Majority Rule Approach of Deep Learning for Time Series Data

Ayahiko Niimi[1], Masato Ikeda[2]
Faculty of Systems Information Science[1], School of Systems Information Science[2]
Future University Hakodate
Kamedanakano, Hakodate
Hokkaido, Japan

Abstract

In this study, two major applications are introduced to develop advanced deep learning methods for credit-card data analysis. The proposed methods are validated by comparing benchmark experiments with other machine learnings. The experiments confirm that deep learning exhibits accuracy similar to that of the Gaussian kernel support vector machine (SVM).The proposed methods are also validated using a large scale transaction data set. Motivated by the experimental results, we propose three different fast approaches of deep learning for time series data. The experimental results failed to demonstrate the effectiveness of the proposed method. However, the author could confirm three challenges that exist with the proposed method: 1) creating a model generating less than 50% error rate; 2) dividing the dataset evenly over time; and 3) considering the class not contained in the divided data. If the created method can overcome these problems, the proposed technique is effective according to ensemble learning theory.

1. Introduction

Deep learning is a state-of-the-art research topic in the machine learning field with applications that can solve a wide variety of problems[1], [2].In this study, we investigate the application of deep learning to credit card data analysis.

Credit card data are mainly used in user and transaction judgments. User judgment determines whether a credit card should be issued to a user satisfying particular criteria, whereas transaction judgment determines whether a particular transaction is valid [3].In our previous study, we determined the deep learning processes required for solving each of these problems and proposed appropriate methods for deep learning[4], [5].

To verify our proposed methods, we compare our methods to benchmark experiments with other machine learning techniques, confirming that the accuracy of the deep learning methods is similar to that of the Gaussian Kernel support vector machine (SVM) method and provide suggestions for future deep learning experiments.

Although previously we only used a small transaction data set for the evaluation experiment [6], in this study, the proposed methods are also validated using a large transaction data set.

Because we have found that the processing of large amounts of a data required for deep learning is time consuming, we propose three different approaches for fast implementation of deep learning for time series data.

The outline of the paper is as follows. First, in section 2, we introduce the characteristics of the credit card data. In section 3, we describe the deep learning approach. Then, we discuss the data processing infrastructure suitable for credit card data analysis in section 4.In section 5, we propose three different approaches for fast implementation of deep learning for time series data. We describe the experiment and the obtained results in section 6 and 7, respectively and thendiscuss the results in section 8.Finally, in section 9, we describe our conclusions and future works.

2. Credit Card Data Set

Credit card information is contained in two data sets:

1. Credit approval data set.
2. Card transaction data set.

2.1. Credit Approval Data Set

For each user applying for a new credit card, there is a record of the decision to issue the card or to reject the application in accordance with the general usage-trend models based on the user's attributes.

However, to reach this decision, it is necessary to combine multiple models, with each model applicable to a different clustered group of users.

2.2.Credit Card Transaction Data

In actual credit card transactions, the data is complex, changes constantly, and continuously arrives online as follows:

1. Approximately one million transactions arrive per day.
2. Each transaction takes less than one second to complete.
3. Approximately one hundred transactions arrive per second during peak time.
4. Transaction data arrive continuously.

Therefore, credit card transaction data can be precisely called a data stream. Although data mining techniques are used for this credit card data, an operator can monitor only around 2,000 transactions per day. Therefore, suspicious transactions must be detected effectively by analyzing less than 0.02% of the total number of transactions. In addition, the amount of detected fraud is extremely low relative to the massive amounts of analyzed transaction data, because real fraud occurs at an extremely low rate, which is 0.02% - 0.05% of all transaction data.

In a previous study, transaction data in CSV format were described as attributed in a time order [3]. Credit card transaction data have 124 attributes of which 84 are transactional data, including an attribute used to discriminate whether the data refers to fraud; the other attributes are behavioral data and refer to the credit card usage. The inflow file size is approximately 700 MB per month.

Mining the credit card transaction data stream presents inherent difficulties, because it requires efficient calculations to be performed on an unlimited data stream with limited computing resources.Therefore, manydata stream mining methods seek an approximate or probabilistic solution instead of an exact one.However, because the actual unauthorized credit card use is quite rare, these imprecise solutions do not allow appropriate fraud detection.

3.Deep Learning

Deep learning is a new approach that recently attracted much attention in the field of machine learning. It significantly improves the accuracy of abstract representations by reconstructing deep structures such as the neural circuitry of the human brain. Moreover, the Deep learning algorithms were honored in various competitions such as the International Conference on Representation Learning.

Deep learning is a generic term for multilayer neural networks that have been researched over several years [1], [2], [7].Multilayer neural networks decrease the overall calculation time by performing calculations on hidden layers. Therefore, these networks were prone to excessive overtraining, because an intermediate layer was often used to approximate each layer.

Nevertheless, technological advances have overcome the problem of overtraining, while the use of GPU computing and parallel processing increased the tractable number of hidden layers

A sigmoid or tanh function was commonly used as the activation function (see Equations (1), (2)),although recently, the maxout function was also used (section 3.1.) and the dropout technique was implemented in order to prevent overtraining (section 3.2.).

$$h_i(x) = \text{sigmoid}(x^T W_{\dots i} + b_i) \quad (1)$$

$$h_i(x) = \tanh(x^T W_{\dots i} + b_i) \quad (2)$$

3.1. Maxout

The maxout model is simply a feed-forward architecture such as a multilayer perceptron or deep convolutional neural network that uses a new type of activation function, the maxout unit [2].

In particular, given an input $x \in \mathbb{R}^d$ (x may be either v or a hidden layer's state), a maxout hidden layer implements the function

$$h_i(x) = \max_{j \in [1,k]} z_{ij} \quad (3)$$

where $z_{ij} = x^T W_{\dots ij} + b_{ij}$, $W \in \mathbb{R}^{d \times m \times k}$ and $b \in \mathbb{R}^{m \times k}$ are learned parameters. In a convolutional network, a maxout feature map can be constructed by taking the maximum across k affine feature maps (i.e., pool across channels, in addition to spatial locations).When training with the dropout, in all cases, we perform element-wise multiplication with the dropout mask immediately prior to multiplication by weights and the inputs not dropped to the max operator. A single maxout unit can be interpreted as a piecewise linear approximation of an arbitrary convex function. In addition to learning the relationship between hidden units, maxout networks also learn the activation function of each hidden unit.

The maxout approach abandons many of the mainstays of traditional activation function design. Even though the gradient is highly spare, the representation produced by maxout is not sparse at all, and the dropout will artificially sparsify the effective representation during training. Although the maxout unit may saturate on one side or another, this is a measure zero event (so it is almost never bounded from above). Because a significant proportion of the parameter space corresponds to the function delimited from below, maxout learning is not constrained.

Moreover, the maxout function is locally linear almost everywhere, in contrast to many popular activation functions that demonstrate significant curvature. Considering all these deviations from thestandard practice, it may seem surprising for the maxout activation functions to work however, we find that they are very robust, easy to train with dropout, and achieve excellent performance.

$$h_i(x) = \max_{j\in[1,k]} Z_{ij} \quad (4)$$

$$z_{ij} = X^T W_{\cdots ij} + b_{ij} \quad (5)$$

3.2. Dropout

Dropout is a technique that can be applied to deterministic feedforward architectures that predict an output y given an input vector v[2].
In particular, these architectures contain a series of hidden layers $h = \{h(1), \cdots, h(L)\}$. Dropout trains an ensemble of models comprising a subset of the variables in both v and h. The same set of parameters θ are used to parameterize a family of distributions $p(y \mid v;\ \theta, \mu)$, where $\mu \in \mathcal{M}$ is a binary mask determining which variables to include in the model. For each example, we train a different submodel by following the gradient $\log p(y \mid v; \theta, \mu)$ for a different randomly sampled μ. For many parameterizations of p (usually for multilayer perceptrons) the instantiation of the different submodels $p(y \mid v;\ \theta, \mu)$ can be obtained by element-wise multiplication of v and h with the maskμ.

The functional form becomes important when the ensemble makes a prediction by averaging the submodels' predictions. Previous studies of bagging averages used the arithmetic mean. However, this is not possible with the exponentially large number of models trained by dropout. Fortunately, some models can easily yield ageometric mean. When $p(y \mid v;\ \theta) = \text{softmax}(v^T W + b)$, the predictive distribution defined by renormalizing the geometric mean of $p(y \mid v;\ \theta, \mu)$ over$\mathcal{M}$ is simply given by $\text{softmax}(v^T W/2 + b)$. In other words, the average exponential prediction for many submodels can be computed simply by running the full model with the weights divided by two. This result holds exactly in the case of a single layer softmax model. Previous work on dropout applies the same scheme to deeper architectures, such as multilayer perceptrons, where the $W/2$ method provides only an approximation of the geometric mean. While this approximation is not mathematically justified, it performed well in practice.

4. Data Analysis Platform

In this section, we consider the data processing infrastructure suitable for credit card data analysis as well as the applications of deep learning to credit card data analysis.

4.1. R

R is a language and environment for statistical computing and graphics [8]. It is a GNU project that is similar to the S language and environment developed at Bell Laboratories (formerly AT&T, now Lucent Technologies) by John Chambers and colleagues. While there are some important differences between them, R can be considered to be different implementation of S, with most of the code written for S running unaltered under R.R is available as Free Software, includes many useful libraries such as multivariate analysis and machine learning, and is widely used and suitable for data mining.

However, R performs the processing in memory, and is therefore unsuitable for processing large amounts of data.

4.2. Google BigQuery, Amazon Redshift

Google BigQuery [9] and Amazon Redshift [10] are systems designed for inquiries that use large amounts of data. These cloud systems can easily store a large amount of data and process it at high speed. Therefore, these systems can be used to interactively analyze data trends. However, data processing such as machine learning needs to be further developed.

4.3. Apache Hadoop

Apache Hadoop is another platform for handling large amounts of data [11].

Apache Hadoop divides the process into mapping and reducing as well as operating in parallel; the Map processes the data, whereas the Reduce summarizes the results. Together, these processes realize high-speed processing of large amounts of data.

However, because processing is performed in batches, the Map/Reduce cycle can be completed before all data are stored. Moreover, it is difficult to apply separate algorithms for different batches of Map/Reduce and in particular, it is difficult to implement the algorithm repeatedly for the same data, as required in machine learning.

4.4. Apache Storm

Apache Storm is designed to process a data stream [12] with data conversion executed for continuously flowing data. The data source is called the Spout and the part that performs the conversion process is called the Blot. Apache Storm is a model

that performs processing by a combination of Bolt from Spout.

4.5. Apache Spark

Apache Spark is another platform for processing large amounts of data [13] using generalized Map/Reduce processing. The processing is performed by caching the work memory and is designed to execute efficient iterative algorithms by maintaining the shared data that is used for repeated processing in the memory. In addition, a machine learning and graph algorithm library is prepared, and it can be used as an easy build environment for data stream mining.

H2O is a library of deep learning for Spark [14], [15].

SparkR is an R package that provides a light-weight R-based frontend for Apache Spark [16]. In Spark 1.5.0, SparkR provides distributed data frame implementation that supports operations, such as selection, filtering, and aggregation, similar to R data frames such as dplyr, but for large data sets. SparkR also supports distributed machine learning using MLlib.

In the present study, we perform credit card data analysis using R and Spark. It is possible to use an extensive R library and obtain high computational performance by parallel and distributed processing of Spark.

5. Fast Deep Learning Methods for Time Series Data

The speed of the deep learning process is an important consideration. Processing of large amounts of multi-attribute data by deep learning requires significant computation time. Therefore, it is difficult for the method to also consider the processing speed. To solve this problem, we propose to apply the concept of data stream mining to processing without the accumulation of data. It is possible to realize accelerated deep learning processing that stores data for a short-term by using the data stream mining concept. We propose three approaches that apply the concept of data stream mining.

- Approach 1: learning by differential data with parameters obtained before the construction of the model
- Approach 2: constructing a new model using differential data and majority rule
- Approach 3: constructing a new model using differential data that failed data judgment

For obtaining the differential data, we consider the window width.We propose two window concepts; separation by a period of time and separation by the number of data. Because the differential data appears as a short-term buffer, it is possible to use recent features for the judgment. Approaches 2 and 3 can construct many models, creating an appearance of long-term buffers that enable the storage of the previous models.

6. Experiments

6.1. UCI Data Set

We used the credit approval data set in the UCI Machine Learning repository to evaluate the experimental results [4].

All attribute names and values were reassigned to meaningless symbols to protect data confidentiality.

In addition, the original data set contained missing values and in the experiment, we used a pre-processing data set [17],as presented in Table 1.

Table 1. UCI Data Set(Credit Approval Data Set)

Number of Instances:	690
Number of Attributes:	15 + class attribute
Class Distribution:	+: 307 (44.5%), -: 383 (55.5%)
Number of Instance for Training:	590
Number of Instance for Test:	100

Deep learning uses the H2O R library [14], [15], which is a library for Hadoop and Spark having an R package.

For comparison, we also used five typical machine learning algorithms. In addition, the deep learning parameters (activation functions and dropout parameter) were changed five times. In this experiment, the hidden layer neurons were set to (100, 100, 200) for deep learning. The parameters used in the experiments are shown in Table 2.

XGBoost is an optimized and parallelized general purpose gradient boosting library [18] that provides an optimized distributed version. It implements machine learning algorithms under the gradient boosting framework, including a generalized linear model and gradient boosted decision trees. Furthermore, XGBoost can be distributed and scaled up to the terascale.

The activation functions used in this study are summarized in Table 3[15].

Table 2. Comparison Algorithms

Deep learning	Rectifier with Dropout
	Rectifier
	Tanh with Dropout
	Tanh
	Maxout with Dropout
	Maxout

Logistic Regression	
Support Vector Machine	Gaussian Radial Basis Function
	Linear SVM
Random Forest	
XGBoost	

Table 3. Activation functions

Function	Formula
Tanh	$f(\alpha) = \frac{e^{\alpha} - e^{-\alpha}}{e^{\alpha} + e^{-\alpha}}$
Rectified Linear	$f(\alpha) = \max(0, \alpha)$
Maxout	$f(\cdot) = \max(w_i x_i + b)$, rescale if $\max f(\cdot) \geq 1$
Tanh with Dropout	Tanh with Dropout
Rectifier with Dropout	Rectified Linear with Dropout
Maxout with Dropout	Maxout with Dropout

Moreover, to ascertain whether there is a bias in the results of the training data and the test data, we performed a 10-fold cross-validation using the entire data set.In this experiment, the hidden layer neurons were set to (200, 200, 200) and we used the following environment.

- AWS EC2 t2.micro
- CPU Intel Xeon 3.3 GHz
- Memory 1GB
- Disk 8GB SSD
- R version 3.2.2
- H2O version 3.0.0.30

6.2. Large-Scale Data Set

In this study, the proposed methods were also validated using a large transaction data set. We constructed this data set from the actual card transaction data set that contained the same number of attributes (130 attributes) with random values within the same range specified for each attribute. The data set had about 300,000 transactions, including about 3,000 illegal usages. For the experiment, we constructed a data set using data of six months data. Because this data set contained random values, it could not be used to evaluate accuracy; instead, we used this data set to estimate machine specifications and calculation times.

The percentage of fraudulent transactions in this data set wad very low. We used all illegal usages (approximately 3,000) and a sampling of normal usages (approximately 3,000) in the experiment.

Moreover, we used the Amazon EC2 r3.8xlarge (32 cores, 244GB memory) for this experiment. As a preliminary experiment, deep learning's parameters of hidden layer neurons (100, 100, 200) and epochs (200) were used, but the learning did not converge.

Therefore, in the experiment, different deep learning parameters (hidden layer neurons (2048, 2048, 4096) and epochs (2000) and hidden dropout ratios (0.75, 0.75, 0.7)) were used, and the "Maxout with Dropout" was used as the activation function.

Then, we analyzed the experimental results.

6.3. Time Series Data Set

For comparison of the proposed method, we evaluate our proposed method by using public time-series benchmark data. We used the gas sensor dataset from the UCI Machine Learning repository [4], [19], [20]. We used the gas sensor array drift dataset at different concentrations data set. The data set contains 10 files which are divided depending on the periods, each file has included some gases and not included other gases. Also, the number of instances of gas is also different depending on the period. In the data set, the number of instances is 13,910 and the number of attributes is 129. Attributes include gas names, which are Ethanol, Ethylene, Ammonia, Acetaldehyde, Acetone, Toluene. Use of this data set is provided as a classification problem, regression analysis, clustering, analysis of causality.

7. Experimental Results

7.1. Comparison of Algorithms Using the UCI Data Set

Table 4 shows the experimental results. We run each algorithm five times and the average values are presented in Table 4.Because the machine learning algorithms that we used do not show initial value dependence, the results of the algorithms are the same for all five times.

Table 4. Result of UCI Data Set

Algorithm	Error Rate
Rectifier with Dropout (Deep Learning)	18.4
Rectifier (Deep Learning)	18.8
Tanh with Dropout (Deep Learning)	17.6
Tanh (Deep Learning)	22.8
Maxout with Dropout (Deep Learning)	12.4
Maxout (Deep Learning)	16.2
Logistic Regression	18.0
Gaussian Kernel SVM	11.0
Linear Kernel SVM	14.0
Random Forest	14.0
XGBoost	14.0

The obtained deep learning results depend on the initial parameters. It is found that deep learning of accuracy with the Maxout with Dropout produces a result close to that obtained by the Gaussian kernel SVM.

7.2. Deep Learning: 10-Fold Cross-Validation

Table 5 shows the results of the 10-fold cross-validation, where N and Y are class attributes.

Table 5. Result of Deep Learning (10-fold cross-validation)

	N	Y	Error	Rate
Totals	332	287	0.138934	=86/619
Totals	337	280	0.132901	=82/617
Totals	333	277	0.136066	=83/610
Totals	318	302	0.135484	=84/620
Totals	325	295	0.156452	=97/620
Totals	306	316	0.180064	=112/622
Totals	338	296	0.140379	=89/634
Totals	336	281	0.136143	=84/617
Totals	327	301	0.141720	=89/628
Totals	318	305	0.146067	=91/623
Average of Error Rate			0.144421	

Stability results are obtained regardless of the data set.

7.3. Transaction Data Set

For Training Dataset, we can obtain nearly 100% correct model which One illegal transaction is miss. For Test Dataset, we can obtain more than 50% correct model, but it's not so high accuracy.

7.4. Majority Rule Method

To verify our approach 3 of "Majority Rule Approach", we apply it to time series data set. The Table 6 shows gas estimation result. According to the procedure of the proposed method, after creating the model using deep learning, we obtained 9 models. The incorrect rate of training data by each model is less than 50%, but incorrect rate of test data by each model is less than 60%.

Table 6. Gas Estimation Result by Proposed Method

	Acetaldehyde	Acetone	Ammonia	Ethanol	Ethylene	Toluene
Acetaldehyde	2	212	349	16	1	20
Acetone	188	70	0	341	0	1
Ammonia	50	28	7	489	0	26
Ethanol	55	287	13	67	174	4
Ethylene	73	83	0	424	0	20
Toluene	89	135	265	11	100	0

When majority voting of the proposed method with these 9 models, the incorrect rate was 95.96% and the model could not estimate gases correctly. The reason why the gases could not be identified is that there were files with no periodicity and gases not included in the file.

8. Considerations

The conducted experiments confirm that deep learning shows the same accuracy as the Gaussian kernel SVM. In addition, the 10-fold cross-validation experiment indicates that deep learning offers a higher precision. In this experiment, we used the H2O library for deep learning, with the deep learning modules, written in Java, activated each time. Therefore, we cannot provide an assessment of the execution time. While adjustment of deep learning parameters is difficult, by optimizing the parameters, it is possible to increase the learning accuracy.

In the model that makes a majority decision of the proposed method, the accuracy of each model may not be so high, but at least 50% is necessary. If the accuracy of the model is less than 50%, correctness of judgment by majority vote cannot be guaranteed.

When dealing with time series data, it is not suitable for the proposed method because it is difficult for the periodic feature to appear in the learning model or the prediction result unless it is a continuous period. There is a possibility that the method proposed in this paper is effective when creating a learning model with data that summarizes a certain period of continuous time. Consecutive periods need to match the characteristics of the data. In the case of seasonal data, if the data is created for each season and the learning model is created, the proposed method may be effective. Considering these points, it is necessary to properly separate the time series data with the proposed method is considered to be possible.

Handling of classes not included in the data set needs to be considered appropriately. For example, it is conceivable to devise measures such as changing weights when majority votes are made according to classes included. Also, when an unclassified class appears in new data, it may be considered to assign it as an unknown class as an exception class by combining analysis as an outlier. By these ideas, it can be considered that it is possible to create a model with certain accuracy even for classes not included in a certain period in the proposed method.

9. Conclusion

In this study, we consider the application of deep learning in credit card data analysis. We introduce two major applications and propose methods for deep learning. To verify our proposed methods, we use benchmark experiments with other machine learnings. Through these experiments, it is confirmed that deep learning has the same accuracy as the Gaussian kernel SVM. The proposed methods are also validated using a large scale transaction data set.

It was found that deep learning requires a significant time to process large amounts of data. Therefore, we proposed three different fast approaches of deep learning for time series data.

The experimental results failed to demonstrate the effectiveness of the proposed method. However, the author could confirm three challenges that exist with the proposed method: 1) creating a model generating less than 50% error rate; 2) dividing the dataset evenly over time; and 3) considering the class not contained in the divided data. If the created method can overcome these problems, the proposed technique is effective according to ensemble learning theory.

10. Acknowledgements

The authors would like to thank Intelligent Wave Inc. for their comments on credit card transaction data sets.

11. References

[1] Y. Bengio, “Learning deep architectures for AI,” Found. Trends Mach.Learn., vol. 2, no. 1, pp. 1–127, Jan. 2009, (Access Date: 16 July, 2016). [Online]. Available: http://dx.doi.org/10.1561/2200000006

[2] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio,“Maxout Networks,” ArXiv e-prints, Feb. 2013.

[3] T. Minegishi and A. Niimi, “Detection of fraud use of credit card byextended VFDT,” in World Congress on Internet Security (WorldCIS-2011), London, UK, Feb. 2011, pp. 166–173.

[4] M. Lichman, “UCI machine learning repository,” 2013, (Access Date:16 July, 2016). [Online]. Available: http://archive.ics.uci.edu/ml

[5] T. J. OZAKI. (2015) Data scientist in ginza, tokyo. (Access Date: 16July, 2016). [Online]. Available: http://tjo-en.hatenablog.com/

[6] A. Niimi, “Deep learning for credit card data analysis,” in WorldCongress on Internet Security (WorldCIS-2015), Dublin, Ireland, Oct.2015, pp. 73–77.

[7] Q. Le, “Building high-level features using large scale unsupervisedlearning,” in Acoustics, Speech and Signal Processing (ICASSP), 2013IEEE International Conference on, May 2013, pp. 8595–8598.

[8] R: The R project for statistical computing. (Access Date: 16 July,2016). [Online]. Available: https://www.r-project.org/

[9] Google cloud platform, what is BigQuery? - Google BigQuery. (AccessDate: 16 July, 2016). [Online]. Available:https://cloud.google.com/bigquery/what-is-bigquery

[10] AWS Amazon Redshift, cloud data warehouse solutions. (Access Date:16 July, 2016). [Online]. Available: https://aws.amazon.com/redshift/

[11] Apache Hadoop, welcome to Apache Hadoop! (Access Date: 16 July,2016). [Online]. Available: https://hadoop.apache.org/

[12] Apache Storm, Storm, distributed and fault-tolerant realtimecomputation. (Access Date: 16 July, 2016). [Online]. Available:https://storm.apache.org/

[13] Apache Spark, lightning-fast cluster computing. (Access Date: 16 July,2016). [Online]. Available: https://spark.apache.org/

[14] 0xdata - H2O.ai - fast scalable machine learning. (Access Date: 16July, 2016). [Online]. Available: http://h2o.ai/

[15] A. Candel and V. Parmar, Deep Learning with H2O. H2O,2015, (Access Date: 16 July, 2016). [Online]. Available: http://learnpub.com/deeplearning.

[16] SparkR (R on Spark) - Spark 1.5.0 documentation. (Access Date:16 July, 2016). [Online]. Available: https://spark.apache.org/docs/latest/sparkr.html

[17] T. J. OZAKI, “Credit approval data set, modified,” 2015, (Access Date: 16 July, 2016). [Online].Available: https://github.com/ozt-ca/tjo.hatenablog.samples/tree/master/r_samples/public_lib/jp/exp_uci_datasets/card_approval

[18] dmlc XGBoost extreme gradient boosting. (Access Date: 16 July,2016). [Online]. Available: https://github.com/dmlc/xgboost

[19] A. Vergara, S. Vembu, T. Ayhan, M. Ryan, M. Homer, and R. Huerta, “Chemical gas sensor drift

compensation using classifier ensembles," Sensors and Actuators B: Chemical, vol.166, no.1, 2012, pp.320-329.

[20] I. Rodriguez-Lujan, J. Fonollosa, A. Vergara, M. Homer and R. Huerta, "On the calibration of sensor arrays for pattern recognition using the minimal number of experiments," Chemometrics and Intelligent Laboratory Systems, vol.130, 2014, pp.123-134.

