Spring 2016

Clustering: Methodology, hybrid systems, visualization, validation and implementation

Dao Minh Lam

CLUSTERING: METHODOLOGY, HYBRID SYSTEMS, VISUALIZATION, VALIDATION AND IMPLEMENTATION

by

DAO MINH LAM

A DISSERTATION

Presented to the Faculty of the Graduate School of the

MISSOURI UNIVERSITY OF SCIENCE AND TECHNOLOGY

in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

in

COMPUTER ENGINEERING

2016

Approved by

Dr. Donald Wunsch, Advisor
Dr. Randy H. Moss
Dr. R. Joe Stanley
Dr. Daryl Beetner
Dr. V.A. Samaranayake

PUBLICATION DISSERTATION OPTION

This dissertation has been prepared in publication option. It is formated according to the style of the journals where the papers are published. Section 1 has been added to supply background information for the whole dissertation. Section 2, 3 and 4 have been added to summarize each paper.

Paper I is entitled ”Clustering Data of Mixed Categorical and Numerical Type with Unsupervised Feature Learning”,and is prepared in the style used by the Institute of Electrical and Electronics Engineers (IEEE) Access journal as submitted on July, 2015.

Paper II is entitled Hidden Markov Model with Information Criteria Clustering and Extreme Learning Machine Regression for Wind Forecasting, and is prepared in the style used by Journal of Computer Science and Cybernetics as published on 2015.

Paper III is entitled Unsupervised Feature Learning Classification with Radial Basis Function Extreme Learning Machine using Graphic Processors, and is prepared in the style used by IEEE Transactions on Cybernetics as submitted on March, 2015.

ABSTRACT

Unsupervised learning is one of the most important steps of machine learning applications. Besides its ability to obtain the insight of the data distribution, unsupervised learning is used as a preprocessing step for other machine learning algorithm. This dissertation investigates the application of unsupervised learning into various types of data for many machine learning tasks such as clustering, regression and classification. The dissertation is therefore organized into three papers. In the first paper, unsupervised learning is applied to mixed categorical and numerical feature data type to transform the data objects from the mixed type feature domain into a new sparser numerical domain. By making use of the data fusion capacity of adaptive resonance theory clustering, the approach is able to reduce the distinction between the numerical and categorical features. The second paper presents a novel method to improve the performance of wind forecast by clustering the time series of the surrounding wind mills into the similar group by using hidden Markov model clustering and using the clustering information to enhance the forecast. A fast forecast method is also introduced by using extreme learning machine which can be trained by analytic form to choose the optimal value of past samples for prediction and appropriate size of the neural network. In the third paper, unsupervised learning is used to automatically learn the feature from the dataset itself without human design of sophisticated feature extractors. The paper points out that by using unsupervised feature learning with multi-quadric radial basis function extreme learning machine the performance of the classifier is better than several other supervised learning methods. The paper further improves the speed of training the neural network by presenting an algorithm that runs parallel on a graphics processing unit (GPU).

ACKNOWLEDGMENTS

I don't know where to start to acknowledge since there are many people that have made my education and this dissertation possible. I am in great luck to have great teachers and be cared by great friends.

The most important contribution for this dissertation are people in grad school. I don't know how to start with my advisor, Donald Wunsch. Dr. Wunsch has given me the most professional guidance throughout choosing research topics, paper writing, technical presentations, staying organized. He has invested a lot of his resources for my education development.

I like to thank Dr. Wei Mingzhen for her financial support. I would to thank other co-authors of my papers, Dr. Shuhui Li and Dr. Tayo, for their constructive discussions about wind forecast and mixed type clustering.

I had a great time with other Ph.D. students in the applied artificial computational intelligence lab. Thanks Sejun Kim, Bryce Schumacher and Islam Elnabarawy for their discussion on topics of machine learning. I also thank Leonardo Silva, Seaar Al-Dabooni, and Yongliang Yang for their fruitful discussion.

I would also like to thank my committee members, Drs. Daryl Beetner, Randy Moss, R. Joe Stanley, V.A. Samaranayake for their precious time in examining this dissertation and their constructive suggestions to my research work.

Last but not least, I would like to thank to my family, my mom and sister. They are the major support for my entire life. Though we live half of the earth apart, their support and faith in me help me a lot when I was in difficulty. In addition to my family, I also thank Tam Cao and her family, Hong Nguyen, Jenny Lam and Henry Nguyen, for their support throughout the last four years of my study. My thanks also goes to Hong Wunsch for her great care of my life in Rolla.

TABLE OF CONTENTS

SECTION

LIST OF ILLUSTRATIONS

Figure Page

PAPER I

PAPER II

PAPER III

LIST OF TABLES

Table Page

PAPER I

PAPER II

PAPER III

1. INTRODUCTION

1.1. THE HETEROGENEITY OF DATA

The amount of data in our life has been exploding, and analyzing large datasets will be a key component for innovation in several areas. The expansion of internet, social network, internet of things, automobiles, and smart energy sensors will fuel the data growth for quite a foreseeable future.

One of the most challenging problems in data analysis is the non-homogeneous property, which is each kind of data has its own format, unique features. The three most popular categories of data are: attribute data, time series data and multimedia data.

Attribute data is the data where each entity is described by a set of attributes often named as features. Many machine learning approaches assume that features have numeric values. However, in practice, mixed, erroneous, and missing data can result from i) errors or mistakes caused by the equipment or humans, or ii) the data attributes, which can be either numerical or categorical. Combinations of these issues can cause data to be mixed-type, multivalued, or missing. Most data pre-processing successfully deals with the erroneous and missing value in the data but hardly can deal with the mix categorical and numerical features.

The second data this dissertation is trying to deal with is time series data. Time series is a collection of observations that are evenly spaced in time and measured successively. Time series collected in this manner often called discrete-time time series. Examples of time series are stock price, wind speed and direction, and hourly readings of air temperature. Time series can be uni-variate and multivariate. In the case of multivariate, a mixed type of both discrete and continuous variables causes a lot of problems for time series analysis since most time series models assume the data to be either discrete or continuous, not a mixture of them. Another challenge for time series analysis is the size of dataset. As time series data is accumulated over time, the size of the dataset will be ever increasing over the time of data collection. This poses the problem of how to handle the large scale of data efficiently.

The third type of data is multimedia data such as texts, sounds, images and videos, which are exploding at an exponential speed. This exploding creates various challenges and opportunities for machine learning applications, especially in the computer vision area. The opportunity here is more and more data help propel a class of machine learning algorithms that attempt to learn the feature automatically from unlabeled data that can be obtained more and more easily. This is the first time a machine learning algorithm can learn the features outside the context of the given dataset. This is also very different from the idea of semi-supervised learning, where the unlabeled data could be labeled, but that label has not been provided. Besides the opportunity, the challenge is also high. The ever-increasing size and complexity of the image datasets creates potential tradeoffs of accuracy and speed in learning algorithms. A good classifier requires progressively more data for training, which also increases the time required for training.

1.2. UNSUPERVISED LEARNING

Unsupervised learning has several alternative names form one discipline to another. In biology, unsupervised learning is known as numerical taxonomy. In graph theory, it is known as partition. In computational intelligence, unsupervised learning is often referred to as clustering. Those different name of the same learning methods reflect the fact that unsupervised learning try to group data into a certain number of clusters but there is no precise definition of the term cluster. However, data in the same cluster should be similar to each other, while data from different clusters should be different from each other.

Unsupervised learning is usually the first learning step in the process of acquiring new knowledge of the dataset. Since unsupervised learning works with unlabeled data, it is used for obtaining insight into the data distribution. However, the most popular application of unsupervised learning is it serves as a preprocessing step for other algorithms. Particularly in this dissertation, unsupervised learning is used for data fusion to reduce the distinction of numerical and categorical features of mixed type data. It is also used as a step of gathering more information from surrounding wind time series in wind forecast. Finally, it is used as a feature learning algorithm where the feature is learned from the dataset itself without a human-designed feature extractor.

1.3. RESEARCH OBJECTIVES AND CONTRIBUTIONS

This dissertation deals with the use of unsupervised learning in various machine learning tasks. In concrete, in the first part of the dissertation, when dealing with mixed-features data, we propose an unsupervised feature learning approach to transform the objects from the mixed-type feature domain to a new numerical domain. By making use of the template matching of adaptive resonance theory architecture (ART), we can rapidly build the prototypes of the dataset. By measuring the distances between each of the objects to those prototypes, we can build new unsupervised learning features for the objects that remove the distinction in treating numerical and categorical features, leading to a better clustering result. The approach is also very scalable for a large scale dataset since the fast learning property of ART. The approach was demonstrated with several real world datasets including credit approval, heart disease and teaching assistant evaluation datasets with ground truth and one noisy, mixed petroleum industry data, confirming the improvement of the presenting method over several other methods. This paper was submitted to IEEE Access in July, 2015. The contributions of this paper are:

- Unsupervised feature learning is applied to mixed type data to achieve sparse representation, which makes it easier for a clustering algorithm to separate the data
- UFL is implemented by using Fuzzy ART

- Apply the methodology to several application fields

The second part of the dissertation deals with time series. The problem this part tries to solve is improving forecast performance of wind speed in the windmills. There are several other methods in the past but they often are either slow or not accurate enough. Our approach to this problem is to make use of the available data of the surrounding wind mills to help better prediction. First we presented a new time series clustering method that can handle mixed type of data using a hidden Markov model HMM. The HMM model is size-optimized by using our modified information criteria. The presented method can handle large scale of data due to the optimized HMM language processing toolbox. Secondly, to forecast the wind with the information from the clustering info, we use a fast extreme learning machine to choose the optimal value of past samples for prediction and appropriate size of the neural network. The result is that we can train the neural network in less than a second for the dataset of 30 years of hourly wind data collection. The second part is published in Journal of Computer Science and Cybernetics in 2014. The contributions of this paper are:

- Propose a modified information criteria to choose the best HMM size and clustering partition
- Develop a new wind time series clustering algorithm using Hidden Markov Models that works with large scale data and mixed type data
- Create a novel fast wind speed forecast by including non-local data from clustering using extreme learning machine and training the neural network in analytic form

The third type of data is multimedia data such as texts, sounds, images, and videos where features are not ready. They are often transformed or processed by feature extractors, which are mostly hand-crafted and often data-specified. So the motivation of the third part of the dissertation is to design a mechanism that can learn the features from the data universally and be performed automatically without any human intervention. It is done by a recent trend of research called unsupervised feature learning, where we learn the feature encoder by clustering the small patches from unlabeled data and then map any new image to feature representation through the learned encoder. Another motivation for this research is the speed and performance of classification for those datasets. As more multimedia data are generated, training a classifier takes more and more time, so there is a strong need to speed up this process. Also deep learning is currently used to improve the accuracy but it takes a lot of time to train the neural network. In this dissertation, we opt for using a extreme learning machine (ELM) with a special kernel known as a multi-quadric radial basis function (RBF) for classification. It is shown that this neural network is more accurate and much faster than many state of the art classifiers. Furthermore, to improve the speed of the classifier, we present a new massive parallel computing CUDA kernel by making use of the device GPU architecture. We test our approach on 2 large datasets CIFAR-10 and MNIST, showing that we achieve better result than many other approaches and have the speed up of 20 times than the CPU program. This part was submitted to IEEE Transactions on Cybernetics on 2015. The contribution of this papers are:

- Use the ELM RBF multi-quadric with unsupervised feature learning (UFL) to achieve better performance
- Implement the ELM RBF kernel using a GPU, in which the hidden layer output is implemented from scratch, using GPU native code.

The dissertation has a several broader impacts. Part I of the dissertation deals with mixed feature data. This type of data happens a lot in practice. By successfully dealing with the real data, the approach of clustering those kind of data presented in dissertation has a lot of application in many areas of not only engineering but also economics and society like: journalism, petroleum, health care, automation, robotics, where one has to deal with both nominal and numerical features. Part II of the dissertation gives better performance of wind forecast. With the increasing concern about environmental pollution, the danger of radiation, and possible energy shortages have urged generating electricity from renewable energy. The percentage of wind power making up the total electrical power supply has increased quickly. The better performance of the prediction wind forecast obtained by the presented method helps the operation of the windmill. The second part of this work with time series prediction can also have a direct application into stock market, health care like EEG, earthquake. Its fast and accurate performance of prediction can help a user have the prompt and correct action to maximize the benefit or minimize the risk. Finally the third part of dissertation, by using the unsupervised feature learning, saves a lot of effort of the user to find out what feature is good. Furthermore the speed up by using GPU help achieve the system work in real time, solving the ever increasing data volume problem. Its application are many, such as image retrieval where query is an image of the similar object, sorting where parcels are sorted by their sites in post office, counting where a biology researcher wants to determine the number of occurrences of a specific subject.

2. UNSUPERVISED LEARNING WITH MIXED TYPE DATA

This chapter results from the collaboration with Professor Mingzhen Wei in the Department of Geological Science & Engineering Missouri S&T. The purpose of this collaboration is to build a clustering methodology for petroleum engineering data. The dataset of interest is a typical dataset of real data where data are corrupted by human and instrument error. The most challenging problem of this dataset is that it contains both numerical and categorical features whereas most of the clustering algorithms require the dataset features be numerical.

This section presents an unsupervised feature learning approach to remove the distinctions between categorical and numerical features. It accomplishes the task by making use of the ability of data fusion of Fuzzy ART clustering, an unsupervised learning algorithm. This is the first time unsupervised feature learning is applied successfully to mixed numerical and categorical data to represent the original data into a new sparse feature representation, which is easier for clustering algorithms to separate the data.

The approach is not only tested on the petroleum engineering data but also several other application fields such as heart disease dataset, teaching assistant evaluation dataset and credit approval dataset.

The author would like to thank Dr. Rui Xu for his Fuzzy ART code. He also thanks Dr. Wei for her constructive discussion of data preprocessing and knowledge in petroleum engineering.

PAPER

I. Clustering Data of Mixed Categorical and Numerical Type with Unsupervised Feature Learning

Dao Lam, Mingzhen Wei and Donald Wunsch

Abstract

Mixed type categorical and numerical data is a challenge in many applications. This general area of mixed type data is among the frontier areas where computational intelligence approaches are often brittle in comparison to the capabilities of living creatures. In this paper, unsupervised feature learning (UFL) is applied to mixed type data to achieve a sparse representation, which makes it easier for clustering algorithms to separate the data. Unlike other UFL methods that work with homogeneous data such as image, and video data, the presented UFL works with mixed type data by using Fuzzy Adaptive Resonance Theory (ART). UFL with Fuzzy ART (UFLA) obtains a better clustering result by removing the differences in treating categorical and numeric features. The advantages of doing this are demonstrated with several real world datasets with ground truth including heart disease, teaching assistant evaluation and credit approval. The approach is also demonstrated on noisy, mixed type petroleum industry data. UFLA is compared with several alternative methods. To the best of the authors' knowledge, this is the first time UFL has been extended to accomplish fusion of mixed data types.

Index Terms

Clustering, unsupervised feature learning, mixed type data, Fuzzy ART.

This work was supported by the Missouri S&T Intelligent Systems Center, and the Mary Finley Missouri Endowment.

Dao Lam and Donald Wunsch are with the Applied Computational Intelligence Lab, Department of Electrical & Computer Engineering, Missouri University of Science & Technology, Rolla, MO 65401 USA. E-mail: {dao.lam, dwunsch}@mst.edu.

Mingzhen Wei is with the Department of Geological Science & Engineering, Missouri University of Science & Technology, Rolla, MO 65409 USA. E-mail: weim@mst.edu.

I. INTRODUCTION

Our work addresses the problem of mixed type categorical and numerical data in clustering. The goal is building a framework that automatically handles the differences in numerical and categorical features in a dataset and groups them into similar clusters.

Clustering is the problem of grouping unlabeled data items into classes based on the similarity of the items [1]. Many clustering algorithms, such as K-means and spectral clustering [2], assume that features have numeric values. However, in practice, mixed, erroneous, and missing data can result from i) errors or mistakes caused by the equipment or humans, or ii) the data attributes, which can be either numerical or categorical. Combinations of these issues can cause data to be mixed-type, multivalued, or missing. This paper presents a mechanism for overcoming these problems.

Many other algorithms, such as those discussed in [3, 4], are designed only for categorical data. Methods for handling mixtures of attributes were researched and discussed in [5, 6], but these approaches are just the beginning of what is needed. They typically group the attributes as categorical or numerical and treat them separately until finally combining them using a distance function. [5] demonstrated a similarity measure based on biometric classification, in which greater weight is assigned to features that are uncommon in the population. Based on that distance, they proposed the hierarchical agglomerative algorithm named Similarity-based Agglomerative Clustering (SBAC). However their quadratic computational cost makes it hard to extend for large datasets. Many researchers, such as [7, 8], have extended the K-means algorithm to work with data containing both numerical and categorical features. In [9], numerical and categorical features were treated separately during the clustering process, and the results then were combined to obtain a better partition using the ensemble learning approach. [10, 11] proposed a variance and entropy clustering for mixed data but it requires domain expertise to build the distance hierarchy for categorical attributes. In this paper, a new approach based on UFL allows a seamless combination of both categorical and numerical features. To our knowledge, this the first time that a method has been developed to extend UFL capabilities to mixed-type data. We accomplish this by combining it with the data fusion capabilities of Fuzzy ART. This is also the first time we are aware of ART being with the UFL technique.

In addition to being mixed, categorical data can have multiple values, and numerical data can have a range of values. Very little research has been conducted regarding this problem. In [12], the investigators addressed the problem of multi-value data in database clustering by building the similarity as the combination of qualitative and quantitative features. Other researchers [13] have used the Hausdorff distance to compute the distance between interval features, followed by a dynamic clustering algorithm to cluster the dataset. Such approaches are limited to either discrete or continuous features.

One of the main challenges of clustering is to determine the true number of clusters in a dataset. Several algorithms, such as K-means and spectral clustering [2], assume that this number must be known a priori. Other methods, such as fuzzy ART clustering[14], do not require this information, but the ideal number of clusters often is determined during the cluster validation process, and numerous studies have contributed to the solution [15, 1].

UFL has been widely used in computer vision [16, 17]. Besides the advantage of removing the labor of designing application specific features, results confirm that UFL shows higher performance than traditional approaches such as discussed in [18, 19]. UFL uses one of the unsupervised learning algorithms, such as K-means or auto encoding, to learn the features but those clustering methods often require numerical data. To the best of our knowledge, there is no previous research on applying UFL to mixed type feature data.

This paper uses ART as the unsupervised learning algorithm to learn the features from the data itself. ART has many attractive characteristics. It scales very well for large scale datasets because of its low computational requirement which is O(NlogN) or it can be further reduced to O(N) [14] when in a one-pass learning mode. The other reason why ART is chosen as UFL for mixed type data is its ability in data fusion [20, 21] by mapping features from multi-modal data simultaneously. Moreover, ART can dynamically and adaptively generate a prototype, which is used in feature encoding, without the requirement of specifying the number of clusters.

In this paper, a novel approach to handling mixed type data clustering is presented by using UFL. The contributions of this paper are:

1. It presents a UFL approach using Fuzzy ART. Unlike other unsupervised learning methods like K-means or auto encoder [22], where one needs large amounts of data, the approach

presented here works for both large and small volumes of data, which can be relevant when some relevant subspaces of the data are large and others are small.

2. UFLA can solve the problem of mixed type data. By learning the higher and sparse feature representation, the distinction between categorical and numerical in the original data becomes less of an obstacle.

The approach is tested on several datasets with mixed features: heart disease, teaching assistant evaluation and credit assignment on UCI repositories [23]. We also test one dataset with no ground truth from the petroleum industry [24].

The rest of the paper is organized as follows: Sec. II includes a review of UFL and Fuzzy ART, and Sec. III presents our novel approach to solving the problem of mixed, erroneous, and missing features. Sec. IV describes our experiments with real data from the UCI machine learning repository and the petroleum industry. Finally, conclusions and some future research directions are discussed in Sec. V.

II. UFLA

A. Novelty and Motivation

Although UFL can be widely applied in many areas, its approach has never been investigated as applicable to mixed type data representing real life datasets.

UFL is known to successfully represent the object in another sparse representation [25]. UFL can discover hidden features in data and represents them in sparse domains, which are more suitable for machine learning tasks than the original data.

The innovation lies in applying the UFL to mixed type data to reduce the distinction between the numerical and categorical. Most UFL has traditionally served as a preprocessing method for supervised learning problems. The transformative possibility here is in its application to a purely unsupervised problem. The primary value then is UFL's contribution to the hard but important problem of clustering when dealing with numeric and categorical data.

This leads to the question of how to apply UFL to this kind of dataset. The novelty of this method is using Fuzzy ART, one method of unsupervised learning, as the method of building a feature encoder.

The difference between categorical and numerical features makes several traditional clustering methods fail since they often work with numbers only. Many approaches try to treat them separately and then combine them in a later step [9] but they are still treated differently, and it is unclear whether the end results are satisfactory. An ideal strategy would be to fuse the two before actual clustering. UFL is very successful in sparse representation of the objects in a different space. Unfortunately, that type of UFL requires a large amount of data to build the encoder. Those large datasets are affordable when working with images, and video, since their natural features are large dimension and require convolutional processing. For those real datasets as shown in the UCI datasets used in this paper's experiments, each sample is represented by less than a few dozen features, and the convolutional operation is unworkable.

This leads to the introduction of Fuzzy ART into the process of UFL. This paper investigates a new method of UFL using Fuzzy ART. ART is known for its fast performance in unsupervised learning. The other biggest advantage of Fuzzy ART is it does not need to specify the number of clusters. The other evidence that ART can help reduce the distinction between the numerical and categorical feature is its capability of data fusion as demonstrated in [21].

The next two sections reviewed the related background that is necessary for the new approach.

B. UFL

Traditional classifiers require a human operator to high-level features, such as the scale-invariant feature transform (SIFT) and histogram of oriented gradients (HOG) [26],[27]. Those approaches are difficult or time consuming to apply to other kinds of data. To tackle the disadvantage of hand-crafted features, several methods of UFL have been researched, such as sparse coding, deep belief nets, auto encoder, and independent subspace analysis [28, 29].

In machine learning, the amount of data is often more important than the choice of algorithm. This is especially true in UFL where simple learning algorithms outperform several handcrafted, carefully designed methods. Recently many researchers have started using UFL in computer vision, e.g. [17] used sparse coding, and [22] used one-layer UFL for classification of text and image.

In tasks such as image classification and object recognition, UFL can be a more attractive approach than those relying on manually-designed features [30, 22]. UFL has also proven to be helpful in greedy layer-wise pre-training of deep architectures [31, 32, 33].

However, a major drawback of many UFL systems is their complexity where parameters like learning rate, momentum, and weight decay must be tuned and network architecture parameters must be cross-validated. This paper investigates a new method of UFL using a simple, fast but effective training by using Fuzzy ART. While most other UFL algorithms focus on applying to classification, this work serves to reduce the gap in numerical and categorical features and work under the clustering domain, an unsupervised learning problem. ART has shown its ability in data fusion by mapping multi-modal features in an incremental manner [20].

More over, UFL often leads to sparse feature representation, as demonstrated in [25]. Sparse representation often has several advantages such as robustness to noise. For clustering, sparse representation is probably easier to separate in higher dimensional space.

An unsupervised learning task often consists of four broad steps: 1)feature extraction 2) feature encoder building 3) feature mapping and 4) feature pooling. Our approach to feature learning removes the feature extraction and feature pooling because they are not relevant to the mixed type data. We only keep feature encoder building with Fuzzy ART clustering and feature mapping with a soft threshold function where the weight below a certain threshold is set to 0 resulting in sparse representation features.

C. Fuzzy ART

ART has been applied successfully to many machine learning applications [34]. It has the advantage of fast and stable learning. It is an online learning algorithm so it can be very scalable for large scale datasets. ART also has noise immunity in document clustering [35].

The other advantage of Fuzzy ART is that ART can be used for data fusion by extending ART from a single input field to multiple ones [20] as Fusion ART. Fusion ART provides a general mechanism for multi-channel features mapping. [21] shows that Fusion ART works successfully in integrating visual and textual features for image text co-clustering.

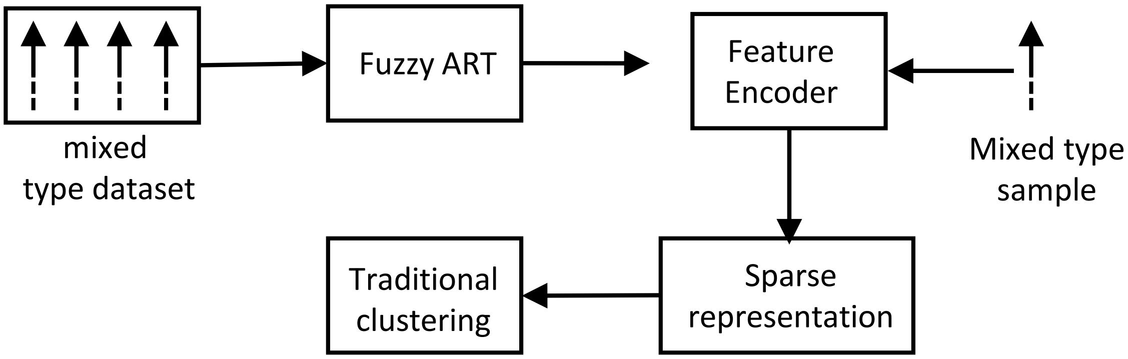


Fig. 1. UFLA framework. The whole dataset are first clustered by Fuzzy ART to produce the prototypes of the dataset. Those prototypes were used as feature encoder to encode individual mixed type data sample to sparse representation domain. After the mapping, the dataset can be clustered by any traditional clustering algorithm.

Let $\mathbf{X} = \{\mathbf{x}_1, ..\mathbf{x}_N\}$ be a set of N samples in the given dataset, where $\mathbf{x}_i = [x_{1,i}, ..x_{d,i}]^T$ is a sample belonging to d-dimensional space R^d.

The basic module of this UFL for mixed type clustering is Fuzzy ART [14]. Fuzzy ART consists of two layers of neurons: the input layer F1 and the clustering representation layer F2. Unlike ART 1, where there are bottom up and top down weight vectors, Fuzzy ART has only one weight vector $\mathbf{w}_j$ for each category j, which is initialized to $w_{j,1} = w_{j,2} = .. = 1$ when the category is uncommitted.

Before the samples can be input to ART, it has to be normalized to [0, 1] and enhanced with complement coding to avoid category proliferation problems. The clusters are formed in layer F2. When an input $\mathbf{x}$ is presented to layer F1, the committed neurons and one uncommitted neuron compete in a winner take all manner to select the one with maximum activation according to the formula below:

$$Tj = \frac{|\mathbf{x} \wedge \mathbf{w}_j|}{\alpha + |\mathbf{w}_j|}, \tag{1}$$

where α is a small real number to break the tie and the fuzzy AND operator $\wedge$ is defined by

$$(\mathbf{x} \wedge \mathbf{y})_i = min(x_i, y_i), \tag{2}$$

and where the norm |.| is defined by

$$|\mathbf{x}| = \Sigma_{i=1}^{d} |x_i|. \tag{3}$$

The winning neuron J, which is $argmax_j T_j$ becomes activated. If neuron J passes the vigilance ρ criterion which is:

$$\rho < \frac{|\mathbf{x} \wedge \mathbf{w}_J|}{|\mathbf{x}|}, \tag{4}$$

then the weight adaption occurs:

$$\mathbf{w}_J(new) = \gamma(\mathbf{x} \wedge \mathbf{w}_J(old) + (1 - \gamma)\mathbf{w}_J(old), \tag{5}$$

where γ is the learning rate parameter.

On the other hand if the vigilance criterion is not met, the current winning neuron is disabled, and the next winning neuron is chosen. If the uncommitted neuron is chosen, a new uncommitted neuron is created for future learning.

The advantage of UFL using Fuzzy ART is the dynamics, which can create many prototypes used for learning feature by just increasing the value of the vigilance threshold ρ.

III. UFLA CLUSTERING WITH MIXED, ERRONEOUS, MISSING FEATURES DATA

Because $\mathbf{x}_i$ has both categorical and numerical features, it can be represented as $\mathbf{x}_i = [xc_{1,i}, xc_{2,i}, ..xc_{r,i}, xn_{1,i}, xn_{2,i}, .., xn_{s,i}]$, where the first part, $[xc_{1,i}, xc_{2,i}, ..xc_{r,i}]$, is categorical, $[xn_{1,i}, xn_{2,i}, .., xn_{s,i}]$ is numerical, r and s are the number of categorical and numerical features, respectively, and $r+s = d$. In other words, the dataset X has fc_1, fc_2,..,fc_r as categorical features and fn_1, fn_2,..,fn_s as numerical features.

The proposed methodology consists of the five steps described in Algorithm 1.

Algorithm 1 UFLA clustering

```
1. Perform data preprocessing to clean up missing, interval, and multi-value data. Perform binary feature mapping on categorical data and normalize numeric data to [0 1].
2. Perform fuzzy ART clustering to obtain a certain number of clusters. Consider the weights from each cluster of the ART as centroids.
3. For each sample $\mathbf{x}$ compute $f(\mathbf{z}) = min(0, mean(\mathbf{z}) - \mathbf{z})$ where z is the distance from $\mathbf{x}$ to centroids.
4. Treating f(z) as an unsupervised learning feature, use VAT or clustering validation to determine number of clusters k.
5. Cluster the new dataset into k clusters via K-means to obtain the final partition.
```

A. Categorical Features and Numerical Features preprocessing

Consider a categorical feature fc_u $(u = 1, ..r)$ that has a domain of l values $\{d_1, d_2, ..d_l\}$. In the binary vector $[b_1, b_2, ..b_l]$, each b_v corresponds to each domain value d_v. A binary feature transform of categorical feature value d_v is the assignment of the categorical value of each sample to a binary vector of d elements $[b_1, b_2, ..b_l]$, where all of the entries are 0, except b_v.

Binary feature transforms are used to handle multi-value categorical features by setting the corresponding entries in the binary vector. Furthermore, missing values can be resolved by setting all of the binary entries to 1.

One form of uncertainty in this feature occurs when data are specified by a range of values, instead of one scalar. To correct this problem, if a numerical feature fn_u has interval data $[a, b]$, it is represented by two numeric features, $fn_{u,1} = a$ and $fn_{u,2} = b$.

Missing values of numeric features are replaced by the average of the observed value or by the k-nearest neighbors.

B. UFL Fuzzy ART clustering Algorithm

For mixed type feature data, the main challenge is to deal with both discrete and continuous variables at the same time in computing the distance between two samples since all the traditional clustering methods treat the features as numerical only. The UFL is used to remove the gap between the discrete and continuous property.

To overcome this hurdle, ART clustering is used since ART can work for mixed type data (after pre-processing), we leverage the advantage of ART clustering as the unsupervised learning algorithm. Fig. 1 depicts this process of UFL.

Moreover, ART is template based learning. The architecture summarizes the data via the examples it has seen, which makes the clusters formed represent the data structure at a specific vigilance threshold.

The algorithm sets the vigilance of the ART module to a moderate high threshold so that numerous representative clusters can be formed. N samples are then fed into the Fuzzy ART module to learn the structure of ART.

After the Fuzzy ART learning, K_F representative clusters are created and each weight $\mathbf{w}_j$ $j = 1..K_F$ connected from one cluster to the ART input is considered as the new representation of the mixed type data. This new representation removes the gap between the numeric and categorical features.

C. Unsupervised feature construction

To construct the unsupervised learning feature representation of a sample $\mathbf{x}$, the distance from $\mathbf{x}$ to w_j $j = 1..K_F$ are used to generate the UFL feature. In particular, the following feature computation is used in this paper

$$z_j = ||\mathbf{x} - \mathbf{w}_j||_2^2 \tag{6}$$

$$f_j = min(0, mean(z_1, z_2, ..., z_{K_F}) - z_j) \tag{7}$$

where $f = [f_1, f_2, ..., f_{K_F}]$ is the UFL feature representation of s and will be used for clustering.

D. Number of clusters

Before the K-means step in Algorithm 1 can be applied, the value of K has to be determined. Estimating the true value of K is a challenge for clustering analysis. To estimate the number of clusters we use a technique called visual assessment of tendency (VAT) [15, 36].

The VAT algorithm works by reordering the distance matrix. Each pixel intensity of the gray scale VAT image represents the dissimilarity between two samples. A black pixel means two samples are close and a white pixel means two samples are far from each other. Each object is identical to itself so the dissimilarity is 0, which is represented by a black pixel along the diagonal that has 0 intensity. The distance matrix is scaled so that the furthest distance corresponds to the white pixel with an intensity of 1. VAT uses a minimum spanning tree algorithm to organize the distance matrix so that the VAT image concentrates the dark block along the diagonal. Those dark blocks represent clusters of objects that close to each other and the white part that are off the diagonal represent the distances between samples in the same clusters to samples outside the cluster. VAT, therefore, can show the number of clusters along the diagonal of the VAT image.

E. Clustering Analysis and Evaluation

A critical step after clustering is analysis. This integrated methodology includes important perspectives from which to look at the clustering results.

From a statistical perspective, the number of each type of categorical feature in each cluster are counted to see which features were dominant. For numerical features, the min, max, average and standard deviation are computed. Good clustering will yield small standard deviations for each cluster, as well as averages that vary greatly from one another.

Two criteria are used to evaluate the performance of clustering. From the classification point of view, the accuracy of grouping the samples that belong to the ground truth class is computed. The resulting clusters can be classified based on the dominant number of true labels in each cluster. The average accuracy of clustering is then defined by:

$$accuracy = \frac{\Sigma_i \frac{corr_i}{N_i}}{C}, \tag{8}$$

where $corr_i$, N_i are the number of correct labels and the number of objects in cluster C_i, respectively; C is the number of clusters in the dataset.

One of the most popular external clustering validation indices is the Rand index, which is defined below.

Assuming that P is the ground truth partition of dataset X with N data objects, which is also independent from a clustering structure C resulting from the use of the UFL Fuzzy ART algorithm, for a pair of data objects $\mathbf{x}_i$ and $\mathbf{x}_j$, we will have four different cases based on how $\mathbf{x}_i$ and $\mathbf{x}_j$ are placed in C and P

Case 1: $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same clusters of C and the same category of P.

Case 2: $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to the same clusters of C but different categories of P.

Case 3: $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to different clusters of C but the same category of P.

Case 4: $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to different clusters of C and different categories of P.

Correspondingly, the number of pairs of samples for the four cases are denoted as a, b, c, and d, respectively. Because the total number of pairs of samples is N(N–1)/2, denoted as L, we have a + b + c + d = L. The Rand index can then be defined as follows, with larger values indicating more similarity between C and P:

$$Rand = \frac{a+d}{L}. \tag{9}$$

IV. EXPERIMENT AND DISCUSSION

This section demonstrates that the methodology can perform well on several real datasets to discover their structure. The approach is first verified with several datasets with known ground truth: heart disease, teaching assistant, and credit assignment datasets from the UCI repository [23]. The method is also applied to a dataset collected from the Enhanced Oil Recovery Project Survey by Oil & Gas Journal [24] to group enhanced oil recovery projects. These clustering results can help petroleum experts to better understand the data they have collected throughout years of oil production.

A. Datasets

1) Dataset with ground truth: StatLog Heart disease dataset [23]: This UCI dataset from Cleveland Clinic has both categorical and numeric features. It has six real value features, one ordered feature (the slope of the peak exercise ST segment), and three binary features (gender, fasting blood sugar > 120 mg/dl, exercise induced angina) which can all be considered as numeric features. The rest are categorical features (resting electrocardiographic results, chest pain type, thal). Totally, it has 303 records with no missing values.

Teaching assistant evaluation dataset [23]: The dataset consists of the evaluations of teaching performance of three regular semesters and two summer semester with 151 teaching assistants at the Statistics Department of the University of Wisconsin. The scores are grouped

TABLE I
ATTRIBUTES IN THE PETROLEUM DATASET

Attribute	Properties
Formation Type	categorical, multi-value, missing
Porosity (%)	numerical, range value, missing
Permeability	numerical, range value, missing, log scale
Depth (ft)	numerical, range value, missing, log scale
Gravity (°API)	numerical, range value
Viscosity (cp)	numerical, range value, log scale
Temperature (°F)	numerical, range value
Residual Oil	numerical, range value

into three groups of low, medium and high. The attributes are i) whether the TA is a native speaker ii) course instructor (25 categories) iii) course (26 categories) iv) summer or regular v) class size. This dataset is challenging since there are a lot of values for two categorical features course instructor and course.

Credit approval dataset [23]: The dataset contains 690 samples having six numeric and nine categorical features. The samples are divided into two groups, 307 approved and 383 rejected. Thirty-seven samples have missing values on seven features. This dataset is well suited for the study because it has both mixed data and missing values.

2) Dataset without ground truth: The petroleum data is collected from the the Oil & Gas Journal [24], biannually published for worldwide enhanced oil recovery projects. The data in the survey were entered manually following the designed data structure. Therefore, there are several data quality problems, such as missing values, inconsistent data, erroneous data, and typos. The survey data recorded the reservoir and petroleum fluid condition, and project start year and project evaluation until the report year. Based on research on enhanced oil recovery (EOR) screening, the domain expert selected a few significant attributes for analysis, which are listed in Table I. Among the numeric attributes, permeability, depth and viscosity had such a large dynamic range that they are represented in log scale. The main purpose of the clustering was to group data collected from several enhanced oil recovery projects. The original dataset contained a total of 460 projects.

TABLE II
VIGILANCE PARAMETER AND NUMBER OF UFL FEATURES IN HEART DISEASE, TEACHING ASSISTANT AND CREDIT APPROVAL DATASETS

	Vigilance	Number of UFL features
Heart disease	.25	29
Teaching assistant	.60	8
Credit assignment	.60	73

TABLE III
PERFORMANCE COMPARISON FOR MIXED TYPE DATA CLUSTERING OF K-PROTOTYPE, K-MEDOIDS, FUZZY ART AND UFL FUZZY ART

	UFL Fuzzy ART		K-prototype		K-medoids		Fuzzy ART	
	Acc	Rand	Acc	Rand	Acc	Rand	Acc	Rand
Heart disease	81.5	69.7	80.0	63.8	76.5	61.1	46.6	50.4
Teaching assistant	52.2	59.1	40.2	55.3	46.1	53.1	44.2	50.9
Credit assignment	86.0	75.0	79	67.1	75.0	62.5	70	50.0

B. Results and discussion of UCI datasets

In all experiments with UFL, the parameter α was fixed at 0.001 because α does not have significant influence on generating clustering nodes in the F2 layer. The learning rate γ is set at 0.9 for moderately fast learning. If γ is set at 1 (fast learning), the number of clusters in Fuzzy ART is often small and the unsupervised learning features are not meaningful. On the other hand if γ is small (slow learning), the unsupervised features are stable but the performance is slow. The main parameter to adjust in UFLA is the vigilance parameter ρ. Unlike the Fuzzy ART clustering problem where the vigilance has to be fine tuned to get the number of clusters equal to the true number of cluster in the dataset, in UFL, the vigilance adjusts roughly so that the number of clusters generated approximates the desired number of features that should be enough for the K-means steps. The vigilance values reported in Table II are representative only, other values of vigilance might result in the same performance. The K-means clustering at step 5 in Algorithm 1 is repeated ten times and the one with the smallest objective function is used for final clustering.

For comparison, several algorithms that can handle mixed type features are applied to the above datasets. These include K-prototypes [6], K-medoids [37]. Furthermore, since the proposed approach is based on Fuzzy ART, Fuzzy ART [14] clustering is also compared to demonstrate how the performance is improved.

Table III shows clearly the superior performance of UFLA clustering compared to the rest of algorithms. For the credit dataset, it has an accuracy of 86% well above the next highest accuracy, which is 79%. The teaching assistant evaluation dataset is a challenging dataset since there are many categorical values but the UFLA is still better than the other algorithms.

It is interesting to compare UFLA clustering with Fuzzy ART itself. All three datasets clearly show the effectiveness of the approach since the unsupervised features have a better representation of the mixed type data than the original data. The reason for this higher performance is the UFL features have removed the gap between the categorical and numerical features leading to a better clustering even when using with K-means in later stage. In the original form of data, although after preprocessing the data is in numerical form, the transformation in many cases make it hard to interpret the distance between the 2 samples [5].

The heart disease dataset has 303 samples and some missing values [23], which make it a good fit for this approach. Before running Algorithm 1, nominal missing values are replaced by the mode of the observed values and numerical missing values are replaced by the mean of the observed values. The FuzzyART clustering is run with vigilance 0.3 to obtain 24 unsupervised learning features.

The heart disease dataset has been used as benchmark in several mixed type data clustering such as COBWEB/3 [38], ECOBWEB [39] and SBAC [5]. Table IV gives the clustering partitions with confusion matrix and average accuracy of UFLA and COBWEB/3, ECOBWEB, SBAC. The UFL approach has the best performance among the algorithms.

C. Results and discussion of petroleum dataset

1) Clustering pre-processing: For numerical features, missing values are populated by the average of the non-missing values of the respective features. Features with interval values were split into two features, one for the lower bound and one for the upper bound. To deal with noisy data, we use whisker plots to define the noisy values and treat them as missing values.

TABLE IV
COMPARISON BETWEEN UFLA WITH COBWEB/3, ECOBWEB AND SBAC FOR HEART DATASET

Algorithm	COBWEB/3	ECOBWEB	SBAC	UFLA
Accuracy	80.7	75.4	75.1	81.5
Confusion matrix	114 33 25 131	119 59 20 105	102 38 37 126	103 21 36 143

For categorical features, missing values are populated by the mode of the observed value of the category.

Binary transform is then applied to formation type features. Permeability, depth and viscosity features are transformed to log scale. All of features are then scaled into range [0, 1] for fuzzy ART clustering.

Then we applied Algorithm 1 to the pre-processed dataset. The vigilance value was set at 0.8. There are 29 clusters formed, corresponding to 29 unsupervised features learned.

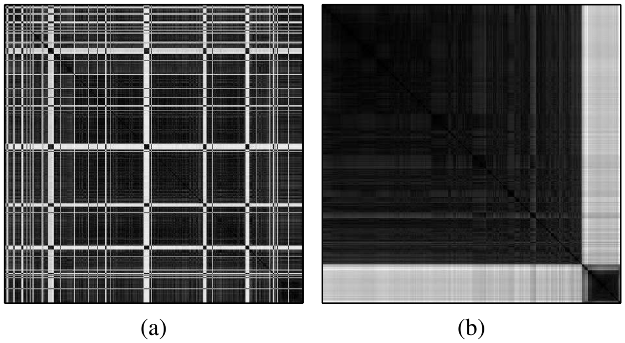
2) Define the number of clusters: VAT can facilitate the estimation of how many clusters exist by allowing the user to count the number of black squares along the diagonal. Fig. 2 shows the rearranged distance matrix resulting from the petroleum dataset according to the VAT algorithm. It clearly indicates two blocks of dark squares along the diagonal of the dissimilarity matrix. It is evident that the petroleum dataset has two clusters.

After the K-means step in Algorithm is performed with k = 2, the dataset is clustered two groups, a group of 400 and a group of 60.

3) Cluster Analysis : To understand more about the partition structure, the distribution of features in each cluster are computed. The statistics of two clusters in the final results are computed and shown in Table V and VI for both categorical and numerical features.

Table V shows the distribution of the Formation Type attribute. Each of the two clusters contained projects from a different formation type; the projects in Cluster 1 were all from sandstone formations, while those in Cluster 2 were from unconsolidated formations. So the formation type feature has a significant discrimination information in the partition structure.

Table VI shows the statistics regarding the numerical attributes of the two clusters, reveal-



(a) (b)

Fig. 2. VAT image of UFL distance matrix of the petroleum dataset: a) before organizing and b) after organizing. It shows that the data forms two clusters as obvious dark squares along the main diagonal.

TABLE V
DISTRIBUTION OF FORMATION TYPE FEATURE IN THE TWO CLUSTERS

	San	Dol	Unc	Tri	Lim	Con	Sha
C1	400	0	0	0	0	2	0
C2	0	3	50	1	2	3	1

ing that many of the attributes yielded compact clusters. For example, for porosity, the deviation was only 3.8 and 8.5, while the values ranged from 18 to 40 and 7.6 to 65 for each of the two clusters.

On the other hand, attributes that had a large dynamic range still yielded a large deviation. For example, the deviation for viscosity in the two clusters was 5×10^4 and 6×10^5, respectively. From a dimensionality reduction point of view, features with large variations tend to contribute

TABLE VI
DISTRIBUTION OF NUMERICAL FEATURES IN THE TWO CLUSTERS

	C1				C2			
	min	max	ave	std	min	max	ave	std
porosity	18	40	32	3.8	7.5	65	33	8.5
permeability	19	2e4	2260	1973	1	1.5e4	3839	4346
depth	100	9000	1491	829	175	5740	1594	1073
API	7	33	13	3.1	8	30	14.8	5.4
viscosity	10	8e5	1e4	5e+4	10	5e6	9e4	6e5
temperature	10	250	102	28	45	400	118	67
oil res.	29	100	67	15	32	100	231	15.2

less significant information to the clustering process [40].

A closer study of the two cluster statistics reveals that permeability and temperature are strong indicators for clustering information since cluster 1 has a lower average and standard deviation but higher number of samples than cluster 2.

V. CONCLUSION

A novel methodology was presented based on UFL that works with noisy, uncertain and mixed data. For mixed-data applications, UFLA was presented. UFLA can learn its features even when the amount of data is small in important subspaces of the dataset. The learned feature representations can remove the distinction in treating categorical and numeric features, leading to a better clustering result. Visual assessment tendency is used to determine the true number of clusters in the dataset when the number of cluster is unknown. Results from the application of this method to several real datasets demonstrate the effectiveness of the approach.

REFERENCES

[1] R. Xu and D. Wunsch, *Clustering*. Wiley-IEEE Press, 2009.

[2] U. Luxburg, “A tutorial on spectral clustering,” *Statistics and Computing*, vol. 17, pp. 395–416, December 2007.

[3] V. Ganti, J. Gehrke, and R. Ramakrishnan, “Cactus clustering categorical data using summaries,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1999, pp. 73–83.

[4] S. Guha, R. Rastogi, and K. Shim, “Rock: A robust clustering algorithm for categorical attributes,” *Information systems*, vol. 25, no. 5, pp. 345–366, 2000.

[5] C. Li and G. Biswas, “Unsupervised learning with mixed numeric and nominal data,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 14, no. 4, pp. 673–690, 2002.

[6] Z. Huang, “Clustering large data sets with mixed numeric and categorical values,” in *Proceedings of the 1st Pacific-Asia Conference on Knowledge Discovery and Data Mining,(PAKDD)*. Singapore, 1997, pp. 21–34.

[7] Z. Huang, “Extensions to the k-means algorithm for clustering large data sets with categorical values,” *Data Min. Knowl. Discov.*, vol. 2, no. 3, pp. 283–304, Sep. 1998.

[8] J. Ji, W. Pang, C. Zhou, X. Han, and Z. Wang, “A fuzzy k-prototype clustering algorithm for mixed numeric and categorical data,” *Knowledge-Based Systems*, vol. 30, pp. 129–135, 2012.

[9] Z. He, X. Xu, and S. Deng, "Clustering mixed numeric and categorical data: A cluster ensemble approach," *arXiv preprint cs/0509011*, 2005.

[10] C.-C. Hsu and S.-H. Wang, "An integrated framework for visualized and exploratory pattern discovery in mixed data," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 18, no. 2, pp. 161–173, 2006.

[11] C.-C. Hsu and Y.-C. Chen, "Mining of mixed data with application to catalog marketing," *Expert Systems with Applications*, vol. 32, no. 1, pp. 12–23, 2007.

[12] T.-W. Ryu and C. F. Eick, "Similarity measures for multi-valued attributes for database clustering," *Proceedings of Smart Engineering System Design Neural Network, Fuzzy Logic, Evolutionary Programming, Data Mining and Rough Sets (ANNIE 98)*, 1998.

[13] M. Chavent, F. d. A. de Carvalho, Y. Lechevallier, and R. Verde, "New clustering methods for interval data," *Computational statistics*, vol. 21, no. 2, pp. 211–229, 2006.

[14] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system," *Neural networks*, vol. 4, no. 6, pp. 759–771, 1991.

[15] J. C. Bezdek, R. J. Hathaway, and J. M. Huband, "Visual assessment of clustering tendency for rectangular dissimilarity matrices," *Fuzzy Systems, IEEE Transactions on*, vol. 15, no. 5, pp. 890–903, 2007.

[16] G. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[17] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: transfer learning from unlabeled data," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 759–766.

[18] J. C. van Gemert, J.-M. Geusebroek, C. J. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *Computer Vision–ECCV 2008*. Springer, 2008, pp. 696–709.

[19] L.-J. Li, H. Su, L. Fei-Fei, and E. P. Xing, "Object bank: A high-level image representation for scene classification & semantic feature sparsification," in *Advances in neural information processing systems*, 2010, pp. 1378–1386.

[20] A.-H. Tan, G. A. Carpenter, and S. Grossberg, "Intelligence through interaction: Towards a unified theory for learning," in *Advances in Neural Networks–ISNN 2007*. Springer, 2007, pp. 1094–1103.

[21] L. Meng, A.-H. Tan, and D. Xu, "Semi-supervised heterogeneous fusion for multimedia data co-clustering," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 26, no. 9, pp. 2293–2306, Sept 2014.

[22] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 215–223.

[23] K. Bache and M. Lichman, “UCI machine learning repository,” 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[24] “Oil and gas journal.” [Online]. Available: http://www.ogj.com/index.html

[25] Y.-l. Boureau, Y. L. Cun *et al.*, “Sparse feature learning for deep belief networks,” in *Advances in neural information processing systems*, 2008, pp. 1185–1192.

[26] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proc. Seventh IEEE Int Computer Vision Conf. The*, vol. 2, 1999, pp. 1150–1157.

[27] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE Computer Society Conf. Computer Vision and Pattern Recognition CVPR 2005*, vol. 1, 2005, pp. 886–893.

[28] Q. V. Le, W. Y. Zou, S. Y. Yeung, and A. Y. Ng, “Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis,” in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 3361–3368.

[29] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[30] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, “Unsupervised learning of invariant feature hierarchies with applications to object recognition,” in *Computer Vision and Pattern Recognition, 2007. CVPR’07. IEEE Conference on.* IEEE, 2007, pp. 1–8.

[31] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, “Greedy layer-wise training of deep networks,” *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[32] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, “Why does unsupervised pre-training help deep learning?” *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[33] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, “Exploring strategies for training deep neural networks,” *The Journal of Machine Learning Research*, vol. 10, pp. 1–40, 2009.

[34] D. S. Levine, *Introduction to neural and cognitive modeling*. Psychology Press, 2000.

[35] A.-H. Tan, H.-L. Ong, H. Pan, J. Ng, and Q.-X. Li, “Towards personalised web intelligence,” *Knowledge and Information Systems*, vol. 6, no. 5, pp. 595–616, 2004.

[36] T. C. Havens and J. C. Bezdek, “An efficient formulation of the improved visual assessment of cluster tendency (ivat) algorithm,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 5, pp. 813–822, 2012.

[37] H.-S. Park and C.-H. Jun, “A simple and fast algorithm for k-medoids clustering,” *Expert Systems with Applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[38] K. McKusick and K. Thompson, “Cobweb/3: A portable implementation,” 1990.

[39] Y. Reich and S. J. Fenves, "The formation and use of abstract concepts in design," in *Concept formation: knowledge and experience in unsupervised learning*. Citeseer, 1991.

[40] A. Zimek and J. Vreeken, "The blind men and the elephant: on meeting the problem of multiple truths in data from clustering and pattern mining perspectives," *Machine Learning*, pp. 1–35, 2013.

II. HIDDEN MARKOV MODEL WITH INFORMATION CRITERIA CLUSTERING AND EXTREME LEARNING MACHINE REGRESSION FOR WIND FORECASTING

Dao Lam[1], Shuhui Li[2] and Donald Wunsch[1]

[1]*Department of Electrical & Computer Engineering, Missouri University of Science & Technology; dlmg4,dwunsch@mst.edu*
[2]*Department of Electrical & Computer Engineering, The University of Alabama; sli@eng.ua.edu*

Abstract

Mixed type categorical and numerical data is a challenge in many applications. This general area of mixed type data is among the frontier areas where computational intelligence approaches are often brittle in comparison to the capabilities of living creatures. In this paper, unsupervised feature learning (UFL) is applied to mixed type data to achieve a sparse representation, which makes it easier for clustering algorithms to separate the data. Unlike other UFL methods that work with homogeneous data such as image, and video data, the presented UFL works with mixed type data by using Fuzzy Adaptive Resonance Theory (ART). UFL with Fuzzy ART (UFLA) obtains a better clustering result by removing the differences in treating categorical and numeric features. The advantages of doing this are demonstrated with several real world datasets with ground truth including heart disease, teaching assistant evaluation and credit approval. The approach is also demonstrated on noisy, mixed type petroleum industry data. UFLA is compared with several alternative methods. To the best of the authors' knowledge, this is the first time UFL has been extended to accomplish fusion of mixed data types.

Index Terms

Clustering, unsupervised feature learning, mixed type data, Fuzzy ART.

1 INTRODUCTION

The importance of time series data has established its analysis as a major research focus in many areas where such data appear. These data continue to accumulate, causing the computational requirement to increase continuously and rapidly. The percentage of wind power in the nation's total electrical power supply has increased quickly. Wind power is, however, known for its variability [1]. Better forecasting of wind time series is helpful to operate windmills and to integrate wind power into the grid [2, 3].

The simplest method of wind forecasting is the persistence method, where the wind speed at time '$t + \Delta t$' is predicted to be the same speed at time 't'. This method is often considered as a classical benchmark. Such a prediction is of course both trivial and useless, but for some systems with high variability it is challenging to provide a meaningful forecast that outperforms this simple approach. Another more useful example of a classical approach is the Box-Cox transform [4], which typically is used to approximate the wind time series to Gaussian marginal distribution before using the auto-regressive-moving-average (ARMA) model to fit the transformed series. However, ARMA models are often outperformed by neural network based methods [5], [6], which represent the approach mentioned in this paper.

The forecasting of time series data using neural networks has been researched widely [7], [8] due to the ability of neural networks to learn the relationship between inputs and outputs non-statistically and their lack of a requirement for any predefined mathematical models. Many wind forecasting methods have used this approach, including [9, 10]. However, training the network takes a long time due to slow convergence. The most popular training method is backpropagation, but it is known to be slow in training, additionally, its wind forecasting performance, in general, has not been as successful as other applications of backpropagation [8]. Radial basis function (RBF) trains faster but with high error and cannot handle a large amount of data due to the memory requirement for each of the training samples. The adaptive neuro-fuzzy interface system (ANFIS) predictor [11] is a fuzzy logic and neural network approach that improves on the persistence method but is still limited in terms of speed when working with large data sets.

A more successful clustering approach is the hidden Markov switching model. In [12], hidden Markov switching gamma models were used to model the wind in combination with additional information. Such approaches, however, have not used clustering techniques to group the data to the same model. Recently, [1] proposed a two-step solution for wind power generation. First, mean square mapping optimization was used to predict wind power, and then adaptive critic design was used to mitigate wind power fluctuations.

Wind speed trends change over time. Therefore, to understand the nature of wind currents, a stochastic model must be built for wind time series. Several approaches have been used in times series data analysis, the most popular of which is the hidden Markov model (HMM) [12]. However, HMM parameter estimation is known to be computationally expensive, and with such a large sequence of National Oceanic & Atmospheric Administration (NOAA) data used to model the wind, the current approaches remain unable to accomplish such estimation.

The goal of this paper is to present an effective solution for forecasting the wind time series, which is achieved by first clustering the time series data using HMM, and then using the clustering results in the extreme learning machine predictor. Therefore, this paper makes valuable contributions. From the clustering perspective, a novel method of clustering time series data is proposed that uses HMM with modified information criteria (MIC) to identify the wind time series clusters sharing the same dynamics. The paper offers the following new features to clustering using HMM: first, it provides a mechanism for handling sequential data that are simultaneously continuous and discrete; second, it proposes a method that probabilistically determines the HMM size and partition to best support clustering; and third, it makes use of the power of the Hidden Markov Model ToolKit (HTK) [13] engine, an open-source speech processing toolkit provided by Cambridge University, to induce the HMM from the time series. One of the primary advantages of the presented method compared to others is its ability to handle a large amount of time series data by leveraging HTK for HMM clustering and the extreme learning machine (ELM) to obtain the analytic solution when training the neural network. Moreover, to forecast wind, a new method for wind time series data forecasting is developed herein based on ELM. The clustering results improve the accuracy of the proposed wind forecasting method.

The paper is organized as follows. Sec. 2 provides a brief review of ELM, model selection and related work. Then, the proposed framework for wind forecasting is presented in Sec. 3. Next, in Sec. 4, the experiment is demonstrated on real data to confirm the success of the clustering approach in clustering. Sec. 5 details the performance of the approach during different seasons and forecast horizons. Finally, Sec. 6 concludes the paper with an idea for future work.

2 BACKGROUND AND RELATED WORK

2.1 Model Selection

From probabilistic mixture model-based selection, it is known that model selection involves finding the optimum number of mixtures by optimizing some criterion. In model-based clustering, mathematical models represent a cluster's structure, and model parameters are chosen to best fit the data to the models. Several criteria have been investigated in the literature, including the Akaike information criterion (AIC) [14] and the Bayesian information criterion (BIC) [15]. In general, no criterion is superior to any other, and criteria selection remains data-dependent.

In HMM clustering, BIC is often used for model selection, e.g., [15, 16, 17]. The basic definition of a BIC measure given a model λ and data X is [15]:

$$BIC = -2log\{P(X|\lambda, \hat{\theta})\} + d \cdot log(L) \tag{1}$$

where d is the number of independent parameters to be estimated in the model. L is the number of data objects, and $\hat{\theta}$ is the parameter estimation of the model λ.

Similarly, the AIC measure [14] is given as :

$$AIC = -2log\{P(X|\lambda, \hat{\theta})\} + 2d \tag{2}$$

Choosing parameters that maximize the criteria allows the best-fitting model to be selected. In both equations, $log\{P(X|\lambda, \hat{\theta})\}$, which is the data likelihood, increases as the model becomes bigger and more complicated, whereas the second term, which is the penalty term, favors simple, small models. For extended series such as wind data, computing $log\{P(X|\lambda, \hat{\theta})\}$ often requires a lot of time. This challenge is met by using the HTK Toolbox in this paper.

A comparison of (1) and (2) reveals a difference in the penalty term. Moreover various forms of BIC measures have been applied successfully in many clustering applications [18].

In addition to the problem of defining the model, HMM clustering also faces the problem of cluster validity, as do other clustering techniques [19]. In the model selection, some existing criteria, techniques, and indices can facilitate the selection of the best number of clusters. This paper follows Bayesian information criteria, which uses the best clustering mixture probability:

$$P(X|\lambda) = \prod_{i=1}^{L} \sum_{k=1}^{K} P_k * P(x_i|\lambda_k) \tag{3}$$

where X is the dataset, λ_k is the model of cluster k, x_i is the i^{th} data point in dataset X, P_k is the likelihood of x_i in cluster k, and L and K are the number of data points and clusters, respectively.

2.2 Extreme learning machine (ELM)

ELM is a feed forward single hidden layer neural network that can approximate any nonlinear function and provide very accurate regression [20, 21]. The most advantageous feature of ELM, however, is the way it is trained. Unlike other neural networks that take hours, or even days to train because of their slow convergence, ELM input weights can be initialized randomly, and ELM output weights can be determined analytically by a pseudo inverse matrix operation [21].

Let $\mathbf{X} \in R^{n\times N} = [\mathbf{x}_1, \mathbf{x}_{2,}..., \mathbf{x}_N]$ be N data used to train the ELM. To take the bias value of the neuron, $\mathbf{X}$ is transformed into $\hat{\mathbf{X}}$ by adding a row vector of all 1s, i.e. $\hat{\mathbf{X}} = [\begin{array}{c}\mathbf{X}\\ \mathbf{1}\end{array}]$.

Denote the expected output of the ELM $\mathbf{T} \in R^{k\times N} = [\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_N]$.

Denote $\mathbf{W}_i \in R^{N_H\times n}$ and $\mathbf{W}_o \in R^{k\times N_H}$as the input weight matrix and output weight matrix of ELM where N_H is the number of neurons in the hidden layer.

Doing so yields

$$\mathbf{H} = g(\mathbf{W}_i * \hat{\mathbf{X}}) \tag{4}$$

where $\mathbf{H} \in R^{N_H\times N}$ is the hidden layer output matrix of ELM and g is the nonlinear activation function of the neuron.

Once $\mathbf{H}$ is obtained, the output of the output layer can be calculated

$$\mathbf{O} = g_2(\mathbf{W}_o * \mathbf{H}) = \mathbf{W}_o * \mathbf{H} \tag{5}$$

(5) occurs because the output node activation function is linear.

For training purposes $\mathbf{O}$ should be as close to $\mathbf{T}$ as possible, i.e. $||\mathbf{O} - T|| = 0$.

ELM theory [21] states that to achieve $||\mathbf{O} - \mathbf{T}|| = 0$, $\mathbf{W}_i$ can be initialized with random value and $\mathbf{W}_o$ is computed as

$$\mathbf{W}_o = pinv(\mathbf{H}) * \mathbf{T} \tag{6}$$

where $pinv(\mathbf{H})$ represents the generalized inverse of a matrix.

Once training completed, ELM can be used for the purpose of regression or classification.

2.3 Related work

The HMM was first developed for speech processing [22], resulting in the two most successful HMM speech engines, HTK [13] and Sphinx [23]. Since then, HMMs have been applied extensively in numerous research studies and applications, including those involving handwriting, DNA, gestures, and computer vision.

In the HMM clustering literature, sequences are considered to be generated from a mixture of HMMs. The earliest work was presented in [24], in which a mixture of HMMs was regarded as a composite HMM. A new metric distance was devised between sequences using the log likelihood and clustered using hierarchical clustering.

Reference [25] extended this work to apply to electrocardiogram (ECG) data using a technique in which observations followed an auto-regressive model rather than a Gaussian mixture. Similarly, in [26] the log likelihood between the sequence and the model was used as the feature vector for the sequence.

To better choose the correct model and number of clusters for HMM clustering, [16] used the BIC. Their approach was not tested on real data and would require some modifications for practical application, as seen in Sec. 2.1. The method used in this paper, while similar to theirs, has advantages. HTK is used to learn HMM parameters and handle time series with multiple features.

To date, wind forecasting approaches have assumed continuous HMMs, but in practice, a wind time series feature vector is simultaneously discrete (for wind direction) and continuous (for wind speed). The method proposed in this paper is able to handle this problem successfully.

3 WIND TIME SERIES FORECASTING USING HMM CLUSTERING AND ELM PREDICTION

This section presents a novel framework for wind time series forecasting. The basic idea is to incorporate data available from different locations in order to achieve better prediction. The framework first clusters the wind time series into groups of similar patterns and then uses data in the same group to train an ELM to improve the prediction result.

3.1 HMM clustering using modified information criteria

Clustering, often known as unsupervised learning, allows objects possessing similar features to be grouped together. This paper presents a new method for clustering wind time series data. Each time series is modeled by an HMM, and clustering is based on the similarity between those models. The algorithm is given in Fig. 1.

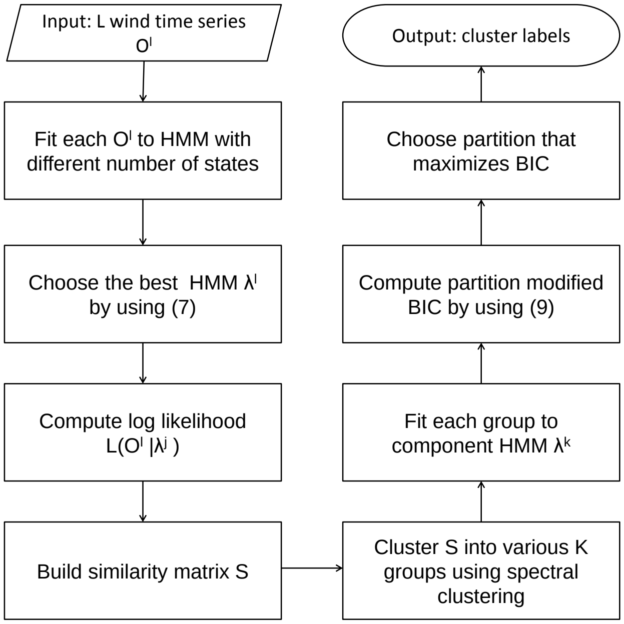

Input: L wind time series O^l
Fit each O^l to HMM with different number of states
Choose the best HMM λ^l by using (7)
Compute log likelihood L(O^l |λ^j)
Build similarity matrix S
Cluster S into various K groups using spectral clustering
Fit each group to component HMM λ^k
Compute partition modified BIC by using (9)
Choose partition that maximizes BIC
Output: cluster labels

Figure 1: Flow chart of time series clustering using MIC HMM. This process removes most of the non-local data but keeps any non-local data that fall into the same cluster as the local data

In the first step, the algorithm searches for the best model for each sequence. Each sequence essentially consists of subsequences, each of which is regarded as a sample in HTK. The HMM is learned using the HTK toolbox. HInit is randomly initialized and the model is later refined by HRest.

The log likelihood of the sequence provided by each model is used to compute the BIC measurement from (7). In this paper, BIC is modified to better work with data from a discrete HMM with numerous observations:

$$MIC = log\{P(X|\lambda, \hat{\theta})\} - \alpha d \tag{7}$$

where α is the adjusted coefficient, which will be defined in Sec. 4. The typical value of α for a discrete HMM is 0.2.

The number of independent parameters in a discrete HMM is calculated using (8):

$$d = Q^2 + QM - Q - 1 \tag{8}$$

where Q represents the number of states and M represents the number of observations.

After the best model for each sequence is found, the log likelihood $L(O^i|\lambda^j)$ is computed as the distance from sequence i to sequence j. The drawback of this step is the cost, $O(N^2)$, but for a small system, this is acceptable. This log likelihood then is used as a similarity between the two sequences i and j. Unfortunately, this likelihood is not symmetric and therefore not applicable for this clustering algorithm unless some transform is undergone. Reference [25] suggests several transforms, but for the current approach, the sum of two likelihoods produces satisfactory results.

The next steps involves finding the best partition by scanning the number of partitions from K_{min} to K_{max}. At each K, a spectral clustering algorithm [27] is used to achieve the partition. Each cluster found using spectral clustering then is modeled by an HMM using the same initialization as in step 1. Next, K component HMMs representing sub-systems are obtained. Finally, (9) is used to compute the BIC criteria, which are used to measure the quality of the configuration.

$$BIC = \sum_{i=1}^{L} log \sum_{k=1}^{K} P_k * P(O_i|\lambda_k) - \beta(K + \sum_{k=1}^{K} d_k) \tag{9}$$

where P_k is the likelihood of data given cluster k. The membership is assumed crisp, so $P_k = 1$ if sequence O_i is in cluster k, and $P_k = 0$ otherwise.

Note that in (9), the first term generally increases as K increases because smaller clusters generally result in better HMMs; therefore, the sum of the log likelihood will be higher. On the other hand, the second term increases linearly with K. Therefore, BIC will reach the peak at some K_{best} and then decrease after that.

β is an important factor to be defined. In this experiment, β depends on both α and the number S of sub-sequences in each sequence:

$$\beta = \frac{\alpha}{1 + log(S)} \tag{10}$$

3.2 Prediction using ELM

The prediction of wind speed a steps ahead is based on past samples from the target wind farms and the changes in wind speed at nearby wind farms caused by meteorological events [28]. For the purpose of wind speed prediction, a specific ELM design is used. The structure of this ELM is described in Fig. 2 as follows: i) The number of ELM prediction inputs is based on the number n of past samples of wind speed used. When predicting with clustering information, m past samples from every other time series are appended to the n past samples of the considered time series. ii) The number F of hidden nodes is defined. Researchers [21] have claimed that the performance of ELM does not depend on the number of hidden nodes if it is large enough. Therefore, in Sec. 5 a small number of hidden nodes are defined but still retain good performance. And iii) Only one output neuron is used for the forecast result.

4 EXPERIMENTAL DESIGN

To verify the proposed approach, the real wind time series is clustered to define its dynamic properties. The work is completed in MATLAB and HTK. This combination is very efficient because MATLAB provides fast programming and analysis of the result, while HTK provides

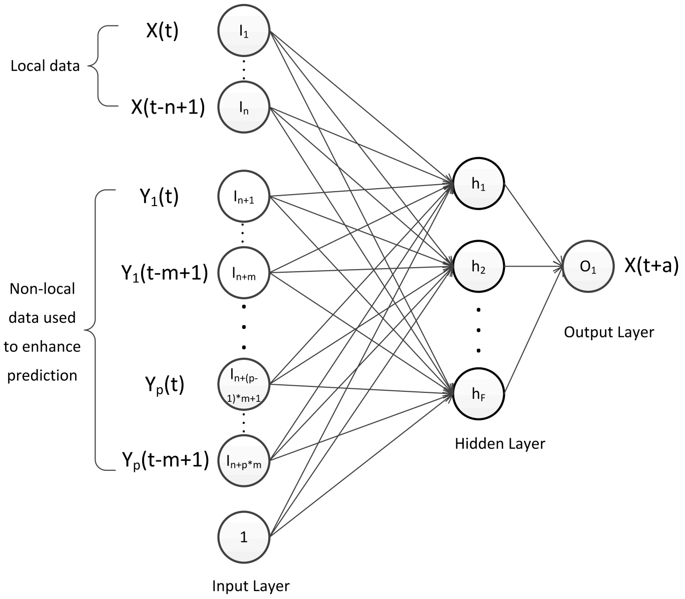


Figure 2: ELM predictor: Input layer serves as a tap delay line, hidden layer has nonlinear activation and linear output layer neuron estimates the predicted value [21]. In this figure, $X(t)$ represents the local time series and $Y_i(t)$, $i = 1..p$ are all the time series data from non-local sites clustered together with local data via the process depicted in Fig. 1. Thus, i is the index of the i^{th} non-local site in the chosen cluster; and p is the number of sites that belong to the same cluster as the local site; n is the number of past samples of local time series used as ELM inputs; m is the number of past samples from every other time series in the same cluster; F is the number of hidden nodes; and a is the number of steps ahead into the future to be predicted.

an engine with which to process the huge HMM computation. The experiment is conducted on a high-performance workstation running Ubuntu with Intel Xeon 6 cores 2.4 GHz CPU, 16 GB RAM. To speed up HMM parameter learning, the MATLAB parallel processing toolbox is utilized.

Before embarking upon a detailed description of the dataset, some important terms are clarified as follows. A site is a geographical location where data are collected. An observation is a measurement of a wind feature at a specific time and location. An observation has some features, including speed and direction. A sequence or time series is the collection of observations at a location over time. A subsequence is a sequence whose observation is limited to one year. In this paper, each site had 35 subsequences corresponding to each year. A cluster is a group of sites that share a similar HMM model.

The wind data set was obtained from the NOAA. The wind time series dataset used in this paper was collected from 15 sites around Vichy, MO, such as St. Louis, MO, Chicago, IL and

Denver, CO. Their locations are mapped in Fig. 5.

Data were collected from 1973 to 2010, and measurements were supposed to have been taken every hour. In total, the number of time samples for each sequence is a few hundred thousand. These are not averaged together by site or time. The duration of data collection is longer than that in many similar wind data studies, but it is appropriate because the clusters vary slowly. Thus, such long period can also be used in the presented approach. There are 11 temporal features for each observation, and wind speed and wind direction are the two most important. While the distribution of wind direction is fairly regular from 10 to 360 degrees, the wind speed distribution appears inconsistent, especially during gusts when the top speed is much higher than the average.

With such a long sequence of hundreds of thousands of observations, calculating the model for each sequence using an EM algorithm would be time consuming. This problem was addressed by dividing the wind time series at each location into smaller sequences according to each year. The sites with incomplete data for the entire period from 1973-2010 were not considered. In all, there were 35 sub-sequences for each location. From a speech processing perspective, each sub-sequence is regarded as a sample of a word or a sentence that can be used to train an HMM corresponding to that word or sentence.

Over years of observation for data collection, many entries in the sequences, both for wind speed and wind direction, were missed. In those instances, the missing entry was populated with the nearest available observation. Moreover, the time series were resampled into paces of one hour by averaging the observations that fell between the two consecutive hours.

The temporal features in the sequences also posed some problems. While in reality, wind direction and speed values are continuous, in the data set, wind direction was recorded as a discrete value in multiples of 10s. This has caused continuous Gaussian mixture HMM approaches, such as in [24, 25],with failure to represent the sequence.

A discrete HMM is used by discretizing the wind speed using a histogram method. Wind speed histogram bins with speeds less than 2m/s are combined with adjacent bins, and those with speeds greater than 30m/s are stored in the same bin. Combining the two discrete wind speed and direction bins yields the observation symbol, with each value corresponding to an observation in the sequence. The dataset contains 2597 different observations.

Following the proposed framework, to find the best model for each sequence, the number of states in the HMM varies between 1, 5, 10, 15, 20, 25, and 30 and the HInit from HTK is run to compute the HMM and the log likelihood of each sequence for each model.

The result of the likelihood is depicted in Fig. 3(a). As illustrated, the log likelihood increases with the size of the model. The BIC is computed using 7 for each model. The value of α changes from .1 to 1 and the value that BIC shows a clear peak is chosen. The result is plotted in Fig. 3b with $\alpha = .2$.

As indicated in Fig. 3(b), when the number of states is small, the log likelihood is dominant in the BIC, which explains the initial increase in the BIC. However, as the number of states increases, the penalty for complex configuration becomes dominant in the BIC, resulting in BIC reductions. For all of the time series, the BIC peaks after a certain number of states. Fig. 3(b) shows that this peak occurrs at Q = 10 for all sequences except for sequence 11. Therefore, an HMM with a size of 10 is used in later steps as the model for deciding the optimum number of clusters.

The dataset itself is very large, but the number of sequences is fairly small (15). Therefore, the number of clusters should be between K_{min} = 2 and K_{max} = 7. Fig. 4 depicts the result of the BIC measurement for different partitions.

Fig. 4 indicates that the BIC is highest when K = 2 and decreases as K increases. Therefore, the ideal number of clusters is 2. The clustering result appears in Fig. 5, where sites in the same cluster have the same shape.

The HMM parameter learning complexity is $O(Tx|Q|^2)$, where T is length of the sequence and Q is number of states. With this large dataset, the average time required to learn the best HMM for each site is 40 minutes, and the total time required to learn the best partition is 3 hours. This is still better than other methods, in which such learning is infeasible due to the large memory footprint. Moreover, the learning only has to be run once.

The sites on the right of the map belong to the same group and share the same dynamic wind properties. The site of particular interest in this work is Vichy, MO because the original goal of obtaining this dataset is to use the data from surrounding locations to help predict the wind

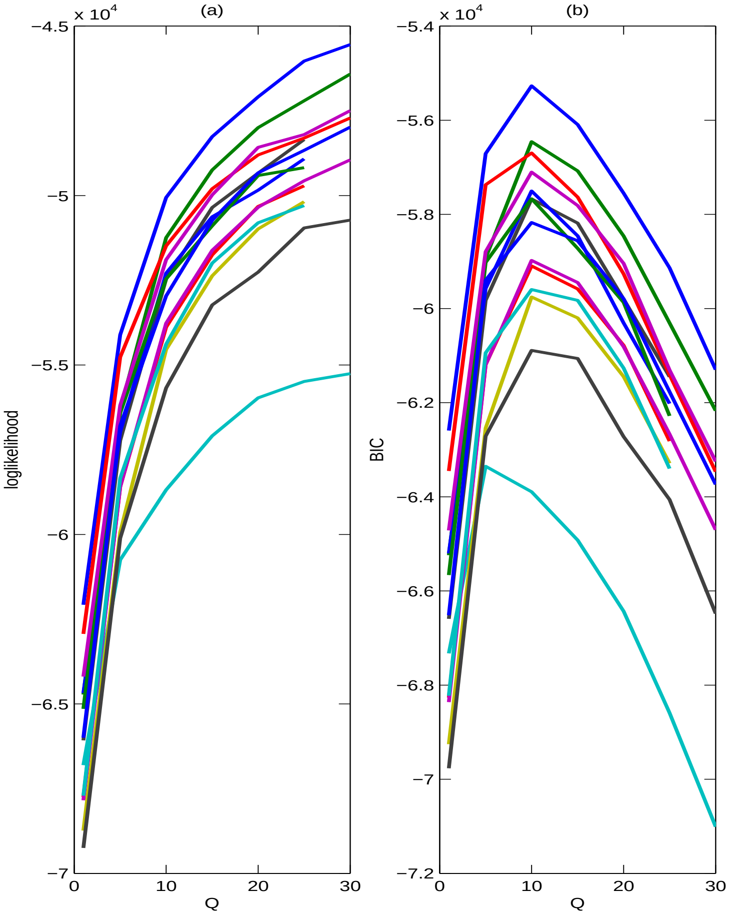


Figure 3: a) Log likelihood and b) BIC of the time sequences with the best estimating HMM plotted vs. the number of states in the HMM. Note that log likelihood increases with the number of states but BIC peaks at an intermediate number of states. Each time series is represented by different color (best viewed in color).

pattern nearby. This clustering result shows that data from other sites in this group can be used to facilitate wind prediction in Vichy, MO as discussed in next section.

The clustering results can be leveraged for wind forecasting. This paper only forecasts the wind speed, but the method can be extended easily to other wind features, such as wind direction. Many of the available wind forecasting approaches only use data collected from a single location. Making use of available data collected from sites belonging to the same clusters can enhance the prediction results, as shown in this section.

A single step ahead scheme is used for prediction, which means the wind speed is predicted one hour ahead. Once the predicted value is actually observed, the observed value is used as a feature in next prediction. The wind time series is arranged into the input matrix as discussed

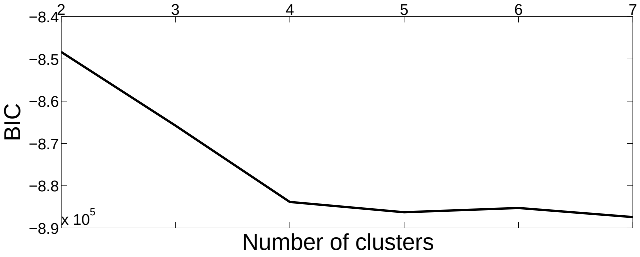


Figure 4: Cluster validity using BIC. After clustering into k clusters using spectral clustering, the partition BIC calculated using (9) is used as a cluster validity index. Higher index indicates better partitioning.

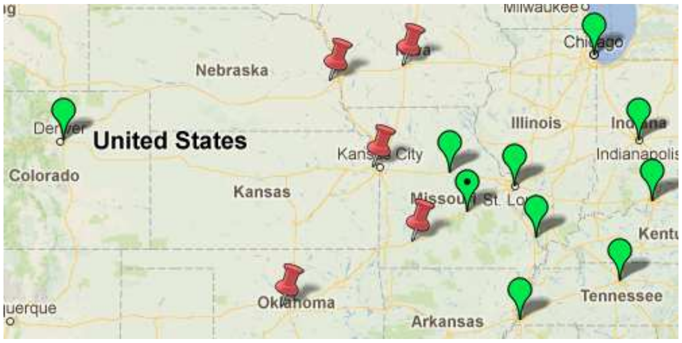


Figure 5: Result of clustering wind location: sites in the pin-shape correspond to the first cluster; sites in the balloon-shape correspond to the second cluster.

in section 3.2. Before proceeding with the prediction, the data is normalized to the range of [-1,1] as required by ELM.

5 FORECAST RESULTS AND DISCUSSION

Four experiments demonstrate the superiority of wind forecasting with clustering over forecasting without clustering and other forecasting methods. Among the 15 sites with available data, Vichy, MO is chosen as the location of the experiment and considered to be local, while the other 8 sites (data from Denver, CO excluded because it is too far from Vichy, MO) in the same cluster as Vichy, MO are considered to be a group. All of the data are divided into seasons. 20% of the final year (432 hours, with data taken equally from each season) is used for testing. The rest of the data from all of previous years constituted the training set (see Table 1). Other parameter settings are listed in Table 1. All ELMs have sigmoid activation function in the hidden layers. All reported results are averaged after 10 runs of ELM regression to take into account the randomness of input weight initialization.

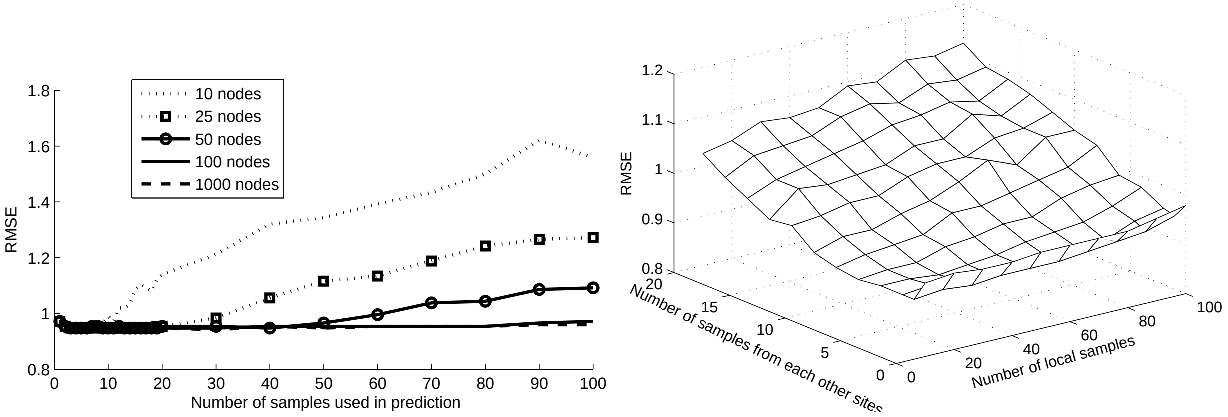
Exp.	F	n	m	Train	Test
Fig. 6	10 .. 100, 1000	1..100	0	Vichy, MO data, autumn 1975..2008+ 80% of 2009	Vichy, MO, autumn 20% of 2009
	100	1..100	1..20	Group data, autumn 1975..2009 (80% of 2009 for local data)	
Table 2	100	50	0	Vichy, MO data, autumn 1975..2008+ 80% of 2009	Vichy, MO, autumn 20% of 2009
	100	30	4	Group data, autumn 1975..2009 (80% of 2009 for local data)	
Table 3	100	50	0	Vichy, MO data; 4 seasons; 80% of 2009 or 3, 5, 10, 20, 35 previous years	Vichy, MO, 4 seasons, 20% of 2009
	100	30	4	group data; 4 seasons; 1, 3, 5, 10, 20, 35 previous years (80% of 2009 for local data)	
Fig. 7	100	30	4	Group data, spring 1975..2009 (80% of 2009 for local data)	Vichy, MO, spring 20% of 2009
	1000	100	40	Group data, spring 1975..2009 (80% of 2009 for local data)	

Table 1: Forecast parameter settings in 4 experiments

The first experiment involves defining the best ELM configuration for prediction with and without clustering information. The parameters to be determined are F, the number of hidden nodes, and n, the number of past wind data samples from Vichy, MO used as ELM inputs. When predicting with clustering information, one additional parameter, m, the number of past samples from each of the locations in the same cluster as Vichy, MO, is used as extra ELM input and has to be estimated.

Fig. 6(a) shows the error of the forecast of wind speed in autumn. For prediction with only local data, F is given representative values 10, 25, 50, 100 and 1000 while n varies from 1 to 100. As Fig. 6(a) indicates, when the number of hidden neurons is small (i.e., about 10) the ELM prediction is unstable. In fact, the root mean square error (RMSE) increases with the number of samples used for prediction. However, as the number of hidden neurons increases, the performance of the prediction becomes stable. At F= 100, the ELM predictor performes very well across a wide range of hidden nodes and the smallest error is obtained when $n = 30$.

For prediction with clustering information, F is fixed at 100 while n and m varies from 1 to 20 and 1 to 100, respectively. Fig. 6(b) shows that the lowest error prediction is reached when m = 4 and n = 30.



(a) (b)

Figure 6: ELM forecasting configuration defined with various numbers of hidden nodes and input nodes. a) Prediction without clustering information. ELM performance is stable as long as F is larger than 100. b) Best number of past samples in the local site defined along with number of past samples in each other site in the same cluster for shared-cluster prediction.

In the second experiment, the performance of the proposed approach is compared with that of other feed forward neural networks such as Backpropagation, RBF, ANFIS, and the persistence method. Backpropagation and RBF are implemented in the MATLAB Neural Network Toolbox, and ANFIS is implemented in the MATLAB Fuzzy Logic Toolbox.

	RMSE (m/s)	MAE (m/s)	MAPE	Training time (s)
Persistence (autumn 2009 data)	1.00	.76	16.27	N/A
ANFIS (2 inputs, autumn 2009 data)	1.05	.75	16.15	1.70
RBF (5 inputs, spread factor =10 with data from last 5 autumns)	.95	.72	15.94	165
Backpropagation (30 inputs, 100 hidden nodes)	1.59	1.14	24.32	6300
ELM local 30 inputs, 100 hidden nodes)	.95	.72	16.08	.24
ELM group (30 local inputs, 4 inputs from each of the sites in the same cluster, 100 hidden nodes)	.90	.70	15.20	.27

Table 2: Performance comparisons among RBF, backpropagation, ANFIS, persistence method, AR and ELM approach

As Table 2 indicates, the proposed approach performed best among the compared algorithms across all of the error performance indices: root mean square error (RMSE), mean absolute error (MAE), and mean absolute percentage error (MAPE). The time requirement is also small, just .24s for local data and .27s for group data.

The performance of the ELM group approach is also tested in various seasons and years. Table 3 shows the wind forecast in spring, summer, autumn and winter of 2009. The number of years of data used to train the ELM is 1, 3, 5, 10, 20, and 35.

RMSE	Years	1	3	5	10	20	35
Autumn	Vichy, MO data	.96	.95	.95	.94	.95	.95
	Group data	.94	.90	.89	.89	.90	.90
Summer	Vichy, MO data	1.08	1.06	1.06	1.06	1.06	1.06
	Group data	1.08	1.02	1.03	1.01	1.01	1.01
Spring	Vichy, MO data	.92	.92	.91	.90	.90	.90
	Group data	.90	.86	.86	.86	.86	.86
Winter	Vichy, MO data	1.40	1.37	1.37	1.36	1.36	1.36
	Group data	1.35	1.29	1.28	1.29	1.28	1.29

Table 3: Forecast performance for various seasons and years.

Different seasons affect the performance of ELM prediction differently. It performs best in spring, for which the RMSE error is lowest at .86 with clustering information. The prediction with clustering information performs better than without it, across all seasons.

Finally, the wind speed is forecast and compared with the true value. Forecasting is performed for both a single step ahead and up to four steps ahead with clustering information. Multiple step ahead prediction can be performed by either of two methods: In the direct method, n past samples are used to directly predict the wind speed 4 hours ahead while in the indirect method, n past samples are used to predict the wind speed one step ahead, and this prediction is recursively used to predict the next step until prediction had been made up to 4 hours ahead. Multiple simulations are performed using two methods; the direct method performed better and therefore is used in this paper.

To increase the performance, F is set at 1000. Similarly to one step ahead prediction, experiments are conducted with different numbers of past samples from the local site and clustering sites to determine that the ELM with n = 100 and m = 40 is the optimal configuration. The wind speed is forecast using this ELM configuration.

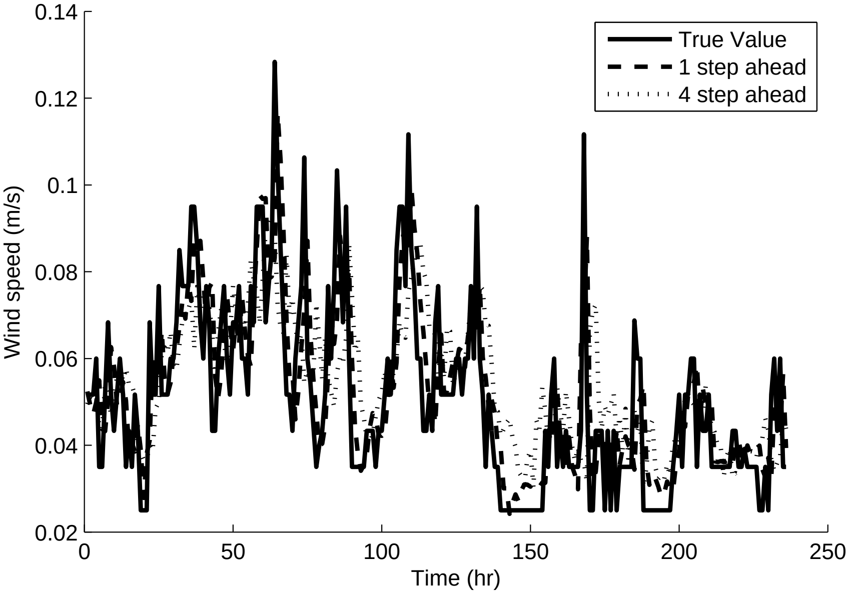


Figure 7: Comparison of forecasting and ground truth over 240-hour duration.

Fig. 7 depicts the 1 hour and 4 hour ahead forecast values and compares with the true value in a 240-hour observation.The RMSE of 4 step ahead prediction is 1.11 m/s while that of 1 step ahead is 0.86 m/s.

One potentially fruitful avenue for future investigation would be to extend the presented technique to online learning. Clustering is widely and correctly considered as a tool for online learning. However, in this application the relationships between clusters vary slowly over time. Therefore, the ELM and its rapid adaptation capability make online learning an intriguing possibility.

6 CONCLUSION

This paper presents a new method for forecasting wind time series data. By combining clustering techniques in HMM and the ELM regression, the proposed method successfully improves forecasting accuracy. The method can handle a large amount of data by leveraging the power of the HTK engine and the analytic solution for training the ELM. In the future, HMM processing will be accelerated using GPU clusters to further reduce the amount of time required for learning HMM parameters.

References

[1] G. Venayagamoorthy, K. Rohrig, and I. Erlich, “One step ahead: short-term wind power forecasting and intelligent predictive control based on data analytics,” *Power and Energy Magazine, IEEE*, vol. 10, no. 5, pp. 70 –78, 2012.

[2] B. Brown, R. Katz, and A. Murphy, “Time series models to simulate and forecast windspeed and wind power,” *Journal of Climate and Applied Meteorology*, vol. 23, pp. 1184–1195, 1984.

[3] S. Li, T. A. Haskew, K. A. Williams, and R. P. Swatloski, “Control of dfig wind turbine with direct-current vector control configuration,” *Sustainable Energy, IEEE Transactions on*, vol. 3, no. 1, pp. 1–11, 2012.

[4] G. Box and G. Jenkins, *Time series analysis, forecasting and control*, 1976.

[5] S. S. Soman, H. Zareipour, O. Malik, and P. Mandal, “A review of wind power and wind speed forecasting methods with different time horizons,” in *North American Power Symposium (NAPS)*, 2010, pp. 1–8.

[6] P. Werbos, “Brain-like prediction: New statistical foundations for prediction in the face of real world complexity,” IEEE Latin American Summer School on Computational Intelligence lecture, 2009.

[7] E. Saad, D. Prokhorov, and I. Wunsch, D.C., “Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks,” *Neural Networks, IEEE Transactions on*, vol. 9, no. 6, pp. 1456 –1470, 1998.

[8] A. Sfetsos, “A comparison of various forecasting techniques applied to mean hourly wind speed time series,” *Renewable Energy*, vol. 21, no. 1, pp. 23–35, 2000.

[9] S. Li, D. C. Wunsch, E. A. O’Hair, and M. G. Giesselmann, “Using neural networks to estimate wind turbine power generation,” *Energy conversion, ieee transactions on*, vol. 16, no. 3, pp. 276–282, 2001.

[10] K. Rohrig and B. Lange, “Application of wind power prediction tools for power system operations,” in *Power Engineering Society General Meeting, IEEE*, 2006.

[11] C. W. Potter and M. Negnevitsky, “Very short-term wind forecasting for Tasmanian power generation,” *Power Systems, IEEE Transactions on*, vol. 21, no. 2, pp. 965–972, 2006.

[12] P. Ailliot and V. Monbet, “Markov-switching autoregressive models for wind time series,” *Environmental Modelling & Software*, vol. 30, no. 0, pp. 92–101, Apr. 2012.

[13] “Htk speech recognition toolkit,” in *http://htk.eng.cam.ac.uk/*.

[14] H. Akaike, “A new look at the statistical model identification,” *Automatic Control, IEEE Transactions on*, vol. 19, no. 6, pp. 716–723, 1974.

[15] G. E. Schwarz, “Estimating the dimension of a model,” *Annals of Statistics*, pp. 461–464, 1978.

[16] C. Li and B. Gautam, “A Bayesian approach to temporal data clustering using hidden Markov models,” in *International Conference on Machine Learning*, 2000, pp. 543–550.

[17] A. Biem, J.-Y. Ha, and J. Subrahmonia, “A Bayesian model selection criterion for hmm topology optimization,” in *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, vol. 1, May 2002, pp. 989–992.

[18] J. Hu and B. Ray, “In interleaved HMM/DTW approach to robust time series clustering,” IBM, Tech. Rep., 2006.

[19] R. Xu and D. Wunsch, *Clustering*. Wiley-IEEE Press, 2009.

[20] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, “Extreme learning machine for regression and multiclass classification,” *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 513–529, 2012.

[21] G. B. Huang, Q. Y. Zhu, and C. K. Siew, “Extreme learning machine: theory and applications,” *Neurocomputing*, 2006.

[22] L. Rabiner, “A tutorial on hidden Markov models and selected applications in speech recognition,” *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, 1989.

[23] CMU, “Sphinx.” [Online]. Available: http://cmusphinx.sourceforge.net/

[24] P. Smyth, “Clustering sequences with hidden Markov models,” in *Advances in Neural Information Processing Systems*. MIT Press, 1997, pp. 648–654.

[25] A. Panuccio, M. Bicego, and V. Murino, “A hidden Markov model-based approach to sequential data clustering.” Springer, 2002, pp. 734–742.

[26] M. Bicego, V. Murino, and M. A. Figueiredo, “Similarity-based classification of sequences using hidden Markov models,” *Pattern Recognition*, vol. 37, no. 12, pp. 2281–2291, 2004.

[27] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: analysis and an algorithm,” in *NIPS*, 2001, pp. 849–856.

[28] R. Schlueter, G. Sigari, and A. Costi, “Wind array power prediction for improved operating economics and reliability,” *Power Systems, IEEE Transactions on*, vol. 1, no. 1, pp. 137–142, 1986.

III. Unsupervised Feature Learning Classification with Radial Basis Function Extreme Learning Machine using Graphic Processors

Dao Lam, *Member, IEEE,* **and Donald Wunsch,** *Fellow, IEEE*

Abstract

Ever-increasing size and complexity of datasets create challenges and potential tradeoffs of accuracy and speed in learning algorithms. This paper offers progress on both fronts. It presents a mechanism to train the unsupervised learning features learned from only one layer to improve performance in both speed and accuracy. The features are learned by an unsupervised feature learning (UFL) algorithm. Then, those features are trained by a fast Radial Basis Function (RBF) Extreme Learning Machine (ELM). By exploiting the massive parallel computing attribute of a modern Graphics Processing Unit (GPU), a customized Compute Unified Device Architecture (CUDA) kernel is developed to further speed up the computing of the RBF kernel in the ELM. Results tested on CIFAR and MNIST datasets confirm the UFL RBF ELM achieves high accuracy, and the CUDA implementation is up to twenty times faster than CPU and the naive parallel approach.

Index Terms

CUDA, ELM, neural network, RBF, SVM.

I. INTRODUCTION

In machine learning, speed and accuracy are often the two main tradeoffs. Traditional classifiers require a human operator to design features that represent high-level features of the objects. These approaches are difficult or time consuming to be applied to other kinds of data.

Dao Lam and Donald Wunsch are with the Applied Computational Intelligence Lab, Department of Electrical & Computer Engineering, Missouri University of Science & Technology, Rolla, MO 65409 USA. E-mail: dao.lam@mst.edu and wunsch@ieee.org.

A good classifier requires progressively more data for training, which also increases the time required for training. In tasks such as image classification and object recognition, UFL have shown to compete well or even outperform manually designed ones [1, 2]. UFL has also proven to be helpful in greedy layer-wise pre-training of deep architectures [3, 4, 5].

However, a major drawback of many UFL systems is their complexity where parameters like learning rate, momentum, and weight decay must be tuned and network architecture parameters must be cross-validated. Surprisingly, a simple feature learning algorithm using k-means as in [2] can perform well. By pursuing larger representations, this algorithm achieves better recognition accuracy for several datasets.

Deep learning is motivated by the need for more accuracy in supervised learning [3]. The idea of deep learning with UFL is to use greedy layer-wise unsupervised pre-training. At each layer UFL creates a set of higher level features. The new learned features are used to train the next layer of the deep network. Finally the set of layers are combined to initialize a deep network classifier such as a Support Vector Machine (SVM) [6] or logistic regression classifier.

However, producing the state of the art results becomes quite computationally intensive [7], e.g., the system in [8] takes 16,000 CPU and runs for 3 days to finish training. Most of the training time is devoted to learning high level features. Therefore, researchers typically reduce the size of datasets and models to be able to train networks in a practical amount of time and these reductions undermine the high level features. Most conventional training methods in deep learning use gradient descent searching and, therefore, have several problems including slow convergence, tuning parameters, local minima traps, or human intervention. ELM [9, 10] aims to overcome these issues. In spite of single-layer learning, ELM can greatly generalize and universally approximate any function [9, 11] and performs well in many aspects.

Several ELM techniques [12, 13, 14] exist to deal with accuracy, overfitting, unsupervised learning, and reducing size but improving the speed of ELM is often neglected. As the data increases, computing the output weights of ELM can sometimes take hours for a large dataset. CUDA has been applied to certain problems in machine learning, including SVM [15], Convolutional Neural Network (CNN) [16], and deep learning [17]. [18] previously attempted to improve the performance of ELM but the accuracy was not so high.

UFL excels at evaluating the efficiency of feature encoders but neglects the merit of the classifier. The approach presented in this paper learns the feature and classifier each in a single layer and still achieves excellent deep learning algorithms. This paper investigates the ELM known as multi-quadric RBF which presents an improvement of the UFL learning. With a surprising simplicity, ELM coupled with UFL yields a significant improvement in accuracy and does much better in training speed.

The main contributions of this work focus on two results:

1. When using ELM with multi-quadric RBF kernel, classification with UFL features gives better results than SVM and many deep learning approaches.

2. Exploiting the power of modern graphics processors GPUs enable a new CUDA kernel using GPU native code that can handle the ELM feature mapping and significantly improves learning speed compared to any other approaches of which the authors are aware.

The rest of the paper is presented as follows. In Sec. II CUDA, the UFL and ELM are briefly discussed. RBF ELM CUDA implementation is presented in Sec. III. In Sec. IV the proposed method is used on two benchmark datasets CIFAR-10 and MNIST.

II. REVIEWS

A. *k-means UFL*

In UFL, images are represented by features learned without class labels. UFL consists of 2 steps: 1) build the encoder and 2) encode the image. Data used to build the encoder are the dataset itself and computer-generated data or data collected from the internet. Building an encoder includes the following steps [2]: 1) extracting a randomly large amount of patches from the data; 2) apply preprocessing to enhance the contrast of patches 3) applying k-means clustering to the patches to obtain k centroids. Those centroids are considered as the filter banks used to encode the images.

In encoding the images into high level features, the full size image is convolved with the filter banks and passed through the triangle activation function $f(z) = max(0, mean(|z|) - |z|)$, where z is the distance from the image patch to each of the k learned centroids. These features are followed by spatial average pooling to achieve invariance to image transformation and more compact representation as well better robustness to noise and clutter. Those pooled features can then be used for classification.

See [19] for a more general discussion of unsupervised learning methods.

B. Radial Basis Function Extreme Learning Machine

In this section we follow the mathematical conventions in [9]. Let $\mathbf{W}$,$\mathbf{V}$,$\mathbf{F}$, $\mathbf{H}$, and $\mathbf{T}$ be the matrices of input weight, output weight, training features, hidden node output, and training labels.

For multi-quadric function:

$$\mathbf{H} = \mathbf{W} \ominus \mathbf{F} \tag{1}$$

where $\ominus$ mean RBF operator between matrices $\mathbf{W}$, $\mathbf{F}$ where

$$\mathbf{H}_{i,j} = \sqrt{||\mathbf{F}_i - \mathbf{W}_j||^2 + b_j^2} \tag{2}$$

where $\mathbf{H}_{i,j}$: entry at i^{th} row, j^{th} column, $\mathbf{F}_i$: feature of sample i^{th}, $\mathbf{W}_j$:weight connecting hidden node j^{th} to input layer ,b_j bias weight of hidden node j^{th}.

ELM theories [9, 10] show that hidden neural weights need not to be tuned and $\mathbf{W}$ can be randomly initialized. $\mathbf{H}$ is actually an n_H-dimension ELM feature mapped from an n_F-dimension input space.

ELM learning becomes a problem of minimizing:

$$min_{\mathbf{V}}||\mathbf{V}||_2^2 + c||\mathbf{HV} - \mathbf{T}||_2^2 \tag{3}$$

where c is the regularization factor between the error and the magnitude of output weight, which can be solved in closed form:

$$\mathbf{V} = \begin{cases} \mathbf{H}^T(\frac{\mathbf{I}}{c} + \mathbf{HH}^T)^{-1}\mathbf{T} & if\, n_S \leq n_H \\ (\frac{\mathbf{I}}{c} + \mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\mathbf{T} & if\, n_S > n_H \end{cases} \tag{4}$$

where n_S,n_H are number of samples and hidden neurons.

C. CUDA GPU Neural Network

An Nvidia GPU (e.g., M2075) has two mechanisms for parallel computing: Physically, multiple processors and stream processors; and programming-wise, massively parallel threads run in streams processors. Those threads are organized into blocks of threads. Each block is run on one multiple processor and within that multiple processor, multiple threads are scheduled to run a stream processor.

Besides the limit on the number of physical processors, a GPU also has a limit on memory. Each thread has a limited number of fast registers to store local variables. Threads in the same block have to share a small but extremely fast cache memory, and it must be synchronized between threads to prevent a racing condition.

The largest memory that can be accessed by all threads is called global memory, however the access time is slow. To compensate for the slowness, global memory is optimized for *coalesced* accesses. Coalesced access happens when consecutive threads in a block access consecutive locations in the global memory. In a highly designed kernel, this must be taken care of.

A good performance CUDA kernel must be designed to optimize the usage number of threads in blocks, as well registers, shared memory and global memory.

III. RBF ELM CUDA KERNEL ALGORITHM

The RBF ELM above requires matrix and vector manipulations. While some of the calculation such as (4) only require a basic GPU library like cuBLAS [20] and MAGMA [21], the major bottleneck of the RBF ELM is the calculation of the RBF kernel in (1) when the input feature matrix is large due to large amount of data or high dimension. In essence, the RBF kernel needs to compute the distance between the i^{th} sample to the weight vector $\mathbf{W}_j$ of the j^{th} hidden node. A naive implementation of CUDA kernel where each thread will compute each distance between the i^{th} sample and weight $\mathbf{W}_j$ will be very slow due to accessing global memory repeatedly.

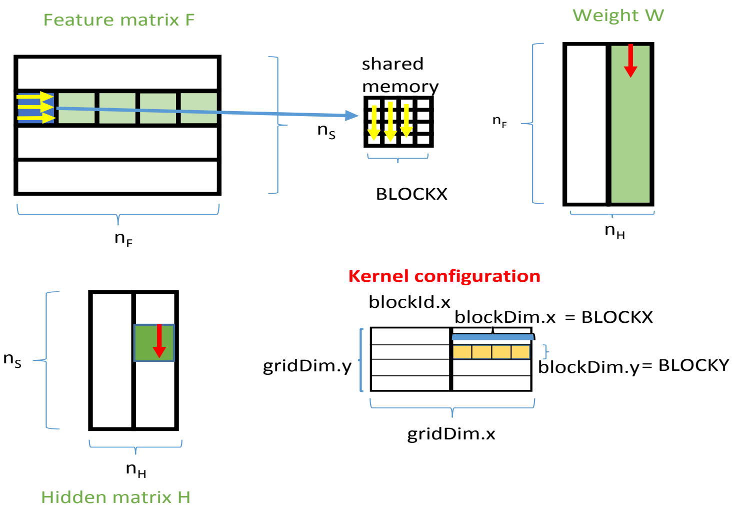


Fig. 1. RBF CUDA kernel implementation. The colored elements of the matrix each represent a block of threads. Arrows mean a portion of row or column, which are used in computing portion of distances. n_F, n_S, n_H are number of features, number of samples, and number of hidden nodes.

A. RBF CUDA kernel Algorithm

One reason for the speed-up of the presented kernel lies in its ability to compute not one distance but multiple distances between the multiple rows in feature matrix **F** and one column in the weight matrix **W** as shown in Fig. 1 by making use of the shared memory in each thread block. To obtain the maximum computing throughput in CUDA, the algorithm divides **F** into row bands and **W** into column bands, where they cooperate to compute the results. Matrix **F** is banded into bands of $BLOCKX$ rows, matrix **W** is banded into bands of $BLOCKX$ columns.

From the CPU host, CUDA kernel configurations are divided into blocks of threads shown as $BLOCKX$ x $BLOCKY$ threads in Fig. 1. The choice of $BLOCKX$ and $BLOCKY$ affects the performance of the CUDA kernel. In each thread, the distance between one column of **W** and $BLOCKX$ rows in **F** are calculated.

Threads in the same block will read elements in the same band in matrix **F** and matrix**W** to perform necessary computation and write the result into respective block in matrix **H** as illustrated in Fig. 1.

As we begin to discuss the needed algorithm, we need to establish some terminologies before getting into the actual steps of the algorithm. Denote **M**(a,b) as the element of matrix **M** at row a and column b; $\mathbf{M}$(a:c, b) as a portion of column b from row a to row c and $\mathbf{M}$(a, b:c) as a portion of row a from column b to column c.

The following Algorithm 1 controls a thread running in the CUDA kernel. Denote this thread as Th, which has $threadIdx.x$ and $threadIdx.y$ and belong to $blockIdx.x$ and $blockIdx.y$.

Thread Th will start by reading the data at location $\mathbf{F}$($blockIdx.y$* $BLOCKX$+ $threadIdx.y$, $threadIdx.x$) and write to $\mathbf{S}$($threadIdx.x$, $threadIdx.y$). It then jumps $BLOCKY$ stride in the same column to read in **F** and write to **S**. It continues to do this $BLOCKX$/$BLOCKY$ times to finish reading the column in the banded section of matrix **F**. It then has to wait for other threads in the same thread block to finish copying data from **F** to **S**. After this, the first chunk in the banded **F** is copied to the shared memory **S**.

Thread Th now computes the distances between a segment of the $\mathbf{W}$(0: $BLOCKX$ -1, $blockIdx.x$* $BLOCKX$* $BLOCKY$+ $threadIdx.y$* $BLOCKX$+ $threadIdx.x$) column in **W** and all of the columns in shared memory **S**. It uses an array C of $BLOCKX$ registers to store the accumulated distances.

Now the thread Th has to wait for other threads in the same block to finish computing their own portion distance. Once synchronization is achieved, the thread moves to the next portion of the respective banded rows and columns.

After the whole row (or column) is processed, the accumulated distance C was added with a constant bias from constant memory and then taken square root and written into the matrix **H** at the location **H** ($blockIdx.y$* $BLOCKX$: $blockIdx.y$*$BlockX$+ $BLOCKX - 1$, $blockIdx.x$* $BLOCKX$ * $BLOCKY$+ $threadIdx.y$* $BLOCKX$+ $threadIdx.x$).

The pseudo code is given in Algorithm 1..

B. Analysis

This section explains how we get the optimal performance for the new CUDA kernel Algorithm 1. The performance of a CUDA kernel depends on:

a. Massive parallelism: Threads are grouped into block of $BLOCKX$ x $BLOCKY$ threads. Threads in different blocks are totally independent so maximum concurrence can be reached. Threads in the same block are cooperative in a optimal manner so the maximum kernel occupation is reached.

Algorithm 1 RBF CUDA kernel

```
Input: Feature matrix F, weight matrixW, threadIdx, blockIdx
Allocate S[$BLOCKX$][$BLOCKX$] in shared memory
Initialize registers $C[BLOCKX]$ = {0}
Repeat {
    1. Copy memory from F to S
    2. Wait for other threads to finish copying.
    3. Compute the partial the distances C between $BLOCKX$ elements of the column of W
and rows in S.
    4. Wait for other threads to finish compute their own portion distances
    5. Increase the portion of the column in W and the row in F
} until the end of column of W
$C$= $sqrt(C)$;
Output: Write $C$ to H
```

b. Shared memory: The kernel makes use of shared memory to reduce the number of times needed to read global memory in matrix **F**. The share memory of $BLOCKX$ x $BLOCKX$ reduces the number of accessing a row in **F** by $BLOCKX$ times.

c. Coalescing: The kernel performs best when read and written to global memory in a coalesced manner. This is achieved in the algorithm. In concrete, for matrix **F**, thread Th reads data at **F**($blockIdx.y$* $BLOCKX$+ $threadIdx.y$, $threadIdx.x$) , thread $Th + 1$ reads data at **F**($blockIdx.y$* $BLOCKX$+ $threadIdx.y$, $threadIdx.x + 1$), which are right next to each other. For matrix **W** thread Th and $Th + 1$ also read data right next to each other at location **W**(0: $BLOCKX{-}1$, $blockIdx.x\ast$ $BLOCKX$* $BLOCKY{+}threadIdx.y\ast$ $BLOCKX$ $+threadIdx.x$) and **W**($0 : BLOCKX{-}1$,$blockIdx.x\ast$ $BLOCKX$* $BLOCKY{+}threadIdx.y\ast$ $BLOCKX + threadIdx.x + 1$). For the matrix **H**, when writing, Th and $Th + 1$ write the data next to each other at **H**($blockIdx.y$* $BLOCKX$*$blockIdx.y$*$BlockX$+ $BLOCKX$-1, $blockIdx.x\ast BLOCKX$*$BLOCKY$ +$threadIdx.y\ast BLOCKX$+$threadIdx.x$) and **H**($blockIdx.y$ *$BLOCKX$: $blockIdx.y$ * $BLOCKX$ + $BLOCKX$-1, $blockIdx.x$ * $BLOCKX$ * $BLOCKY$ + $threadIdx.y$ * $BLOCKX$ +$threadIdx.x$+1). Therefore, all write and read operations to the global memory are coalesced.

d. Register memory: When $BLOCKX$ is small, all the local memory in thread Th is stored in registers, which makes the memory access in the whole thread extremely fast.

C. RBF ELM CUDA Algorithm

Once the hidden output $\mathbf{H}$ in (1) is obtained, the output weight $\mathbf{V}$ can be computed using standard GPU LAPACK routine cuBLAS and MAGMA as in Algorithm 2.

```
Algorithm 2 RBF ELM CUDA algorithm
Input: Feature matrix $\mathbf{F}$, weight matrix $\mathbf{W}$
    1. Compute $\mathbf{H}$ using Algorithm 1
    2. Initialize $\mathrm{B} = \mathbf{I}/c$, where $\mathbf{I}$ is the identity matrix
    3. Compute $\mathrm{B}$=($\mathbf{I}$/c+$\mathbf{HH}^T$) : cublasgemm( $\mathbf{H}$, $\mathbf{H}^T, \mathrm{B}$), where $^T$means transpose
    4. Inverse $\mathrm{B}$: magma_getrf_gpu($\mathrm{B}$) and magma_getri_gpu($\mathrm{B}$)
    5. Compute $\mathbf{H}^T\mathrm{B}$: cublasgemm($\mathbf{H}^T$, $\mathrm{B}$, $\mathrm{B}_1$)
    6. Compute $\mathbf{V}$ = cublasgemm( $\mathrm{B}_1$, $\mathbf{T}$, $\mathbf{V}$)
Output: output weight $\mathbf{V}$
```

IV. EXPERIMENT

The UFL-RBF-ELM is implemented on a PC with Intel Xeon E5645 CPU 2.4GHz 12GB RAM, running Ubuntu 12.04 LTS. The machine is equipped with a GPU Nvidia Tesla M2075 with 6GB RAM, 448 cores clocked at 1.5 GHz, 32 KB shared memory, 63 registers in each thread. Those GPU specifications are important for selection of the best configuration of the kernel code. The CUDA version in use is CUDA 5.5 Toolkit. The RBF-ELM-CPU is implemented using linear algebra C++ Eigen library.

Since ELM is a randomization algorithm, the result discussed below is obtained after averaging 10 runnings.

A. Dataset and feature learning CIFAR-10

The dataset used to test our approach is CIFAR-10 [22], which consists of 60,000 32x32 color images in 10 classes, with 10,000 images per class, that are mutually exclusive. 50,000 of these images were used for training, each class has 10,000. The remaining 10,000 are used for testing, each class has exactly 1000 images. The dataset is organized into a 50,000 x 3072 (3072 = 32x32x3) matrix for training and 10,000 x 3072 for testing. The images have various poses, appearances, and backgrounds. Some have severe distortion. This makes it popular in the computer vision and machine learning community.

For learning the UFL feature, the implementation in [2] is used. A window with a size of 6x6 pixels is used to randomly sample the training dataset to collect 400,000 patches. Each patch then is vectorized into a column. Then, those patches are normalized and whitened to increase the contrast. We then apply k-means into the enhanced patch matrix to learn K 800/1600 centroids. Those centroids are considered as the filter banks to encode the objects.

For learning the high feature representation for each image, the full size image is convolved with the filter banks and passed through the triangle activation function $f(z) = max(0, mean(|z|) - |z|)$. These features are followed by spatial average pooling 2x2 before feeding into the ELM.

B. RBF ELM Accuracy

First the RBF ELM is compared with sigmoid ELM as well as several other state of the art algorithms to confirm the superiority of RBF ELM. The sigmoid ELM is implemented as in [18]. There are several RBFs but according to several experiments, when the number of unsupervised learning features is large, the multi-quadric function is more stable when computing the inverse than the Gaussian function. The inverse multi-quadric has the same performance as the multi-quadric but it involves an inversion which slows down the process of output weight computing in the CUDA kernel. From now on, multi-quadric function is used as default RBF.

As we increase the number of hidden neurons, the performance of ELM becomes better. At some value, the performance is not improved anymore no matter how many nodes are added to ELM. At the highest accuracy, RBF beats both SVM with a slim margin and Sigmoid ELM with a large margin as shown in the Table I. UFL RBF ELM with 6400 features achieves the highest accuracy among many state of the art algorithms such as Mean-covariance Restricted Boltzmann Machine and SVM. Some top deep learning methods such as [23, 16, 24] are more accurate than this ELM approach. However, they come with the disadvantage of long training times, while the merit of the presented approach is that it is extremely fast and the result is still comparable.

C. Single precision vs double precision CUDA

The sigmoid ELM has poorer performance than RBF ELM but its performance is stable when implemented in CUDA using single precision as discussed in [18]. On the other hand,

TABLE I
TEST ACCURACY OF UFL RBF ELM PERFORMANCE ON CIFAR-10 DATASET

Algorithm	Test Accuracy
Mean-covariance Restricted Boltzmann Machine (3 layers) [25]	71.0
Improve local coordinate coding [26]	74.5
UFL SVM L2 6400 features [2]	77.9
UFL Sigmoid ELM with 3200 features	63.1
UFL RBF ELM with 3200 features (no tune up)	77.2
UFL RBF ELM with 6400 features (c =1)	78.1
Sum-Product Networks [23]	84.1
Deep Convolutional Neural Networks [16]	87.0
Multi-Column Deep Neural Networks [24]	88.8

the RBF ELM works well in MATLAB using double precision. However, when implementing in CUDA using single precision to save memory used in GPU, the result is not stable. A cross check with MATLAB implementation at each step in RBF as in (4) shows that the matrix inversion in (4) is not stable with single precision. To confirm this hypothesis, two experiments were performed. In one experiment, every calculation is single precision except the inversion in (4) , and the result is stable. In the other experiment, every calculation is double precision except the inversion in (4), and the result becomes unstable again. Therefore, double precision must be used in the RBF ELM CUDA.

D. New kernel performance with regard to $BLOCKX$ and $BLOCKY$ parameters

As stated in the algorithm description section, the performance of the RBF kernel depends upon the kernel configuration. This section investigates that dependency. For this purpose, the ELM structure is fixed to 2048 hidden nodes. The $BLOCKX$ and $BLOCKY$ are varied (but $BLOCKX$ is still a multiple of $BLOCKY$).

For Nvidia Tesla M2075, the maximum number of threads per block is 1024. Therefore, the maximum of $BLOCKX$ is 32. Fig. 2 plots the training time of RBF ELM under different values where $BLOCKX$ = 32, 16 and 8.

When $BLOCKX$=32, we have more concurrent threads working at the same time; however, the training speed is slow. There are two main reasons explaining the setback. First, more concurrent threads in a kernel means threads have to wait longer for other threads to achieve synchronization in writing into shared memory and computing the portion distance. Second, there is not enough register memory to store the temporary memory C as well other variables in the kernel of a thread (requires at least 64 registers but M2075 has only 63). As a result of that, thread has to use global memory to store C, which significantly reduces speed. Due to the two above reasons, $BLOCKX$=32 results in the slowest kernel.

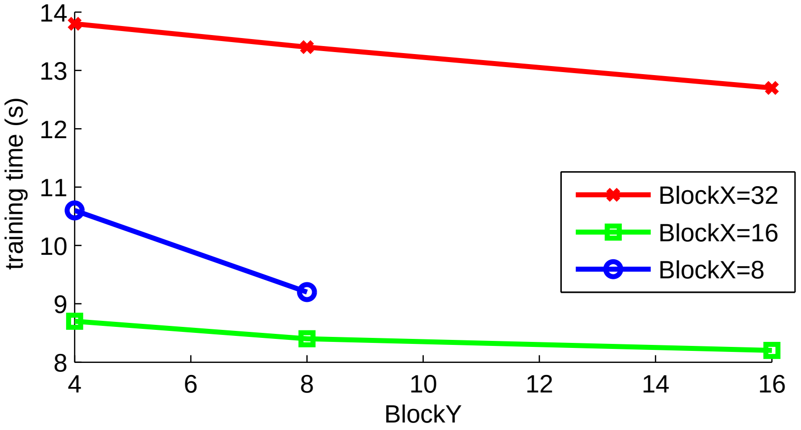


Fig. 2. Effect of parameter $BLOCKX$ and $BLOCKY$ in RBF ELM CUDA kernel. As $BLOCKY$ increases toward $BLOCKX$ the training time reduces. CUDA kernel with $BLOCKX$= $BLOCKY$ =16 achieves the best performance.

When $BLOCKX$=8, all the memory is in register but we have a smaller number of concurrent threads running at the same time. This reduces the speed of the kernel.

When $BLOCKX$=16, we get the balance of number for concurrent threads and register memory in capacity. In this case different $BLOCKY$ values do not have much effect on the speed. However, when $BLOCKY$=16, the kernel achieves the best performance. Hence $BLOCKX$=16 and $BLOCKY$=16 are chosen as the optimal configuration for the new kernel and used in later experiments.

Fig. 2 also shows that with the same size of $BLOCKX$, the kernel performs faster when $BLOCKY$ increases. Since an increase in $BLOCKY$ not only means an increase in the number of threads in block but also an increased synchronization waiting time, it means the number of concurrent threads outweighs the fact of thread synchronization in the presented kernel.

E. Speed-up

The presented kernel is compared with a GPU implementation proposed in [27], where two shared memories are used to store a portion of rows and columns to reduce the accessing of global memory. It turns out this method suffers from spending more time waiting for threads to complete copying from the global memory to two shared memories, instead of just one in the presented kernel. The presented kernel is also compared with the CPU version implemented by using highly optimized linear algebra package C++ Eigen.

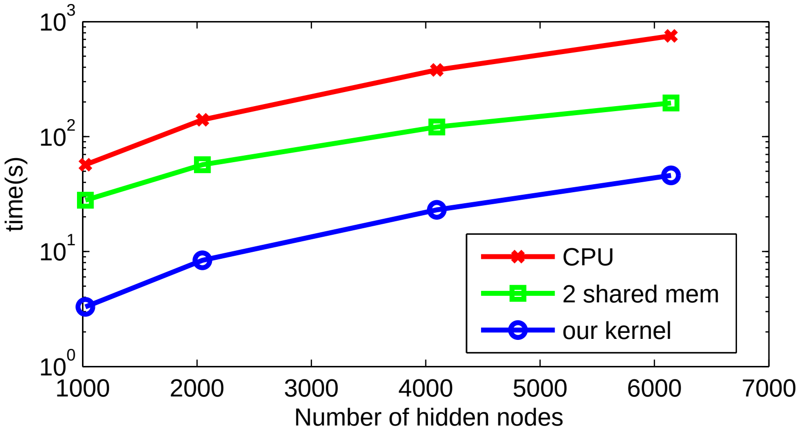


Fig. 3. Speeding up RBF ELM using CUDA. The GPU algorithm is about 4x faster than [27], and 20x faster than CPU.

As can be seen from Fig. 3, with a wide range number of hidden neural nodes in ELM , the presented GPU implementation of RBF ELM maintains a great speed-up. For example, when number of hidden of neurons is 6000, the GPU is 4x faster than [27], and about 20x faster than CPU version.

F. Experiment on MNIST

MNIST [28] was created by Yann LeCun while working on hand writing recognition using neural network. It is a historical dataset for benchmarking handwritten digit recognition. MNIST is constructed out of the original MNIST that has 60,000 training images and 10,000 test images of ten digits from 0 to 9. All the black and white digits are size normalized and centered lying at the center of the image with 28x28 pixels. Comparing to CIFAR-10, MNIST is a more standardized dataset and is better suited for testing the new algorithms. The result reported in this paper doesn't use any preprocessing for distortion like deskewing, blurring, etc.

The UFL feature learning is 6 local receptive patch, 1 pixel stride, 3200 and 6400 features.

Fig. 4 shows the performance of of ELM with different kernels when UFL has 3200 features. In particular, multi-quadric RBF is compared with sigmoid kernel. As the n_H increased, the performance of ELM also increased. In contrast with concern that the ELM may suffer from an ill-conditioned matrix when computing pseudo inverse, several runnings of multi-quadric RBF ELM with both MNIST and CIFAR show that the performance of ELM to be very stable. It is interesting to see that multi-quadric RBF had the best performance without the need to tune the regularization parameter in this case.

Sigmoid ELM performs worse than multi-quadric RBF by about 1%. For multi-quadric RBF, as n_H increases, its performance improves even over state of the art algorithms.

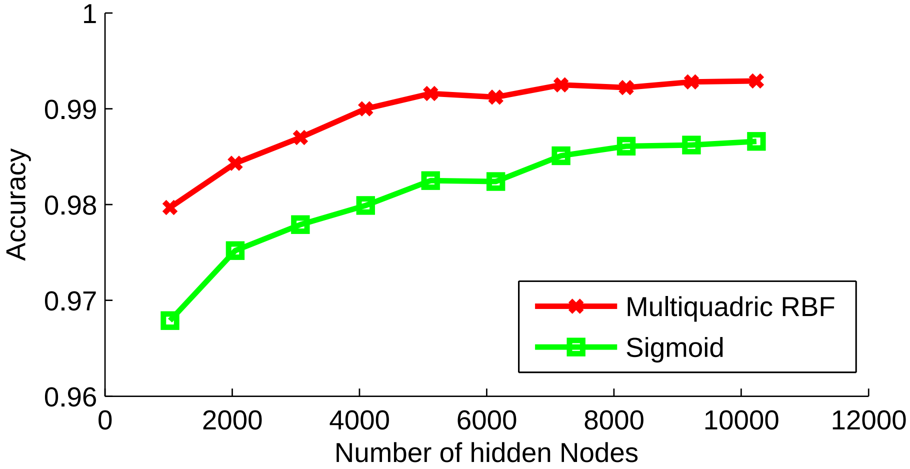


Fig. 4. Comparing Multi-quadric RBF ELM with Sigmoid ELM in MNIST dataset

Table II shows the UFL RBF ELM with 10,000 hidden nodes and compares performance with other state of the art algorithms on the MNIST. As shown, UFL-RBF-ELM outperforms CNN, Deep Belief Network (DBN), Convolutional DBN and Multi-layer ELM. Those are the multi-layer deep and large neural networks and require a long time to learn, while RBF ELM is just a single hidden layer. Although its size is large (nearly a million weights), due to ELM pseudo inversion property, it can be learned faster than any other methods in comparison. For

MNIST, the ELM algorithm does not even need to be tuned with any parameters or with any cross validation.

TABLE II
ACCURACY OF RECOGNITION OF UFL RBF ELM AND OTHER METHODS FOR MNIST DATASET

Algorithm	Test Accuracy
CNN LeNet-5 [28]	99.05
Convolutional DBN[29]	99.18
DBN [30]	99.05
Multi-layer ELM [31]	99.03
UFL Invariant [1]	99.36
ConvNet L-BFGS [32]	99.31
UFL 6400 features SVM L2 [2]	98.91
Raw pixel RBF ELM 3200 features	98.01
UFL RBF ELM with 3200 features	99.29
UFL RBF ELM with 6400 features	99.39

It can be observed from the Table II that representing images by UFL features improved the classification (from 98.01 to 99.39), and UFL training with SVM has a higher classification error than training by RBF ELM.

V. CONCLUSION

This paper presents an improvement of classification in speed and accuracy. The performance of the approach is comparable with other top-performing algorithms in accuracy but with improved speed. To achieve those advantages and to make learning the features easier, the approach makes use of the features learned from UFL k-means for universal feature learning. For faster training of the classifier, the approach uses ELM with a CUDA kernel. The test on MNIST shows that the approach beat several state of the art algorithms even when using the deep learning method. Tests on CIFAR-10 shows the approach not far from many others in accuracy, but speed is greatly improved up to twenty times faster than the CPU version using an optimized linear algebra package or four times faster than other CUDA codes.

ACKNOWLEDGMENTS

We would like to thank Adam Coates for his public codes. We also gratefully acknowledge partial support from the National Science Foundation, the Missouri S&T Intelligent Systems

Center, and the Mary K. Finley Missouri Endowment.

REFERENCES

[1] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on.* IEEE, 2007, pp. 1–8.

[2] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *International Conference on Artificial Intelligence and Statistics*, 2011, pp. 215–223.

[3] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle *et al.*, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[4] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *The Journal of Machine Learning Research*, vol. 11, pp. 625–660, 2010.

[5] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *The Journal of Machine Learning Research*, vol. 10, pp. 1–40, 2009.

[6] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[7] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, 2012, pp. 2951–2959.

[8] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on.* IEEE, 2013, pp. 8595–8598.

[9] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 513–529, 2012.

[10] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *Neural Networks, IEEE Transactions on*, vol. 17, no. 4, pp. 879–892, 2006.

[11] Y. Bengio, Y. LeCun *et al.*, "Scaling learning algorithms towards ai," *Large-scale kernel machines*, vol. 34, no. 5, 2007.

[12] Z. Bai, G.-B. Huang, D. Wang, H. Wang, and M. Westover, "Sparse extreme learning machine for classification," *Cybernetics, IEEE Transactions on*, vol. 44, no. 10, pp. 1858–1870, Oct 2014.

[13] Z. Sun, K.-F. Au, and T.-M. Choi, "A neuro-fuzzy inference system through integration of fuzzy logic and extreme learning machines," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 37, no. 5, pp. 1321–1331, 2007.

[14] G. Huang, S. Song, J. Gupta, and C. Wu, "Semi-supervised and unsupervised extreme learning machines," *Cybernetics, IEEE Transactions on*, vol. 44, no. 12, pp. 2405–2417, Dec 2014.

[15] B. Catanzaro, N. Sundaram, and K. Keutzer, "Fast support vector machine training and classification on graphics processors," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 104–111.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[17] A. Coates, B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew, "Deep learning with cots hpc systems," in *Proceedings of The 30th International Conference on Machine Learning*, 2013, pp. 1337–1345.

[18] D. Lam and D. Wunsch, "Unsupervised feature learning classification using an extreme learning machine," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–5.

[19] R. Xu and D. Wunsch, *Clustering*. Wiley-IEEE Press, 2009.

[20] NVIDIA, "Cuda basic linear algebra subroutines (cublas)." [Online]. Available: https://developer.nvidia.com/cuBLAS

[21] K. Innovative Computing Laboratory, University Tennessee, "Magma: Matrix algebra on gpu and multicore architectures." [Online]. Available: http://icl.cs.utk.edu/magma/

[22] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Computer Science Department, University of Toronto, Tech. Rep*, vol. 1, no. 4, p. 7, 2009.

[23] R. Gens and P. Domingos, "Discriminative learning of sum-product networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 3248–3256.

[24] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3642–3649.

[25] M. Ranzato and G. E. Hinton, "Modeling pixel means and covariances using factorized third-order boltzmann machines," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2551–2558.

[26] K. Yu and T. Zhang, "Improved local coordinate coding using local tangents," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 1215–1222.

[27] D.-J. Chang, M. M. Kantardzic, and M. Ouyang, “Hierarchical clustering with cuda/gpu.” in *ISCA PDCCS*, 2009, pp. 7–12.

[28] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[29] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[30] R. Salakhutdinov and G. E. Hinton, “Deep boltzmann machines,” in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.

[31] E. Cambria, G.-B. Huang, L. L. C. Kasun, H. Zhou, C.-M. Vong, J. Lin, J. Yin, Z. Cai, Q. Liu, K. Li *et al.*, “Extreme learning machines.” *IEEE Intelligent Systems*, vol. 28, no. 6, pp. 30–59, 2013.

[32] J. Ngiam, A. Coates, A. Lahiri, B. Prochnow, Q. V. Le, and A. Y. Ng, “On optimization methods for deep learning,” in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 265–272.

SECTION

3. UNSUPERVISED LEARNING WITH TIME SERIES DATA

Time series is a popular type of data. It differs from other kinds of data that the number of objects is often small but the number of observations can become really large as its observations are accumulated over time. The wind time series dataset investigated in this section is obtain from NOAA. It was collected from 1973 to 2010 from 15 sites around Vichy, MO. Each of the time series contained a few hundred thousands of observations, making it a large dataset. The dataset also suffered from the mixed feature problem as described in the first paper, where each observation has both numerical and categorical features.

The purpose of this work is to improve the forecast performance of wind speed in the windmill at Vichy, MO. There are several other methods in the past but they are often either slow or not accurate enough. Our approach to this problem is to gather more information into the forecast by applying unsupervised learning to group those wind time series similar to each other into the same group. This work chooses HMM to build the clustering algorithm. To be able to select the optimal model, the paper introduces the modified information criteria for both individual time series and group time series as in Equations (7) and (9). While building the HMMs, we leverage the power of the HTK toolkit to process a large amount of data of the wind time series.

The clustering information can be used for improving wind forecast. We propose a new forecast based on an extreme learning machine where the input is not only local data but also non-local data from the time series in the same cluster. The extreme learning machine provides an analytic form of training a neural network in a few seconds whereas the iterative training methods often take hours to finish.

Experiments are set up to select the best number of inputs and the appropriate size of the neural network. The performance of the forecast is tested on various seasons, years with one step or multiple steps ahead.

The author wants to thank the people at the Machine Intelligence Laboratory of the Cambridge University Engineering Department for providing the HTK toolkit. The author also want to thank Dr. Shuhui Li for his helpful discussion about wind energy and paper organization.

4. UNSUPERVISED FEATURE LEARNING WITH IMAGE CLASSIFICATION

Images as well as texts and videos are exploding with exponential speed. They differ from the two previous kinds of data type in which the features representing the images are not available. The images are often transformed into higher representation for the machine learning tasks by feature extractors, which are hand-crafted and data-specified. The exponential exploding of this kind of data also creates the challenges of tradeoff between speed and accuracy in image classifying.

The motivation of this section is to build universal feature learning without human design feature extractor. This feature learning also has a better performance than traditional feature extractor. The more important motivation is creating a faster image classifier to handle a large number of images in the training process.

The first motivation is achieved by using unsupervised feature learning technique, where the feature encoder is learned by clustering the small patches for unlabeled data. The feature encoders are used as filter banks to represent the image in a higher representation where the features are also sparse, which helps the image classifiers easier to recognize the objects.

The improvement of the training speed is obtained using an extreme learning machine where the neural network is trained using an analytic form, which results in a much faster training compared to backpropagation of gradient descent. The contribution of this work is two-fold. First, the paper points out that by combining unsupervised learning with an extreme learning machine of multi-quadric radial basis function, the classifying result is more accurate than most of state of the art algorithms. Second, although the use of an extreme learning machine makes the training fast, it can be even faster when implemented in a GPU. This work describes in details how to organize the number of threads, shared memory, register memory and coalescing memory access to obtain the maximal speed.

We test our approach on two large datasets CIFAR-10 and MNIST, showing that we achieve a better result than many other approaches and have a speed-up of 20 times compared to the CPU program.

The author would like to thank Adam Coates for his unsupervised feature learning codes.

VITA

Dao Lam received the B.S. degree from Post and Telecommunications Institute of Technology, Ho Chi Minh, Vietnam in 2003. He got his M.S. from Waseda University, Japan in 2008 under Japanese government scholarship. He is a member of Applied Computational Intelligence Lab. His research topics are computer vision, machine learning using computational intelligence. During his PhD, he completed several projects: image classification using CUDA, wind forecast, petroleum project categorization, video compressive sensing. He got an internship at Intelligrated Inc. as a software developer, developing robotics vision and path planning packages. He graduated with Ph.D. degree of Computer Engineering in May 2016.