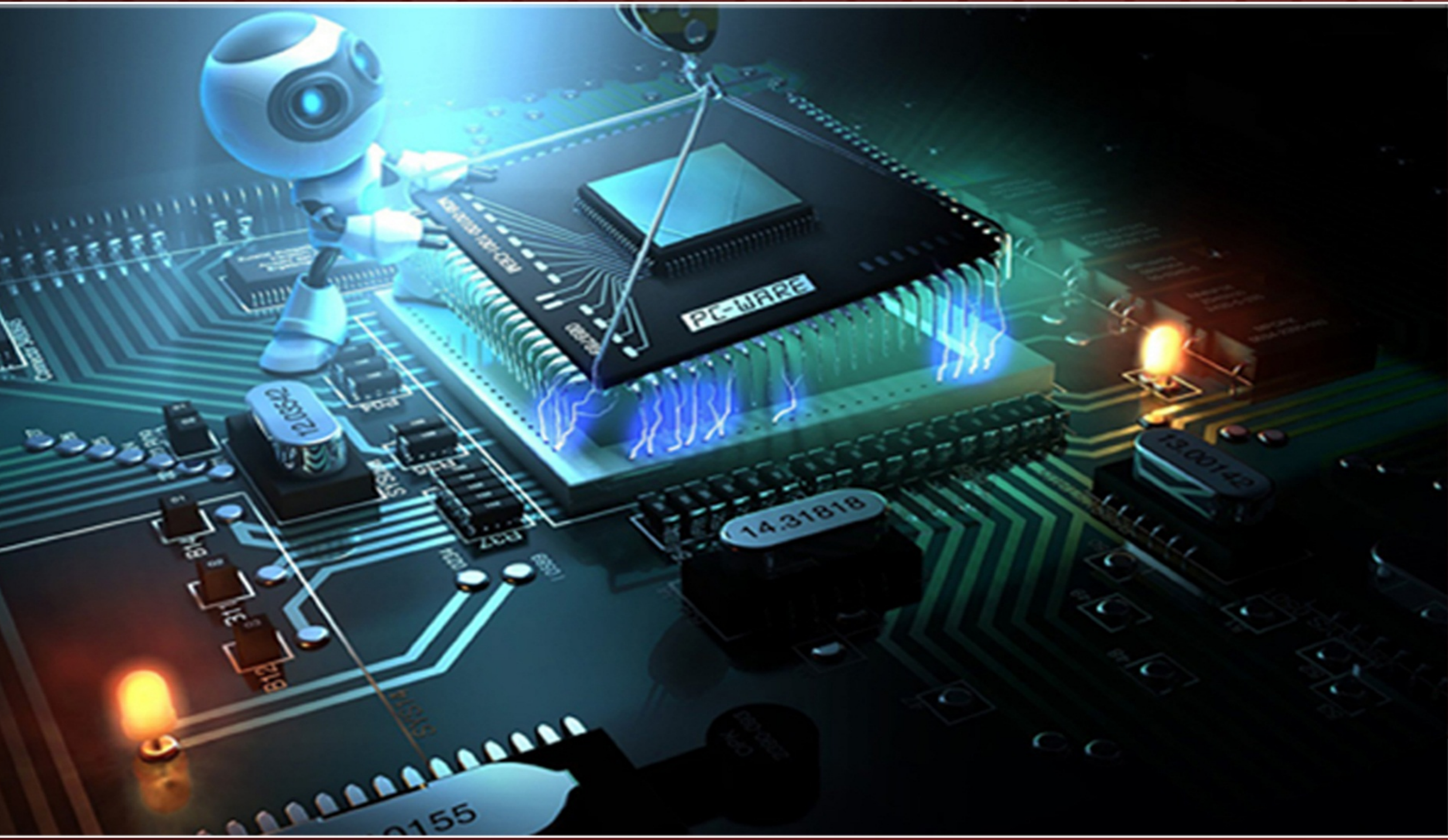
ISSN: 2054-7390

Volume 4 Issue 1, February 2016

Transactions on Machine Learning and Artificial Intelligence



Society for Science and Education
UNITED KINGDOM

TABLE OF CONTENTS

EDITORIAL ADVISORY BOARD	I
DISCLAIMER	II
An Information Reinstatement Dealing with Machine Learning Firoj Parwej and Hani Alquhayz	1
Implications of System Identification Techniques on ANFIS E-learners Activities Models-A Comparative Study Isiaka Rafiu Mope, Omidiora Elijah Olusayo, Olabiyisi Stephen O., Okediran Oladotun O., and Babatunde Ronke Seyi	15
Using Machine Learning Algorithms for Cloud Client Prediction Models in a Web VM Resource Provisioning Environment Samuel A. Ajila and Akindele A. Bankole	28
The Adept K-Nearest Neighbour Algorithm - An optimization to the Conventional K-Nearest Neighbour Algorithm Anjali Jivani, Karishma Shah, Shireen Koul and Vidhi Naik	52
Brain Tumor Segmentation using Multiobjective Fuzzy Clustering Olfa Mohamed Limam	58

EDITORIAL ADVISORY BOARD

SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
ISSN 2054-7390

An Information Reinstatement Dealing with Machine Learning

Firoj Parwej[1] **and Hani Alquhayz**[2]
[1]*Department of Computer Science, College of Science, Al-Zulfi, Majmaah University, Majmaah, Kingdom of Saudi Arabia (KSA)*
[1]f.hussain@mu.edu.sa; [2]h.alquhayz@mu.edu.sa

ABSTRACT

Information retrieval using probabilistic techniques has attracted significant attention on the part of researchers in information and computer science over the past few decades. The process of machine learning is similar to that of data mining. Both systems search through data to look for patterns. Machine learning programs detect patterns in data and adjust program actions accordingly. In this paper, we are exploring the use of machine learning techniques for information retrieval and we are using machine learning algorithms that can benefit from limited training data in order to identify a ranker likely to achieve high retrieval performance over unseen documents and queries. This problem presents novel challenges compared to traditional learning tasks, such as regression or classification. We are investigating the discriminative learning of ad-hoc retrieval models. For that purpose, we propose different models based on kernel machines or neural networks adapted to different retrieval contexts. The proposed approaches rely on different online learning algorithms that allow efficient learning over large collection and finally approaches rely on discriminative learning and enjoy efficient training procedures, which yields effective and scalable models.

Keywords: Soft Computing, Machine Learning, Information Retrieval (IR), Gaussian Mixture Model (GMM), Unsupervised learning, Supervised learning.

1 Introduction

The history of Information Retrieval (IR) parallels the development of libraries. The first civilizations had already come to the conclusions that efficient techniques should be designed to fully benefit from large document archives.

As early as 5,000 years ago, the Sumerian librarians were already describing and categorizing official transaction records, legends and theological documents in indexes. Information retrieval (IR) systems were [1] originally developed to help manage the huge scientific literature that has developed since the 1940s. Today, numerous university, corporate, and public libraries now use information retrieval systems to provide access to books, journals, and other documents.

The first automated information retrieval systems were introduced in the 1950s and 1960s. By 1970 several different techniques had been shown to perform well on small text corpora such as the Cranfield collection (several thousand documents). Large-scale retrieval systems, such as the Lockheed Dialog system, came into use early in the 1970s. In 1992, the US Department of Defense along with the National Institute of Standards and Technology (NIST), cosponsored the Text Retrieval Conference (TREC) as part

DOI: 10.14738/tmlai.41.1691
Publication Date: 6th January, 2016
URL: http://dx.doi.org/10.14738/tmlai.41.1691

of the TIPSTER text program [2]. The objective of this was to look into the information retrieval community by supplying the infrastructure that was needed for evaluation of text retrieval methodologies on a very large text collection. This catalyzed research on methods that scale to huge corpora. The introduction of web search engines has boosted the need for very large scale retrieval systems even ahead. Information Retrieval has primordially changed with the advent of computers. Digital technologies give a unified infrastructure to store, exchange and automatically process big document collections. The search for information consequently developed from the manual examination of brief document abstracts within [3] predefined categories to algorithms searching through the whole content of each archived document. Nowadays, automatic retrieval systems are widely used in several application domains (e.g. web search, book search or video search) and there is a constant need for improving such systems. The main tasks of information retrieval, the so-called ad-hoc retrieval task which target at finding the documents episodic to submit queries.

Machine Learning proposes and studies algorithms that allow computer systems to automatically improve through experience, i.e. from training data. Learning systems are commonly used for various perception tasks, [1] such as automatic face detection or automatic speech recognition. In this paper firstly our task corresponds to a ranking problem, which insinuates that the performance for a given query cannot be formalized as a sum of a measure of performance estimated for each corpus document. Secondly, most retrieval queries, current a highly disorganized setup, with a set of relevant documents, accounting only for a very small fraction of the corpus [4]. Thirdly, ad-hoc retrieval corresponds to a kind of dual generalization problem, since the learned model should not only confrontation new documents but also new queries. Finally, our task also presents stimulating efficiency compellable, since ad-hoc retrieval is typically applied to enormous corpora.

2 The Machine Learning

Machine Learning is the field of study that intends to give the computers or the machines the ability to learn from data without being explicitly programmed and act according to this information learnt [5]. It is considered a branch within Artificial Intelligence, which reckon with the idea of giving human-like intelligence to software-defined machines. Machine learning also helps us find solutions too many problems in vision, speech recognition, and robotics. Machine learning is programming computers to optimize a performance criterion using example data or past experience [6]. We have a model defined up for some parameters, and learning is the execution of a computer program to optimize the parameters of the model using the training data or past experience. The model may be predictive to make predictions on the forthcoming, or descriptive to gain knowledge from data, or both. The range of applications of Machine Learning is very comprehensive including.

- Prediction, where some current variable outputs are computed according to prior acquired results of the same variables. One area where prediction is applied is to make weather forecasts.
- Classification, were given an input point the algorithm tries to classify it into one of the groups of the model previously defined. It might be used to build classifiers for spam or fraud detection for instance.
- Pattern recognition, with applications in many fields such as text recognition (OCR), object recognition in computer vision, medicine (ECG measurements), face recognition, etc.

- Recommender platforms, which intend to predict the preference of a user with respect to a certain product, based on latest purchases or liked the items. They have become extremely famous recently, with the exponential evolution of Internet advertising companies or Amazon-like websites.
- Data mining, where the goal is to extract valuable information (patterns) out of large data sets. It is a widely used term to refer to many applications, with search engines and customer data extraction being two of the most famous.

There are several ways in which machine learning can be organized. The machine learning techniques classify techniques and algorithms based on whether the desired output is known. If this is the case, we call the problem a supervised learning problem. In this case, we possess a number of training examples, for which we know the desired output. If the desired output is not known, the problem is an [7] unsupervised learning problem. In this case, we ask the machine to optimize some function, but we do not really know what the output should be.

2.1 Supervised learning

Supervised learning problems are typical problems that humans can solve or know the answer to, but don't really know how to solve. Examples may be speech recognition or weather prediction. These techniques are often classified according to their purpose. We can discriminate between two great families of supervised machine learning techniques based on this distinction. The classification techniques occasionally, the purpose of a machine is to discriminate between a finite number of classes. For example, we may want an intelligent car to be able to discriminate between pedestrians and road signs, or a quality control robot to discriminate between correctly manufactured items and manufacturing failures, etc. This is a broad class of problems, and many different machine learning techniques focused only on these. Next regression techniques in other settings, the purpose of a machine will be to output continuous values. For example, we may want an intelligent car to be able to control the steering in such a way as to stay on the road, or to control speed so as to bring us to our destination in [8] a speedy, comfortable and safe way, or we may want a machine to be able to predict the amount of rainfall for tomorrow. The problem of supervised learning can be approached from different angles, which again leads to a classification of methods for non-parametric methods as we mentioned, it is not feasible to store all the measurement values that might ever occur, and to store the desired answer for that measurement value. However, we do have some example values with corresponding label, we could store those and perform classification or regression on new measurements based on how similar those measurements are to the stored values. These methods are called non-parametric methods.

In discriminant functions another possible approach is to select a function that will provide an output of the required format for a given input of the measurement's format. For example, in a two-class classification problem, we may choose a function that returns $f(x) = 0$ for a measurement x in class B1 and $f(x) = 1$ for a measurement in class B2. Learning then consists of finding values for the parameters of this function, so that it performs correctly on the largest possible number of examples. The model-based approaches Instead of finding a function that will optimistically provide us with the right output given some input and we could also look at what the inputs look like for each possible output.

We build a model of the data, and use it to perform the task at hand. We need to assume that the data's distribution can be delineated by a probability density function of certain family and we then need to estimate the parameters of that distribution. When we know the distribution of the data for each possible class, we can compute the probability that a new data point should be seen if it belonged to any of the classes, and based on that we can compute the probability that a data point belongs [9] to any given class. The massive mileage of this method is that it is possible to not only provide the most likely classification, but also a assuredness estimate of that classification.

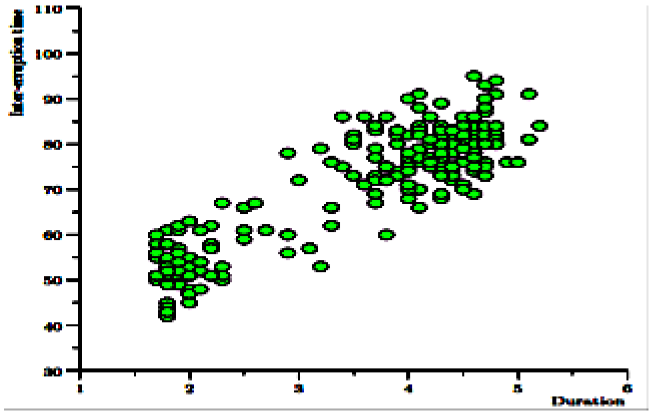


Figure 1. The some measurements that might be used for clustering. Humans will readily detect two clusters in this data. But how did we do that? Why don't we see three clusters in there?

2.2 Unsupervised learning

The unsupervised learning techniques are given the measurements, but no information as to what we expect to obtain from this data. Such techniques can be divided into two broad categories, firstly clustering and secondly dimensionality reduction.

In clustering deals with the problem of trying to group data by recognizing some structure in the data, when no corresponding labels are known. This is something that humans excel at, but that is very hard to evaluate objectively. Finding the correct number of clusters is an especially challenging task. Clustering can be extremely utilized as a form of data compression. It is often not necessary to know what the measurement was in order to perform a given task knowing which cluster it belonged to is often enough. For example, consider the task of predicting a plane's speed from its size. It is perhaps enough to know that a plane's size falls in the tourism, fighter jet or commercial transport category in order to be able to make the prediction. The precise size measurements are not occasional.

In dimensionality reduction it is over and over again possible to make many measurements simultaneously, crummy very high-dimensional data. As an example, consider a digital camera. Such a camera may have a few million sensors, each registering how much light fell on them during a given time span. Every picture taken with such a camera may therefore be seen as a very high-dimensional vector of measurements. We cannot list all the possible images that such a sensor could take, because there are far too many possibilities even though this number is finite. We do know that actual images will have characteristics that limit the number of possible valid images. For example, most pairs of neighboring pixels in an actual image will have very similar color and intensity. We can therefore detract the dimensionality of the data point without renouncing any occasional information. These automated techniques to do this are called dimensionality reduction techniques.

3 The Information Retrieval

An information retrieval system is a system that is capable of storage, retrieval, and maintenance of information. Information in this context can be composed of text (including numeric and date data), images, audio, video and other multi-media objects. The form of an object in an information retrieval system is diverse the text aspect has been the only data type that lent itself to fully functional processing. The other data types have been treated as highly informative sources, but are primarily linked for retrieval based upon a search of the text.

An information retrieval system consists of a software program that facilitates a user in finding the information the user needs [10]. The system may use standard computer hardware or specialized hardware to support the search sub function and to transform non-textual sources to a searchable media (e.g., transcription of audio to text). The gauge of the success of an information system is how well it can minimize the overhead for a user to find the needed information. From a user's perspective is the time required to find the information needed, excluding the time for actually reading the relevant data. Hence search composition, search execution, and reading non-relevant items are all aspects of information retrieval overhead. The general objective of an information retrieval system is to minimize the overhead of a user in locating needed information. The favorable outcome of an information system is very subjective, based upon what information is needed and the willingness of a user to accept overhead.

The total information storage and retrieval system is collected of four major functional processes: Item Normalization, Selective Dissemination of Information (i.e., "Mail"), archival Document Database Search, and an Index Database Search along with the automatic file build process that supports index files. The commercial systems have not integrated these capabilities into a single system, but supply them as [11] independent capabilities. In figure 2 shows the logical view of these capabilities in a single integrated information retrieval system. The boxes are used in the diagram to represent functions while disks describe data storage.

3.1 Item Normalization

The first step in any integrated system is to normalize the incoming items to a criterion format. In addition to translating multiple external formats that might be received into a single consistent data structure that can be manipulated by the functional processes, item normalization provides logical restructuring of the item. Additional operations during item normalization are needed to create a searchable data structure identification of processing tokens (e.g., words), [12] characterization of the tokens, and stemming (e.g., removing word endings) of the tokens.

3.2 Selective Dissemination of Information

The selective dissemination of information process provides the potential to dynamically differentiation newly received items in the information system against standing statements of interest of users and deliver the item to those users whose statement of interest matches the contents of the item. The Mail process is collected in the search process, user statements of interest and user mail files. The every item is received, it is processed against every user's profile. A profile contains a typically broad search statement along with a list of the user mail files that will receive the document if the search statement in the profile is satisfied. The user search profiles are different than ad hoc queries in that they contain notably more search terms and cover a wider range of interests.

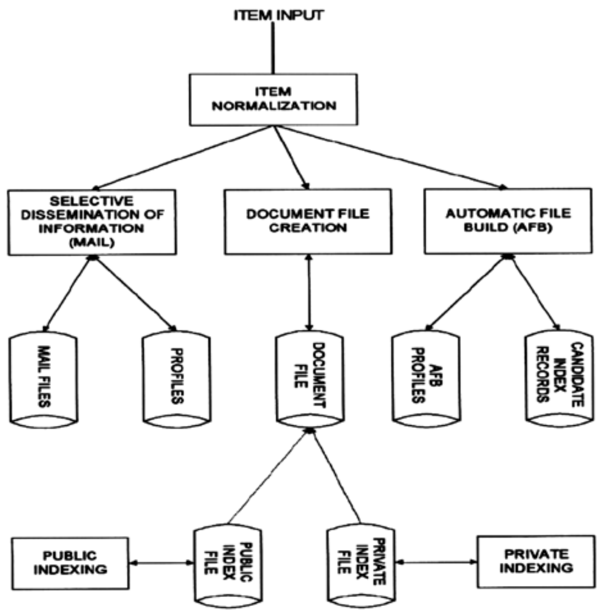


Figure 2. The Total Information Retrieval System

3.3 Document Database Search

The document database search process provides the impressibility for a query to search against all items received by the system. The document database search process is composed of the search process, the user entered queries and the document database which contains all items that have been received, processed and stored by the system. It is the retrospective search source for the system. If the user is on-line, the selective dissemination of information system delivers to the user items of interest as soon as they are processed into the system. Whichever search for information that has already been processed into the system can be considered a retrospective search for information. This does not prevent the search to have search statements constraining it to items received in the last few hours.

3.4 Index Database Search

When an item is determined to be of interest, a user may want to save it for future reference. This is in effect filing it. In an information system this is versed via the index process. In this process the user can logically store an item in a file along with additional index terms and descriptive text the user wants to associate with the item [13]. It is also possible to have index records that do not reference an item, but contain all the substantive information in the index itself.

3.5 Multimedia Database Search

From a system perspective, the multimedia data is not logically its own data structure, but an augmentation to the existing structures in the information retrieval system. It will consist almost entirely in the area described as the document database [14]. The specialized indexes allow search of the multi-media (e.g., vectors representing video and still images, text created by audio transcription) will be augmented search structures. The original source will be kept as a normalized digital real source for access possibly in their own specialized retrieval servers.

4 The related work

The core objective of keyword spotting is to discriminate between the segments of the signal belonging to a keyword utterance and the others. The first approaches based on dynamic time warping (DTW) proposed to compute the alignment distance between a template utterance of the keyword and all possible subsequences of the test signal [15]. The keyword is considered as detected for the subsequences for which the distance is below some predefined threshold. Such approaches are, however greatly affected by speaker mismatch and varying recording conditions between the template sequence and the test signal. The discrete HMMs were introduced to ASR [16], and then for keyword spotting [17]. A discrete HMM assumes that the observations of a sequence of discrete events are independent conditioned on a hidden state variable that follows a Markov process. This type of model introduces several advantages compared to dynamic time warping based approaches, including an improved robustness to speak and channel changes, when several training utterances of the targeted keyword are available.

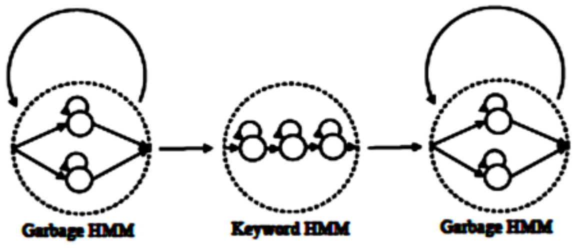


Figure 3. The HMM topology for keyword spotting with a likelihood of the sequence given the keyword is uttered

The HMMs remove the need for acoustic vector quantization, as the distributions associated with the HMM states are continuous densities, generally Gaussian Mixtures. The learning of both the Gaussian Mixture parameters and the state transition probabilities is performed in a single integrated framework, maximizing the likelihood of the training data given its transcription through the Expectation-Maximization algorithm [18]. It is now the most widely used approach for both ASR and keyword spotting.

To gain some robustness, likelihood ratio approaches have been proposed [19]. In this case, the confidence score outputted by the keyword spotter corresponds to the likelihood ratio estimated by an HMM requiring an occurrence of the keyword, and an HMM excluding it, see figure 3 and 4. The performed by comparing the outputted score to a predefined threshold. The fewer studies have proposed discriminative parameter training approaches to circumvent this weakness [19]. The maximize the likelihood ratio between the keyword and garbage models for keyword utterances and to minimize it over a set of false alarms generated by a first keyword spotter. [20] We are proposing to apply minimum classification error (MCE) to the keyword spotting problem. Other discriminative approaches have focused on combining different HMM-based keyword detectors. For instance, trains a neural network to combine likelihood ratios from different model [21].

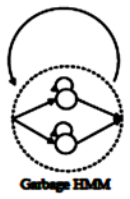


Figure 4. The HMM topology for keyword spotting with a likelihood of the sequence given the keyword is not uttered

The support vector machines to combine different averages of phone-level likelihoods. Both of these approaches propose to minimize the error rate, which equally weights the two possible spotting errors, false positive and false negative in other words missing a keyword occurrence, often called keyword deletion. This measure is however barely used to evaluate keyword spotters [22], due to the unbalanced nature of the problem. The maximization of the AUC would hence be an appropriate learning objective for the discriminative training of a keyword spotter.

5 The Information Retrieval Using Machine Learning

We focus on the learning of ranking functions for information retrieval systems. This means that we are interested in identifying a ranking function f from a set of training data, such that its expected performance on a new ranking problem is high. The two main types of learning approaches have been applied in this context, supervised and unsupervised learning. In the case of supervised learning, the training data consists of both documents and queries along with the corresponding relevance assessments [23]. This means that the learning procedure should generalize to a new ranking problem, while having access to the desired output on the training data. In the unsupervised case, the training data simply consist in a set of documents, without queries and relevance assessments. As the learning procedure has no access to examples of the desired strategy [24], it should discover some hidden structure of the data, from which it is possible to identify an effective ranking function.

In latent semantic analysis (LSA) aims at modeling term correlation [25], to overcome the term mismatch problem. For instance, one of LSA's goals is to assign a high RSV to a document which does not use any query term, but only related terms or synonyms. For that purpose, LSA assumes that the vocabulary-sized vectors actually originate from a lower dimensional space ($k < T$, the vocabulary-size), to which orthogonal noise has been added. Given a set of n training documents, represented as a matrix

$$D = [d_1, \ldots, d_n] \in \mathbf{R}^{T \times n},$$

LSA solves the least square problem,

$$D^k = \text{argmin}_{X:\text{rank}(X)=k} \, ||D - X||_2^2.$$

The replaces D with $D^k = [d^k_1, \ldots, d^k_n]$ as the denoised representation of documents. The substitution of D with Dk actually projects each document to a k dimensional subspace, and LSA hence assumes that the term mismatch problem can be solved through linear projection. The probabilistic latent semantic analysis, PLSA [26], proposes a probabilistic interpretation of the notion of topics in text documents to address the term mismatch problem. The documents can be decomposed as a mixture of aspects, where each aspect defines a multinomial over the vocabulary terms. In this model, documents and terms are considered as the observation of two discrete random variables D and T. The occurrence of a term t in a document d corresponds to the observation of the pair (t, d), which is modeled by the joint probability.

$$P(t,d) = \sum_i P(z_i)P(t|z_i)P(d|z_i),$$

Where the discrete random variable Z, of values $z_1, \dots, z_k$, is called the aspect variable. The term variable T is conditionally independent from the document variable D, given the aspect variable Z. The parameters of the model, i.e. $P(z_i)$, $P(t|z_i)$, $P(d|z_i)$ for all aspects z_i, all vocabulary terms t and all corpus documents d are learned to maximize the likelihood of the pairs (t, d) occurring in the training corpus, relying on the expectation maximization (EM) algorithm [26]. The pair classification formalizes the learning of ranking functions as a binary classification problem. Given a query q and a document d, the ranking function f should determine whether (q, d) is a positive pair, i.e. d is relevant to q, or a negative pair, i.e. d is not relevant to q. Inter-query discrimination refers to an intrinsic problem of the pair classification framework, which presents a more difficult problem to the classifier than the actual retrieval task. In this framework, the classifier should output a positive score f (q, d) > 0 for any positive pair (q, d) and a negative score f (q0, d0) < 0 for any negative pair (q0, d0).

6 The Proposed Information Reinstatement with Machine Learning

In the keyword spotting task, we are provided with a speech utterance along with a keyword k, and we should determine whether k is uttered in ̄x. The keyword spotter f can be evaluated relying on the receiver operating curve (ROC). This curve plots the true positive rate (TPR) as a function of the false positive rate (FPR). The TPR measures the fraction of keyword occurrences correctly spotted, while the FPR measures the fraction of negative utterances yielding a false alarm. The points on the curve are obtained by sweeping the threshold b from the largest value outputted by the system to the smallest one. These values, hence correspond to different trade-offs between the two types of errors a keyword spotter can make, i.e. missing a keyword utterance or rising a false alarm. In order to evaluate a keyword spotter over various trade-offs, it is common to report the area under the ROC (AUC). The AUC can be written as,

Where | · | refers to set cardinality and ‖· refers to the indicator function. The A_k hence estimates the probability that the score assigned to a positive utterance is greater than the score assigned to a negative utterance. This quantity is also referred to as the Wilcox on Mann Whitney statistics. where w is a vector of importance weights, ϕ(̄x , ̄p^k , ̄s) is a feature vector, measuring different characteristics related to the confidence that ̄p^k is pronounced in ̄x $\bar{x}$ with the segmentation ̄s. In other words, our keyword spotter outputs a confidence score by maximizing a weighted sum of feature functions over all possible segmentations. This maximization corresponds to [27] a search over an exponentially large number of segmentations. Nevertheless, it can be performed efficiently by selecting decomposable feature functions, which allows the application of dynamic programming techniques, like for HMMs our keyword spotter f is parameterized as

$$f_{\mathbf{w}}(\bar{x},\bar{p}^k) = \max_{\bar{s}} \mathbf{w} \cdot \phi(\bar{x},\bar{p}^k,\bar{s}),$$

Our objective now is to identify the vector w minimizing a regularized version of the loss $L(f_w)$ to avoid over fitting,

$$L^{\text{Reg}}(f_{\mathbf{w}}) = \|\mathbf{w}\|^2 + C \sum_{(k,\bar{x}^+,\bar{x}^-)\in T_{\text{train}}} \beta_k \; l(\mathbf{w};\bar{p}^k,\bar{x}^+,\bar{x}^-),$$

Where

$$l(\mathbf{w};\bar{p}^k,\bar{x}^+,\bar{x}^-) = \left|1 - \max_{\bar{s}} \mathbf{w}\cdot\phi(\bar{x}^+,\bar{p}^k,\bar{s}) + \max_{\bar{s}} \mathbf{w}\cdot\phi(\bar{x}^-,\bar{p}^k,\bar{s})\right|_+$$

The C is a hyper parameter setting the importance of the training loss versus the regularized. One can note that $L^{\text{Reg}}(f_w)$ is not a convex function of w. This section explains why the minimization of $L_{\text{Reg}}(f_w)$ corresponds to a large margin approach. $L^{\text{Reg}}(f_w)$ combines two terms, the regularized and the loss. The loss sums lc (f_w;¯ p^k ,¯x+ ,¯x-) over the training triplets (¯ p^k ,¯x+ ,¯x-) ε T_{train}. For each term, the lowest possible value (f_w;¯ p^k ,¯x+ ,¯x-)=0 is reached when,

$$\mathbf{w}\cdot\phi(\bar{x}^+,\bar{p}^k,\bar{s}^+) - \max_{\bar{s}} \mathbf{w}\cdot\phi(\bar{x}^-,\bar{p}^k,\bar{s}) > 1,$$

Which is equivalent to

$$\forall \bar{s}, \quad \mathbf{w}\cdot\phi(\bar{x}^+,\bar{p}^k,\bar{s}^+) - \mathbf{w}\cdot\phi(\bar{x}^-,\bar{p}^k,\bar{s}) > 1.$$

These inequalities can be rewritten as,

$$\forall \bar{s}, \quad \mathbf{u}\cdot\phi(\bar{x}^+,\bar{p}^k,\bar{s}^+) - \mathbf{u}\cdot\phi(\bar{x}^-,\bar{p}^k,\bar{s}) > \frac{1}{\|\mathbf{w}\|},$$

7 The Experiment Result

We conducted two types of experiments to evaluate the proposed discriminative approach. First, we learned the parameters of our model over the training set of TIMIT dataset, and compared its performance against an HMM baseline over the test set of TIMIT dataset. The corpus provides manually aligned phoneme and word transcriptions for each utterance. It also provides a standard split into training and testing data. From the training part of the corpus, we extract three disjoint sets consisting of 1500, 300 and 200 utterances.

7.1.1 The TIMIT Experiments

The GMM (Gaussian Mixture Model) [28] corresponds to a Bayes classifier combining one GMM per class and the phoneme prior probabilities, both learned from the training data. In this case, the log posterior of a phone given the frame vector is used as the function g. We compare the results of both models against an HMM baseline, in which each phoneme is modeled with a left-right HMM of 5 emitting states. The density of each state is modeled with a 40-Gaussian GMM.

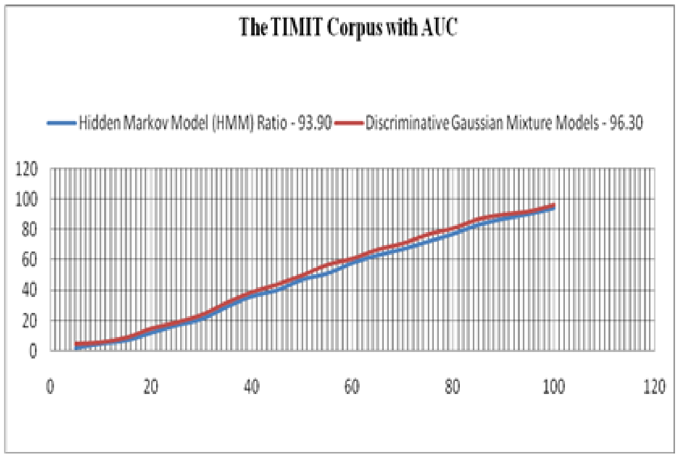


Figure 5. The TIMIT Corpus with AUC Using Discriminative Gaussian Mixture Model and HMM

Training is performed over the whole TIMIT training set. Embedded training is applied, i.e. after an initial training phase relying on the provided segmentation, a second training phase which dynamically determines the most likely segmentation is applied. The hyper parameters of this model are selected to maximize the likelihood of a held-out validation set. The evaluation of discriminative and HMM-based models is performed over 80 keywords, randomly selected among the words occurring in the test set of TIMIT. This random sampling of the keyword set aims at evaluating the expected performance over any keyword. The table 1 reports the AUC results, averaged over the 80-word test set, for the evaluated models. These results show the advantage of our approach. The two HMM based solutions are outperformed by the keyword spotters relying on our discriminative learning approach. The improvement introduced by our discriminative training algorithm can be observed when comparing the performance of the Discriminative Gaussian Mixture Model to the performance of the HMM spotters.

Table 1. The Area under the Curve (AUC) over the TIMIT Corpus

Model	The TIMIT Corpus with AUC
Hidden Markov Model Ratio	93.90%
Discriminative Gaussian Mixture Model	96.30%

8 Conclusion

Today, scenario information retrieval (IR) is a multidisciplinary field. The Humans retrieve information every time they ask a question and receive a response that addresses their question and adds to their knowledge about the queried topic. Information requests vary widely in their complexity and in the quantity of potentially relevant material that can be retrieved, as well as in the effort required to retrieve satisfactory information. Today, much of our business and cultural information is being recorded on electronic media and stored in multiple electronic databases. In order to make this information available to an information seeker, there must be an electronic information retrieval system that facilitates location and retrieval of documents that are relevant to the information seeker's question. Since information can be recorded on various media types, such as tables, images, text, audio and video, the retrieval system must be able to retrieve information from varying media representations. This work proposed a learning algorithm, which aims at maximizing the AUC over a set of training spotting problems. Our strategy is based on a large margin formulation of the task, and relies on an efficient iterative training procedure. The resulting model contrasts with standard approaches based on Hidden Markov Models (HMMs), for which the training procedure does not rely on a loss directly related to the spotting task.

Compared to such alternatives, our model is shown to yield significant improvements over various spotting problems on the TIMIT and the WSJ corpus. For instance, the best HMM configuration over TIMIT reaches 93.90% AUC, compared to 96.30% for the best Discriminative Gaussian Mixture Model spotter.

REFERENCES

[1] Frakes, William B. (1992). Information Retrieval Data Structures & Algorithms. Prentice-Hall, Inc. ISBN 0-13-463837-9.

[2] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? Communications of the ACM, 35(12):29–38, 1992.

[3] Singhal, Amit (2001). "Modern Information Retrieval: A Brief Overview". Bulletin of the IEEE Computer Society Technical Committee on Data Engineering 24 (4): 35–43.

[4] STANFILL, C. (1990a). Information Retrieval Using Parallel Signature Files. IEEE Data Engineering Bulletin, 13 (1), 33-40.

[5] C. M. Bishop, "Pattern Recognition and Machine Learning (Information Science and Statistics)," Aug. 2006.

[6] Dr. Yusuf Perwej, (2015), "An Evaluation of Deep Learning Miniature Concerning in Soft Computing" , International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE), Vol 04, Issue 02, pp 10 – 16, 28, ISSN (Print) 2319-5940, ISSN (Online) 2278-1021, with Impact Factor = 2.117 DOI : 10.17148/IJARCCE.2015.4203

[7] Christopher M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[8] C. Ji and S. Ma, "Performance and efficiency: Recent advances supervised in learning," Proc. IEEE, vol. 87, pp. 1519–1535, Sept. 1999.

[9] S. Zribi Boujelbene, D. Ben Ayed Mezghani, and N. Ellouze, "Support Vector Machines approaches and its application to speaker identification", IEEE International Conference on Digital Eco-Systems and Technologies DEST-09, Turkey, pp. 662-667, Jun 2009.

[10] Belkin, N. and W. Croft, "Retrieval Techniques", in Annual Review of Information Science and Technology, Elsevier Science publishers, New York, 1989, pages 109-145.

[11] Card, K., "Visualizing Retrieved Information: A Survey", IEEE Computer Graphics and Applications, Vol. 16, No. 2, March 1996, pages 63-67.

[12] Chalmers, M. and P. Chitson, "Bead: Explorations in Information Retrieval", Proceedings of SIGIR 92, Copenhagen, Denmark, June 1992, pages 330-337.

[13] Crew, B. and M. Gunzburg, “Information Storage and Retrieval”, U.S. Patent 3, 358, 270, December 12, 1967.

[14] Leek, T., Miller, D. and R Schwartz, "A Hidden Markov Model Information retrieval Ssystem", In Proceedings of the 22nd Annual ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pages214-221.

[15] J. S. Bridle. An efficient elastic-template method for detecting given words in running speech. In British Acoustic Society Meeting, pages 1–4, London, UK, April 1973.

[16] L. R. Bahl, P. F. Brown, P. de Souza, and R. L. Mercer. ,“Maximum mutual information estimation of hidden markov model parameters for speech recognition” , In International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pages 49–52, Tokyo, Japan, April 1986.

[17] J. G. Wilpon, L. R. Rabiner, C. H. Lee, and E. R. Goldman. ,“Automatic recognition of keywords in unconstrained speech using hidden markov models”, IEEE Transactions on Acoustics, Speech and Signal Processing (TASSP), 38 (11):1870–1878, 1990.

[18] J. A. Bilmes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical Report TR-97-021, International Computer Science Institute, Berkeley, CA, USA, 1998.

[19] M. Weintraub. LVCSR log-likelihood ratio scoring for keyword spotting. In International Conference on Acoustics, Speech, and Signal Processing (ICASSP), volume 1, pages 297–300, Detroit, MI, USA, May 1995.

[20] E. D. Sandness and I. Lee Hetherington. Keyword-based discriminative training of acoustic models. In International Conference on Spoken Language Processing (ICSLP), volume 3, pages 135–138, Beijing, China, October 2000.

[21] Y. Benayed, D. Fohr, J. P. Haton, and G. Chollet. Confidence measures for keyword spotting using support vector machines. In International Conference on Acoustics, Speech, and Signal Processing (ICASSP), volume 1, pages 588–591, Hong Kong, China, April 2003.

[22] R. A. Sukkar, A. R. Seltur, M. G. Rahim, and C. H. Lee. Utterance verification of keyword strings using word-based minimum verification error training. In International Conference on Acoustics, Speech, and Signal Processing (ICASSP), volume 1, pages 518–521, Atlanta, GA, USA, May 1996.

[23] Dr. Yusuf Perwej, “The Bidirectional Long-Short-Term Memory Neural Network based Word Retrieval for Arabic Documents” Transactions on Machine Learning and Artificial Intelligence (TMLAI) which is published by Society for Science and Education, Manchester, United Kingdom (UK), Volume 03, No.01, Pages 16 – 27, 02 February 2015, ISSN 2054 - 7390, DOI : 10.14738/tmlai.31.863

[24] Forsyth, R. and R. Rada, "Adding an Edge", in Machine Learning: application in expert systems and information retrieval, Ellis Horwood Ltd., 1986, pages 198-212.

[25] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. Journal of the American Society of Information Science, 6(41):391–407, 1990.

[26] T. Hofmann. Unsupervised learning by probabilistic latent semantic analysis. Machine Learning, 42(1):177–196, 2001.

[27] C. Cortes and M. Mohri. Confidence intervals for the area under the roc curve. In Advances in Neural Information Processing Systems (NIPS), Vancouver, Canada, December 2004.

[28] Reynolds, D.A., Rose, R.C.: Robust, "Text-Independent Speaker Identification using Gaussian Mixture Speaker Models", IEEE Transactions on Acoustics, Speech, and Signal Processing 3(1) (1995

Society for Science and Education United Kingdom
TMLAI Transactions on Machine Learning and Artificial Intelligence
ISSN 2054-7390

Implications of System Identification Techniques on ANFIS E-learners Activities Models-A Comparative Study

[1]Isiaka Rafiu Mope, [2]Omidiora Elijah Olusayo, [3]Olabiyisi Stephen O., [4]Okediran Oladotun O., and [5]Babatunde Ronke Seyi

[1,5] Department of Computer Science, College of Information and Communication Technology, Kwara State University, Malete, Ilorin, Nigeria;

[2,3,4] Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria;

abdulrafiu.isiaka@kwasu.edu.ng; eoomidiora@lautech.edu.ng; soolabiyisi@lautech.edu.ng; oookediran@lautech.edu.ng; ronke.babatunde@kwasu.edu.ng

ABSTRACT

Efficient e-learners activities model is essential for real time identifications and adaptive responses. Determining the most effective Neuro- Fuzzy model amidst plethora of techniques for structure and parameter identifications is a challenge. This paper illustrates the implication of system identification techniques on the performance of Adaptive Network based Fuzzy Inference System (ANFIS) E-learners Activities models. Expert knowledge and Historical data were used to formulate the system and their performances were compared. Similarly, comparison was made between memberships functions selected for Historical data identification. The efficiencies of the simulated models in MATLAB editor were determined using both classification uncertainty metrics and confusion matrix–based metrics. The classification uncertainty metrics considered are Mean Absolute Error (MAE) and Root Mean-Squared Error (RMSE). The confusion matrix-based metrics used are Accuracy, Precision and Recall. It was discovered that the model based on Experts Knowledge after training outperformed those based on Historical Data. The performances of the Membership Functions after ranking are Sigmoid, Gaussian, Triangular and G-Bells respectively.

Keywords: Neuro Fuzzy Model; E-learners Activities; System Identification Technique; Dataset Normalization.

1 Introduction

Adoption of Artificial Intelligent (AI) techniques for modelling complex problems is gaining prominence over mathematical models and traditional statistics [9]. Also, within the AI contemporary researches, the Soft computing techniques is becoming the most implemented when compared with the Hard computing techniques. Reasons adduced to this phenomenon essentially are the inherent features of the former which include tolerance of imprecision, incomplete or corrupt input data, capability to solve problems through repeated observation and adaptation [4], others are inspiration by natural processes and availability of simulation tools. The Soft computing technique which is also known as the Scruffy

DOI: 10.14738/tmlai.41.1799
Publication Date: 21st February, 2016
URL: http://dx.doi.org/10.14738/tmlai.41.1799

Techniques is less provable but are yielding useful and significant results [8]. The technique is the fusion of Fuzzy Computing, Evolutionary computing, Artificial Neural Networks and Probabilistic computing [15].

The Soft Computing techniques have their individual strengths and weaknesses which could be optimised by hybridization. Neuro Fuzzy techniques for instance integrate the learning capability, generalization capacity and standard architecture of Artificial Neural Networks (ANN) with the linguistic rule base and explicit internal operations of Fuzzy Inference System (FIS) [10]. The Hybrid intelligent systems built on these techniques have proven to be one of the best solutions in data modelling due to their capability to reason and learn in an environment of uncertainty and imprecision [3]. However, being a Scruffy technique, researches are being intensified at determining the most efficient approaches for optimizing the various operational components.

Identification of appropriate structure of ANN that could maximally model a scenario is a challenge, most a times the Trial and Error approach are adopted [1]. It could by incremental approaches [5], using number of rules, number of hidden layers or input clustering techniques [2]. This limitation could be ameliorated with a fusion of the ANN with FIS to form a hybrid Neuro Fuzzy model. The FIS built for the problem in accordance with available human expert knowledge suggests the structure for the ANN. Though the FIS model on its own may not be optimized due to human incapability to identify and or represent all possible instances. Even more that, no standard methods exist for transforming human knowledge or experience into the rule base and database of FIS [14]. Nonetheless, the fused ANN optimizes the system by adjusting the parameters during training. Among other Neuro Fuzzy techniques that have been implemented, ANFIS seems to be the most widely applied [7]. Its application specifically and that of ANN generally covers areas such as Function Fitting, Pattern recognition, Data clustering and Time series analysis.

This work demonstrates the implementation of this approach, the implication of each stage from formulation of FIS, transformation to ANFIS, the training and the effects of varied membership functions on the performance of E-Learners Activities Model.

2 Modelling E-Learner's Concentration and Exploration

E-Learners modelling are commonly meant for performance prediction or adaptive system. This study is a diversion from the common trends, in that it modelled quality of learner's experiences on a course. It was premised on the fact that formal knowledge requires formal experience [13]. Hence, the extent of knowledge acquired could be presumed from the quality and quantity of experiences acquired. The two behaviours considered were the learner's extent of content exploration and level of concentration. These behaviours are composite and of low bandwidth that cannot be measured directly, they can only be deduced from simple and measurable activities [11]. For actualization, the learner's Exploration was measured using the time spent on the course and extent of course content coverage. Other variables used for measuring the Concentration are Participation Index and Diagnosis Assessment Grade.

3 Methodology

3.1 Adoption of Expert Knowledge Techniques for Model Identification

Expert knowledge which is one of the common approaches for system identification in user modelling was employed as the base. The Expert knowledge based fuzzy system was built as described in [11], [6]. There, the number (N) of observable behaviours for modelling was two. Given as:

N = 2 = {N1, N2} = {Exploration, Concentration}, where N1 (Exploration) = {B1, B2} = {total time spent reading, content completion status} and N2 (Concentration) = {B3, B4} = {the Participation Index, diagnosis assessment remarks}.

Model Term set $\mathrm{T} = \{\mathrm{T}(\mathrm{B}_1), \mathrm{T}(\mathrm{B}_2), \mathrm{T}(\mathrm{B}_3), \mathrm{T}(\mathrm{B}_4)\}$ where

T(B1) = {total time spent reading} = {L,M,H} = B1f1 = 3 linguistic values,

T(B2) = {content completion status} = {NA, A} = B2f2 = 2 linguistic values,

T(B3) = {the Participation Index } = {L,M,H} = B3f3 = 3 linguistic values and

T(B4) = {diagnosis assessment remarks} = {U, S} = B4f4 = 2 linguistic values.

Hence, the model's term set is T(N) = {(L,M,H), (NA, A), (L,M,H), (U, S)}.

The Cartesian product of this (3 x 2 x 3 x 2) premises generated 36 rules and 36 consequents.

However, in order to provide training capability for the model such that it could adjust to the subjective decisions of various tutors on students' activities. The fuzzy student model was implemented in a connectionist adaptive network as Hybrid/Fussed Neuro-Fuzzy System. The fuzzed neural network was built on the following principles as in [12].

I. The number of cells in the input layer is equal to the number of input values which is four.
II. The number of cells in the fuzzification layer is equal to the number of fuzzy set which is ten
III. The number of cells in the premise layer, normalization layer and consequent layer is equal to the number of rules which is thirty six
IV. The output layer has a single node for the final inference

The structure of the neural network for implementing the fuzzy student model for learning activities quantitative and qualitative exploration is shown in Figure 1.

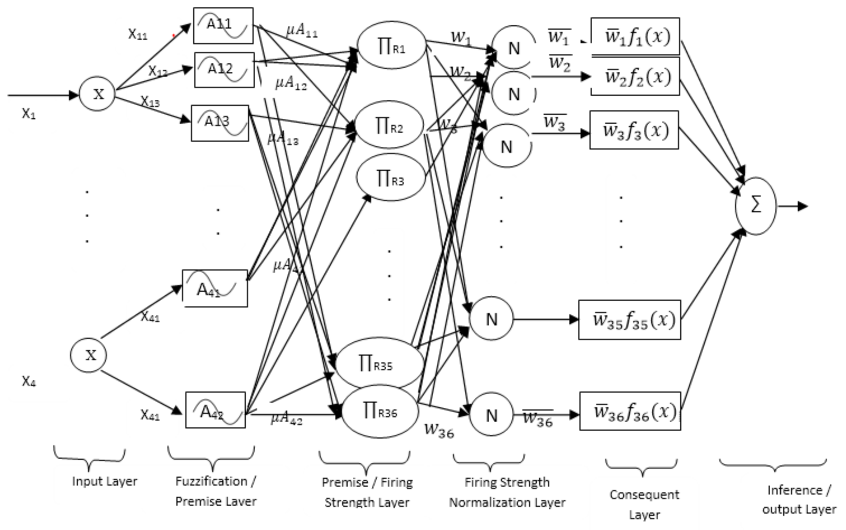


Figure 1: The Neuro-Fuzzy Model for Student Exploration and Concentration

The Neuro Fuzzy model shown is a six layer feed-forward hybrid network. Being an adaptive network, it consists of nodes and directional links. The network is adaptive because some of the nodes have adjustable parameters; such nodes are indicated with circular/oval shape in the network. Other nodes in square /rectangle shape are fixed nodes.

The Input layer (L1) provides the system with the base value for each of the features contributing towards the students evaluation. For instance the total time spent on the course may be based on seconds say 10800s for a 3 hour course. So the input layer send out the external crisp values to the next fuzzification layer without any processing. Hence, the neurons in this layer passively transmit the external input (crisp) signals to the next layer. The output of layer1 is given as:

$$O_i^1 = x_i \tag{1}$$

The total number of neurons in the input layer is four (4), one each for four linguistic variables B1-B4. Hence, for every instance (a student), four crisp values will be available as input for the purpose of determining the status of the student.

The Fuzzification layer (L2) is the first hidden layer; it has node for every categories of expert's description and classification for each of the variables. Neurons in this layer represent antecedent fuzzy sets of the fuzzy rules. This design adopts low, medium, and high fuzzy set, for B1 and B3. It uses Non Adequate (NA), Adequate (A), Unsatisfactory (US) and Satisfactory (S) for B2 and B4 respectively. For each input value entering the system, the membership degree to which such input belongs will be estimated. Common functions are Bell shape, Triangular, Gaussian and Sigmoid. For this work the Gaussian transfer function was used because of its amenability to training via adjustment of the parameters.

$$\text{The Gaussian MF} = \mu_{A_{ij}}\left(x_j; C_{ij}, \sigma_{ij}\right) = e^{\left(\frac{-\left(x_j - c_{ij}\right)^2}{2\sigma_{ij}^2}\right)} \tag{2}$$

Where x is the input, c is the centre and σ is the width, i is the input number and j is the terms number.

The output of the Fuzzification layer is the degree of membership of the input values. It can be given as:

$$O_{ij}^2 = \mu_{A_{ij}}(x_i) \tag{3}$$

The Premise layer (L3) is the second hidden layer. Each neuron in the layer corresponds to a fuzzy rule in the system. It receives signals only from relevant fuzzification neurons and calculates the activation of premises of the fuzzy rules. It uses minimum type t-norm to implement AND operators in each of the units. The t-norm operator for determining firing strength of each rule is given as:

$$O_j^3 = w_j = \prod_{j=1}^{36} \mu_{A_j}(x_i) \tag{4}$$

The fourth layer (L4) is the third hidden layer. Each neuron in this layer receives signals from all rule neurons in layer 3. It then calculate the normalized firing strength for a given rule to a close range [0 1]. It is given as:

$$O_j^4 = \overline{w}_j = O_j^3 / \sum_{j=1}^{36} O_j^3 = \prod_{j=1}^{36} \mu_{A_j}(x_i) / \sum_j^{36} \prod_{j=1}^{36} \mu_{A_j}(x_i) \tag{5}$$

The consequent layer is the defuzzification layer. Each neuron in this layer receives the initial input signals and connected to the respective normalization neuron in the preceding fourth layer. It adaptively multiplies O_j^4 with the consequent parameters as shown in (14).

$$O_j^5 = O_j^4 f_j(x) = \bar{w}_j f_j(x) = \bar{w}_j(p_j x_1 + q_j x_2 + r_j x_3 + s_j x_4 + t) \quad (6)$$

Here, the p, q, r, s, and t for each defuzzification neuron are the consequent parameters; they are to be determined by training. The $x_1, x_2, x_3,$ and x_4 are the initial inputs received.

The sixth layer is the output layer. This is a layer of single neuron that calculates the sum of the weighted consequent value from the defuzzification layer. For our model with 36 rules the overall output is given as:

$$O^6 = \sum_{j=1}^{36} \bar{w}_j f_j(x) = \sum_{j=1}^{36} \bar{w}_j \left(p_j x_1 + q_j x_2 + r_j x_3 + s_j x_4 + t\right) = y \quad (7)$$

3.2 The Network Training

The training techniques used for the network parameter identification is the Hybrid learning algorithm. It is a two pass algorithm; the forward pass and the backward pass. This option is opted for because the use of back propagation (Steepest Descent Method) - a one pass algorithm alone as shown in Table 2 converges with high error estimates. The hybrid learning uses combines Steepest Descent (SD) with Least Squares Estimation (LSE). This learning method is possible because output of the adaptive network is linear in some of the network's parameters as shown in (6) and (7). The two passes of the Hybrid learning algorithm is known as forward propagation and backward propagation respectively. In the forward pass, on the condition that the premise (nonlinear) parameters are fixed, the input vector is propagated through the network layer by layer until the last layer in which the consequent parameters are estimated by linear Least-Square Method (xxx). The sum of squared residuals or the sum of squared errors is given as:

$$\text{The Sum of Squared Error} = \sum_{i=1}^{36} (y_i - y_i^*)^2 \quad (8)$$

The y_i is the desired output and y_i^* is the observed output from the system. The essence of squaring the errors is to prevent cancellation of error value with opposite signs, thereby gives the sum of error made by the network in the course of identifying the training dataset. The network error is a function of the weight (w) of the network. If the error is unsatisfactory, the network weight needs to be adjusted in a manner to minimize the error. This can be achieved by using Gradient descent method. The Gradient descent method is 'a going downhill in small steps' method until the bottom of the error surface is reached. As a back propagation technique, the weight update of Gradient descent is scaled by a learning rate η as show in (9).

$$w_{ji\ new} = w_{ji\ old} + \eta(y_j - y_j^*)x_i \quad (9)$$

The w_{ji} represents the synaptic weight to jth neuron in the output layer from the ith neuron in the preceding layer. The parameter η is the speed at which error correction is made. Since changing the weight value could lead to convergence or divergence from the local minima of the curve. It is essential therefore to take steps proportional to the negative of the gradient at the point of estimation.

3.3 Models Implementation

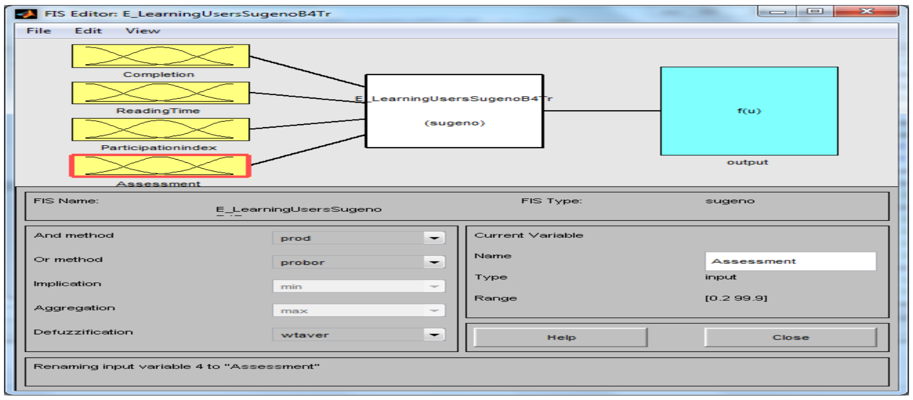
Each of the stages in the Neuro-Fuzzy Network development as discussed where simulated in the MATLAB Version 7 environment. The following are the descriptions of the implementations.

3.3.1 Sugeno Fuzzy Model Simulation

The Sugeno Fuzzy Inference System was produced by a direct transformation of the Mamdani Fuzzy in (Isiaka. Et.al, 2014). The Mamdani Fuzzy model which was developed based on expert knowledge has the same premise as the Sugeno but the consequent of the later is in the form of First Order Polynomial. The MATLAB command used for the transformation is

>> a=readfis('e_learningusers') ; >> sugenoelearningusers=mam2sug(a).

Figure 2 shows the Sugeno Model in MATLAB FIS Editor



FIS Name: E_LearningUsersSugeno
FIS Type: sugeno
And method: prod
Or method: probor
Implication: min
Aggregation: max
Defuzzification: wtaver
Current Variable
Name: Assessment
Type: input
Range: [0.2 99.9]
Renaming input variable 4 to "Assessment"

Figure 2: Formulated Sugeno FIS Model

The Editor shows the four input variables, the Sugeno Inferences, the resultant linear output and selected training parameters.

3.3.2 Neuro-Fuzzy Model Simulation

The Sugeno FIS Model was converted to its Adaptive Neuro-Fuzzy Inference System (ANFIS) equivalent with the command:

>> anfisedit e_learninguserssugeno

The generated Neuro-Fuzzy Structure for the E-learning Users Model is shown in figure 3.

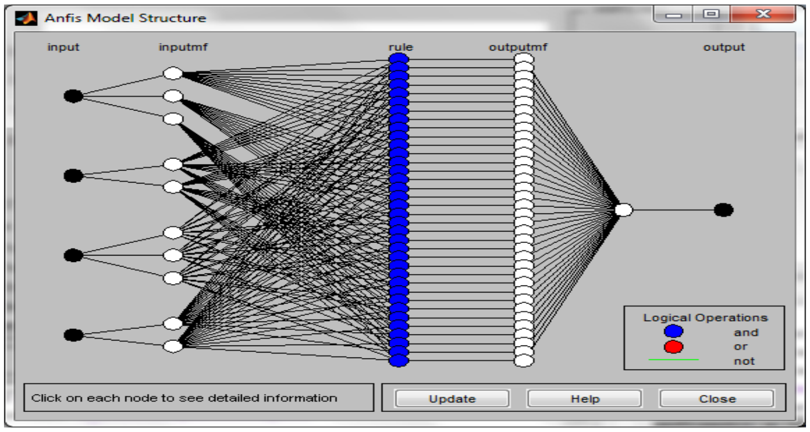


Figure 3: Generated Neuro-Fuzzy Structure E-learning Users Model

The system auto-generated fuzzy inference structure shown in figure 3 is the equivalent of the structure in figure 1. The figure shows the four neurons of the input layer, one for each of the input variables. The second layer shows the ten (10) fuzzification neurons, one for each of the fuzzy-terms. The third or premise layer has the thirty six (3 x 2 x 3 x 2) neurons, one for each of the rules. The blue colour shows that 'AND' is the aggregation operator used. The thirty six normalization neurons and the thirty six consequent neurons in Figure 1 are merged in Figure 3. Finally, the single neuron for the inference or output is shown. Details of the neurons are revealed in figure 1 but they are hidden in figure 3, they can be revealed as a screen tip when mouse is pointed at them.

3.4 Input Data Generation and Dataset Normalization

In the early stage of the model development as described in Isiaka (2014) simulated dataset where used for the model training, validation and testing. However, the actual target dataset for the study was generated in the Rain Semester of 2013-2014 academic sessions. It was the Learning activities of Sixty (60) B.Sc Computer Science students in the Department of Computer, Library and Information Science, Kwara State University, Malete, Nigeria that enrolled for System Analysis and Design (CSC 306) in the session. The lesson which spanned for the period of fourteen weeks entailed utilization of several learning activities and resources online as provided in Moodle LMS. Moodle by defaults logs (stores) users and details of all their activities into corresponding mdl_tables. The mdl_user table stores one record for each user in the system. In all it has sixty five (65) users, five (5) for the Admin and the Teachers, the remaining sixty (60) are students records. The approached used for estimation of the four variables are follows:

3.4.1 Time Variable Estimation

The estimation of the total time spent by the student on the platform throughout the duration is the most challenging of all. The mdl_log table logged every user's actions as far as possible. The mdl_log table for this research had the total of twenty thousand seven hundred and seventy four (20,774) records. The attributes considered in the table are userid, time, ip, course and action. The four patterns of online sessions (usage) identified are Login-logout session, Login-login session, Login_ip - logout_ip session and Login_in - logout_end session.

a. Login-logout session: here the user successfully logged out after series of actions. This is the ideal and expected situations. The time spent in such session is given as Tlogout – Tlogin. The sum of all such time is given as:

$$\sum_{i=1}^{p}(Tlogout - Tlogin) \tag{10}$$

b. Login-login session: here the user could not log out of the session as expected. This situation may be as result of Internet connectivity failure or care-free attitude. In that case, the time of the last valid action that precedes the second login following the first is considered as the logout time for that session. The time spent in such session is given as (Tlogin2-2) – Tlogin1. The sum of all such time is given as:

$$\sum_{j=1}^{q}((\mathrm{Tlogin2} - 2) - \mathrm{Tlogin1}) \tag{11}$$

c. Login_ip - logout_ip session: this is the kind of situation in which a login or a logout is indicated by a change in the internet protocol (ip) addresses. Hence, the end of a previous session (which is also the beginning of a new session) is signified by the difference in the successive ip addresses. The time spent in such session is given as Tlogout_ip – Tlogin_ip. The sum of all such time is given as:

$$\sum_{k=1}^{r}((\mathrm{Tlogout_{ip}} - Tlogin) + (\mathrm{Tlogout_{ip}} - \mathrm{Tlogin_{ip}})) \quad (12)$$

d. Login_in - logout_end session: this category of session represent a situation when the last action for a student’s record set in the mdl_log is not a logout. The time spent in such session is given as Tlogout_end - Tlogin. The sum of all such time is given as:

$$\sum_{l=1}^{s}(\mathrm{Tlogout}_{\mathrm{end}} - Tlogin) \quad (13)$$

The total time spent by the student in relation to the threshold is

$$\mathrm{x_t} = \frac{\sum \mathrm{te}}{\mathrm{T}} \times 100 \quad (14)$$

where $\sum \mathrm{te}$ is the total time spent by the learner on the lesson and T is the total threshold for the time in seconds. Equation for the total time spent by the student is:

$$\sum \mathrm{te} = \sum_{i=1}^{p}(Tlogout - Tlogin) + \sum_{j=1}^{q}((\mathrm{Tlogin2} - 2) - \mathrm{Tlogin1}) + \sum_{k=1}^{r}\left((\mathrm{Tlogout_{ip}} - Tlogin) + (\mathrm{Tlogout_{ip}} - \mathrm{Tlogin_{ip}})\right) + \sum_{l=1}^{s}(Tlogout_{end} - Tlogin) \quad (15)$$

Where p,q,r and s are the upper bounds for their respective type of sessions. The i,j,k and l are the counters for the various sessions.

3.4.2 Completion Variable Estimation

The table that stores the completion state (completed or non_ completed) of all modules specified by the teacher against all registered students is the mdl_course_modules_completion table. The attributes considered in this table are *userid, coursemoduleid* and *completionstate*. The total instance of module completion status in this lesson was four hundred and sixty eight (468) out of this number, four hundred and forty nine (449) while the status of nineteen (19) of them are non_completed. Completed status had status (1) while the status (0) was assigned to non completed status.

The completion value for a student is estimated based on the equation

$$\mathrm{x_c} = \frac{\sum \mathrm{Ce}}{\mathrm{C}} \times 100 \quad (16)$$

where $\sum \mathrm{Ce}$ is the total number of modules completion records with ‘completed status’ that is available for the student and the C is the completion threshold set by the teacher.

3.4.3 Participation Index Variable Estimation

The measure for student participation in a lesson is a composite function. In this research, participation is treated as a function of assignment submission, approved project proposal, survey participation, message initiation and reply to message, and forum post.

The mdl_assign_submission is the log table for monitoring student assignment submission. The attributes considered include *userid, assignment_id and the assignment_statusfields*. The assignment_statusfields could either be submitted (1) or draft (0). The total number of submissions in this table was two hundred and fifty seven (257), the number with submitted status was two hundred and fifty six while the status of only one record reads draft.

The mdl_data_records table was used for database. It was used in this work to process the projects topics submitted by the students. The attributes used include *userid, dataid, and approval*. A submission that was approved had its status set to 1 otherwise it was set to 0. The table had fifty seven (57) submissions, fifty three (53) were approved while four (4) were not approved. The total number of students that participated in the survey as shown in the mdl_feedback_tracking table was forty five (45).

Furthermore, the mdl_message_read table was used to track student to students' interactions. The table had three hundred and thirty records (330), two hundred and eighty one (281) of them had notification status 1the remaining forty nine (49) had notification status 0. Those with notification status 1 were the system notifications message they were not included in the estimation of participation index. Only those with status 0 were included in the estimation because they were the messages exchanged between students.

Finally, students' forum posts were included in the participation index. The mdl_forum_post table logs the record of students' forum threads either as parent or as children. The total number of forum threads was thirty four (34). The students that participated in messaging and forum had the points included in their grading. If u,v,w,x,y, is used respectively for tables mdl_assign_submission, mdl_data_records,mdl_feedback_tracking, mdl_message_read and mdl_forum_post. The total participation index for a student x can be estimated using

$$\mathrm{x_p} = \frac{\sum \mathrm{pe}}{\mathrm{P}} \times 100. \tag{17}$$

Where the $\sum \mathrm{pe} = \sum \mathrm{u, v, w, x, y}$ and P is the threshold set by the teacher for Participation Index.

3.4.4 Diagnosis Assessment Estimation

The students Diagnosis Assessment is extracted from the mdl_quiz_grades. The table has the quiz value for all students that participated in the quiz. The overall quiz grade can be estimated using

$$\mathrm{x_d} = \frac{\sum \mathrm{q}}{\mathrm{Q}} \times 100 \tag{18}$$

Where q is the number of correct diagnostic questions, Q is the number of diagnostic questions available.

4 Performance Evaluation Metrics

The efficiency of the model was determined using both classification uncertainty metrics and confusion matrix–based metrics. The classification uncertainty metrics considered are Mean Absolute Error (MAE) and Root Mean-Squared Error (RMSE). The confusion matrix-based metrics used are Accuracy, Precision and Recall. The 70% of the dataset were used for training the models, 15% were used for the model validation and remaining 15% were used for the testing. Details of the evaluation procedures are as follows.

4.1 Evaluation of the Sugeno Fuzzy Models Based On Aggregation and Defuzzification Techniques

The Sugeno Fuzzy Model described in 3.3.1 was evaluated using the Aggregation and Defuzzification Techniques. The two Aggregation techniques (Prob_ Probor and Max_Min) where combined with the Wtsum and Wtaver Defuzzification Techniques. The result is as shown in table 1.

Table 1: Sugeno Fuzzy Model Evaluation Based On Aggregation and Defuzzification Techniques

Aggregation Techniques	Prob_Probor		Max_Min	
Defuzzification Techniques	Wtsum	Wtaver	Wtsum	Wtaver
(MAE)	2190.5	3489.4	1342.8	13936.4
(RMSE)	46.9	59.1	36.7	118.1

Table 1 shows the performance of four types of Sugeno Fuzzy models. In all, the models with Wtsum deffuzification are the best. They have the least MAE and RMSE values when compared with those of Wtaver deffuzification. Attempt was made to convert the best of the models (MAE: 1342.8 and RMSE: 36.7) with Max-Min Aggregation and Wtsum deffuzification to Adaptive Neuro-Fuzzy Inference System (ANFIS) model. However, it was realized that ANFIS support only Sugeno with Wtaver deffuzification. To this end, the Sugeno model with the least value (MAE: 3489.4 and RMSE: 59.1) and Prob_Probor Aggregation Technique made the choice for building the ANFIS models whose performances are further shown in Table 2.

4.2 Evaluation of ANFIS Based on Training Methods

The effect of training and the choice of training methods were determined by taken two copies of the untrained ANFIS model which was the direct transformation of the Sugeno. The first copy was trained using the Back propagation method while the second copy was trained by the Hybrid (Least Square Estimation and Gredient Descent) method. The result is as shown in Table Tb.

Table 2: Effects of Training Methods on ANFIS

ANFIS (Prob_Probor/Wtaver)			
Metrics	Before Training	After Back Propagation Training	After Hybrid Training
(MAE)	2190.5	2074.8	127.4
(RMSE)	46.9	45.6	11.3

As shown, table 2 reveals the performance of three categories of the ANFIS models. The MAE and RMSE of the untrained ANFIS are 2190.5 and 46.9 respectively. This significant improvement over the Sugeno equivalent with MAE (3489.4) and RMSE (59.1) even before training of the model reveals the strength of the ANFIS architecture. The quest for optimized model however necessitated training model using the Back Propagation and the Hybrid Training Techniques. The performance of the Back propagation trained model is 2074.8 for the MAE and 45.6 for the RMSE. These error figures are very high when compared with those obtained when the Hybrid training methods were used. In the Hybrid model which is the best, the MAE was 127.4 and the RMSE was 11.3. This information established the fact that adaptive network trained by the Hybrid algorithm (Forward and Backward – two passes) will outperform the untrained and a one pass trained network of the same architecture. This deduction provides the bases for using the Hybrid training for the ANFIS model built on Expert knowledge and four other ANFIS models built on Historical data (Isiaka, 2014) and default membership functions. The details of the performances of these five (5) models follow.

4.3 Evaluation of ANFIS Models Based on Identification Techniques and Membership Functions

The knowledge discovered in the results of the experiment on the best Sugeno fuzzy model in table 1 and that of table 2 on the appropriate choice of training algorithm came into play in designing equivalent ANFIS models. The models were developed based on Historical data and varied four (4) default membership functions (MF). The membership functions considered are Triangular MF, Gaussian (MF), G-Bells MF and Sigmoid MF. The performances of these models were compared with the developed experts' knowledge based model. The Precision Accuracy of the five models shown in Table 3.

Table 3: Comparison of the Models by Precision Accuracy

Model Types: ANFIS Partitioning Methods: Grid Optimization Techniques: Hybrid			
Membership Functions	MAE	RMSE	Ranking
Developed Model	127.4	11.3	1
Triangle	163.2	12.8	4
Gaussian	160.4	12.7	3
G. Bells	168.0	13.0	5
Sigmoid	157.6	12.6	2

Table 3 shows the MAE and RMSE for the Expert based model and those based on Historical data. The G-Bells model display the worst performance, it was ranked 5th, because its performance estimates for the MAE and the RMSE are 168.0 and 13.0 respectively. The model with Triagular MF was ranked 4th, its MAE is 163.2 and its RMSE is 12.8. The 3rd in the ranking is the Gaussian MF based model. The table shows that the 2nd best model is the Sigmoid MF based model while best of the five models in consideration is the formulated model. The MAE and RMSE for the 2nd best are 157.6 and 12.6 respectively. The MAE and RMSE for the Expert based which is the best of the models are 127.4 and 11.3 respectively. Furthermore, in Table 4, the superiority of the developed model was further demonstrated.

Table 4: Comparison of Neuro-Fuzzy Models Complexity

Model Types: ANFIS Partitioning Methods: Grid Optimization Techniques: Hybrid						
Membership Functions	Training Time (sec)	Epoch	Training Errors	Training Data Errors	Validation Data Errors	Ranking
Developed Model	5	40	10.6538	10.6506	9.6042	1
Triangular	5	40	12.5577	12.5566	13.2108	3
Gaussian	4	40	12.7555	12.7529	13.3295	4
G-Bells	5	40	13.2678	13.2591	13.9856	5
Sigmoid	5	40	12.7382	12.5532	12.6077	2

Table 4 shows that the developed model has the least complexity figures, follow by Sigmoidal MF models. There is a swap of ranking positions between the Triangle MF model and the Gaussian MF model. In complexity the Triangle MF model outperformed the Gaussian. The G-Bells model retained the worth performance ranking of 5th position.

Finally, the Classification Accuracy of the models were determined as shown in table 5 and table 6

Table 5: Measures of Classification Accuracy of the Models

Positive (Proceed) = 67.0 Negative (Repeat) = 83.0 Total = 150.0					
Expected Values	**Develop Model**	**Triangle Model**	**Gaussian Model**	**G-Bells Model**	**Sigmoid Model**
TP	67.0	66.0	67.0	67.0	64.0
TN	74.0	65.0	66.0	68.0	67.0
FP	9.0	18.0	17.0	15.0	16.0
FN	0.0	1.0	0.0	0.0	3.0
Metrics					
Accuracy **(TP +TN) / (P+N)**	0.9	0.9	1.0	1.0	1.0
Precision **TP / (TP+FP)**	0.9	0.8	0.8	0.8	0.8
Recall **TP / (TP+FN)**	0.5	0.5	0.5	0.5	0.5

The analysis in table Table 5 shows that Gaussian, G-Bells and Sigmoidal membership function based model have accuracy value of 1.0 which is higher by 0.1 when compared with the 0.9 of the developed and Triangle MF models. The less accuracy value of the developed model does not discredit it supremacy, this is because accuracy has been adjudged to be a non sensitive metric especially when handling unbalanced dataset (). Secondly and most importantly is the precision value in which the developed model has 0.9 where the value for other models is 0.8. Finally, the developed model is in per with others in the Recall capability. Table 6 provides the compacted view of the classification accuracy of the models in a Confusion Matrix.

Table 6: The Confusion Matrix of Classification Models

Confusion Matrix										
	Developed		Triangle		Gaussian		G-Bells		Sigmoid	
	True Class		True Class		True Class		True Class		True Class	
Predicted Class	P(+ve)	N(-ve)	P(+ve)	N(-ve)	P(+ve)	N(-ve)	P(+ve)	N(-ve)	P(+ve)	N(-ve)
T (True)	TP (67.0)	FP (9.0)	TP (66.0)	FP (18.0)	TP (67.0)	FP (17.0)	TP (67.0)	FP (15.0)	TP (64.0)	FP (16.0)
F (False)	FN (0.0)	TN (74.0)	FN (1.0)	TN (65.0)	FN (0.0)	TN (66.0)	FN (0.0)	TN (68.0)	FN (3.0)	TN (67.0)
Total	P=TP +FN (67.0)	N=FP +TN (83.0)	P=TP +FN (67.0)	N=FP +TN (83.0)	P=TP +FN (67.0)	N=FP +TN (83.0)	P=TP +FN (67.0)	N=FP +TN (83.0)	P=TP +FN (67.0)	N=FP +TN (83.0)

The Confusion Matrix as shown in Table 6 provides another evidence for rating the developed model higher than others. As indicated, the developed model has the best True Positive value of 67.0 which is the highest, though it shares this figure with Gaussian and G.Bell models. However, its strength over these two is vivid from its 9.0 False Positive (FP) value which is very low when compared with 17.0 and 15.0 for Gaussian and G.Bell models respectively. Furthermore, the developed model did not have a single False Negative and its True Negative (74.0) is the best. The implications of this figures among others is that the system shall not in any way disadvantage any student.

5 Conclusion

This work demonstrates an approach to overcome the challenges of system identification by Trial and Error method. It recommends that a FIS of a problem should be created using whatever level of expert knowledge that may be available. Such model should then be transformed to ANFIS via its Sugeno equivalent. The transformed model can trained for better performance if historical data is available. The hybrid training technique is recommended since it has proven to be the most efficient. Moving forward, this approach shall be tested in different domain and on other Neuro Fuzzy Model.

REFERENCES

[1] Ahmed, A., Jun, S., Rami, A., and Jun, Y., *Modeling and simulation of an adaptive neuro-fuzzy inference system (ANFIS) for mobile learning*. IEEE Transactions and Learning Technologies, 2012. 5(3): p. 226 – 237.

[2] Babuska, R., and Verbruggen, H.B., *Fuzzy set methods for local modeling and identification*. In Multi model Approaches to Modeling and Control, Yaylor & Francis. 1997. P. 75 – 100.

[3] Bodyanskiy Y., and Dolotov A., *Methods and Instruments of Artificial Intelligence*. Rzeszow-Sofia, Bulgaria: ITHE, 2010. p. 17–24.

[4] Ebru, A., and Parvinder, S.S., *A soft computing approach for modeling of severity of faults in software systems.* International Journal of Physical Sciences, 2010. 5(2): p. 74-85.

[5] Hoogendoorn, R.G., Van Arem, B., and Hoogendoorn, S.P., *A neeurofuzzy apprach to modeling longitudinal driving behaviour and driving task complexity*. International Journal of Vehicular Technology. 2013. P. 1-12.

[6] Isiaka, R.M., Omidiora, E.O, Olabiyisi, S.O, and Okediran, O.O*., Mamdani fuzzy model for learning activities evaluation.* International Journal of Applied Information Systems (IJAIS). Foundation of Computer Science FCS. 2014. New York, USA. 7(3): p. 1-8

[7] Jang, J.R., Sun , C.T., and Mizutani, E., *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*. 1997. USA: Prentice-Hall,Inc.

[8] Jones, M.T., *Artificial intelligence: A systems approach*. Infinity Science Press LLC, 2008. New Delhi

[9] Lan, T.H., Lo, E.W., Wu, M.S., Hu, T.M., Chou, P., Lan, T.Y., and Chiu, H.J., *Performance of a neuro-fuzzy model in predicting weight changes of chronic schizophrenic patients exposed to antipsychotics*.Molecular Psychiatry, 2008. 13: p. 1129-1137.

[10] Nikam, S.R., Nikumbh P.J., and Kulkarni, S.P*., Fuzzy logic and neuro-Fuzzy modelling*. Recent Trends in Computing. International Journal of Computer Applications (IJCA), 2012. 22: p. 22-31.

[11] Omidiora, E.O, Olabiyisi, S.O, Okediran, O.O., and Isiaka, R.M., *Learner activities evaluation model: A neuro-fuzzy approach*. International Journal of e-Education, e-Business, e-Management and e-Learning. 2013. 3(5): p. 421-424.

[12] Sevarac, Z., *Neuro fuzzy reasoner for student modeling*. Advanced Learning Technologies, 2006. IEEE Sixth International Conference. P. 740-744.

[13] Sun, Z., and Finnie, G*., Experience management in knowledge management*. Information Technology Papers. 2005. Paper 103. Retrieved on 23/04/2010 from http://epublications.bond.edu.au/ infotech_pubs/103.

[14] Yuanyuam, C., Limin, J., and Zundong, Z., *Mamdani model based adaptive neuro fuzzy inference system and its application.* World Academy of Science, Engineering and Technology. 2009. 3: p. 743- 750.

[15] Zadeh, L.A., *Outline of a new approach to the analysis of complex systems and decision processes*. IEEE Trans. on systems, Man and Cybernetics, 1999. 3 (1): p. 28-44.

SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM

TMLAI TRANSACTIONS ON
MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE

ISSN 2054-7390

Using Machine Learning Algorithms for Cloud Client Prediction Models in a Web VM Resource Provisioning Environment

Samuel A. Ajila and Akindele A. Bankole
Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa K1S 5B6, ON Canada,
ajila@sce.carleton.ca

ABSTRACT

In order to meet Service Level Agreement (SLA) requirements, efficient scaling of Virtual Machine (VM) resources in cloud computing needs to be provisioned ahead due to the instantiation time required by the VM. One way to do this is by predicting future resource demands. The existing research on VM resource provisioning are either reactive in their approach or use only non-business level metrics. In this research, a Cloud client prediction model for TPC-W benchmark web application is developed and evaluated using three machine learning techniques: Support Vector Regression (SVR), Neural Networks (NN) and Linear Regression (LR). Business level metrics for Response Time and Throughput are included in the prediction model with the aim of providing cloud clients with a more robust scaling decision choice. Results and analysis from the experiments carried out on Amazon Elastic Compute Cloud (EC2) show that Support Vector Regression provides the best prediction model for random-like workload traffic pattern.

Keywords— Cloud Computing, Resource Provisioning, Prediction, Machine Learning, Support Vector Machine, Neural Network, Linear Regression

1 Introduction

The three main markets associated with cloud computing include Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS) and Software-as-a-service (SaaS) [35]. Popular providers of these services are Amazon Elastic Compute Cloud (Amazon EC2), Google App engine and Salesforce respectively. These services (IaaS, PaaS and SaaS) can be made accessible to the public, otherwise called public cloud or restricted for private use (private cloud). Sometimes, these services can be hosted on a hybrid cloud which is a composition of both public and private clouds. One area that researchers have focused on is resource management. Quiroz et al [37] described four stages of data center resource management: Virtual Machine (VM) Provisioning, Resource Provisioning (includes mapping and scheduling requests onto distributed physical resources), Run-time Management and Workload Modeling. In this work, focus will be on VM Provisioning. In trying to meet up with both client Service Level Agreement (SLA) for Quality of Service (QoS) and their own operating cost, cloud providers are faced with the challenges of under-provisioning and over-provisioning Under-provisioning often leads to SLA penalty resulting in revenue loss on the part of the cloud providers [7], [19], [20] and also a poor Quality of Experience (QoE) for the cloud clients. On the other hand, over-provisioning can lead to excessive energy consumption, culminating in high operating cost and waste of resources [7], [19], [20]; though this has no negative impact on the client.

DOI: 10.14738/tmlai.41.1690
Publication Date: 27th February, 2016
URL: http://dx.doi.org/10.14738/tmlai.41.1690

Accurate VM provisioning is a challenging research area that seeks to address the two extremes especially where user workload requirements cannot be determined a priori. Furthermore, VM boot up time has been reported to span various time durations before it is ready to operate [3], [28], [32], [35], [37], [39]; specifically from between 5 and 10 minutes [3], [32], and between 5 and 15 minutes [39]. It is believed that during this time of system and resource unavailability, requests cannot be serviced which can lead to penalty on the part of the cloud providers. Multiplying this lag time over several server instantiations in a data center can result in heavy cumulative penalties. These penalties or compensations to the client cannot redeem the poor QoE the customers must have perceived. To this end, cloud clients can take a proactive step to mitigate reputational loss by controlling their VM provisioning using the Cloud provider's API. One of the numerous strategies available to the client is a predictive approach wherein insight into the future resource usage (CPU, memory, network and disk I/O utilization) may help in scaling decisions ahead of time, thus, compensating for the start-up lag time [13]. Present monitoring metrics made available to clients are limited to CPU utilization, memory and network. These may not give a holistic view of the QoS. For instance, a web server may not necessarily be saturated for an SLA breach to occur. Therefore, CPU based scaling decisions may not achieve the goal of accurate VM provisioning. Several predictive resource usage approaches exist, such as pattern matching and machine learning. In fact, the use of machine learning as a predictive tool to allow dynamic scaling is one way of mitigating the challenge of resource scaling [6]. In this work, we evaluated the ability in forecasting future resource usage in a multi-tier web application of the following machine learning techniques: Neural Network (NN), Linear Regression (LR) and Support Vector Regression (SVR). In addition, the cloud watch metrics is extended by including two business level metrics: Throughput and Response time.

We used Amazon EC2, a public cloud for resource provisioning. The selection of this cloud provider is based on the availability of documentation, open source Application Programming Interface (API) and a vast array of instance types to select from (representing either on-demand, reserved or spot instances). Finally, being an early entrant in providing IaaS, Amazon EC2 boasts of a very good technical support team.

The rest of this paper is organized as follows. Section II discusses the background, state of the art, and the related work, while section III presents the methodology employed in this research work. Section IV discusses the experimental setup by emphasizing feature selection, data collection, feature reduction, CPU utilization and training, parameter selection for response time and throughput. Section V presents the simulation and analyses the results for each model (LR, NN, and SVR). A comparison of the models is carried out and a sensitivity analysis using Little’s law is done. Section VI gives a conclusion and suggestion for future work.

2 Background, State Of The Art and Related Works

2.1 Background Works

In our exploratory work [8], we developed and evaluated cloud client prediction models for TPC-W benchmark web application based on Neural Network (NN), Linear Regression (LR) and Support Vector Machine (SVM), and using a linear traffic workload (TPC-W) pattern and 170 minutes of experiment time in terms of data collection. This initial exploratory work sets the stage for a second phase based on a more random and non-linear traffic pattern [1] implemented on a public cloud environment: Amazon Elastic Compute Cloud (Amazon EC2) infrastructure. In this phase, we maintained the same architecture as the first phase and increased the experiment time to 532 minutes (compared to 170 minutes for the first phase). We noticed that during the second phase of the research work the feature selection, parameter setting for LR, NN, and SVR, and data training play crucial roles in the simulation results. Due to the big variance between the actual and predicted resource provisioning (cf. Appendix C and especially in the case of LR and NN), we decided to take a further study of the feature selection, parameter setting, and training to make sure the final result is as close to reality as possible. So, the content of this paper is a more complete research that includes:

- We maintained the same architecture as phases 1 and 2.
- We retained the inclusion of two SLA metrics: response time and throughput.
- We studied the effects of the various TPC-W workloads used in the experiment on the system performance of the database tier.
- We carried out simulations for the selection of parameters and features for the purposes of data training and learning.
- We maintained the same evaluation of the prediction capability of Support Vector Machine, Neural Network and Linear Regression using three benchmark workloads from TPC-W in:
    - An extended experimentation time frame of 532 minutes as opposed to 170 minutes in the first phase.
    - Traffic patterns wherein workloads spike up and down and then stabilize in a randomized manner; a pattern we reckon to be somewhat realistic.

Amazon EC2 offers three different instance purchasing options: On-Demand Instances, Reserved Instances and Spot Instances.

2.2 State of the Art

2.2.1 Amazon Elastic Compute Cloud (Amazon EC2)

Amazon EC2 is a web service that provides resizable compute capacity in the cloud. Amazon EC2 benefits include: elasticity, multiple instance types, operating systems and software packages, and a commitment to 99.95% availability for each EC2 Region. Finally and importantly, EC2 offer a very low per hour pay rate for the compute capacity consumed. Amazon EC2 has a range of instance types [3]. TABLE 1 summarizes some instance types and their specifications.

Table 1 Amazon Instance Type Specifications

Instance Type	Platform	CPU	Memory (GB)	Disk (GB)	Cost/Hr ($)
M1.Small	32 or 64-bit	1 ECU	1.7	160	0.060
M1.Medium	32 or 64-bit	2 ECU	3.7	410	0.120
M1.Large	64-bit	4 ECU	7.5	850	0.240
M1.Extra Large	64-bit	8 ECU	15	1690	0.480
T1.micro	32 or 64-bit	Up to 2 ECU	0.613	8	0.020
High Memory Extra Large	64-bit	6.5 ECU	17.1	420	0.410
High-CPU Medium	32 or 64-bit	5 ECU	1.7	350	0.145

2.3 Machine Learning

According to Wang and Summers [46], machine learning is the study of algorithms that run on computer systems which can learn complex relationships or patterns from empirical data and make accurate decisions. It is classified machine learning into supervised learning, semi-supervised learning and unsupervised learning. Supervised learning is to deduce a functional relationship from training data that generalizes well to testing data. Unsupervised learning on the other hand seeks to discover relationships between samples or reveal the latent variables behind the observations. Semi-supervised learning falls between supervised and unsupervised by utilizing both labeled and unlabeled data during the training phase [46]. Supervised learning has been employed because its purpose matches the problem area of this work.

2.4 Related Works

Several authors have worked in the area of resource provisioning using different approaches. This section presents some of the related techniques (Threshold-based, Control Theory, Reinforcement Learning, and Time Series).

Han et al [24] proposed and implemented a lightweight approach to enable cost-effective elasticity for cloud applications. Their solution which was centered on the cloud provider's side employed two scaling techniques to support QoS requirements of the application owner: Self-healing and Resource-level scaling. For self-healing, idle resources of one VM can be used to release the overloaded resources in another while resource-level scaling is based on using unallocated resources at a particular physical machine to scale up a VM executing on it. Though their scaling technique (scale up or down) can be completed very fast; in a matter of milliseconds, the reactive scaling mechanism employed would definitely lead to SLA penalty when a new VM provisioning is required. Furthermore, some resource providers may choose not to export the access to hypervisor-level actuators of the cloud computing infrastructure, such as controlling the CPU and memory allocations [34]. This constraint makes their work restrictive and not easily generalized. The work by Hasan, M.Z. et al [25] provided cloud clients (tenants) the ability to set policies which indicated conditions under which resources should be auto-scaled. Their Integrated and Autonomic Cloud Resource Scaler (IACRS) integrates performance metrics from other multiple domains (compute, network and storage) in making scaling decisions. The proposed algorithm is a departure from scaling decisions using the regular singular metric (CPU), the algorithm has not been implemented and thus provides no performance evaluation of any kind. In addition, their approach was also reactive and VM boot up or lag time would result in SLA penalty.

According to Lorido-Botran, T. et al [35], control theory has been applied to automate the management of web server systems and data centers, and it shows interesting results in cloud computing. Control systems are either closed loop or open loop systems. For the open loop system, control action does not depend on the system output (non-feedback) while in the case of closed loop; the controller output $\mu(t)$ tries to force the system output $C(t)$ to be equal to the reference input $R(t)$ at any time t irrespective of the disturbance ΔD [5]. Ghanbari, H. et al [26] used control theory to find a proper reservation action (immediate, in-advance, best effort or auction based reservation) at any given time based on the current system state. This approach tried to minimize the average response time and at the same time minimize resource cost by selecting the most appropriate reservation action. Though the authors agreed that best effort or on-demand reservation may not be provisioned in a timely manner, no discussion on how to handle this possible reservation action was mentioned. Lim, H.C. et al [34] proposed that cloud customers should be empowered to operate their own dynamic controllers, outside or as extensions to the cloud platform itself. The solution centered on adapting the control policy for cases where fine grained actuators for adjusting CPU entitlements are not made available by cloud providers. They introduced proportional thresholding policy which modifies an integral control by using a dynamic target range (CPU utilization for example), instead of a single target value.

Reinforcement learning (RL) is another type of automatic decision-making approach that can be applied to VM provisioning [35]. It is well suited to cloud computing as it does not require a priori knowledge of the application performance model, but rather learns it as the application runs [18]. Dutreilh, X. et al [18] used the Q-learning algorithm for their work as the Q-function is easy to learn from experience. The approach is; given a controlled system, the learning agent repeatedly observes the current state (workload, number of VMs and performance SLA), takes an action and then a transition to a new state occurs. The new state and corresponding reward is then observed. However, because defining the policy from which decisions can be chosen can take a long time (exploration and exploitation [38]) the authors introduced a convergence speedup phase at regular intervals to hasten the learning process.

Time series analysis could be used to find repeating patterns in the input workload or try to forecast future values. For example, a certain performance metric, such as average CPU load (utilization) will be periodically sampled at fixed intervals. The result will be a time-series X containing a sequence of the last w observations [35]: $X = x_t, x_{t-1}, x_{t-2}, \dots, x_{t-w+1}$ Several authors have used time series analysis for dynamic VM provisioning; for example, the work presented by Sadeka, I. et al [39] also concerns the use of time-series analysis for adaptive resource provisioning in the cloud. Their proposed prediction framework used statistical models that are able to speculate the future surge in resource requirement; thereby enabling proactive scaling to handle temporal bursty workload. The authors evaluated the prediction capabilities of two machine learning algorithms: Neural Network and Linear Regression. Historical data was first collected by using the TPC-W benchmarking e-commerce application hosted on Amazon cloud. The sampled CPU utilization dataset was then used to train both learning algorithms after which a forecast of the future CPU utilization on a 12 minute interval (the average boot up time for a new VM instance) was carried out. The same training procedure was employed with the sliding window technique which works by anchoring the left point of a potential segment at the first data point of a time series, then attempting to approximate the data to the right with increasing longer segments [29]. Performance evaluation of the two learning algorithms showed that Neural Network demonstrated enhanced accurate prediction capability compared to Linear Regression.

Our work aims at analyzing the problem of resource provisioning from the Cloud client's perspective with the ability of the hosted application to make scaling decisions by not only evaluating the future resource utilization but also considering business SLA metrics of response time and throughput; thus providing a tripartite auto-scaling decision matrix. The prediction model that we used to achieve this in a multi-tier web application is a set of machine learning techniques – Neural Network, Linear Regression and Support Vector Machine. These techniques would be evaluated using Mean Absolute Percentage Error (MAPE), Mean Absolute Error (MAE) Root Mean Square Error (RMSE) and PRED (25).

3 Methodology and Tools

In this section, we describe our methodology for predictive resource provisioning for multi-tier web applications using machine learning to develop the performance prediction model (Fig. 1). NN and LR have been widely explored by several authors in building prediction models [13], [11], [19], and [9]. Recently SVM, a powerful classification technique [11] has been gaining significant popularity in time series and regression prediction [8], [10], and [14]. We introduce these learning techniques below.

3.1 Linear Regression

A linear regression model assumes that the regression function $E(Y|X)$ is linear in the input $X_1, \ldots, X_p$. It is simple and often provides an adequate and interpretable description of how the inputs affect the output [45]. It is one of the staple methods in statistics and it finds application in numeric prediction especially where both the output or target class and the attributes or features are numeric [47]. The linear regression model has the form:

$$f(X) = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \ldots \qquad (1).$$

The β_j's are the unknown parameters or coefficients, and the variables X_j are the quantitative inputs or attributes. Typically, the parameters β are estimated from a set of training data $(x_1, y_1) \ldots (x_N, y_N)$ [45], [47]. Each $(x_{i1}, x_{i2}, \ldots, x_{ip})^{\top}$ is a vector of feature measurements for the ith case. The most popular estimation method is least squares, wherein we pick the coefficients $\beta = (\beta_0, \beta_1, \ldots, \beta_p)^{\top}$ to minimize the residual sum of squares (RSS) [58]

$$RSS(\beta) = \sum_{i=1}^{N} (y_i - f(x_i))^2 = \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j))^2 \qquad (2)$$

Linear regression is an excellent, simple scheme for numeric prediction. However, linear models suffer from the disadvantage of non-linearity: if the data exhibits a non-linear dependency, the best fitting straight line will be found [47].

3.2 Neural Network

A neural network (NN) is a two-stage regression or classification model, typically represented by a network diagram [58]. Several variants of neural network classifier (algorithm) exist, some of which are; feed-forward, back-propagation, time delay and error correction neural network classifier. According to Trevor, H. et al [45], there is typically one output unit Y_1 at the top i.e. $K = 1$ for regression problems though multiple quantitative responses can be handled in a seamless fashion. Derived features Z_m are created from linear combinations of the input, and then the target Y_k is modeled as a function of linear combinations of the Z_m,

$$\begin{aligned} Z_m &= \sigma(\alpha_{0m} + \alpha_m^T X), m = 1, \ldots, M, \\ T_k &= \beta_{0k} + \beta_k^T Z, k = 1, \ldots, K, \\ f_k(X) &= g_k(T), k = 1, \ldots, K, \end{aligned} \qquad (3)$$

Where $Z = (Z_1, Z_2, \ldots, Z_M)$, and $T = (T_1, T_2, \ldots, T_K)$. The activation function $\sigma(v)$ is usually chosen to be the sigmoid $\sigma(v) = \frac{1}{(1+e^{v})}$. Sometimes, Gaussian radial basis functions are used for $\sigma(v)$, producing what is known as a radial basis function network [45]. The units in the middle of the network, computing the derived features Z_m, are called hidden units because the values Z_m are not directly observed. The neural network model has unknown parameters, often called weights, and we seek values for them that make the model fit the training data well. The complete set of weights θ, consists of

$$\{\alpha_{0m}, \alpha_m; m = 1,2,\dots,M\} \quad M(p+1) \text{ weights and } \{\beta_{0k}, \beta_k; k = 1,2,\dots,K\} \quad K(M+1) \text{ weights} \tag{4}$$

For regression, the sum-of-squares errors is used as the error function [45]

$$R(\theta) = \sum_{k=1}^{K}\sum_{i=1}^{N}(y_{ik} - f_k(x_i))^2. \tag{5}$$

The generic approach to minimizing $R(\theta)$ is by gradient decent, called back-propagation.

3.3 Support Vector Machine

According to Sakr, G.E et al [40], Support Vector Machine (SVM) is a machine learning algorithm that uses a linear hyperplane to create a classifier with a maximal margin. For cases where the data is not linearly separable, the SVM maps the data into a higher dimensional space called the feature space. It has the advantage of reducing problems of overfitting or local minima. In addition, it is based on structural risk minimization as opposed to the empirical risk minimization of neural networks [30]. SVM now finds application in regression and is termed Support Vector Regression (SVR). The goal of SVR is to find a function that has at most ε (the precision by which the function is to be approximated) deviation from the actual obtained target for all training data with as much flatness as possible [41], [42].

Given training data (x_i, y_i) $(i = 1, \dots l)$, where x is an n-dimensional input with $x \in R^n$ and the output is $y \in R$, the linear regression model can be written as [22], [42]:

$$f(x) = < w, x > + b, \; w, x \in R^n, b \in R \tag{6}$$

where $f(x)$ is the target function and $<.,.>$ denotes the dot product in R^n. To achieve the flatness, w is minimized i.e. $||w||^2 = < w, w >$. This can further be written as a convex optimization problem: minimize $\frac{1}{2}||w||^2$ subject to the constraint

$$\begin{cases} y_i - < w, x_i > - b \le \varepsilon \\ < w, x_i > + b - y_i \le \varepsilon \end{cases} \tag{7}$$

Equation (7) assumes that there is always a function f that approximates all pairs of (x_i, y_i) with ε precision. However, according to Dashevskiy and Luo [17]; this function may not be obtainable thus necessitating the introduction of slack variables γ_i, γ_i^* to handle infeasible constraints. Therefore, equation (7) leads to

Minimize $\frac{1}{2}||w||^2 + \text{C}\sum_{i=1}^{n}(\gamma_i + \gamma_i^*)$ subject to

$$\begin{cases} yi - < w, x_i > - b \le \varepsilon + \gamma_i \\ < w, x_i > + b - yi \le \varepsilon + \gamma_i^* \\ \gamma_i, \gamma_i^* \ge 0 \end{cases} \tag{8}$$

The constant $C > 0$ determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated. Equation (8) can be reformulated and solved to give the optimal Lagrange multipliers α *and* α^* with w and b given as

$$w = \sum_{i=1}^{n}(\alpha - \alpha^*)\, x_i \text{ and} \tag{9}$$

$$b = -\frac{1}{2} < w, (x_r + x_s), x_r and\ x_s \text{ are the support vectors.} \quad (10)$$

Inserting (9) and (10) into (6) yields

$$f(x) = \sum_{i=1}^{n}(\alpha - \alpha^*) < x_i, x > +b \quad (11)$$

This generic approach is usually extended for nonlinear functions. This is done by replacing x_i with $\varphi(x_i)$; a feature space that linearizes the relation between x_i and y_i [20].

Therefore, (11) can be re-written as:

$$f(x) = \sum_{i=1}^{n}(\alpha - \alpha^*)\ K < x_i, x > +b \quad (12)$$

where $K < x_i, x >$ = $< \varphi(x_i), \varphi(x)$ is the so called kernel function.

The four basic kernels used are [20]:

- Linear: $K(x_i, x) = x_i^T x$
- Polynomial: $K(x_i, x) = (\gamma x_i^T x + r)^d$, $\gamma > 0$.
- Radial basis function (RBF): $K(x_i, x) = \exp(-\gamma||x_i - x||^2)$, $\gamma > 0$.
- Sigmoid: $K(x_i, x) = \tanh(\gamma x_i^T x + r)$.

Where $\gamma, r\ and\ d$ are kernel parameters

4 Setup of the Experiment

4.1 System Architecture

The cloud client prediction model for cloud resource provisioning in a multitier web application environment has the following components in the overall architecture (Figure 1):

- Client infrastructure: This is a High-CPU Instance with 1.7 GB of memory, 5 EC2 Compute Units (2 virtual cores with 2.5 EC2 Compute Units each) and 350 GB of instance storage. The TPC-W emulator is executed on this infrastructure
- Web server infrastructure: This is a 3.75 GB of memory, 2 EC2 Compute Unit (1 virtual core with 2 EC2 Compute Unit) and 410GB instance storage. The Java implementation of the TPC-W benchmark is deployed on a Tomcat web server environment
- Database server infrastructure: This is a 7.5 GB of memory, 4 EC2 Compute Units (2 virtual cores with 2 EC2 Compute Units each) and 850GB instance storage. MYSQL is the relational database management system used.

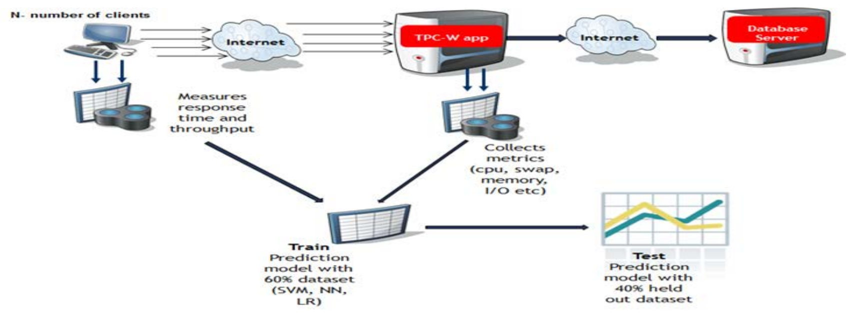

N- number of clients
Internet
TPC-W app
Internet
Database Server
Measures response time and throughput
Collects metrics (cpu, swap, memory, I/O etc)
Train Prediction model with 60% dataset (SVM, NN, LR)
Test Prediction model with 40% held out dataset

Figure 1. Architecture of the System

The Waikato Environment for Knowledge Analysis (WEKA) is used to train and test the three machine learning techniques. WEKA is a Java data mining software that has a collection of machine learning algorithms including SVR, NN and LR [22]. In this work the Explorer application (a GUI option) which is an environment for exploring data with WEKA is selected. The choice of WEKA is hinged on its open source availability and rich suite of several learning algorithms including SVR, NN and LR.

4.2 Experimental Setup

4.2.1 Feature Selection

We base our prediction models on a continuous observation of a number of specific features [40]. The following initial features are selected for the three target values (CPU utilization, response time and throughput) [2]:

- DiskReadOps: This metric identifies the rate at which an application reads a disk.
- DiskWriteOps: This metric identifies the rate at which an application writes to a hard disk.
- DiskReadBytes: This metric is used to determine the volume of the data the application reads from the hard disk of the instance.
- DiskWriteBytes: This metric is used to determine the volume of the data the application writes onto the hard disk of the instance.
- NetworkIn: This metric identifies the volume of incoming network traffic to an application on a single instance.
- NetworkOut: This metric identifies the volume of outgoing network traffic to an application on a single instance.
- Memory Utilized: This metric collects and sends the memory utilization excluding cache memory and buffers.
- Memory Available: This metric collects and sends available memory used by the operating system and the application.
- Swap Utilized: The amount of swap spaced utilized.

4.2.2 Data collection using TPC-W benchmark

TPC-W has been used by several authors [12], [48] for resource provisioning and capacity planning [39]. Similar to Sadeka et al. [39], a Java implementation of TPC-W that emulates an online bookshop is used. It is deployed on a-two-tier architecture as depicted in Fig. 1. The system resource metrics like CPU utilization and memory used are collected from the web server while the response time and throughput are measured from the “client’s” end. TPC-W has a remote browser emulator (RBE) that allows a single node to emulate several clients. The response time in this context is the time lag between when a page request is made to the reception of the last byte of the HTML response page. Similarly, the throughput is the total number of web interactions completed during an experimental run. TPC-W has 14 web interactions characteristics of which 6 belong to the Browsing category and the other 8 to the Ordering category. The three workload mixes used by TPC-W: Browsing, Shopping and Ordering are made up of a combination of Browsing and Ordering categories. For instance, Browsing mix is made up of 95% Browsing category (consists of 6 web interactions that make up the 95%) and 5% Ordering category (consists of 8 web interactions that make up the 5%). For this experiment, the N – number of clients in Fig. 1 refers to the number of users participating in any of the three workload mixes. Table 2 shows some randomly selected workload mix used in the course of the experiments. During the 1st to 7th minute, there are 84 Shopping mix users, 52 Browsing mix users and 52 Ordering mix users simultaneously making requests to the Web server (a total of 188 user requests). Each workload mix runs for 7 minutes and the choice of this time interval is intuitive as there is no documented time frame for how long workload mix should run. By adjusting the number of emulated clients in a random pattern, a changing workload that sends requests to the web server in a continuous fashion throughout the duration of the experiment is created. Amazon EC2 has a web service that enables monitoring, managing and publishing of various metrics [2]. The traditional ***Top*** command in Linux is not used as this command give metrics for the underlying host and not the actual instance [4]. Feature readings (defined in Section IV) are collected every 60 seconds by some customized Java batch scripts. For instance, to collect CPU utilization the Amazon EC2 API is: “*"mon-get-stats CPUUtilization --start-time 2013-01-08T19:17:00 --end-time 2013-01-08T19:50:00 --period 60 --statistics “+stat+” --namespace “+namesp+" --dimensions “+ dimen”*. The “*--statistics”* parameter returns the average reading over 60 seconds while “*--dimensions*” is the instance-id; the “*--namespace”* is a conceptual container for metrics [2] and for this work the EC2 namespace is used.

Table 2-Experimental Workload Mix for Some Selected Time

Time (minutes)	1-7	56-63	154-161	350-357	490-497	504-511
Shopping mix users	84	168	16	180	248	160
Browsing mix users	52	112	36	320	192	160
Ordering mix users	52	108	28	224	268	160
Total user Requests	**188**	**388**	**80**	**724**	**708**	**480**

Table 3 - Performance Metrics and their Calculations

Metric	Calculation
MAPE (Mean Absolute Percentage Error)	$\frac{1}{n}\sum_{i=1}^{n}\frac{\lvert a_i-p_i\rvert}{a_i}$ where a_i and p_i are the actual and predicted values respectively
RMSE (Root Mean Square Error)	$\sqrt{\frac{\sum_{i=1}^{n}(a_i-p_i)^2}{n}}$
MAE (Mean Absolute Error)	$\frac{1}{n}\sum_{i=1}^{n}\lvert p_i-a_i\rvert$
PRED 25	No. of observations with relative error ≤ 25% / No. of observation

The duration for the entire experiment is 532 minutes. The data is then used to build the prediction model from which forecast can be made for future resource requirement and business level metrics of the web server.

4.2.3 Feature reduction

The importance of selecting the right features for prediction modeling is very critical to reducing the potential source of error as the data mining principle of “junk in, junk out” means erroneous predictions can occur even if the prediction algorithm is optimal [40]. The Weka tool [23] is used to determine the relevance of each feature in an instance to the target class (CPU, response time and throughput). Using attribute selection functionality, the least correlated attributes are eliminated.

4.2.4 Data preprocessing

During this phase, the 6 input features (including CPU utilization, Response Time and Throughput) are scaled to values between 0 and 1. Normalization or scaling is carried out by finding the highest value within each input in the 532 dataset, and dividing all the values within the same feature by the maximum value. The main advantage for normalizing is to avoid attributes in greater numeric ranges dominating those in smaller numeric range [16].

4.2.5 Training of Dataset

As discussed earlier, the goal of this work is to build prediction model that can forecast future resource requirement (using CPU utilization) and two business level SLA – response time and throughput. Towards this end, the normalized sampled dataset is used to train the prediction model. First, training with CPU utilization as the target class is done using the three machine learning techniques discussed above. Next, using the same dataset, models for both response time and throughput are trained. The metrics in Table 3 are used to evaluate both training and testing results of the models. These metrics have been used by other authors [28], [39], and since this work seeks to compare predicted and actual target values, these metrics are good fit.

4.2.6 CPU utilization

- **Neural Network**: Using the Weka tool, the model is trained with the following parameters: learning rate ρ = 0.38, number of hidden layers = 1, number of hidden neurons = 4, momentum = 0.2 and epoch or training time =10000. These parameters gave the best results after several trials based on simulations. Parameter selection is usually based on heuristics as there is no mathematical formula or theory that has been proposed to select the best parameters
- **Linear Regression**: The Weka tool is also used to train the model. The only parameter set was the ridge parameter which was set to the default of 1.0E-8. The ridge parameter minimizes the penalized residual sum of squares. The parameter controls the amount by which the regression coefficients are shrank. The larger the ridge, the greater the shrinkage [45]. Varying the value during simulation had no significant impact on the target value.
- **Support Vector Regression**: SVR has four kernels that can be used to train a model. They are: Linear, Polynomial, Radial Basis Function (RBF) and Sigmoid [30]. The four different kernels were tried with RBF returning the most promising result with the least MAPE value. This is expected as RBF can handle the case when the relationship between features and target value is nonlinear [16]. Before training, the Grid Parameter Search for Regression with cross validation is used (v-fold cross validation) [15] to estimate the C and γ. Cross-validation is a technique used to avoid the over fitting problem [16], [30]. The search range for C was between 2^{-3} to 2^{5} and that of γ between 2^{-10} and 2^{2}. These values are purely heuristics [16], [30]. The search returns the optimal C and γ by using the Mean Square Error to evaluate the accuracy of the various C and γ combinations. The best C and λ was 14 and 0.0092. Using these parameters, the model was trained with the Radial Basis Function (RBF) Kernel.

4.2.7 Response time and Throughput

The business SLA metrics were approached in a similar way as CPU utilization (above). For Throughput, SVR's C and γ were 8 and 0.009 respectively. NN values for ρ, hidden layer, hidden neurons and momentum were 0.4, 1, 3 and 0.2 respectively. Finally the ridge parameter for LR was 1.0E-8. Also, for Response time; SVR's C and γ were 1.05 and 0.009 respectively. NN values for ρ, hidden layer, hidden neurons and momentum were 0.5, 1, 3 and 0.2 respectively. The ridge parameter for LR was 1.0E-8. Tables IV, V and VI list the final parameters used for training the SVM, NN and LR model.

Table 4 - Final Parameters of the SVR CPU Utilization and SLA Prediction Model

Metric	CPU Utilization	Response time	Throughput
C parameter search range	2^{-3} - 2^{5}	2^{-3} - 2^{5}	2^{-3} - 2^{5}
γ parameter search range	2^{-10} - 2^{2}	2^{-10} - 2^{2}	2^{-10} - 2^{2}
C	14	1.05	8
γ	0.0092	0.009	0.009

Table 5 - Final Parameters of The NN CPU Utilization and SLA Prediction Model

Metric	CPU Utilization	Response Time	Throughput
Learning Rate	0.38	0.5	0.4
Number of Hidden Layers	1	1	1
Number of Hidden Neurons	4	3	3
Momentum	0.2	0.2	0.2

This step is very significant as it is possible to obtain impressive results for training data but dismal results when it comes to testing. Furthermore, prediction accuracy is based on the held out test dataset. A training-to-testing ratio of 60%:40% (319:213 minutes) was used as this gave the optimal prediction output for the models after several simulations. A 12 minute prediction interval is adopted. This is based on reports from previous works [3], [32] regarding VM boot up time and motivation from the work of [39]. The prediction trend at the 9th, 10th, 11th and 12th minute is included in Section V to check for consistency and reliability in the prediction models of SVR, NN and LR.

Table 6-Final Parameters Of The LR CPU Utilization And SLA Prediction Model

Metric	CPU Utilization	Response Time	Throughput
Ridge Parameter	1.0E-8	1.0E-8	1.0E-8

5 Simulation Results and Analysis

This section presents the results of the various experimental simulations for determining the prediction capability of the three machine learning techniques: SVR, NN and LR. The objective is to evaluate the accuracy of the selected machine techniques in forecasting future resource usage for random workload traffic patterns over an extended period of time. In addition, the inclusion of business level metrics to the prediction is considered. Results include both training and test datasets.

5.1 Linear Regression Models

5.1.1 CPU utilization training and test results

Table 7 shows the training and testing performance metric results for the LR. Furthermore, Fig. 2 and 3 present the graphical representation of the actual and predicted CPU utilization for the 319 minutes of training and 213 minutes of testing respectively. The training MAPE value was about three times that of testing. The reason for this is that the training dataset's values are steeper than the test dataset. For instance, at the 137th minute, the CPU utilization is approximately 65 percent and this falls to about 5 percent at the 139th minute. Fig. 2 shows this graphically. Aside the test model MAPE result, the other three metric values are worse than the training model result. The reason for this is attributed to the negative predicted values and also the general poor forecasting ability of LR in a non-linear traffic pattern as captured in Figure 3.

Table 7-CPU Utilization Training and Test Performance Metric

Model	MAPE	RMSE	MAE	Pred (25)
Training	113.31	14.70	11.11	0.51
Test	36.19	22.13	15.98	0.36

Table 8-Throughput Training and Test Performance Metric

Model	MAPE	RMSE	MAE	Pred (25)
Training	75.25	4.45	3.22	0.57
Test	24.62	3.72	2.87	0.63

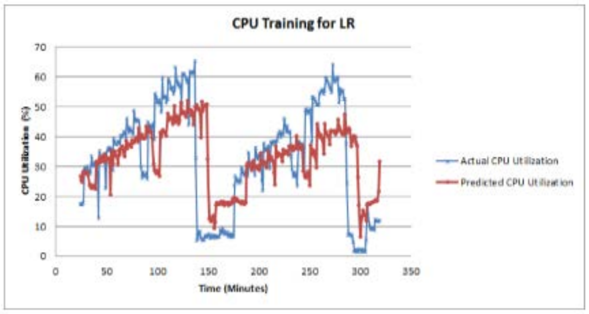


Figure 2. CPU Utilization Actual and Predicted Training Output using LR

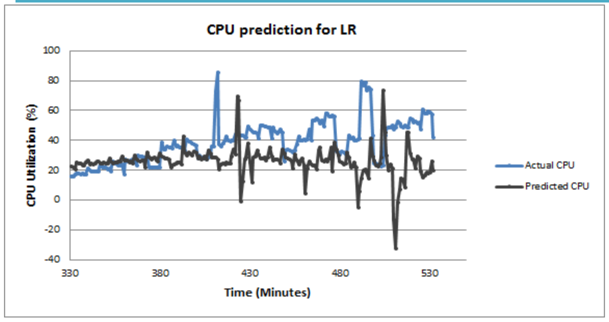


Figure 3. CPU Utilization's Actual and Predicted Test Output using LR

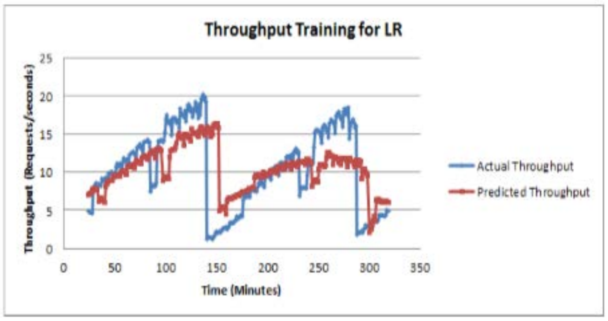


Figure 4. Throughput's Actual and Predicted training using LR

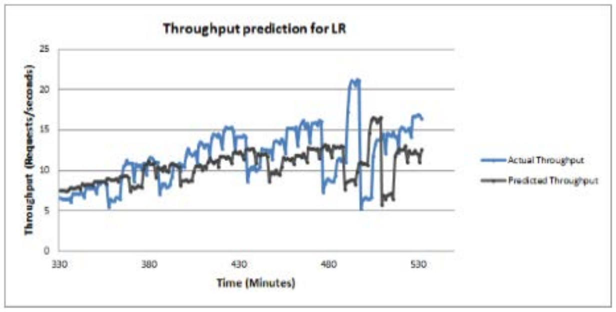


Figure. 5. Throughput Actual and Predicted test output using LR

5.1.2 Throughput training and test results

The Throughput's training and test performance metric results are shown in Table 8. In addition, Fig. 4 and 5 present the graphical representation of the actual and predicted Throughput values for both training and test dataset respectively. The training and test interval are the same as that of CPU Utilization. All test metric results are better than training results. The variances in the actual and predicted values are not as wide as that of CPU utilization. It can also be observed that there are more spikes in the training dataset than in the test dataset, thus further making the test result better than the training result. Fig. 5 shows the Throughput forecast.

5.1.3 Response time training and test results

The Response time's training and test performance metric results are shown in Table 9. Furthermore, Fig. 6 and 7 present the graphical representation of the actual and predicted Response time values for both training and testing dataset respectively. The difference in the training and test results is very close as the traffic patterns are almost similar. The accuracy for this metric is very impressive, though it can be said that the variance between the actual and predicted values are lower than that of the CPU utilization. The minimum and maximum response time values are between 1-12 seconds.

Table 9 - Response Time Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	17.58	1.24	0.81	0.90
Test	12.35	1.39	1.11	0.91

Table 10 - CPU Utilization Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	105.63	14.08	9.48	0.59
Test	50.46	31.08	19.82	0.34

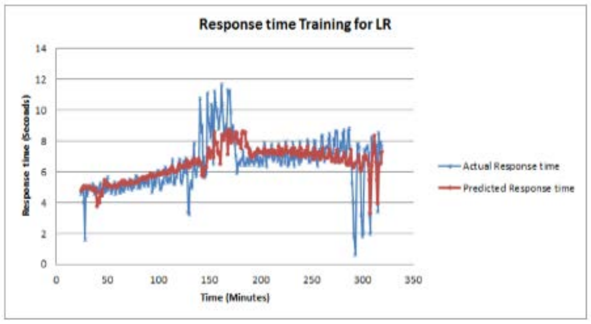


Figure 6. Response time Actual and Predicted training output using LR

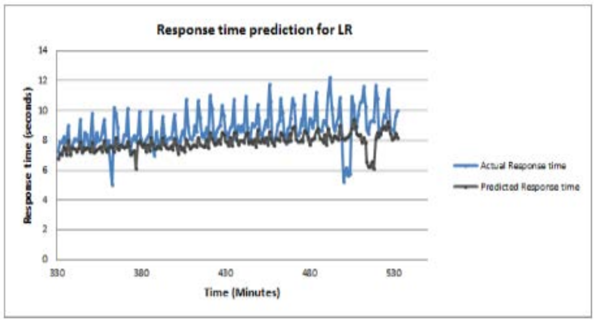


Figure 7. Response time Actual and Predicted test output using LR

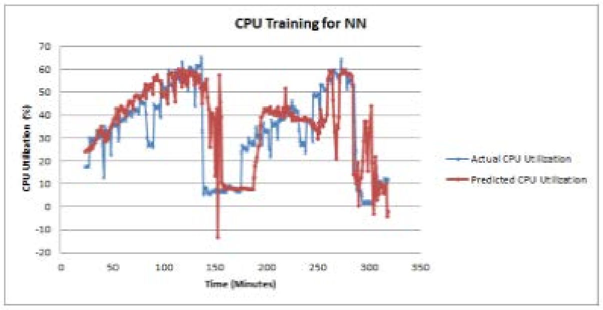


Figure 8. CPU Utilization Actual and Predicted training output using NN

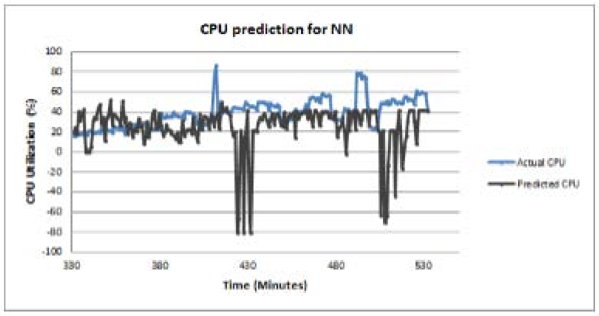


Figure 9. CPU Utilization Actual and Predicted test output using NN

5.2 Neural Network Models

5.2.1 CPU Utilization training and test results

The CPU utilization training and test performance metric results for NN model is shown in Table 10. Fig. 8 and 9 present the graphical representation of the actual and predicted CPU utilization for the 319 minutes of training and 213 minutes of testing respectively. The training MAPE value is also very high and the reason for this is similar to the explanation given in sub-section on LR. It is also observed that the test dataset has some series of negative values in its prediction as shown in Fig. 9. The number of negative predicted values which is more than that of LR contributed to the poorer test metric values.

5.2.2 Throughput training and test results

The Throughput's training and test performance metric results are shown in Table 11. In addition, Fig. 10 and 11 present the graphical representation of the actual and predicted Throughput values for both training and test dataset respectively. Comparing the training and test model results, the test model's metric values are quite better than the training model. Some predicted throughput values in the training model are negative (Fig. 9). However, negative prediction is absent for the test dataset (Fig. 10).

Table 11 - Throughput Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	56.46	6.85	4.96	0.30
Test	38.90	6.12	4.46	0.47

Table 12 - Response Time Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	36.28	3.51	2.38	0.58
Test	17.84	2.02	1.64	0.75

5.2.3 Response time training and test results

The Response time's training and test performance metric results are shown in Table 12. Furthermore, Fig. 12 and 13 present the graphical representation of the actual and predicted Response time values for both training and test dataset respectively. The test model's metric values are better than that of the training model as Fig. 12 shows more erroneous prediction than Fig. 13. That is, the high training metric values (MAPE, RMSE and MAE) are attributed to the large variations in some predicted and actual Response time as shown in Figure 12.

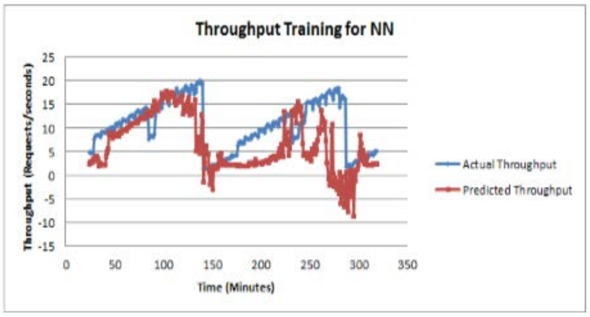


Figure 10 Throughput Actual and Predicted training for NN

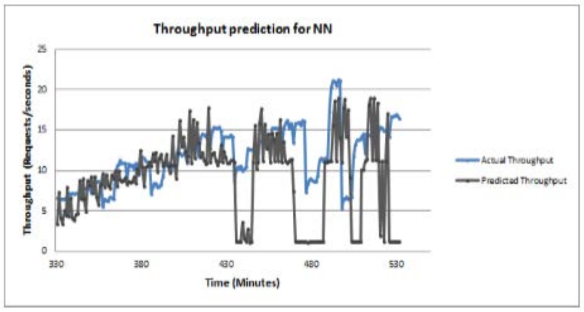


Figure 11 Throughput Actual and Predicted test output for NN

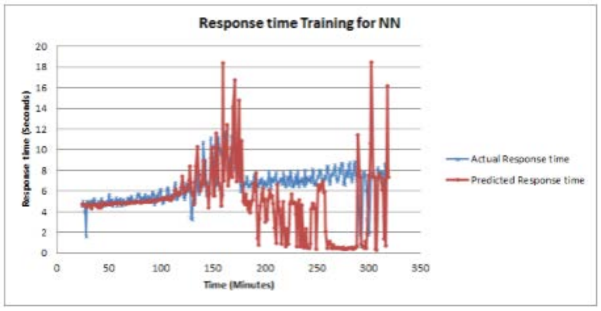


Figure 12 Response time Actual and Predicted training for NN

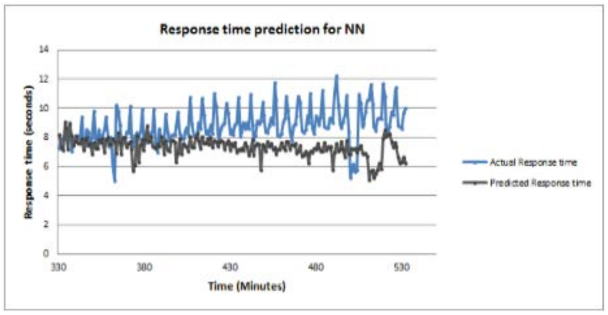


Figure 13 Response time Actual and Predicted test for NN

5.3 Support Vector Machine (Regression) Models

Similar to the two previous sub-sections, CPU utilization, Throughput and Response time training and testing dataset results are presented.

5.3.1 CPU Utilization training and test results

The CPU utilization training and test performance metric results for SVR model is shown in Table 13. Fig. 14 and 15 present the graphical representation of the actual and predicted CPU utilization for the 319 minutes of training and 213 minutes of testing respectively. As discussed in previous sub-sections (LR and

NN), the training model also had a very high MAPE value (107.8). A significant improvement is however observed in the test dataset metric. Fig. 15 shows that all predicted values are positive and very close to the actual values. However, some sudden spikes result into substantial variation in values.

Table 13 - CPU Utilization Training And Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	107.80	15.48	10.09	0.64
Testing	22.84	11.84	8.74	0.64

Table 14 - Throughput Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	78.78	4.74	2.80	0.70
Testing	22.07	3.22	2.41	0.67

5.3.2 Throughput training and test results

The Throughput's training and test performance metric results are shown in Table 14. Fig. 16 and 17 present the graphical representation of the actual and predicted Throughput values for both training and test dataset respectively. The significant difference in the training and test MAPE value is also attributed to the spikes as shown in Fig. 16. For instance, at the 256th minute, the actual Throughput value is approximately 16 requests/second while at the next minute; it drops to approximately 1.8 requests/second. The predicted value at this point is about16 requests/second. The large variation lasted for about 12 minutes before the gap was closed.

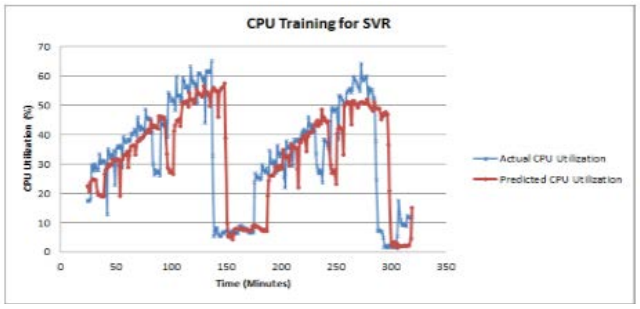


Figure 14. CPU Utilization Actual and Predicted training for SVR

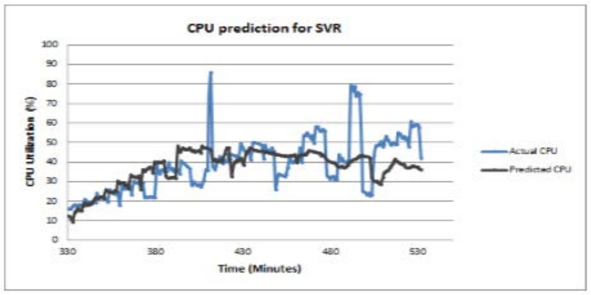


Figure 15. CPU Utilization Actual and Predicted test for SVR

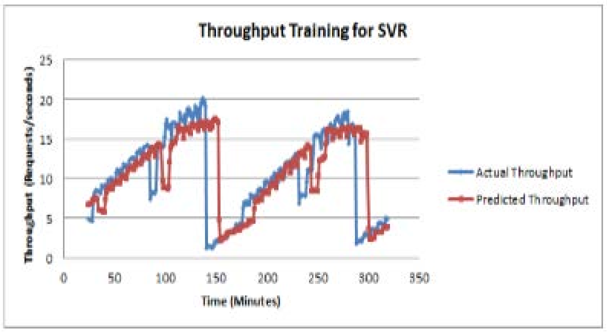


Figure. 16 Throughput Actual and Predicted Training for SVR

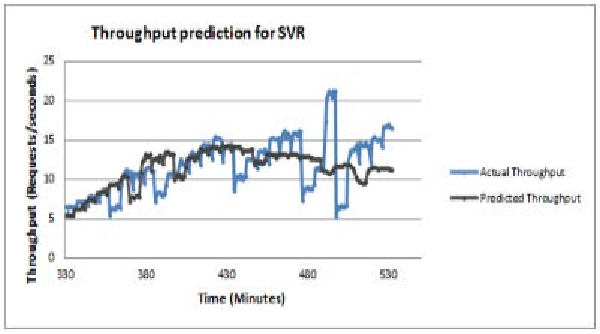


Figure. 17. Throughput Actual and Predicted Test for SVR

5.3.3 Response time training and test results

The Response time's training and test performance metric results are shown in Table 15. Additionally, Fig. 18 and 19 present the graphical representation of the actual and predicted Response time values for both training and test dataset respectively. The training and test models present similar metric values. The graph in Fig. 18 shows some variation in the predicted and actual Response time values especially between the 140th and 160th minute and also towards the end of the training dataset. In the case of Fig. 19, test dataset, the predicted values follow the trend of the actual values. Less traffic surge is also observed in comparison to Fig. 18. Again, the range of the test dataset is between 5.5 and 12 seconds unlike the wider range for the training dataset (1 to 12 seconds). Therefore, the test metric output result is much better than that of the training.

5.4 Comparison of Prediction Models

The overall CPU utilization values range from 1.73% to 85.96%. The training dataset presented in Tables 7, 10 and 13 show that the MAPE values are above 100 percent with LR having the highest of 113.31. This abnormally high performance metric value is attributed to the fact that the traffic pattern of the workload for the experiment is random. For instance, at the 140th minute, there is a drop from 65% to about 5% CPU utilization. This drop lasted for about 40 minutes after which it surged again. The training behavior of the three models (SVR, NN and LR) for this scenario is shown in Fig. 20. It can be observed that NN shows a zigzag prediction pattern between the 132nd to about the 155th minute after which it gave a near perfect prediction of the CPU utilization. SVR and LR present a better and stable CPU utilization prediction than NN during this same interval. Isolating this randomness would significantly reduce the MAPE values; however, one of the goals of this work is to study how these learning techniques would perform in an almost realistic workload scenario. The PRED (25) metric for SVR reported the highest value of 0.64 or 64%. More importantly, the forecasting (prediction) ability of these techniques gives a more interesting trend.

Table 15 - Response Time Training and Test Performance Metric

Model	MAPE	RMSE	MAE	PRED(25)
Training	19.24	1.43	0.88	0.84
Test	9.92	1.21	0.87	0.93

Table 16 - CPU Utilization Step Prediction For MAPE

Model	9-min	10-min	11-min	12-min
SVR	22.31	22.69	22.78	22.84
NN	53.07	49.90	45.62	50.46
LR	34.43	35.14	35.92	36.19

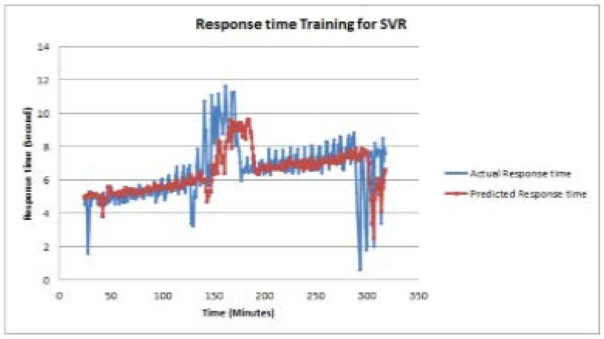


Figure 18. Response time Actual and Predicted training for SVR

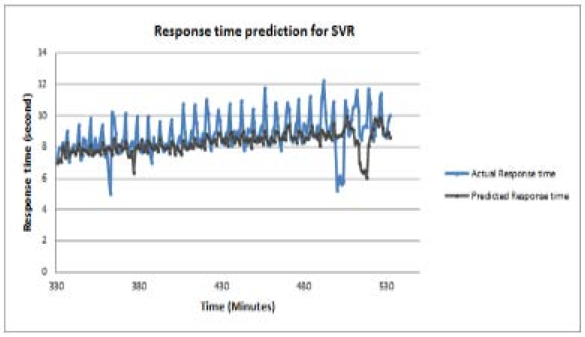


Figure 19. Response time Actual and Predicted test for SVR

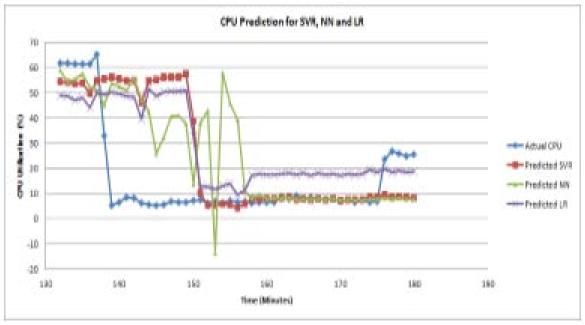


Figure 20. CPU utilization training prediction for SVR, NN and LR at selected time interval

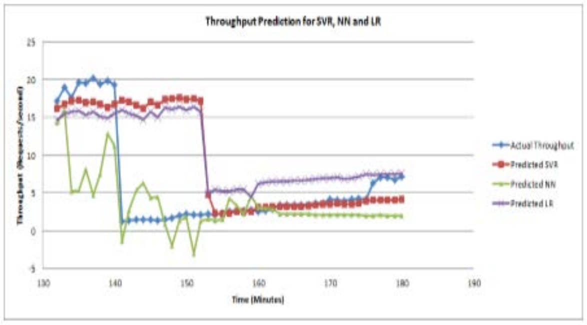


Figure 21. Throughput training prediction for SVR, NN and LR at selected time interval

SVR significantly outperforms the other two models when MAPE, RMSE and MAE performance metrics are considered. Interestingly, the test dataset was made up of short burst of high-low CPU utilization as shown in Fig. 3, 9 and 15. The generalization capability of SVR is brought to fore as it is the least susceptible to the high/low test dataset values that should result in poor forecasting output. NN and LR reports negative CPU utilization, an anomaly that exposes their weakness in random workload forecasting. The MAPE and RMSE step predictions in Tables 16 and 15 respectively show a prediction reliability of SVR and LR as opposed to NN. SVR yields the least MAPE, RMSE and MAE error. Therefore, a conclusion may be drawn in favor of SVR as the strongest and superior prediction model for CPU utilization with LR following closely.

Moving on to the business level metrics of which the Throughput model is analysed first; the throughput values had a range between 1.25 and 21 requests/second. Again, Figure 4-20 shows the selected throughput training result between the 132nd and 180th minute. The SVR and LR models could not adjust immediately to the sharp drop at the 141st minute thus accounting for the high MAPE value. SVR and LR took about 12 minutes to significantly reduce the variance between the predicted and actual throughput values though LR's prediction was not as close to the actual compared to SVR. Fig. 20 and 21 show negative prediction values for NN even though NN had the best training MAPE value of 56.46. Fig. 21 explains the reason for this as though NN had some negative predictions, the variance between predicted and actual throughput is the least. The step prediction outputs for test dataset are summarised in Tables 18 and 14. The dataset is also a mix of high and low throughput values corresponding to the random workload pattern employed in this work. SVR metrics proved to be the best by displaying a strong generalizing attribute, i.e. using the trained model to forecast unseen data (test) in a non-fitting manner. Fig. 5, 11 and 17 show the graph plots of the forecasting ability of LR, NN and SVR respectively. LR comes second behind SVR.

Table 17 - CPU Utilization Step Prediction For RMSE

Model	9-min	10-min	11-min	12-min
SVR	11.86	11.96	11.92	11.84
NN	31.56	29.69	27.64	31.08
LR	20.97	21.43	21.84	22.13

Table 18 - Throughput Step Prediction for MAPE

Model	9-min	10-min	11-min	12-min
SVR	21.38	21.62	21.80	22.07
NN	38.84	36.87	37.39	38.90
LR	25.60	25.25	25.01	24.62

Table 19 - Throughput Step Prediction For RMSE

Model	9-min	10-min	11-min	12-min
SVR	3.17	3.18	3.20	3.22
NN	5.94	5.94	5.97	6.12
LR	3.78	3.75	3.73	3.72

Finally on the business level metric wherein Response time model is analysed; the overall Response time value had a range from about 0.6 to 12 seconds. From the results obtained for the test dataset presented in Tables 9, 11 and 15, SVR performed best in comparison to LR and NN. Furthermore, the step prediction output for the test dataset shown in Tables 20 and 21 reveals the superior prediction capability of SVR. Fig. 19 shows the prediction trend for SVR. The test result shows an impressive forecasting behaviour for the test dataset for SVR. LR had a better prediction result than NN. LR and NN's test performance metric can be seen to be close to that of SVR. The reason for this is the seemingly closeness of the dataset to a linear pattern unlike the throughput and CPU utilization. Furthermore, with less variance, comes the tendency of linearity; an area LR and NN performs well. Response time model has the least range difference in comparison to CPU utilization and Throughput. In spite of this, SVR still shows superiority across board. The prediction consistency of SVR is also brought to fore in Tables 20 and 21.

Table 20 - Response Time Step Prediction for MAPE

Model	9-min	10-min	11-min	12-min
SVR	10.99	11.07	10.49	9.92
NN	15.73	16.40	17.09	17.84
LR	11.81	12.08	12.17	12.35

Table 21 - Response Time Step Prediction For RMSE

Model	9-min	10-min	11-min	12-min
SVR	1.28	1.27	1.25	1.21
NN	1.85	1.94	2.00	2.02
LR	1.37	1.39	1.39	1.39

5.5 Sensitivity analysis

In this sub-section, the validity of the experimental results is analyzed using the Little's law. Customers (user requests) arrive at the system (web server), stay for a while (receiving service) and leave. Little's law states that the average number of users in a system is equal to the departure rate of the user requests from the system multiplied by the average time each user request spends in the system. This can be summarized as [42]: $N = \lambda R$ ... (14); where N = number of users in the system, λ=throughput and R = Response time

Little's law is quite general and requires few assumptions. It applies to all stable systems that may contain an arbitrary set of components such as CPU. Using Little's law, the consistency of the measurement data obtained from the experiment can be validated. Due to the large sample space (532 data points), the data from Table 2 is used as subset of the entire dataset in checking the consistency of the measurement data. From equation (14), the Number of users in the system is the Total User Request; the departure rate is the Throughput. The average time spent in the system is calculated and compared with the response time measured experimentally. During the 1st to 7th minute interval, the average throughput measured was 5.29requests/second. The average number of users during this time interval is $\frac{188}{7} \sim 27$. Using equation(14), the average time spent is $\frac{27}{5.29} = 5.1\ seconds$. The measured average response time during this period was 4.66 seconds. The percentage variance would be:

$Variance = \frac{(Average\ time\ spent - measured\ time\ spent)}{measured\ time\ spent} x\ 100$. With this example, Table 22 is completed.

Table 22 - Data Consistency Measurement

Time (minute)	1-7	56-63	154-161	350-357	490-497	498-503	504-511
Average total user requests	27 (188/7)	55 (388/7)	11 (80/7)	103 (724/7)	101 (708/7)	95 (664/7)	69 (480/7)
Average Throughput (Requests/second)	5.29	10.52	2.47	15.92	10.00	12.23	13.27
Average time spent (seconds)	5.10	5.23	4.45	6.47	10.01	7.77	5.2
Measured time spent	4.66	4.92	9.92	7.53	9.06	9.01	9.45
Time variance (%)	9.44	6.31	55.14	14.08	10.49	13.76	44.97

It can be observed that the results at the 154^{th}-161^{st} minute and that of the 504^{th}-511^{th} minute had a high percentage variance. While the period between 154-161 minute duration has an acceptable average throughput (based on numbers of user requests), the latter (504-511) has an unusually high average throughput for the number of users during the 7 minute window. For the 504-511 time interval, the logical explanation for this anomaly could be that the web server is still processing user requests from the 498-503 window when the 504-511 user request batch started sending requests. The measured response time also shows that more requests i.e. greater than the actual average of 69 users must have been requesting for service at the web server. However, the anomaly during the 154-161 window could be attributed to experimental error.

6 Conclusion

In this work, three forecasting models are built using Linear Regression (LR), Neural Network (NN) and Support Vector Regression (SVR) for a two-tier TPC-W web application. Asides from the traditional single metric prediction using CPU utilization, the monitoring metric is extended to include response time and throughput (business SLA metrics). This three-factor combination in the prediction model provides a broader view of the QoS. The user workload traffic employed is random, an approach to simulate a realistic workload pattern. After an extensive simulation lasting about 10 hours, the three machine learning techniques are trained and validated with 60% and 40% of the historical dataset respectively. The performance of SVR, LR and NN are measured using four metrics; MAPE, RMSE, MAE and PRED (25).

Overall, Support Vector Regression (SVR) model displayed superior prediction accuracy over both Neural Network (NN) and Linear Regression (LR) in a 9-12 minute window. Specifically and in terms of the MAPE performance test metric the following key observations from the simulation results are presented.

- In the CPU utilization prediction model, SVR outperformed LR and NN by 58% and 120% respectively
- For the Throughput prediction model, SVR again outperformed LR and NN by 12% and 76% respectively; and finally,
- The Response time prediction model saw SVR outperforming LR and NN by 26% and 80% respectively.
- The clear prediction superiority of SVR shows strong generalization ability in a non-linear model (random-like workload pattern). SVR can optimally map the non-linear input data to a higher dimension feature space via the Kernel function (RBF in this case), then perform linear regression

in the higher dimensional feature space [41]. The absence of the Kernel function in LR and NN makes it difficult for them to perform well in non-linear models.

Therefore, based on these experimental results SVR may be accepted as the best prediction model. Consequently, cloud clients can employ SVR to build their prediction models. Furthermore, the addition of business level SLA metrics (response time and throughput) into the prediction model paves the way for a three-factor combination decision matrix for scaling VM resources. The inclusion of response time and throughput further broadens the view of the QoS of client applications as these business level metrics may have degraded long before an application reaches its set CPU utilization threshold.

In this study, forecasting future resource usage using machine learning techniques has shown promising results. However, some areas have been identified for further research and they are presented in this section.

- This study has focused only on the web server tier. Further work to include the database tier may be worth investigating. With this inclusion, unsaturated/saturated webserver and database combination can be modeled and subsequent forecasting made using the same machine learning techniques.
- Investigating the combination of SVR and other predicting techniques that may further increase the prediction accuracy is another future direction.
- In order to further validate the forecasting strength of machine learning techniques and specifically SVR, the use of other application workloads that are not web based is another interesting investigation that may be pursued.

REFERENCES

[1] Ajila A. S. and Bankole A. A., Cloud Client Prediction Models Using Machine Learning Techniques, COMPSAC 2013 - The 37th Annual International Computer Software & Applications Conference, Kyoto, Japan July 22-26, 2013

[2] “Amazon CloudWatch Developer Guide API Version 2010-08-01”, 2013. [Online]. Available: http://awsdocs.s3.amazonaws.com/AmazonCloudWatch/latest/acw-dg.pdf.

[3] “Amazon elastic compute cloud (amazon ec2)”, 2013. [Online]. Available: http://aws.amazon.com/ec2.

[4] “Amazon Web services Discussion Forums”, 2013. [Online]. Available: https://forums.aws.amazon.com/thread.jspa?threadID=67697

[5] Ali-Eldin, A., Tordsson, J and Elmroth E., “An adaptive hybrid elasticity controller for cloud infrastructures”. IEEE Network Operations and Management Symposium (NOMS) pp 204-212. Hawaii, USA. April, 2012.

[6] Armbrust, M. et al., “A view of cloud computing”. Commun. ACM. 53, 4 pp. 50–58, April, 2010.

[7] Armbrust, M. et al., Above the Clouds: A Berkeley View of Cloud Computing. 2009

[8] Bankole A., and Ajila S.A., “Cloud Client Prediction Models for Cloud Resource Provisioning in a Multitier Web Application Environment”, in 7th IEEE International Symposium on Service-Oriented System Engineering (IEEESOSE 2013), San Francisco Bay, USA March 25 – 28, 2013.

[9] Bertholon, B., Varrette, S., Bouvry, P., “Certicloud: A Novel TPM-based Approach to Ensure Cloud IaaS Security”. IEEE International Conference on Cloud Computing (CLOUD). pp. 121–130. 2011.

[10] Boniface, M. et al., “Platform-as-a-Service Architecture for Real-Time Quality of Service Management in Clouds”. 5th International Conference on Internet and Web Applications and Services (ICIW). pp. 155–160, Barcelona, Spain. May, 2010.

[11] Borgetto, D., Maurer, M., Da-Costa, G., Pierson, J., and Brandic, I., “Energy-Efficient and SLA-Aware Management of IaaS clouds”. 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy). pp. 1–10. Madrid, Spain. May, 2012.

[12] Cain , H. W., Rajwar, R., Marden, M., and Lipasti, M., “An Architectural Evaluation of Java TPC-W” in Proceedings of the Seventh International Symposium on High- Performance Computer Architecture, Nuevo Leone, Mexico. January, 2001.

[13] Caron, E., Desprez, F., and Muresan, A., "Forecasting for Grid and Cloud Computing On-Demand Resources Based on Pattern Matching" 2nd International Conference on Cloud Computing Technology and Science (CloudCom). pp.456-463, Indianapolis, USA. November, 2010.

[14] Chieu, T.C., Mohindra, A., Karve, A.A., and Segal A., “Dynamic Scaling of Web Applications in a Virtualized Cloud Computing Environment”. IEEE International Conference on e-Business Engineering ,ICEBE ’09. pp 281-286. Macau, China. October, 2009.

[15] Chih-Chung, C. and Chih-Jen , L., “LIBSVM : a library for support vector machines”. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

[16] Chih-Wei, H., Chang, C.C, and Lin C., “A practical guide to support vector classification”. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, 2003. [Online]. Available: http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

[17] Dashevskiy, M. and Luo, Z., "Time series prediction with performance guarantee". IET Communications. Vol. 5, Issue 8, pp. 1044–1051. 2010.

[18] Dutreilh, X., Kirgizov, S., Melekhova, O., Malenfant, J., Rivierre, N., and Truck I., “Using Reinforcement Learning for Autonomic Resource Allocation in Clouds: Towards a Fully Automated Workflow” Proceedings of the 7th International Conference on Autonomic and Autonomous Systems. pp 67-74. Mestre, Italy. May, 2011.

[19] Fang, W., Lu, Z., Wu, J and Cao Z., “RPPS: A Novel Resource Prediction and Provisioning Scheme in Cloud Data Center”. IEEE Ninth International Conference on Services Computing (SCC). pp.609 –616, Washington DC, USA. June, 2012.

[20] Gandhi, A., Chen, Y, Gmach, D., Arlitt, M., and Marwah, M., “Hybrid resource provisioning for minimizing data center SLA violations and power consumption”. Sustainable Computing: Informatics and Systems. pp. 91–104. Orlando, Florida, USA. June, 2012.

[21] Ghanbari, H., Simmons, B., Litoiu, M., Barna, C., and Iszlai, G., "Optimal autoscaling in a IaaS cloud". Proceedings of the 9th international conference on Autonomic computing. pp 173-178. San Jose, California, USA. September, 2012.

[22] Guosheng, H., Hu, L., Li, H., Li, K., and Liu, W., "Grid Resources Prediction with Support Vector Regression and Particle Swarm Optimization," 3rd International Joint Conference on Computational Science and Optimization (CSO), vol.1, pp.417-422, China. May, 2010.

[23] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I.H., "The WEKA Data Mining Software: An Update", SIGKDD Explorations, Volume 11, Issue 1. 2009.

[24] Han, R., Guo L., Ghanem, M.M., and Guo, Y., "Lightweight Resource Scaling for Cloud Applications". 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp 644 –651, Ottawa Canada, May. 2012.

[25] Hasan, M.Z., Magana, E., Clemm, A., Tucker, L., and Gudreddi, S.L.D., "Integrated and autonomic cloud resource scaling". IEEE Network Operations and Management Symposium (NOMS). pp 1327-1334. Hawaii, USA. April, 2012.

[26] Hilley, D., Cloud Computing: A Taxonomy of Platform and Infrastructure-level Offerings: 2009. https://smartech.gatech.edu/handle/1853/34402. Accessed: 2013-01-24.

[27] Holehouse A., "Stanford Machine Learning". [Online]. Available: http://www.holehouse.org/mlclass/index.html

[28] Imam, M.T., Miskhat, S.F., Rahman, R.M., and Amin, M.A., "Neural network and regression based processor load prediction for efficient scaling of Grid and Cloud resources". 14th International Conference on Computer and Information Technology (ICCIT). pp 333-338, Bangladesh, India. December, 2011.

[29] Keogh, E., Chu, S., Hart, D., and Pazzani, M., "An online algorithm for segmenting time series". Proceedings of IEEE International Conference on Data Mining. pp 289-296. San Jose, California, USA. November, 2001.

[30] Khashman, A. and Nwulu, N.I., "Intelligent prediction of crude oil price using Support Vector Machines", in IEEE 9th International Symposium on Applied Machine Intelligence and Informatics (SAMI), pp.165-169, Smolenice, Slovakia. January, 2011.

[31] Kulkarni, P., "Reinforcement and Systematic Machine Learning For Decision Making", Wiley-IEEE Press, 2012.

[32] Kupferman, J., Silverman, J., Jara, P., and Browne, J., "Scaling Into the Cloud". University of California, Santa Barbara, Tech. Rep. http://cs.ucsb.edu/~jkupferman/docs/ScalingIntoTheClouds.pdf. 2009.

[33] Lim, H.C., Babu, S., and Chase, J.S., "Automated control for elastic storage". Proceedings of the 7th international conference on Autonomic computing. pp 1-10. Washington DC, USA. June, 2010.

[34] Lim, H.C., Babu, S., and Chase, J.S., "Automated control in cloud computing: challenges and opportunities". Proceedings of the 1st workshop on Automated control for datacenters and clouds. pp 13-18. Barcelona, Spain. June, 2009.

[35] Lorido-Botran, T., Alonso-Miguel, J., and Lozano, J.A., “Auto-scaling Techniques for Elastic Applications in Cloud Environments” Department of Computer Architecture and Technology, University of Basque Country, Tech. Rep. EHU-KAT-IK-09-12. September, 2012.

[36] Muppala, S., Zhou X., and Zhang, L., “Regression-based resource provisioning for session slowdown guarantee in multi-tier Internet servers”. Journal of Parallel and Distributed Computing. pp. 362-375 March, 2012.

[37] Quiroz, A., Kim, H., Parashar, M., Gnanasambandam, N., and Sharma N., “Towards autonomic workload provisioning for enterprise Grids and clouds” in Grid Computing, 2009 10th IEEE/ACM International Conference. pp 50-57, Banff, Alberta, Canada. October, 2009.

[38] Richard S. S and Andrew B.B., “Reinforcement Learning an Introduction” http://www.scribd.com/doc/92878651/Reinforcement-Learning-an-Introduction-Richard-S-Sutton-Andrew-G-Barto. Accessed: 2013-01-02.

[39] Sadeka, I., Keung, J., Lee, K., and Liu, A., “Empirical prediction models for adaptive resource provisioning in the cloud”, Future Generation Computer Systems, vol. 28, no. 1, pp 155 – 165, January, 2012.

[40] Sakr, G.E., Elhajj, I.H., Mitri, G., Wejinya, U.C., "Artificial intelligence for forest fire prediction" IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), pp.1311-1316, Montreal, Canada. July, 2010.

[41] Sapankevych, N and Sankar, R., "Time Series Prediction Using Support Vector Machines: A Survey," Computational Intelligence Magazine, IEEE, vol.4, no.2, pp.24-38, May 2009.

[42] Smola, A.J and Scholkopf, B., “A Tutorial on Support Vector Regression” in Statistics and Computing vol 14, pp. 199 – 222, August, 2004.

[43] Tian, C., Wamg, Y., Qi, F., and Yin, B., "Decision model for provisioning virtual resources in Amazon EC2". 8th International Conference on Network and Service Management (CNSM), pp. 159–163, Las Vegas, USA. October, 2012.

[44] TPC, TPC-W Benchmark, Transaction Processing Performance Council (TPC), San Francisco, CA, USA, 2003.

[45] Trevor, H., Tibshirani, R., and Friedman, J., “The Elements of Statistical Learning: Data Mining, Inference, and Prediction”, New York: Springer, February, 2009.

[46] Wang, S. and Summers, R.M. "Machine learning and radiology". Medical Image Analysis. Vol 16, Issue 5. pp. 933-951. 2012.

[47] Witten, I. H and Frank, E., “Data Mining Practical Machine Learning Tools and Techniques with Java Implementations”, San Diego: Academic Press, 2000.

[48] Wood, T., Cherkasova, L., Ozonat, K., and Shenoy, P., “Profiling and Modeling Resource Usage of Virtualized Applications” Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, pp. 366-387, New York, USA. December, 2008.

SOCIETY FOR SCIENCE AND EDUCATION
UNITED KINGDOM
TMLAI TRANSACTIONS ON
MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE
ISSN 2054-7390

The Adept K-Nearest Neighbour Algorithm - An optimization to the Conventional K-Nearest Neighbour Algorithm

Anjali Jivani[1], Karishma Shah[2], Shireen Koul[3] and Vidhi Naik[4]
Department of Computer Science and Engineering, Maharaja Sayajirao University of Baroda, Vadodara, India
[1]anjali_jivani@yahoo.com; [2]12.karishma@gmail.com; [3]koulshireen@gmail.com;
[4]vidhinaik2@yahoo.com;

ABSTRACT

This research aims to study the efficiency of a well-known classification algorithm, K-Nearest Neighbour, and suggest a new classification method, an optimised version than one of the existing classification method. The purpose of this research is to reduce the time taken by the existing K- Nearest Neighbour Classification method. The classification algorithm's purpose is to identify the characteristics that indicate the class to which each document belongs. This pattern not only helps in understanding the existing data but also to predict how new instances will behave. Classification algorithms create classification models by examining already classified data (cases) and inductively finding a predictive pattern.

Keywords: Supervised Learning, classification, k Nearest Neighbour, Cosine Similarity

1 Introduction

Classification [1] is a data mining (machine learning) technique used to predict group membership for data instances. For example, you may wish to use classification to predict whether the weather on a particular day will be sunny, rainy or cloudy. Popular classification techniques include decision trees and neural networks.

Following are the examples of cases where the data analysis task is Classification −

- A bank loan officer wants to analyse the data in order to know which customers (loan applicant) are risky or which are safe.
- A marketing manager at a company needs to analyse a customer with a given profile, who will buy a new computer.

Classification is a supervised learning data mining techique.

2 k-Nearest Neighbour Algorithm (kNN)

Simply stated, kNN is an algorithm that classifies the new cases based on similarity measures[2] or distance measures of pair of observations such as euclidean, cosine, etc. kNN algorithm is a lazy learner i.e. it does not learn anything from the training tuples and simply uses the training data itself for classification [5]. It is a non-parametric method used for classification and regression. [3]

DOI: 10.14738/tmlai.41.1876
Publication Date: 27th February, 2016
URL: http://dx.doi.org/10.14738/tmlai.41.1876

The newly arrived document is classified on the basis of the majority occurring class of its neighbours, with the document being assigned to the class most frequent among its *k* nearest neighbours. Figure 1[4] depicts how kNN works.

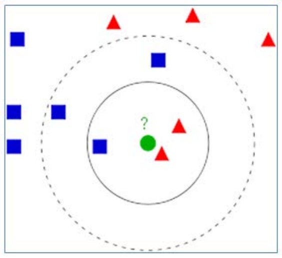


Figure 1: Example of kNN

The test document of green circle needs to be classified; it can either be classified into red triangle or Blue Square depending on the distance between each of them. If k=3 (shown by a solid line), green circle will be classified to red triangle class as out of the three, red triangle occurs two times and blue square occurs one time. But, if k=5(shown by a dotted line), then the green circle will be classified to blue square for similar reasons.

Line up of sub-procedures explains how kNN method is implemented on a broader aspect.

A newly arrived document is fed into an algorithm that removes all the stop words present in the text file and the intermediate result is then sent to porter stemming algorithm. The sub-procedure at that point calculates tf-idf score and hence cosine similarity between the newly arrived document and all the cases which were previously classified into specific classes. An ordered sorted list is prepared with cases arranged in the descending order fashion of cosine similarity. Cosine similarity ranges between 0 and 1 where cosine similarity of 0 means that the two compared cases are not at all similar and a cosine similarity of 1 means that the two compared cases are completely similar. 'k' in k Nearest Neighbour method is a user input. The procedure then selects the top k neighbours from the ordered list. The newly arrived document is then classified in to that class which occurs maximum number of time in the top k-selected neighbours.

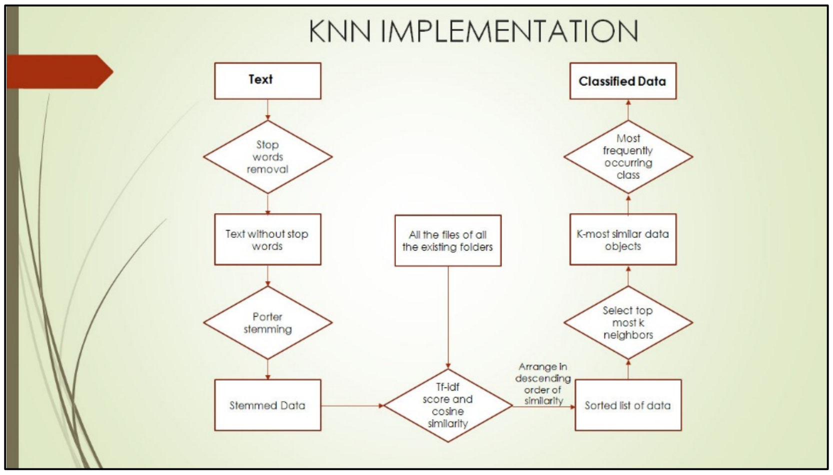

KNN IMPLEMENTATION
Text
Stop words removal
Text without stop words
Porter stemming
Stemmed Data
All the files of all the existing folders
Tf-idf score and cosine similarity
Arrange in descending order of similarity
Sorted list of data
Select top most k neighbors
K-most similar data objects
Most frequently occurring class
Classified Data

Figure 2: Flowchart showing stepwise implementation of kNN

2.1 Advantages of kNN

The advantages of the kNN method are as follows:

1. Analytically tractable
2. Simple implementation
3. Nearly optimal in the large sample limit ($N \rightarrow \infty$)
4. Uses local information, which can yield highly adaptive behaviour
5. Lends itself very easily to parallel implementations

2.2 Drawbacks of kNN

The drawbacks of the kNN method are as follows:

1. kNN algorithm is that it is a *lazy learner*, i.e. it simply uses the training data itself for classification.
2. Result of this is that the method does not learn anything from the training data, which can result in the algorithm not generalizing well. Further, changing K can change the resulting predicted class label.
3. Also, algorithm may not be robust to noisy data.
4. To predict the label of a new instance the kNN algorithm will find the K closest neighbours to the new instance from the training data, the predicted class label will then be set as the most common label among the K closest neighbouring points.
5. The algorithm needs to compute the distance and sort all the cases at each prediction, which can be slow if there are a large number of training examples.
6. Large storage requirements.
7. Computationally intensive recall.
8. Highly susceptible to the curse of dimensionality.

3 The Adept k Nearest Neighbour Algorithm

Here we describe the proposed algorithm. We have considered our input data as text documents. These documents belong to three classes ARTS, SCIENCE and FINANCE with a number of documents already been classified into one of these three classes. In the document of the conventional kNN as we have seen above, after the removal of stop words and obtaining the stemmed output, the intermediate result will be compared with the existing classes of documents, in order to find the cosine similarity between all of them.

The newly suggested method, Adept kNN, emphasises on the fact that most of the comparisons or similarity checks go in vain. It is suggested that in contrast of comparing a document with each existing or previously classified document, it will be better to compare the document to be classified with three cases only (or files as in our example) each document being an intermediate representation of corresponding class. This intermediate representative document or text file is subjected to change over time.

Adept kNN is just an optimisation of kNN. The flowchart of the Adept kNN is as shown in Figure 3.

The intermediate file is designed in such a way that it contains only the important or frequently occurring words/tags. To determine the frequently occurring words, a counter for each word is maintained which is incremented by one each time the word occurs in the text. Once, the counter exceeds a value X (which is a user defined value), the counting procedure for that particular word stops as the word has already been put in the file or list of frequently occurring words or tags. A newly arrived document is then compared with only 3 such files/cases instead of being compared to 30 files as we were doing it previously. Also with arrival of each new document, the intermediate representative file is updated with concatenation of new frequently occurring tag, if any.

The line-up of sub-procedures in Adept kNN is similar to the one in conventional kNN.

Firstly, the document is fed into procedure for stop word removal and then into porter stemming method. Then the cosine similarity between the intermediate stemmed output and representative files is determined. The document to be classified is classified into the class which is most similar to it, that is the one which has highest cosine similarity. This can also be inferred that there is no question of k (user input) in this method.

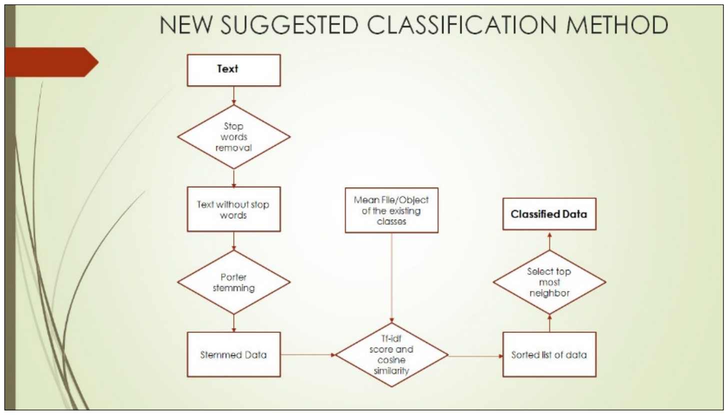


Figure 3: Flowchart showing stepwise implementation of Adept kNN

3.1 Advantages of Adept- kNN over kNN

1. The algorithm needs to compute the distance with only the representative files and hence reduce the time required to perform this task significantly.
2. Accuracy of this method is almost comparable to that of the original kNN.
3. Lesser number of comparisons or similarity checks are needed to be performed as compared to that of kNN.
4. There is no need of a user input K as the document is directly classified to that class whose representative file is most similar to the newly arrived document i.e. where cosine similarity is highest between the new document and the representative file.

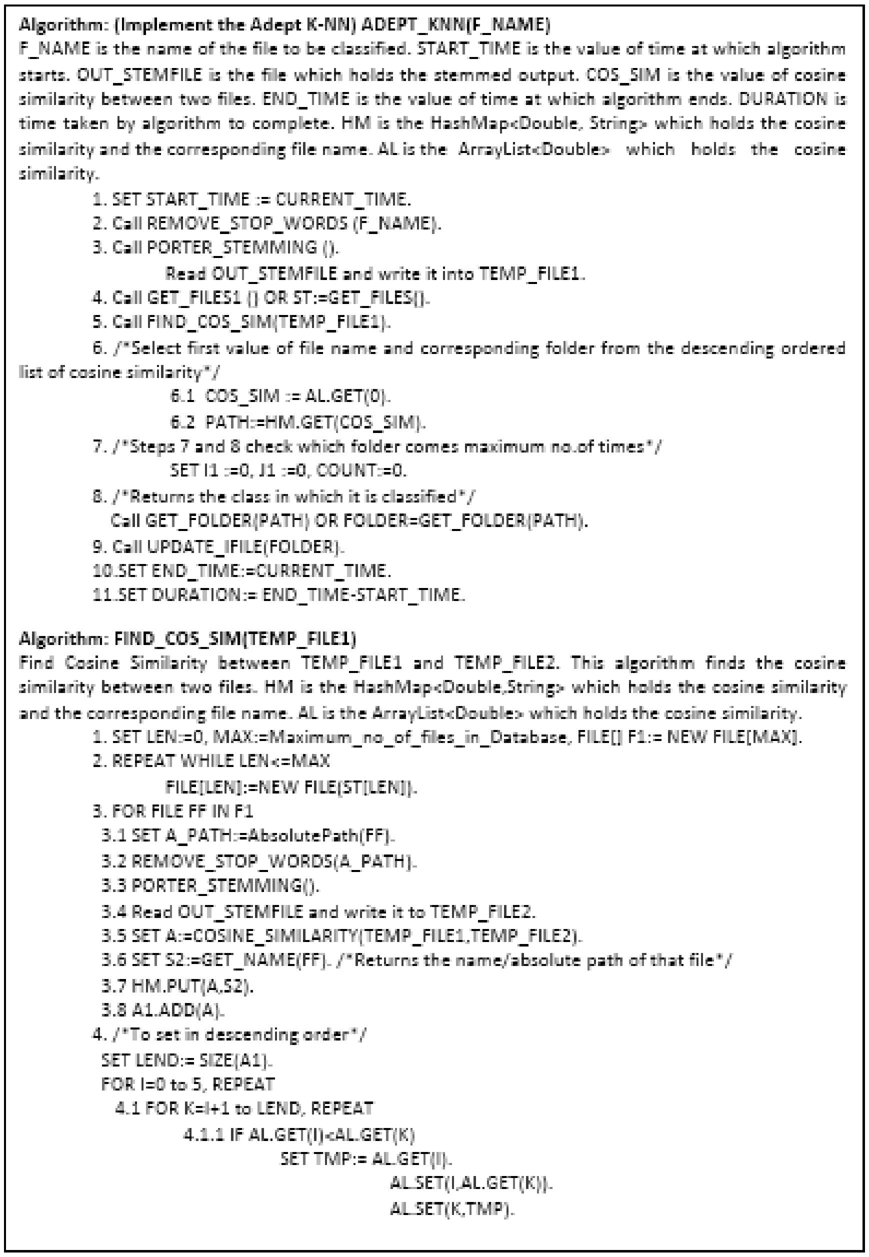
3.2 Disadvantages of Adept kNN

1. There is an overhead of maintaining an additional file in addition to maintain the already residing files or cases.
2. When the count of previously classified files will increase to millions, the size of intermediate representative file will also become huge.
3. In addition to maintain the representative file, Adept kNN also needs to consistently maintain a counter of each word or tag. If at any moment it fails to do so, the algorithm may result in an

inconsistent classification as it no longer maintains the perfect list of frequently occurring words or tags.

The algorithm for our proposed Adept kNN method is as shown is Figure 4. The details about each procedure are described as per the pre-processing done on the documents and the classification process itself is also described in detail.

The documents are first tokenized and converted to a bag of words. After that the stop words have been removed by comparing the tokens from the documents with the list of stop words available on the net. The Porter's stemming algorithm has been used to perform stemming on the remaining words. The comparison between the bag of words of documents has been done using the cosine similarity.



Algorithm: (Implement the Adept K-NN) ADEPT_KNN(F_NAME)

F_NAME is the name of the file to be classified. START_TIME is the value of time at which algorithm starts. OUT_STEMFILE is the file which holds the stemmed output. COS_SIM is the value of cosine similarity between two files. END_TIME is the value of time at which algorithm ends. DURATION is time taken by algorithm to complete. HM is the HashMap<Double, String> which holds the cosine similarity and the corresponding file name. AL is the ArrayList<Double> which holds the cosine similarity.

```
    1. SET START_TIME := CURRENT_TIME.
    2. Call REMOVE_STOP_WORDS (F_NAME).
    3. Call PORTER_STEMMING ().
            Read OUT_STEMFILE and write it into TEMP_FILE1.
    4. Call GET_FILES1 () OR ST:=GET_FILES().
    5. Call FIND_COS_SIM(TEMP_FILE1).
    6. /*Select first value of file name and corresponding folder from the descending ordered
list of cosine similarity*/
            6.1  COS_SIM := AL.GET(0).
            6.2  PATH:=HM.GET(COS_SIM).
    7. /*Steps 7 and 8 check which folder comes maximum no.of times*/
            SET I1 :=0, J1 :=0, COUNT:=0.
    8. /*Returns the class in which it is classified*/
      Call GET_FOLDER(PATH) OR FOLDER=GET_FOLDER(PATH).
    9. Call UPDATE_IFILE(FOLDER).
    10.SET END_TIME:=CURRENT_TIME.
    11.SET DURATION:= END_TIME-START_TIME.
```

Algorithm: FIND_COS_SIM(TEMP_FILE1)

Find Cosine Similarity between TEMP_FILE1 and TEMP_FILE2. This algorithm finds the cosine similarity between two files. HM is the HashMap<Double,String> which holds the cosine similarity and the corresponding file name. AL is the ArrayList<Double> which holds the cosine similarity.

```
    1. SET LEN:=0, MAX:=Maximum_no_of_files_in_Database, FILE[] F1:= NEW FILE[MAX].
    2. REPEAT WHILE LEN<=MAX
            FILE[LEN]:=NEW FILE(ST[LEN]).
    3. FOR FILE FF IN F1
      3.1 SET A_PATH:=AbsolutePath(FF).
      3.2 REMOVE_STOP_WORDS(A_PATH).
      3.3 PORTER_STEMMING().
      3.4 Read OUT_STEMFILE and write it to TEMP_FILE2.
      3.5 SET A:=COSINE_SIMILARITY(TEMP_FILE1,TEMP_FILE2).
      3.6 SET S2:=GET_NAME(FF). /*Returns the name/absolute path of that file*/
      3.7 HM.PUT(A,S2).
      3.8 A1.ADD(A).
    4. /*To set in descending order*/
      SET LEND:= SIZE(A1).
      FOR I=0 to 5, REPEAT
        4.1 FOR K=I+1 to LEND, REPEAT
              4.1.1 IF AL.GET(I)<AL.GET(K)
                      SET TMP:= AL.GET(I).
                           AL.SET(I,AL.GET(K)).
                           AL.SET(K,TMP).
```

Figure 4: Algorithm of Adept kNN

4 Future Scope

The area of application of Adept kNN is almost similar to that of the original kNN

1. Text Mining
2. Agricultural Area
3. Financial Departments
4. Medicinal Field
5. Large Organizations

REFERENCES

[1] Based on study from http://www.academia.edu/4607757/Application_of_K-Nearest_Neighbor_kNN_Approach_for_Predicting_Economic_Events_Theoretical_Background

[2] Biju Issac, Nauman Israr. " Case Studies in Intelligent Computing: Achievements and Trends"

[3] Altman, N. S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression".

[4] Antti Ajanki AnAj. http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[5] Based on the study from Jiawei Han, Micheline Kamber. "Data Mining Concepts and Techniques"

[6] http://research.cs.tamu.edu/prism/lectures/pr/pr_l8.pdf

[7] http://www.nickgillian.com/wiki/pmwiki.php?n=GRT.kNN

Society for Science and Education United Kingdom
TMLAI Transactions on Machine Learning and Artificial Intelligence
ISSN 2054-7390

Brain Tumor Segmentation using Multiobjective Fuzzy Clustering

Olfa Mohamed Limam
Institut Supérieur d'Informatique, University Tunis EL Manar, Tunis, 2080, Tunisie
limemolfa@yahoo.fr

ABSTRACT

The segmentation of magnetic resonance images plays a crucial role in medical image analysis because it extracts the required area from the image. Despite intensive research, it still remains a challenging problem and there is a need to develop an appropriate and efficient medical image segmentation method. In this paper, we propose a clustering approach for brain tumor segmentation to diagnose accurately the region of cancer. Applied to magnetic resonance image brain our method provides better identification of brain tumor.

Keywords: Brain tumor image segmentation, fuzzy clustering, multiobjective optimization, genetic algorithm.

1 Introduction

Image segmentation is an essential process in most medical image analysis, particularly in the study of some brain disorders. Interpretation process of complex medical structures, like tumor, cannot be identified without automatic image segmentation [1]. For example, the tumor portion of an image is defined as the irregular development of cells' growth inside the brain. Hence, image segmentation is needed to extract the required area from images [2]. As a common data analysis tool, clustering is a vital technique to segment a given data set into a distinct clusters such that similar patterns are assigned to a group, which is considered as a cluster [3]. Clustering works efficiently in discriminating between different tissues from medical images [4]. Conventional clustering methods assign points exclusively to one cluster. However, real medical image data sets present overlapping gray-scale intensities for different tissues [4]. In particular, borders between tissues are not clearly defined and memberships in the boundary regions are fuzzy. Fuzzy sets define the idea of uncertainty of belonging by a membership function [4]. Fuzzy clustering is considered to be the most suitable to model this situation in decision oriented applications such as tumor detection and tissue classification [5]. Fuzzy clustering technique provides a means of classifying pixel values with a good accuracy.

Several studies of brain tumor segmentation have been proposed in the literature. Methods based on statistical models such as Markov random models and Gaussian intensity models work well in normal brain image segmentation but not for abnormal tissues [6].

In pathological cases, methods based on classification techniques are widely used in image segmentation and have shown their robustness [6]. Fuzzy c-means (FCM) is the most popular clustering for image

DOI: 10.14738/tmlai.41.1840
Publication Date: 27th February, 2016
URL: http://dx.doi.org/10.14738/tmlai.41.1840

segmentation [1]. Li et al. [7] segmented brain images using FCM algorithm. Maksoud et al. [8] used K-means clustering technique integrated with FCM algorithm. It is followed by thresholding and level set segmentation stages to provide an accurate brain tumor detection. Pham and Prince [9] used also FCM to deal with magnetic resonance imaging (MRI). Udupa and Pnuam [10] used the fuzzy connectedness for abnormal tissue segmentation. Menon et al. [11] and Alsmadi [12] proposed a MRI brain image segmentation method using the hybridization of FCM and artificial bee colony algorithm.

Fuzzy clustering has been successfully applied, even though, it is greatly affected by noise and being very sensitive to its initialization [13].

MRI segmentation using thresholding is thought to be a very simple method to section an image that has dark backgrounds. Natarajan et al. [14] used thresholding based method for proficient recognition of a brain tumor image. Thapaliya et al. [15] used also thresholding for brain tumor image. Whereas, the determination of thresholding for the distribution of tissue intensities is very complex [1]. Therefore, segmentation techniques based on thresholding are simple and effective but they don't consider busyness of gray levels [16].

Region growing is also an important segmentation approach used for tumor detection. Weglinski et al. [17] proposed a region growing approach to segment brain tumor images. Won and Kim [18] proposed a similar technique to detect a brain ventricle in MRI brain images. Viji and JayaKumari [19] developed a texture based region growing technique for brain image segmentation. Gibbs et al. [20] used region growing for tumor volume determination. Lachmann et al. [21] used means of the texture analysis for brain tissue classification. Region growing is also effective but not a fully automatic segmentation of tumor tissues.

Numerous neural network based approaches have been proposed to segment brain images [22]. Hall et al. [23] used artificial neural network for segmentation of MRI brain images. Wells et al. [24] and Leemput et al. [25] used expectation-maximization (EM) for classification of MRI brain images. While, a common disadvantage of EM algorithm is the use of the normal distribution to model the intensity distribution which is untrue for noisy images [25].

Despite the potential of these approaches, some of them need manual guidance for their initialization with some manual learning [6]. In conclusion, fuzzy clustering is the most widely used technique [13].

However, fuzzy clustering methods optimizing a single objective fail to give satisfactory results since a single validity measure is not able to deal with different kinds of data sets. Moreover, the wrong selection of a validity measure leads to poor results [5]. Therefore, it is reasonable to consider more than one objective. Acharay et al. [26] used simulated annealing as the multiobjective optimization strategy for classification of tissue samples from cancer data sets. However, most multiobjective fuzzy clustering techniques are usually used in the segmentation of brain images but less for brain tumor detection [26]. Therefore, this paper proposes an adapted multiobjective fuzzy clustering algorithm for brain tumor segmentation of MRI images to accurately diagnose the region of cancer [27].

This paper is organized as follows. Section 2 details our proposed approach. Section 3 describes our experimental analysis and displays a visual comparison results with other fuzzy clustering techniques along with a quantitative comparison. Section 4 gives a conclusion and some perspectives.

2 The Multiobjective Fuzzy Clustering Model

A brain tumor segmentation consists of separating the different tissues such as solid tumor, edema and necrosis from the normal brain tissues as same as the tissue classes of brain MRI, gray matter, white matter and cerebrospinal fluid [6].

We propose a multiobjective fuzzy clustering method which simultaneously optimizes the fuzzy spatial compactness of the clusters and the fuzzy spatial separation among the clusters. In brain MRI image, neighboring pixels have a strong correlation. Thus, we propose to incorporate the spatial information among objective functions to generate more accurate clusters. Moreover, we use FCM for encoding initial centers instead of randomly extracting them from the data set. For the final generation, our multiobjective approach produces a set of non-dominated solutions, from which the best solution is chosen to be the final image segmentation based on the I index measure. In the following section, we discuss the different steps of our proposed method.

2.1 Feature extraction

Original brain MRI is a gray-level image deficient to support fine features [2]. To obtain more useful features and improve the visual density, the proposed method adopts the standard RGB color space. In medical image segmentation, color features could be considered [28].

2.2 Pattern proximity

The pattern proximity is calculated using a distance function defined on pairs of patterns, namely the Euclidean distance [29]. In general, the distance between two pixels *x:* $(x_1, ..., x_n)$ and *y:* $(y_1, ..., y_n)$ in an Euclidean n-space is defined as follows:

$$d(x,y) = |x-y| = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \qquad (1)$$

2.3 Multiobjective fuzzy clustering algorithm

NSGA-II is used as the underlying optimization multiobjective strategy. NSGA-II inputs are the data set, population size, maximum generation and an upper bound on the number of clusters. This algorithm requires a number of individuals as potential solutions to be initialized at the beginning of the algorithm.

2.3.1 Population Initialization

In NSGA-II based clustering, real-valued chromosomes representing the coordinates of the partitions (centers) are used. Centers encoded in a given chromosome in the initial population are computed using FCM instead of being randomly extracted from the data set to give a more robust partition. FCM produces C cluster centers and C x N membership matrix U(x). The FCM output is developed in a fuzzy representation where each pixel has a variable degree of membership to each of the output centers encoded in a chromosome.

2.3.2 Computation of the objectives

NSGA-II fitness functions, fuzzy global spatial compactness and fuzzy spatial separation are simultaneously optimized and computed for each chromosome. To compute the objective functions, we extract the centers $V = \{v_i\}$ for $i \in \{1, ..., C\}$ encoded in a given chromosome.

The first objective, global spatial compactness τ, is given by :

$$\tau = \sum_{i=1}^{C} \frac{\sigma_i}{n_i} = \sum_{i=1}^{C} \frac{\sum_{k=1}^{N} u_{ik}^{*m} D(v_i, x_k)}{\sum_{k=1}^{N} u_{ik}^{*m}} \tag{2}$$

where σ_i denotes the variation, $D(v_i, x_k)$ is the Euclidean distance between ith cluster center and kth data point, $m \in \{1,\infty\}$ is the fuzzy exponent and n_i is the fuzzy cardinality of the ith cluster such that

$$n_i = \sum_{k=1}^{N} u_{ik}^{*m}, 1 \le i \le C \tag{3}$$

The second objective, fuzzy spatial separation S, is defined as :

$$S = \sum_{i=1}^{C} \sum_{i=1, j \neq i}^{C} u_{ij}^{*m} \; D(v_i, v_j). \tag{4}$$

The measure τ should be minimized while the fuzzy separation S should be maximized, as follows

$$\begin{cases} \text{Min } \tau \\ \text{Max } S \end{cases}$$

We propose to integrate the relative locations of neighboring pixels assuming that a pixel has a dependent intensity to its surrounding pixels. However, a pixel is too small to represent a part of an image and it is affected by neighborhood intensity [1]. The notion of spatial coherence for brain tumors is carried out in order to segment different varieties of brain tumors. This is a challenging issue because tumors can appear in many different sizes and shapes [30]. The segmentation process is therefore decided not only by the pixel intensities but also by the neighboring pixels intensities. A pixel gets high membership value to a cluster when its neighborhood pixels have a higher membership value to that cluster [31].

The spatial function h_{ik} represents the probability that pixel x_k belongs to the ith clustering. The $NB(x_k)$ represents a square window centered where pixel x_k is in the spatial domain. A window of 3x3 is used throughout this work. The modified membership function of the current pixel, that includes the spatial information, is defined as follows:

$$u_{ik}^{*} = \frac{u_{ik} h_{ik}}{\sum_{j=1}^{C} u_{jk} h_{jk}}, \tag{5}$$

where u_{ik} is computed as follows.

$$u_{ik} = \frac{1}{\sum_{j=1}^{C} \left(\frac{D(v_i, x_k)}{D(v_j, x_k)} \right)^{\frac{2}{m-1}}} \tag{6}$$

For 1 ≤i ≤C; 1 ≤ k ≤ N.

The spatial function is defined as:

$$h_{ik} = \sum_{k \in Nb(x_k)} g_{ik}, \tag{7}$$

where, for l={1,...,C}

$$g_{ik} = \begin{cases} 1 & if \quad u_{ik} = \max\{u_{lk}\}, \\ 0 & otherwise \end{cases} \tag{8}$$

2.3.3 Genetic Operators

Crowded binary tournament selection is adopted to generate the mating pool of chromosomes, followed by conventional crossover and mutation, is used here. The last part of the NSGA-II is its elitism operation, where the non dominated solutions among the parent and child populations are propagated to the next generation. This algorithm produces a large number of non-dominated solutions on the final Pareto optimal front.

2.3.4 Optimal solution

In the last stage of the proposed approach, it is obvious to choose the appropriate solution from the set of non-dominated solutions based on a validity measure namely the I index [32].

The pseudo code of our proposed algorithm is given in Algorithm 1.

Algorithm 1

1. Initialization process: FCM to compute the cluster centers of the partition encoded in chromosomes for the initial population.
2. Multiobjective optimization: Multiobjective clustering technique NSGA-II produces a set of near Pareto optimal solutions.
3. Partition validation: Maximum I index value.

3 Experimental Analysis

In order to show the performance of our method, we conducted an experimental study on two simulated brain tumor image data sets representing two different tumor shapes. Moreover, we compare our segmented image to other images produced by competing algorithms respectively, k-means algorithm, histogram clustering and FCM clustering algorithm [2]. Then, a quantitative comparison is conducted between clustering algorithms based on performance accuracy and execution time.

The platform for conducting the experiments is a Core 2 Duo 2.10 GHz laptop with 3 GB RAM using Windows 7 32bit. The program was implemented using MATLAB development environment. The simulated T-weighted MRI brain data sets were downloaded from Brainweb [33]. Brainweb provides a simulated brain database including a set of realistic MRI data volumes produced by an MRI simulator.

3.1 Visual Comparison

Figure 1 contains a low grade tumor, while, Figure 2 contains a large, partially enhancing tumor inside the brain stem. Figure 1 and Figure 2 (a) show the original MRI brain tumor image and (b) display the image segmented by our approach.

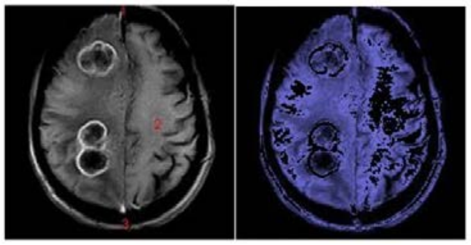


Figure 1 (a) Original MRI brain tumor image with a low grade tumour (b) Corresponding image segmented by our method

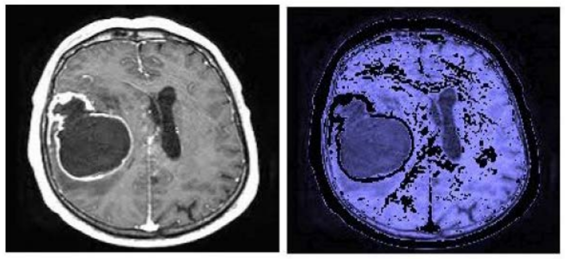


Figure 2 (a) Original MRI brain tumor image with a large grade tumour (b) Corresponding image segmented by our method

In Figures 1, 2, our method succeed to locate the tumor part of the brain and clearly identifying from the other part of the brain.

Figure 3 shows the segmentation results of four different clustering algorithms including our proposed method. Figure 3 shows (a) The original MRI brain image, (b) K-means segmented image, (c) Histogram segmented image, (d) FCM segmented image and (e) Our image segmented.

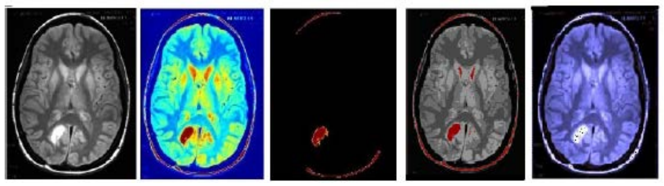

(a) (b) (c) (d) (e)

Figure 3 (a) The original MRI brain image, (b) k-means segmented image, (c) histogram segmented image, (d) FCM segmented image and (e) Our image segmented

From Figure 3, it can be noticed that the k-means and FCM clustering algorithms identify efficiently the different regions of the brain image, while the Histogram algorithm does not. Our proposed method succeed in identifying the different regions, showing the ability of fuzzy clustering for brain image segmentation and brain tumor detection.

It can be seen through comparison that our method gives better result as compared to the original image.

In conclusion, based on experimental study, we show the effectiveness of our method in segmenting the brain tumor with different tumor grades. Also, our proposed method gives better results compared with three state-of-the-art algorithms: K-means, Histogram and FCM.

3.2 Quantitative comparison for brain tumor images

Moreover, to show the performance of our proposed method, we conducted a comparative study on two different brain tumor image data sets. Based on a performance measure namely accuracy [32], defined as follows

$$Accuracy(T,C) = \frac{a+b}{c} \times 100, \tag{9}$$

The accuracy of segmentation is obtained as the ratio between the sum of correctly classified pixels to the total number of pixels [8], where T denotes the true clustering of data set based on domain knowledge, C the partition given by a clustering algorithm, a the number of pairs of points that belong to the same clusters in both T and C, b the number of pairs of points that belong to different clusters in both T and C, and c the total number of pairs of points. A high value of accuracy (≥ 65%) denotes better matching between T and C [35]. Thus, high values of accuracy mean that we have an optimal matching between the clustering solution and the known partition.

Also, our comparison is based on the execution time. Tables 1 and 2 show the results of a comparison between our proposed method with K-means (KM), Expectation Maximization (EM), Mean shift clustering technique (MS), FCM, K means integrated with fuzzy c means (KFCM), particle swarm optimization (PSO) [8] and our proposed method (PM), based on the performance accuracy and the execution time, respectively.

Table 1 The performance accuracy of fuzzy clustering algorithms for brain tumor images

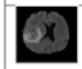
DS	KM	EM	MS	FCM	KFCM	PSO	PM
	85.7	66.6	85.7	85.7	90.5	95.5	95.75
	95.06	95.06	95.06	100	100	100	100

Table 1 illustrates the evaluation results using the performance measure of accuracy for different clustering algorithms. Where the accuracy refers to the percentage of correct predictions made by the model compared to the known partition in the test data.

The performance accuracy is enhanced compared to other algorithms. The performance accuracy results show the capacity of our proposed method in providing better results.

Table 2: The Execution time of fuzzy clustering algorithms for brain tumor images

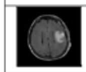
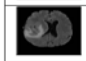
DS	KM	EM	MS	FCM	KFCM	PSO	PM
	7.5	34.47	59.52	12.87	34.58	9.23	20.45
	4.34	32.06	6.89	3.46	6.26	6.01	3.56

Table 2 shows the execution time (seconds). For the first brain image, KM takes less processing time but it does not give accurate results as in the second brain image. However, the obtained results indicate the success of our proposed method in detecting clearly the tumor site in the image. Therefore, our proposed method provides good results compared to other competing algorithms in terms of execution time.

4 Conclusion

This paper proposes an improved detection method of brain tumor using clustering technique. We used a multiobjective fuzzy clustering that integrates the spatial information and a robust initialization process. The proposed method is applied to segment MRI brain image for recognition of tumor. Both quantitative and visual comparison show the superiority of our proposed method which has the ability of finding tumor locations in MRI brain. Our method can be extended to detect tumors on other views of brain.

REFERENCES

[1] Shen Shan, Sandham William, Granat Malcolm, and Sterr Annette. Mri fuzzy segmentation of brain tissue using neighborhood attraction with neural-network optimization. IEEE Transactions on Information Technology in Biomedicine, 9(3):459–467, 2005.

[2] P.Tamije Selvy, V. Palanisamy, and T. Purusothaman. Performance analysis of clustering algorithms in brain tumor detection of mr images. European Journal of Scientific Research, 62(3):321–330, 2011.

[3] Suchita Yadav and Meshram Sachin. Brain tumor detection using clustering method. International Journal of Computational Engineering Research, 3(4):11–14, 2013.

[4] Miin-Shen Yang, Yu-Jen Hu, Karen Chia-Ren, and Charles Chia-Lee Lin. Segmentation techniques for tissue differentiation in mri of ophthalmology using fuzzy clustering algorithms. Magentic resonance Imaging, 20:173–179, 2002.

[5] Sanghamitra Bandyopadhyay, Anirban Mukhopadyay, and Ujjwal. Maulik. Combining multiobjective fuzzy clustering and probabilistic ann classifier for unsupervised pattern classification: Application to satellite image segmentation. IEEE on Evolutionary Computation, pages 877–883, 2008.

[6] Weibei Dou, Su Ruan, Daniel Bloyet, and Jean-Marc Constans. A framework of fuzzy information fusion for the segmentation of brain tumor tissues on mr images. Image and Vision Computing, 25(2):164–171, 2007.

[7] Chunlin Li, DB Goldgof, and LO Hall. Knowledge-based classification and tissue labeling of mr images of human brain. IEEE transactions on Medical Imaging, 12(4):740–750, 2002.

[8] Eman A. Abdel Maksoud, Mohammed Elmogy, and Rashid Mokhtar Al-Awadi. MRI brain tumor segmentation system based on hybrid clustering techniques. In Advanced Machine Learning Technologies and Applications - Second International Conference, AMLTA, pages 401–412, 2014.

[9] L. Dzung Pham and Jerry L. Prince. Knowledge-based classification and tissue labeling of mr images of human brain. IEEE transactions on Medical Imaging, 18(9):737–752, 1999.

[10] Jayram K. Udupa and K. Saha Punam. Fuzzy connectedness and image segmentation. Proceedings of the IEEE, 91:1649–1669, 2003.

[11] R Sivakumarl Neeraja R Menon, M Karnan. Brain tumor segmentation in mri image using unsupervised artificial bee colony and fcm clustering. International Journal of Computer Science and Management Research, 2:2450–2454, 2013.

[12] Mutasem K. Alsmadi. Mri brain segmentation using a hybrid artificial bee colony algorithmwith fuzzy-c mean algorithm. Journal of Applied Sciences, 15:100–109, 2015.

[13] Zhanshen Feng and Boping Zhang. Fuzzy clustering image segmentation based on particle swarm optimization. Telecommunication Computing Electronics and Control, 13(1):128–136, 2015.

[14] P. Natarajan, N. Krishnan, N.S. Kenkre, S. Nancy, and B.P. Singh. Tumor detection using threshold operation in mri brain images. IEEE International Conference on Computational Intelligence and Computing Research (ICCIC), pages 1–4, 2012.

[15] K. Thapaliya, , and Kwon Goo-Rak. Extraction of brain tumor based on morphological operations. 8th International Conference on Computing Technology and Information Management (ICCM), 20:515 – 520, 2012.

[16] Anam Mustaqeem, Ali Javed, and Tehseen Fatima. An efficient brain tumor detection algorithm using watershed thresholding based segmentation. Interntional Journal of Image, Graphics and Signal Processing, 10:34–39, 2012.

[17] Tomasz Weglinski and Anna Fabijanska. Brain tumor segmentation from mri data sets using region growing approach. Proceedings of VIIth International Conference on Perspective Technologies and Methods in MEMS Design, pages 185 – 188, 2011.

[18] Won Chulho and Kim Dong-Hun. Region growing method using edge sharpness for brain ventricle detection. SICE, Annual Conference, pages 1930–1933, 2007.

[19] K.S.A Viji and J. JayaKumari. Modified texture based region growing segmentation of mr brain images. IEEE Conference on Information Communication Technologies (ICT), pages 691–695, 2013.

[20] Peter Gibbs, David L. Buckley, Stephen J Blackband, and Anthony Horsman. Tumour volume determination from mr images by morphological segmentation. Physics in Medicine and Biology, 41:2437–2446, 1996.

[21] Frederic Lachmann and Christian Barillotr. Brain tissue classification from mri data by means of texture analysis. Medical Imaging VI: Image Processing, 72, 1992.

[22] D.M. Joshi, N. K. Rana, and V.M. Misra. Classification of brain cancer using artificial neural network. International Conference on Electronic Computer Technology (ICECT), pages 112–116.

[23] L.O Hall, A.M. Bensaid, L.P. Clarke, R.P. Velthuizen, and M.S. Silbige rand James C. Bezdek. A comparison of neural network and fuzzy clustering techniques in segmenting magnetic resonance images of the brain. IEEE Transactions on Neural Network, 3(5):672–682, 1992.

[24] W. M. Wells, W. L. Grimson, R. Kikinis, and F. A . Jolesz. Adaptive segmentation of mri data. IEEE Transactions on Medical Imaging, 15(4):429–442, 1996.

[25] K. V. Leemput, F. Maes, D. Vandermeulen, and P. Suetens. Automated model-based tissue classification of mr images of the brain. IEEE Transactions on Medical Imaging, 18(10):897–908.

[26] Sudipta Acharya, Yamini Thadisina, and Sriparna Saha. Multi-objective clustering of tissue samples for cancer diagnosis. International Conference on Advances in Computing, Communications and Informatics, pages 1059–1064, 2014.

[27] Olfa Limam and Fouad Ben Abdelaziz. Multicriteria fuzzy clustering for brain image segmentation. 2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), pages 1019 – 1031, 2013.

[28] Bong Chin-Wei and Mandava Rajeswari. Multiobjective optimization approaches in image segmentation - the directions and challenges. International Journal of Advances in Soft Computing and Its Applications, 2(2):41–64, 2010.

[29] Anil Kumar Jain, M Narasimha Murty, and Patrick Joseph Flynn. Data clustering: A review. ACM Computing Surveys, 31(3):265–323, 1999.

[30] Marcel Prastawa, Elizabeth Bullitt, Sean Ho, and Guido Gerig. A brain tumor segmentation framework based on outlier detection. Medical Image Analysis, 8(3):275–283, 2004.

[31] Xiang-Yang Wang and Juan Bu. A fast and robust image segmentation using fcm with spatial information. Digital Signal Processing, 20:1173–1182, 2010.

[32] Jie Liu, Jigui Sun, and Shengsheng Wang. Pattern recognition: An overview. International Journal of Computer Science and Network Security IJCSNS, 6(6):57–60, 2006.

[33] Brainweb: Simulated brain database, 2011, available at http://www.bic.mni.mcgill.ca/brainweb.

[34] K.Y. Yeung and Walter L. Ruzzo. An empirical study of principal component analysis for clustering gene expression data. 2001.

[35] Maulik Ujjwal and Mukhopadhyay Anirban. A multiobjective approach to mr brain image. Applied Soft Computing 11, 872–880.