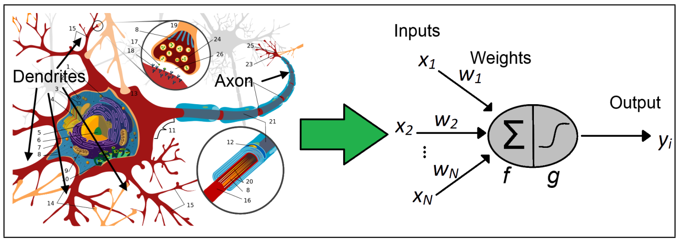

FIG. 1: A diagram of a real neuron and an artificial neuron.

ANNs are arranged into layers. A network typically consists of three layers: input layer, hidden layer, and output layer. First, the input layer feeds input to the hidden layer. The input layer does no calculations; thus, it is not actually composed of neurons. There is one node for each output. Next, the hidden layer receives from the input layer and outputs to the output layer. This is where most of the calculations are done. The activation function of neurons in the hidden layer is usually the logistic function. The number of neurons in the hidden layer is chosen by the user. Having more neurons yields better accuracy, but increases computational complexity. The hidden layer can contain multiple layers of neurons within itself. Finally, the output layer does a final round of calculations to produce the final output of the network. The activation function of neurons in the output layer is often taken to be linear. The output layer has one neuron for each output.

We wanted to use ANNs to predict data with a regression model. Our method was to use 85% of the data points we had to train the network and fit the model. The remaining 15% percent were used to test the model made during training. We used the root-mean-square error between the network output and the actual sensor data for those points as our metric.

C. Fuzzy clustering

Fuzzy clustering is a variant on classical set theory. In classical set theory, whether an element x of the universe of discourse X is a member of a given set A is given by the *characteristic function*, χ, of A,

$$\chi_A(x) = \begin{cases} 1 \text{ iff } x \in A \\ 0 \text{ iff } x \notin A \end{cases} . \tag{3}$$

In other words, an element either is a member of a set or it is not. Fuzzy set theory, on the other hand, allows for a continuum of membership values given by the *membership function*

$$\mu_A(x) : X \to [0, 1]. \tag{4}$$

When μ is zero, x is not an element of A. When it is one, x is completely a member of A. For values between zero and one, x can be said to be “sort of” an element of A.[5]

The difference between classical sets and fuzzy sets can be seen in fig. 2. The upper half is a classical set A inside a universe of discourse. The universe is crisply divided into regions of A and NOT A. In the fuzzy region in the lower half of the figure, however, there is a gradient of membership in A seen by the fading out of black.

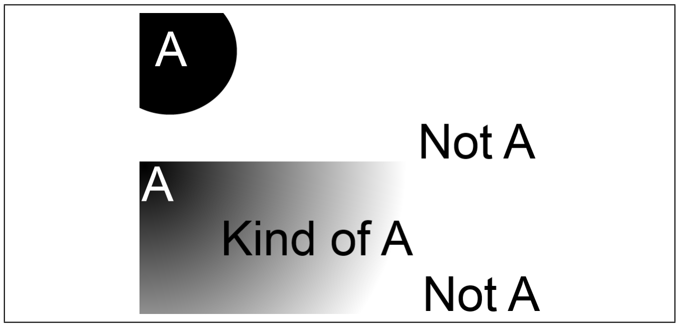

FIG. 2: Comparison of classical sets (upper half) and fuzzy sets (lower half).

We wanted to use fuzzy clustering to validate the data of target sensors. Our method was as follows. We used clustering to isolate incorrect, or “errant” points in their own cluster, leaving “correct” points in their own cluster. To make errant points, we performed the following operation to randomly-selected data points:

$$x_e = x + N\sigma \tag{5}$$

where x is the original value of the data point, x_e is the new, errant value, σ is the standard deviation of the original data set, and N is a scalar dubbed the *errancy factor*. By choosing random points to be errant and increasing N, we can see how far the data must be made errant before the fuzzy clustering algorithm will group errant points mostly (or entirely) into their own cluster.

To test how well this clustering happened, we counted the total number of data points in each cluster, T, and the number of errant points in each cluster, e. Ideally, for the correct cluster we should have $e/T = 0$, and for the errant cluster we should have $e/T = 1$. To determine whether a given point was “in” a given cluster, we used a threshold value, t. If for a given point x it is true that $\mu_{errant}(x) > t$, then x is in the errant cluster. Similarly, if $\mu_{correct}(x) > t$, then x is in the correct cluster. Note that as long as t is greater than 0.5, a point cannot be in both clusters, since $\mu_{errant}(x) + \mu_{correct}(x) = 1$.

We used the fuzzy c-means algorithm as described by Bezdek et al.[7]

III. RESULTS

A. Artificial neural networks

ANNs proved to be quite effective at predicting data for three of the five sensors: temperature, humidity, and pressure. As fig. 3 on the next page shows, for these three sensors, the root-mean-square error (RMSE) between the network output and the actual target data for 500 networks was mostly less than 8% of the respective sensor's mean value.

For the other two sensors, liquid flow and refrigerator energy, this was not the case, as fig. 4 shows. For the energy sensor, the RMSEs are mostly around 152%. We suspect this poor result is due to the energy sensor's somewhat erratic distribution of values. For the flow sensor, its RMSEs are nearly all very low. However, this sensor has many readings of 0 – dozens in a row, in fact. With this sort of distribution, a constant 0 function would probably yield a low RMSE, but may not necessarily reflect a good prediction.

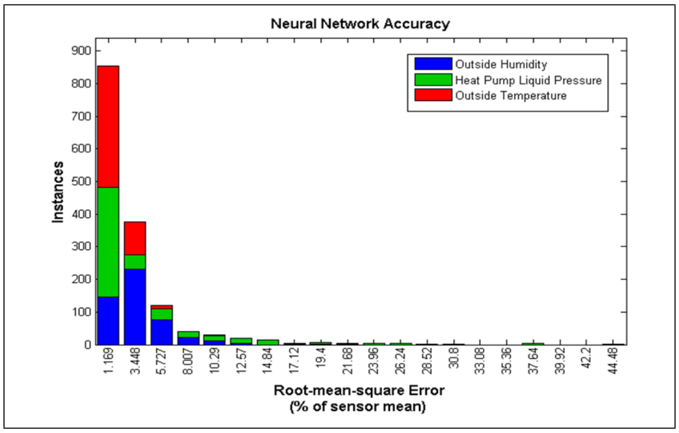

FIG. 3: Histogram of root-mean-square error (RMSE) of 500 networks each for temperature, humidity, and pressure sensors. The RMSE values are expressed as a percentage of their respective sensor's mean value.

B. Fuzzy clustering

Our fuzzy clustering validation method proved not to be useful. In fig. 5, we see the concentration of errant points, e/T, as a function of the errancy factor for the correct and errant clusters with differing values of t for the temperature sensor. In this particular simulation, 20% of the total population was errant. As the figure shows, the concentration for the errant cluster does not get close to 1 until the errancy factor is about 4, meaning the errant data is about four standard deviations above its original value. When the concentration is not near 1, there are still a lot of correct points in the errant cluster. This means that this validation method is not effective until a large chunk of data are off by several standard deviations. A validation method that only works when data is wrong by that amount isn't of much use. We obtained similar results with the other sensors. When we decreased the number of errant points, the data had to be even more wrong before the errant points were clustered by themselves – as high as seven standard deviations when 5% of the population was errant.

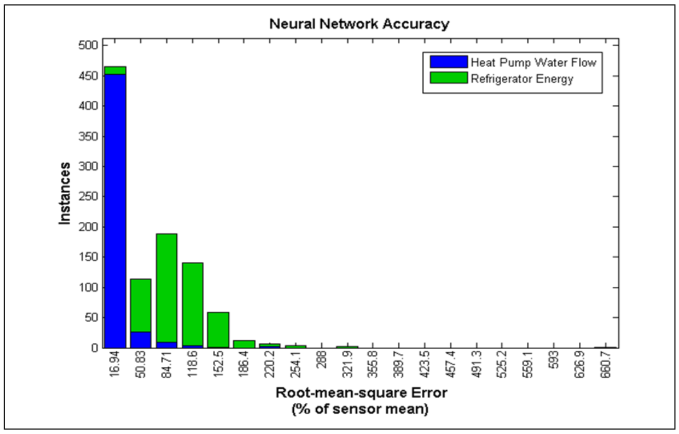

FIG. 4: Histogram of root-mean-square error (RMSE) of 500 networks each for liquid flow and energy sensors. The RMSE values are expressed as a percentage of their respective sensor's mean value.

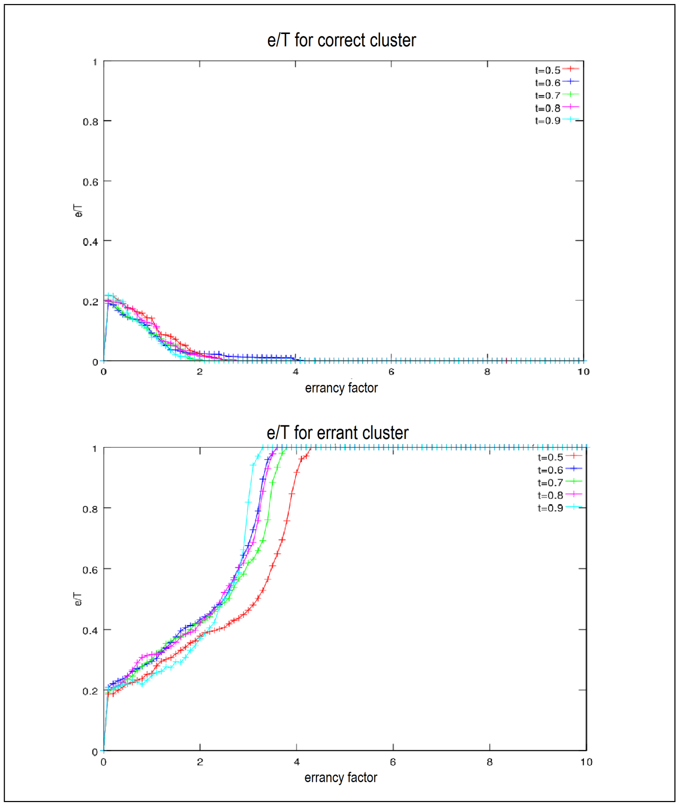

FIG. 5: Concentration of errant points, e/T, for correct and errant clusters

IV. CONCLUSIONS

In this paper, we have shown that artificial neural networks can be good predictors of sensor data for some sensors. We demonstrated a fuzzy clustering validation method which was unsuccessful.

V. ACKNOWLEDGEMENTS

This work was supported in part by the U.S. Department of Energy, Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships program (SULI).

[1] Department of Energy, "Buildings energy data book," [Online], Available: `http://buildingsdatabook.eren.doe.gov/ChapterIntro1.aspx`.

[2] C. C. Castello and J. New, in *Energy Informatics* (Atlanta, Georgia, 2012).

[3] `http://www.intropsych.com/ch02_human_nervous_system/how_neurons_communicate.html`.

[4] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Prentice Hall, 1995).

[5] L. H. Tsoukalas and R. E. Uhrig, *Fuzzy and Neural Approaches in Engineering* (Wiley-Interscience, 1997).

[6] K. Murphy, "A brief introduction to graphical models and bayesian networks," `http://people.cs.ubc.ca/~murphyk/Bayes/bnintro.html`.

[7] J. C. Bezdek, R. Ehrlich, and W. Full, Computers & Geosciences **10**, 191 (1984).