Dissertation

Brenton Cromwell

September 2018

Contents

1 Introduction

1.1 Energy Lancaster Systems and Services

The application of machine learning ,and other emerging technologies ,to energy systems is a topic that is being researched and implemented all over the world. This has resulted in the production of so called "smart meters" that collect the a massive amount of data that aids in the application of machine learning techniques. However, what do we do when an energy system consists of sensors that date back to the 1960's and were never built to be easily compatible with big data analysis. This is the system that exists at Lancaster University and is overseen by Energy Lancaster.

Energy Lancaster is a cohort of professors, researchers, and students whose goal is to develop new energy solutions as well as optimizing current, antiquated systems. To achieve the later goal Energy Lancaster is updating their system with modern machine learning, database architecture (Hadoop, Spark, etc), and optimization techniques. This is all in attempt to understand the data collected, bring down energy costs and develop new techniques in energy regulation.

1.2 Current State of Energy Lancaster Systems

Energy Lancaster presides over all energy usage and generation at Lancaster University. To monitor this vast and complex system over 300,000 sensors record data every ten minuets. These sensors cover everything from the position of a switch that controls a fan in a restaurant to a thermometer that records the temperature inside of a classroom. The complexity and diversity of this system is the largest stumbling block when it comes to applying statistical and machine learning techniques. On top of the complexity of the raw data, the meta data recorded on each of the sensors was unclear and unstructured. This added an additional dimension of complexity that needed to be addressed before any machine learning techniques could be applied. This process is outlined in chapter 2.

1.3 Similar Problems In Energy and Time Series Data

Talk about the personal sensors

1.4 Aims and Objectives

At the start of the project, Energy Lancaster outlined two main issues to be addressed:

1. Leveraging metadata to find new insights.
2. Quantifying the quality of data being collected by sensors.

To address these problems, this paper will introduce two novel approaches in turn.

1.4.1 Meta-data Analysis and Enrichment

In the process of exploratory data analysis of the metadata files an additional problem was discovered. The information recorded was unstructured and in many places unreadable by humans. This is further addressed in chapter 2 but it did add a third problem that is addressed in this section.

Objectives:

- Analyze the metadata for quality and completeness.
- Find a way to correct/update the metadata files
- Determine a method to classify and cluster sensors.

1.4.2 Data Quality Analysis

In such a large data set with so many varied data streams being collected it can be rather challenging to determine the quality of the incoming. For instance you could have a sensor that is reading 0 if it is a switch that is hardly every turned on it is an expected behavior but if it's a room temperature sensor in the middle of summer there is probably something amiss.

Objectives:

- Develop a method to analyze incoming data and determine its quality/

- Develop algorithms to find inconsistencies in data that could point to poor data quality.
- Put a system in place to flag poor data streams that require attention of system controllers.

1.5 Report Overview

Since there are two problems addressed in this project the layout of the report will be as follows. Chapter 3 will introduce the data sets and go over the generic data cleaning and feature engineering methods applied to both projects. This is done to eliminate the need to repeat the process in either project's chapter. Chapter 4 will go through the unique background, methods, results, and discussion related to the metadata and classification problem. Chapter 5 will mirror the sections of chapter 4 but will present the data quality problem. Chapter 6 will contain an overall discussion and conclusion of the projects, detail how they interact, and proposed future work.

2 The Data: Layout, Cleaning, and Feature Engineering

2.1 Data Sets

Energy Lancaster collected data is stored under the files of Building Management Systems (BMS), Energy Management System(EMS), Wi-fi Data, Weather Station Data, and Synetica Data depending on the type, location and purpose of the sensors. All these data sets are saved on a Comprehensive Knowledge Archive Network (CKAN) behind the universities firewall. While in future all of these data sets will be considered in the process of optimizing the energy system, this project is only concerned with BMS and EMS.

	BMS	EMS
Size/Month	5Gb	5Gb
Amount of NA/ Missing Data	50	1
Time Between Collections	10 min	30 min
Type of Collector	Instantaneous Values by Controller	Cumulative Valuesby Pulse Logger

Table 1: Comparison of BMS and EMS data

2.1.1 Building Management System (BMS)

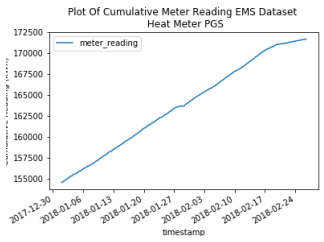
As we can see from 1 the BMS data is the larger and more complex of the two. It is collected more frequently and from many more sensor types. This data also had multiple layers of identification codes. There was a Logger Asset Code that identifies the logger that is collecting the data. However each logger has multiple sensors attached to it so each sensor is given a Logger Channel Number. These two numbers were combined to produce a new "unique key" that could be used to separate out each unique sensor. To build this "unique key" we first had to take the Logger Asset Code from its original form (0F850990-984A-49FC-B351-82CCFD6A644B) and place it in format that would not present any issues of being read by python code as a string. To do this a simple function was applied to remove all symbols and leave just numbers and letters. To complete the key we simply added the Logger Channel Number to the cleaned Logger Asset Code. This new unique key does not only allow for the separation of data by sensor at the lowest level but it also allows for the data to be bound to the metadata files.

The BMS data for the whole campus for one month had over 43 million rows of data which made performing a simple join (in this case left join) with the metadata file a computationally expensive task. In fact, the computer being used for these calculations had 16Gb of RAM and an i7 processor and it was still not able to complete the join given 5 hrs of run time. In order to counteract the issue with the size of the data a new strategy was required. Instead starting with the metadata file, a building was selected to be the focus of this project. Then the unique key was created using the same method for the BMS data. The relevant keys were then recorded and used to slice the BMS data. Once the dimensionality was reduced by taking only one building worth of data, the metadata was easily joined to create a complete data file.

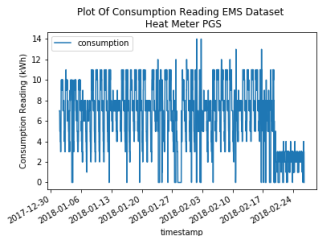
2.1.2 Energy Management System

The EMS data is much cleaner than the BMS data and has a unique key built in called "metadata id", but it has its own drawbacks. It is a cumulative sensor which means that only two readings are recorded: the change in the reading over the previous 30 minutes and the cumulative value on that meter. The difference in these values can be seen in 2.1

Since the EMS data is much smaller than the BMS data no additional methods were needed to reduced the size of the data files. A simple left join was utilized to bind the data with the metadata and in this project three months



(a) Example of cumulative value



(b) Example of consumption value

Figure 2.1: Examples from EMS Data

of data is considered.

2.2 Generic Data Cleaning

Before any cleaning is done to the data it is important to first explore the data to find any easily found quality issues. For instance (INSERT FIGURE WITH FLAT LINES) is an prime example of the type of data that the data quality process will want to discover. Using algorithm (name algorithm) determines the number of non zero first derivative values exist in the data. This determines if the data is flat or not. By taking a step further and applying it to every month for each sensor we are able to see if the sensor has any strange behavior like in (POINT TO THAT FIGURE AGAIN)

After the binding the data and recording all the pre-cleaned data characteristics, both data sets were standardized by using the pandas time series function resample which takes a times series and places all the points at a certain time interval. This is useful as it creates a uniform time difference between all points, simplifies the time stamps (10:11:04 becomes 10:10 for example), and fills any missing data points using interpolation.

Once the data is in the pandas time series indexed format and the missing values are interpolated we can move on to normalizing the data. For both data sets the sensor collected data was normalized to values between 0 and 1 using (Equation 1)

2.3 Generic Feature Engineering

Both the BMS and EMS data sets are time series which means that several additional features need to be created in order to aid the machine learning process. The mean, standard deviation, first derivative, second derivative,

day, month, week number, day of the week, hour, second, and minutes were calculated using functions built into pandas.

2.3.1 Data Frame Types

Finally, the natural way of representing this data is as a time series. However, time series data creates a bit of a challenge when it comes to classification and clustering. Therefore a lower dimensional representation of the data was implemented. In the this lower dimensional version the data is grouped by sensors and the data is represented by their descriptive statistical values. This layout also allows for information about the sensor to be stored once in the row representing that sensor, which aids in visualization.

This paper will deal primarily with the lower dimensional data in chapter 3 and the time series data in chapter 4.

3 Classifier System

[?]

3.1 Background Research

In this chapter both an ensemble classifier and an unsupervised clustering algorithm are introduced along with all the machine learning algorithms involved in the ensemble.

3.1.1 Cost Function Optimization Algorithms

Before the introduction of the various algorithms and theories utilized in this paper it would be useful to briefly go over some of the optimization algorithms employed by sklearn to learn the parameter weights for several of the machine learning techniques.

1. "adam"
2. "sgd"
3. "sag"

4. "saga"
5. "lbfgs"
6. "local search"
7. "argmax"

3.1.2 Ensemble Learning

Ensemble learning is based on the idea that several weak learners together can outperform not only each of those learners individually, but also outperform more complex models. According to Dietterich(CITE HERE) there are three main reasons that the ensemble system, in theory, should have this heightened performance.

- The first reason is a statistical one. In the hypothesis space (space where the desired solution resides) the ensemble classifier will cover more area around the desired solution. Even though none of the individual solutions may land on the optimal solution the average of all the ensemble outputs will be much closer.
- The second reason arises from the way these learning algorithms are trained. They all rely on some version of gradient decent, local search, or other maximization/minimization method. These methods are very susceptible to getting stuck in local minimums instead of the desired global minimum. This results in non-optimal solutions. This problem has several proposed solutions but they are often complicated and take a lot of computation time. Instead, if you use an ensemble system starting from several different start points in the cost function space, you can get a better idea as to which point the algorithms are converging towards.
- The last reason is a bit more abstract but it comes from a well known principle of representing a point in space by the combination of several, more base elements. For instance, a sound wave can be a very complex thing that can have multiple periodicity's, trends, and amplitudes. However, in Fourier space these complex waves can be represented by several more basic waves. The same thing can be seen in ensemble learning. If the solution exists in an obscure region of the hypothesis space, a combination of solutions in a more basic region could help describe it.

The fist step in constructing and ensemble system is the select the algorithms that will be used to vote. Hanson and Salomon (CITE) state that the algorithms used for ensemble learning should be "diverse and accurate". They

go on to define "accurate" as meaning the algorithm does better than random guessing on new data. This ensures that the algorithms included are capable of contributing to an increase in the ensemble classifier accuracy. The "diverse" requirement ensures that the algorithms approach the solution in different ways in order to cover the weaknesses of one another by requiring that the algorithms "make different errors on new data points"(CITE).

The second step in building an ensemble classifier is to determine the method for combining the results of the individual learning algorithms. For the purposes of this paper two methods of "voting" were considered.

1. **Hard Voting:** This method tallies up the classifications from each of the learning algorithms in the ensemble and picks the one with the most votes. This method is straight forward but is weak if using an even number of classifiers (potential tie).

2. **Soft Voting:** Also known as "weighted average probabilities voting", soft voting does just that. First, a weight is assigned to each classifier either by hand or through a scoring algorithm based on accuracy. Next the probabilities are extracted from the algorithm for each classifier. Finally, the probabilities are multiplied by the weights and averaged for each classifier. A simple argmax algorithm (appendix) is then used to determine the winning class.

In this paper both hard and soft voting are explored in order to determine the most viable option.

3.1.3 One Vs All Logistic Regression

Logistic regression is a supervised learning algorithm that comes from the family of generalized linear models (GLM). However, unlike most of its brothers and sisters it is used for classification instead of regression. The GLM family all start out with a linear hypothesis function (3.1):

$$hypothesisfunction = \alpha + \beta^T x_i \tag{3.1}$$

$$where : \alpha + \beta^T x_i = \alpha + \beta_0 x_0 + \beta_1 x_1 + ... + \beta_n x_n$$

α = bias value

β = vector of parameters

x = vector of feature values

i = number of data inputs/outputs

In basic linear regression this hypothesis function is then fit to the data by updating the parameters in the β vector by first building a cost function and then using a multidimensional minimization/maximization method like gradient decent[maybe put a link here] to update the values. In the case of logistic regression you instead apply a link function to the hypothesis function in order to transform the outputs to values between 0 and 1 that can be interpreted as probabilities. It does this with the inverse logit link function (NOTE: also known as the Sigmoid function) (3.2):

$$logit^{-1}(t) \equiv \frac{1}{(1+e^{-t})} = \frac{e^t}{(1+e^t)} \tag{3.2}$$

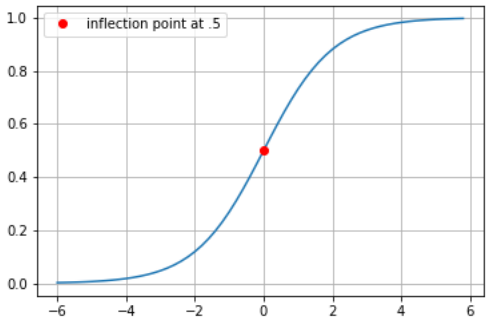


Figure 3.1: Decision Boundary Shape for Inverse-Logit Function

As seen in 3.1 the inverse logit function forces the input (along the x axis) to return a value on the closed interval [0,1]. It is this characteristic that makes logistic regression so ideal for classification. Formalizing the conditional probability of y given the data we find:

$$P(y_i|x) = [logit^{-1}(\alpha + \beta^T x_i)]^{y_i} * [1 - logit^{-1}(\alpha + \beta^T x_i)]^{1-y_i} \tag{3.3}$$

$$\text{with i} \in \{0, 1\}$$

$$\therefore P(y_i|x) = \begin{cases} P(y=0|x) = logit^{-1}(\alpha + \beta^T x_i) \\ P(y=1|x) = 1 - logit^{-1}(\alpha + \beta^T x_i) \end{cases} \tag{3.4}$$

Now looking at (3.4) notice that it has a very similar form to the general Bernoulli distribution (**??**) , treating the sigmoid function output as a probability, and as such writing down the maximum likelihood estimation is simple (3.1.3) and this will be our cost function.

$$g(x, \beta) = logit^{-1}(\alpha + \beta^T x_i) \tag{3.5}$$

Simplification of inverse logit function notation

$$J(\beta) = -\frac{1}{m} \sum y_i log(g(x_i, \beta)) + (1 - y_i)(1 - log(g(x_i, \beta))) \tag{3.6}$$

MLE solution of Bernoulli distribution with (3.1.3) as probability

Looking at the intuition of (3.1.3) shows that in the case of y = 1 and the equation (3.1.3) returns a value close to 1 the overall cost will approach zero. The cost function not only behaves exactly as desired but it is also parametric. This means that applying basic gradient decent will be sufficient for finding the parameter values. [PUT IN SOMETHING ABOUT GRAD DECENT?]

The last part that is needed for the application of logistic regression to the Energy Lancaster data is to adopt a One vs All approach so the algorithm can be used to classify over multiple classes. The process is described in algorithm BLAHHHH PUT ALGORITHM IN HERE

3.1.4 Decision Tree

The decision tree algorithm is based on information theory as opposed to logistic regression which is based on errors and k-means which is based on similarity. It can be used for both classification and regression, this paper will address only classification. The anatomy of a decision tree can be seen in figure(BLAH). It consists of interior, leaf and a root nodes that are connected by branches.

- A root node is defined as having no incoming branches and zero or more outgoing branches
- An interior node is defined as having one incoming branch and two or more outgoing branches
- A leaf node is defined as having only one incoming branch and zero outgoing branches
- (PuT REFERENCE TO BOOK HERE)

The example in figure (BLAH) represents a data set used to determine, based on weather, if a game of football should be played or not. The toy data set has mixed variables and a binary response.Figure 3.2 also shows two possible outcomes for a decision tree built on the data. Both trees are equally accurate when predicting data but

Tree 1 is more efficient. It correctly outlines the fact that Sunny and Rain are essentially the same feature, since the data does not show a time where it is both sunny and raining. This means that it is possible to exclude sunny when using rain or vise-versa.

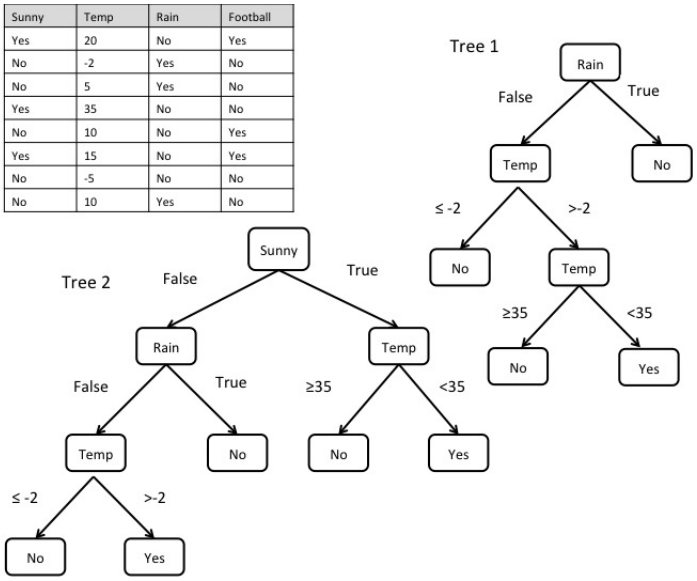


Figure 3.2: Example data set with two possible decision tree outcomes

The splitting of the tree at each node is determined by the impurity or heterogeneity of the resulting child nodes. The splits that result in child nodes containing a higher purity of data are favored over others. In the decision tree family there are several metrics used to quantify this impurity but in this case entropy is used. Entropy comes from Shannon's information theory and measures information gain in bits. It was inspired by the equation of the same name from the study of thermodynamics. Shannon's entropy is represented by (3.11).

$$H(t) = -\sum_{i=0}^{c-1} p(i|t) \log_2 p(i|t) \tag{3.7}$$

$$where:$$

$p(i|t)$represents the fraction of the data belonging to class i at node$t$

$$c = \text{number of classes}$$

The decision tree algorithm uses this entropy metric to create all nodes and branches.

3.1.5 K-Means

The unsupervised learning algorithm K-means groups data by its proximity to centroids (points that will be at the center of your cluster). The number of centroids indicate the number of groups and is set by the user. The algorithm follows this pattern:

K-Means Algorithm

1. Set k number of centroids randomly (k set by user).
2. Calculate the distance between all points and each centroid. Assign each data point to a centroid.
3. Move the centroids to the average center of all the data points assigned to it.
4. Repeat the above steps until the change in position of the centroids is negligible (a threshold can be manually set).

the distance between the points and the centroids is calculated numerous different ways but for most cases a simple Euclidean distance is used.

3.1.6 Gaussian Mixture Models

As stated above, GMMs build upon the basic framework of the K-means clustering theory. Instead of having the centroids be random points in space, GMMs place Gaussian distributions at those points. This allows for a probabilistic approach to clustering that can add additional layers to analysis. For instance, image you have a point that has been clustered in cluster A but it sits near the boarder of cluster B. In k-means it would be given the same output as a point that is located near the centroid of cluster A and thus the user would have to use visualization techniques to know that it was very near cluster B and might be wrongly classified. However, in multidimensional space this visualization can be very challenging. Instead if the user chose to implement GMM for clustering the point would have been given a probability of existing in cluster A and cluster B. The user could easily notice that the point is on the boarder of two clusters, a fact that could be pivotal to understanding the data.

With this increased functionality comes a more complex method. The probability distribution for each point is parameterized by three values: mean μ_k, variance/co-varianceσ_k, and the mixture component weightsϕ_i. The mean

and variance should be familiar but the components weights are the probabilities of the data point coming from each Gaussian centroid in the mixture model and has the constraint that $\sum_{i=1}^{K} \phi_i = 1$. This ensures that each point is 100 percent represented by a combination of all centoroids. The number of centroids K is set by the user just as in the k-means algorithm. To build the model formally:

$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x}|\vec{\mu_i}, \Sigma_i) \tag{3.8}$$

$$\mathcal{N}(\vec{x}|\vec{\mu_i}, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} exp(-\frac{1}{2}(\vec{x} - \vec{\mu_i}^T \Sigma_i^{-1} (\vec{x} - \vec{\mu_i} \tag{3.9}$$

$$\sum_{i=1}^{K} \phi_i = 1 \tag{3.10}$$

It now comes down to how this model is trained. Gaussian mixture models are trained using a method called expectation maximization(EM). EM is described as "a popular tool for simplifying difficult maximum likelihood problems" in (SPRINGER). (SPRINGER) also contains a great representation of the EM algorithm for a GMM.

EM Algorithm

1. Initialize $\vec{\mu_i}$, $\vec{\Sigma_i}$, and ϕ_i randomly.

2. Expectation step: compute the responsibilities

$$\gamma_{ik} = \frac{\phi_k \mathcal{N}(\vec{x_i}|\vec{\mu}_k, \vec{\Sigma}_k)}{\sum_{j=1}^{K} \phi_j \mathcal{N}(\vec{x_i}|\vec{\mu}_j, \vec{\Sigma}_j)} \tag{3.11}$$

3. Maximization Step: compute the weighted means and variances:

$$\phi_k = \sum_{i=1}^{N} \frac{\gamma_{ik}}{N} \tag{3.12}$$

$$\vec{\mu}_k = \frac{\sum_{i=1}^{N} \gamma_{ik} x_i}{\sum_{i=1}^{N} \gamma_{ik}} \tag{3.13}$$

$$\vec{\sigma}_k = \frac{\sum_{i=1}^{N} \gamma_{ik} (x_i - \vec{\mu_k})^2}{\sum_{i=1}^{N} \gamma_{ik}} \tag{3.14}$$

$$\vec{\Sigma}_k = \vec{\sigma}_k \mathbb{I} \tag{3.15}$$

4. Iterate steps 2 and 3 until convergence

By using the EM algorithm the multiple parameters of the GMM can be found.

3.1.7 Neural Network

Inspired by the inner workings of the neurons found in the human brain, neural networks were first introduced in 1943 by Warren S. McCulloch and Walter H. Pits (insert citation here). Though the theory was proposed in the 1940's it was tabled until the late 20th century due to the lack of computation power available. Now practitioners have access to GPUs and parallel computing which allow for the development of new types of neural networks and the emergence of deep learning(cite). In this instance a multi-layer perceptron (MLP) (feed forward neural network) was employed for classification.

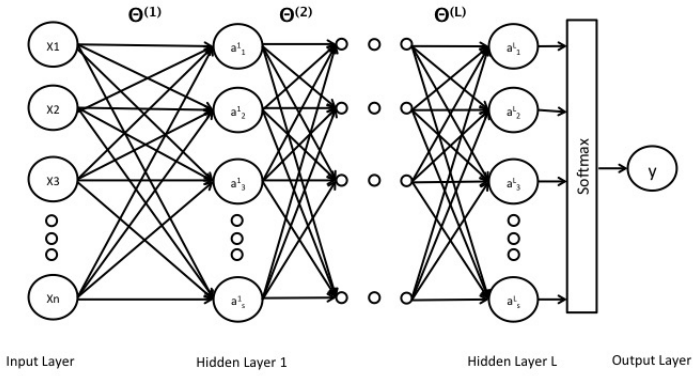


Figure 3.3: Multi-layer Perceptron

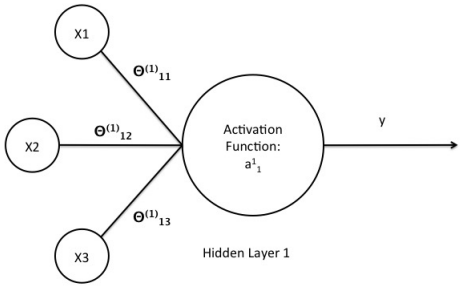


Figure 3.4: Single Neuron Example

First figure 3.3 presents the full anatomy of a MLP and acts for as a reference as the notation used is explained.

First set the notation, referencing the full MLP in figure 3.3 a MLP usually has one input layer with n number of cells where n is the number of features in each row of the data set. The each cell in the input layer is then fully connected to each neuron in the first hidden layer. In figure 3.3 L is the total number of hidden layers in the MLP. Each hidden layer has s neurons in it, s is again a hyper-parameter that again needs to be tuned using an optimization method. Each neuron in the hidden layer applies an activation function to all incoming values and the weights found in the Θ matrix. The basic activation function used is a sigmoid function (input reference to sigmoid) but there is a whole family of other activation functions that excel in different circumstances. Lastly, to explore the notation for the activation neurons and the $Theta$ weight matrix it is easier to reference figure 3.4 and look at the mathematical notation:

$a_s^{(l)} =$ activation neuron

$\Theta_{ij}^{(l)} =$ weight matrix

where: $l = 1, 2, 3....., L$

where: $L =$ number of layers

$i =$ ith neuron in layer l

$j =$ jth neuron in layer $l-1$ that is sending its value to neuron i

$s =$ activation neuron$s$in layer l

The first step in training a neural network is to perform forward propagation. In the case of neural networks it is convention to use 'forward' to define processes where the algorithm moves from left to right (while looking at figure 3.3) and 'backward' for algorithms that move from right to left. Forward propagation in this case will mirror the calculations of logistic regression introduced in 3.1.1 minus the gradient decent. Mathematically it is expressed:

$$define: \ g(x) = \text{activation link function}$$

$$z_i^{(l)} = \Theta^{(l-1)} X \text{if } l = 1 \tag{3.16}$$

where: X is the input vector

$$z_i^{(l)} = \Theta^{(l-1)} a^{(l-1)} \text{if } l \text{ ¿ } 1 \tag{3.17}$$

$$a_s^{(l)} = g(\Theta_{s0}^{(l-1)} x_0 + \Theta_{s1}^{(l-1)} x_1 + ... + \Theta_{sn}^{(l-1)} x_n) \tag{3.18}$$

$$\therefore a^{(l)} = g(z^{(l)}) \tag{3.19}$$

These equations give an outline of the forward propagation process. This continues through the hidden layers until the softmax layer is reached. It is in this layer where the one verse all approach of logistic regression is mirrored. A probability is calculated for each of the possible classification outcomes and the one with the highest probability is then sent through to the output node. It is only once the output node is reached that the back propagation process begins.

Back propagation is the process by which the weights of the Θ matrix are updated and the MLP is able to become a predictive tool for classification. It is achieved by determining how a change in particular weights will effect the cost function. As an example a cost function $C_0 = (a_j^{(l)} - y_j)^2$ is used. This cost function represents the "error" between the last hidden layer, before softmax, and the known solution of the problem. In the case of multi-class classification the last hidden layer outputs are considered probabilities of the input belonging to each class and the y vector will contain a 1 for the correct class and 0's for all others. To explore how this cost function is affected by the terms in the previous layers a partial derivative is used:

$$C_0 = (a_j^{(l)} - y_j)^2 \tag{3.20}$$

Taking the derivative with respect to the weights and applying the chain rule:

$$\frac{\partial C_0}{\partial \Theta_{ij}^{(L)}} = \frac{\partial z_i^{(L)}}{\partial \Theta_{ij}^{(L)}} \frac{\partial a_s^{(L)}}{\partial z_i^{(L)}} \frac{\partial C_0}{\partial a_s^{(L)}} \tag{3.21}$$

Taking derivatives from (3.8)(3.9) and (3.11) we get:

$$\frac{\partial C_0}{\partial \Theta_{ij}^{(L)}} = 2a_s^{L-1} g'(z^{(L)})(a^{(L)} - y) \tag{3.22}$$

By combining forward and back propagation it is possible to train a MLP to learn to correctly classify data. The last step in for a basic MLP has to do with weight initialization and hyper parameter fitting. Weight initialization has to do with what values the Θ matrix begins with. If they all start at zero it is rather challenging for the MLP to learn anything (as all the equations in forward propagation will be multiplied by zero). To avoid any problems in the training process it is common practice to initialize the weights randomly. One of the oldest techniques for achieve this was to sample the weights from a normal distribution. Even this basic step vastly aids in the training of the MLP.

3.1.8 Determining the Validity of Classifiers

The performance ensemble classifier, and all the classifiers contained within, will be measured by three metrics:

- **Accuracy** is defined, in this instance, as the number of correctly classified sensors divided by the total number of sensors in the data set and is represented by equation (3.23).
- **Precision** is defined as the number of true positives divided by the total number of values predicted to be positive. In other words, it is the percentage of values predicted to be sensor type 1 that are actually sensor type 1. Mathematically it is represented as (3.24).
- **Recall** is defined as the as the number of true positives divided by all the positives in the data. Put another way,it would be the number of sensors correctly identified as sensor type 1 out of all the sensor type 1s in the data. Represented by equation (3.25).

$$Accuracy = \frac{\sum TruePositive + \sum TrueNegative}{TotalPopulation} \tag{3.23}$$

$$Precision = \frac{\sum TruePositive}{\sum TruePositive + \sum FalsePositive} \tag{3.24}$$

$$Recall = \frac{\sum TruePositive}{\sum TruePositive + \sum FalseNegative} \tag{3.25}$$

The problem with only using accuracy is that it gives no insight into proficiency of the classifier to identify true/false negatives or positives. This is the reason for the inclusion of precision and recall as metrics. In various real world applications precision and recall can dictate the way that the algorithms are trained and optimized. For instance, while working to classify bank fraud it is much more desirable for the trained algorithm to better perform when scored with recall (assuming a fraudulent charge is considered a positive value) then precision. This is because failing to flag a fraudulent transaction (a false negative) would be much more detrimental the bank than flagging a normal transaction as fraudulent (false positive). In the second case a customer might be inconvenienced but that is preferable to the alternative.

However, in the case of the classification of sensors precision and recall played a different role. In this case there

was no extreme detrimental effects from miss classifying the sensors so the precision and recall were introduced to give the system administrators and idea as to how well the trained ensemble classifier performs when it comes to different sensors. This is a crucial piece of information as it dictates the amount of faith given to the algorithm. As for their use in model selection, it was advantageous to select models who performed similarly in both precision and recall.

3.2 Methodology

The method introduced in this chapter is designed to not only gain insight into the data from what little usable metadata exists, but also aims to update the existing metadata files in order to aid future analysis.

In order to do this a system combining an ensemble classifier, unsupervised clustering, and expert reinforcement was constructed ("EUR" for short).

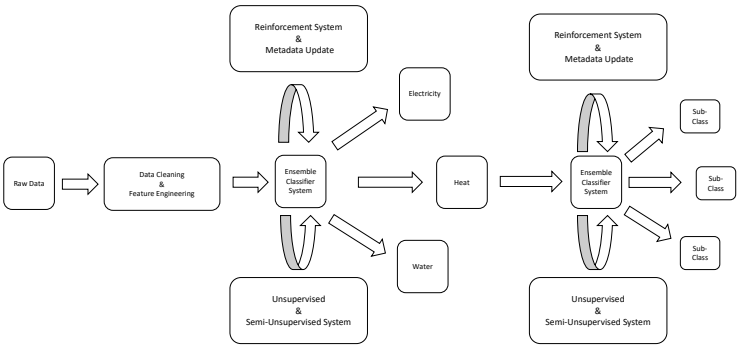


Figure 3.5: Flow Diagram of Classification System

The pipeline in 3.5 shows that the data went through the cleaning and feature engineering phase (pre-processing) before entering the "EUR". The data was then labeled using a basic system unique to EMS and BMS. For the EMS data the highest labels in the hierarchy were clear (INSERT FIGURE OF EMS DATA). Basic natural language processing (NLP) techniques were used to label sensors that had clear labels (POINT TO CODE). This only labeled about 30% of the EMS sensors. For the BMS data the initial labels were less clear. Instead another NLP technique was used to find the frequency of words found the descriptions of the BMS sensors. Excluding number labels and labels consisting of only one letter, the 5 most frequent words with no overlaps were chosen for the initial labels.

The ensemble classifier was then trained on this data and the accuracy, precision, and recall tested and recorded. At this stage the chosen classification algorithms that make up the ensemble classifier are re considered to ensure that they fulfill the diversity and accuracy requirements. In this case the MLP, OvR logistic regression and entropy decision tree were chosen. The trained ensemble classifier was then used to classify all the sensors in the data frame.

Next the GMM applied to both the parameters of the lower dimensional data frame and the full time series data for each sensor. The two different clustering were performed as a check to determine 1) the usefulness of GMM on time series 2) to determine which would perform better compared to the classifier. A separate clustering assignment was given for each process and recorded in the appropriate rows. The clustering was then checked against the classification to determine if the sensors classified together by the ensemble system were also clustered together by the GMM. A simple "YES" or "NO" was assigned to each sensor to determine if its clustering matched its classification. This was done by a script that would compare a group of sensors that had been classified together and see of those sensors had also been clustered together.

There are then two outputs that were sent to the expert.

- A csv file of the data frame including: sensor id, sensor description, statistical parameters, location, and classification prediction.
- A directory containing a file for each centroid trained by the time series GMM. Each file has images of each time series in the cluster graphed with the centroid of that cluster and a label of which sensor it is.

These files are then used for the reinforcement part of the "EUR". The csv file is available for the expert to peruse in excel and correct any labels they may see that are glaringly wrong. Any corrections made are saved to a excel file to be recycled back into the pipeline. The image files are viewed using a UI (currently under construction at Energy Lancaster) and using the principles of semi-unsupervised learning an expert is able to label a whole cluster of images very quickly. These results are recorded and also returned to the pipeline in the form of a text file.

The returned data is then integrated with the metadata file, with preference given to areas changed by the expert, and the training process is then repeated with a fresh ensemble system. The outputs of this system are then compared with the older one to ensure the metrics of accuracy are improving.

The unsupervised learning system may seem a bit redundant in the first stages of this "EUR" but it will become

more relevant in the future of this project (point to future work). It will allow for label easy labeling of sensors for classification further down the hierarchy of sensors.

3.3 Results

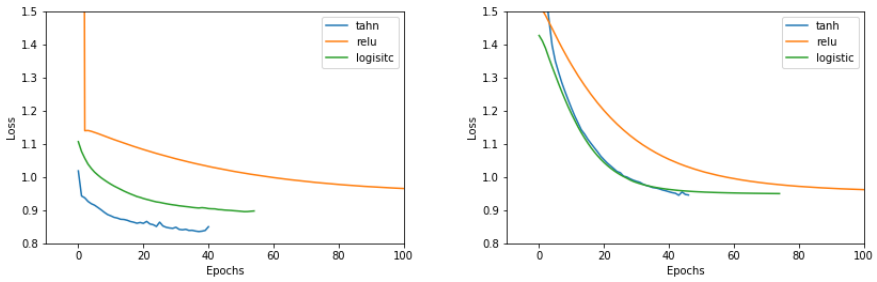
This section will present the results of hyperparameter fitting for the algorithms involved in the ensemble classifier before providing the remaining results. The tuning for the decision tree will be absent as the only hyperparameter of interest is which method of splitting will be used and due to its roots in information theory this approach will only be concerned with the entropy method as introduced in 3.2.3.

3.3.1 Neural Network

As stated in section 3.1.6 the MLP has several hyper-parameters that need to be fit before it can become part of the ensemble classifier. Using the sklearn neural network library only provides three hyperparameters that are in need of tuning:

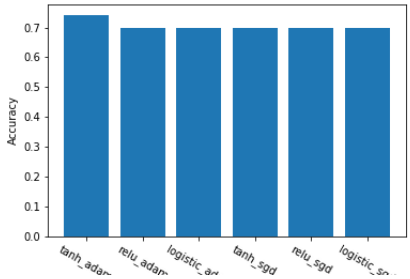
- Solver: This determines which algorithm is used for the back propagation. Choices are "adam" and "sgd"
- Activation: This tells the algorithms which activation function to use. Choices are 'tanh','relu',and 'logistic'
- Number and shape of hidden layers: This informs the architecture of the MLP by allowing for the number of hidden layers and number of neurons per hidden layer to be specified.

Since the purpose of this approach is to utilize the power of the ensemble method the hidden layers hyperparameters will be fixed at 2 hidden layers with 100 neurons each. This allows for easier iterative optimization of the other hyperparameters. This processes will need to be done separately for each the EMS and BMS data.



(a) Loss Comparison using 'adam'

(b) Loss Comparison using 'sgd'



(c) Accuracy Scores for activation/solver pairs

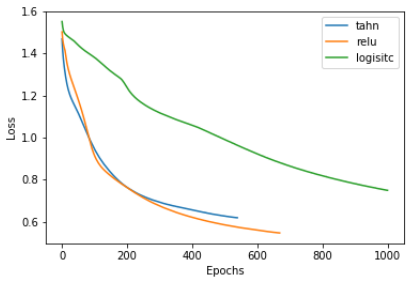
Figure 3.6: MLP model performance under various Hyperparameters (EMS)

3.3.2 Logistic Regression

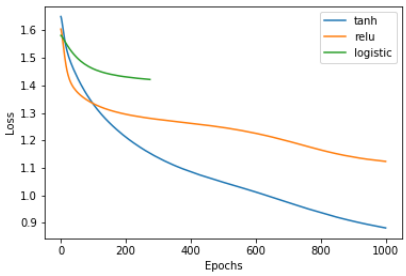
3.3.3 Ensemble

3.3.4 Gaussian Mixture Model

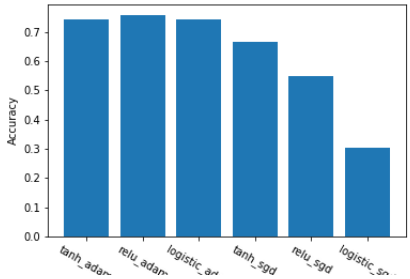
As stated in the methodology for this chapter, the GMM algorithm is the only one that was applied to both the reduced dimension data and the time series data. This was done in order to facilitate the use of the semi-unsupervised learning methods. This being the case, the findings will be represented separately first.



(a) Loss Comparison using 'adam'



(b) Loss Comparison using 'sgd'



(c) Accuracy Scores for activation/solver pairs

Figure 3.7: MLP model performance under various Hyperparameters (BMS)

GMM for Time Series Data

GMM for Low Dimension Data

3.4 Discussion

4 Data Quality

4.1 Background Research

4.2 Statistical Quality Control

Statistical quality control is the name given to the application of statistical methods to monitor the quality of a process. This is a very basic yet effect principle that is in use in both simple and complex systems. One of the tools of statistical quality control is the use of control charts. [**?**]

One Vs All Logistic Regression (Solver = "newton")			
Classifier	Precision	Recall	Instances
Electric	0.74	0.99	472
Gas	0.67	0.04	45
Heat	0.20	0.05	42
Water	0.68	0.15	103
Avg/Total	0.69	0.73	662
Accuracy	0.731		

(a) Logistic Regression using Adam

One Vs All Logistic Regression (Solver = "sag")			
Classifier	Precision	Recall	Instances
Electric	0.71	1.00	472
Gas	0.50	0.04	45
Heat	0.00	0.00	42
Water	0.00	0.00	103
Avg/Total	0.54	0.71	662
Accuracy	0.712		

(b) Logistic Regression using 'sag'

One Vs All Logistic Regression (Solver = "saga")			
Classifier	Precision	Recall	Instances
Electric	0.71	1.00	472
Gas	0.50	0.04	45
Heat	0.00	0.00	42
Water	0.00	0.00	103
Avg/Total	0.54	0.71	662
Accuracy	0.7129		

(c) Logistic Regression using 'saga'

One Vs All Logistic Regression (Solver = "lbfgs")			
Classifier	Precision	Recall	Instances
Electric	0.74	0.99	472
Gas	0.67	0.04	45
Heat	0.00	0.00	42
Water	0.55	0.16	103
Avg/Total	0.66	0.73	662
Accuracy	0.731		

(d) Logistic Regression using 'lbfgs'

Figure 3.8: Outputs from the 4 options of logistic regression

Decision Tree (criteria = "entropy")			
Classifier	Precision	Recall	Instances
Electric	0.92	0.88	472
Gas	0.53	0.58	45
Heat	0.44	0.55	42
Water	0.73	0.78	103
Avg/Total	0.83	0.82	662
Accuracy	0.821		

Figure 3.9: Caption

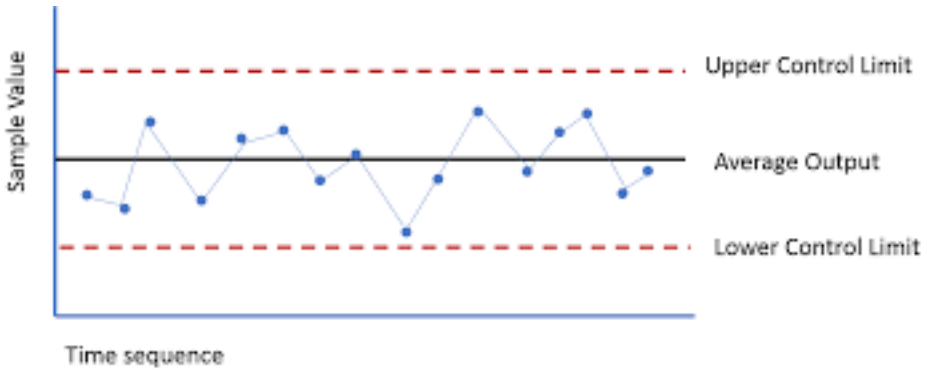


Figure 4.1: Basic Control Chart (CITE SITE)

Control charts consist of data, a line representing the mean of the data,and representations of the upper and lower control limits. It is these upper and lower limits that are useful when attempting to detect outliers that may indicate low quality data. The upper and lower control limits are found as follows:

$$Upper\ Control\ Limit(UCL) = 2 * \sigma + \mu \tag{4.1}$$

$$Lower\ Control\ Limit(LCL) = 2 * \sigma - \mu \tag{4.2}$$

A metric for data quality can be determined by counting the number of instances where data points stray outside the bounds of the UCL and LCL. This method can even be 'windowed' in order to produced a moving average, UCL and LCL that is capable of capturing periodicity in time series data.

4.2.1 Bayesian Inference

Bayesian statistics stem from the use of Bayes' Theorem (4.1) to express the degree of belief or disbelief in an event occurring. It differs from a frequentest approach by using the prior distribution (4.2) to represent the initial uncertainty in the parameters and the posterior distribution (4.5) to represent the uncertainty after being exposed to the data(SPRINGER).

$$P(\theta|X) = \frac{P(X|\theta)P(\theta)}{P(X)} \tag{4.3}$$

$$P(\theta) = \text{Prior Distribution for the parameters} \tag{4.4}$$

$$P(X|\theta) = \text{Likelihood} \tag{4.5}$$

$$P(X) = \text{Marginal Likelihood} \tag{4.6}$$

$$P(\theta|X) = \text{Posterior Distribution} \tag{4.7}$$

Using the Bayesian approach also allows for the calculation of a predictive distribution (4.6) that allows for the probability that a new value comes from the posterior distribution to be calculated. This calculation also has an advantage over the frequentest approach of maximum likelihood inference because the calculation results in a probability instead of a confidence interval or a p-value.

4.3 Methods

The approach

In Chapter 3 this paper introduced a new method for the classification of sensors based on their data. However, all those machine learning techniques are useless if the data the sensors collected is corrupted or inaccurate. This leads to the problem of data quality and how to quantify it. Statistical quality control techniques provide a time tested baseline for this process but the goal of this project was to attempt to improve upone these methods. To that end bayesian inference was implemented on the data.

4.4 Results

5 Conclusion

hello all