# A Fast Subspace Text Categorization Method Using Parallel Classifiers

Nandita Tripathi[1], Michael Oakes[1] and Stefan Wermter[2]

[1]Department of Computing, Engineering and Technology, University of Sunderland, St Peters Way, Sunderland, SR6 0DD, United Kingdom
(Nandita.Tripathi@research.sunderland.ac.uk, Michael.Oakes@sunderland.ac.uk)

[2]Institute for Knowledge Technology, Department of Computer Science, University of Hamburg, Vogt Koelln, Str. 30, 22527 Hamburg , Germany
(wermter@informatik.uni-hamburg.de)

**Abstract.** In today's world, the number of electronic documents made available to us is increasing day by day. It is therefore important to look at methods which speed up document search and reduce classifier training times. The data available to us is frequently divided into several broad domains with many sub-category levels. Each of these domains of data constitutes a subspace which can be processed separately. In this paper, separate classifiers of the same type are trained on different subspaces and a test vector is assigned to a subspace using a fast novel method of subspace detection. This parallel classifier architecture was tested with a wide variety of basic classifiers and the performance compared with that of a single basic classifier on the full data space. It was observed that the improvement in subspace learning was accompanied by a very significant reduction in training times for all types of classifiers used.

**Keywords:** Text Categorization, Subspace Learning, Semantic Subspaces, Maximum Significance Value, Conditional Significance Vectors

## 1 Introduction

The huge amount and variety of data available to us today makes document search and classifier training a lengthy process. Due to the ever increasing volume of documents on the web, classifiers have to be periodically retrained to keep up with the increasing variation. Reduced classifier training times are therefore a big asset in keeping classifiers up to date with the current content. Classifier application efficiency (test efficiency) is also very important in returning search results.

Retrieving a relevant document quickly in the presence of millions of records (the web) is an essential characteristic for a search engine today. In addition to this, the *curse of dimensionality* [1] degrades the performance of many learning algorithms. The large number of dimensions reduces the effectiveness of distance measures [2]. Today's data also contains a large number of data domains which can be as diverse as medicine and politics. These data domains can be considered as independent subspaces of the original data.

Independent data domains give rise to the idea of using parallel classifiers. Instead of training a single classifier on the full dataset, we can use many classifiers in parallel to process these independent subspaces. Classifier performances can be improved further by using only a subset of the dimensions. Active research is going on in the area of dimension reduction [3].

Random Projections [4] have also been used in dimensionality reduction. In the Random Subspace Method (RSM) [5], classifiers were trained on randomly chosen subspaces of the original input space and the outputs of the models were then combined. However, random selection of features does not guarantee that the selected inputs have the necessary distinguishing information. Several variations of RSM have been proposed by various researchers such as Relevant Random Feature Subspaces for Co-training (Rel-RASCO) [6], Not-so-Random Subspace Method (NsRSM) [7] and Local Random Subspace Method [8].

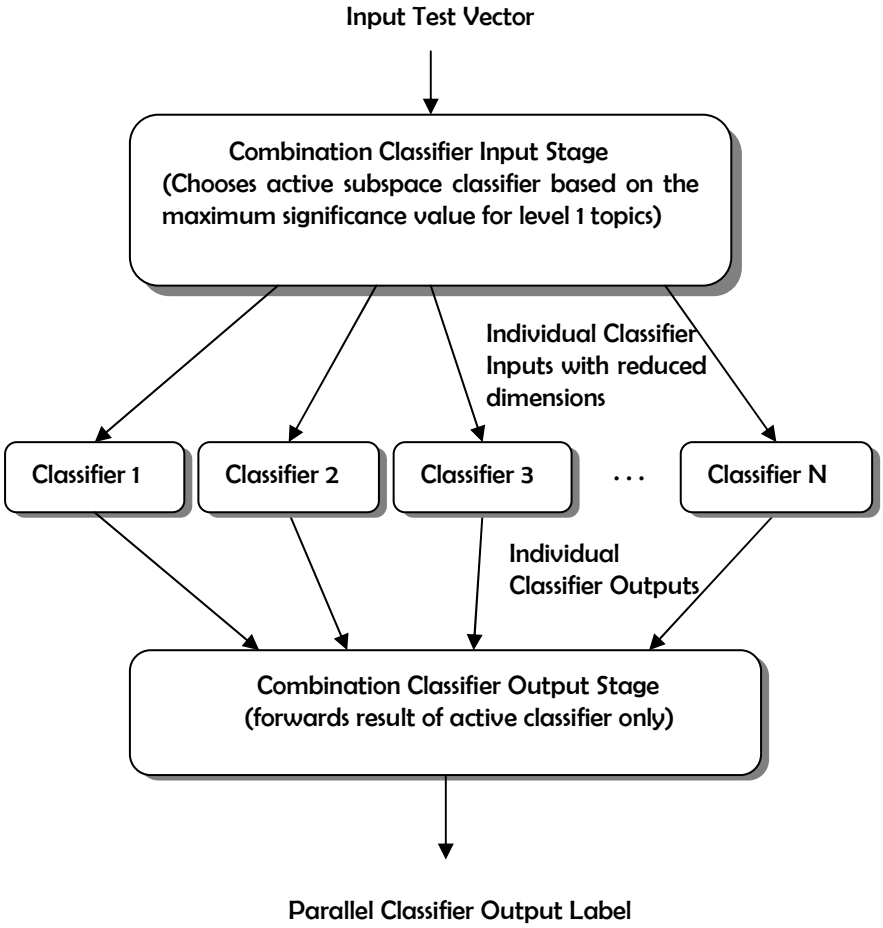
There are many methods of classifier combination. One method is to use many classifiers of the same or different types on different portions of the input data space. The combining classifier decides which part of the input data has to be applied to which base classifier. Two special types of classifier combinations are Bagging [9] and Boosting [10] which use a large number of primitive classifiers of the same type (e.g. a decision stump) on weighted versions of the original data.

Many classifier combination methods have been applied to text categorization. In one method [11], text and metadata were considered as separate descriptions of the same object. Another text categorization method [12] was based on a hierarchical array of neural networks. The problem of large class imbalances in text classification tasks was addressed by using a mixture-of-experts framework [13].

In the real world, documents can be divided into major semantic subspaces with each subspace having its own unique characteristics. The above research does not take into account this division of documents into different semantic subspaces. Therefore, we present here a novel parallel architecture (Fig. 1) which takes advantage of the different semantic subspaces existing in the data. We further show that this new parallel architecture improves subspace classification accuracy as well as it significantly reduces training time. For this architecture, we use parallel combinations of classifiers with a single type of base classifier. We use the conditional significance vector representation [14] which is a variation of the semantic significance vector

[15], [16] to incorporate semantic information into the document vectors. The conditional significance vector enhances the distinction between subtopics within a given main topic. The region of the test data is determined by the maximum significance value which is evaluated in O(k) time where k is the number of level 1 topics and thus can be very effective where time is critical for returning search results.

Fig. 1 : Parallel Classifier Architecture for Subspace Learning



## 2 Methodology and Architecture

In our experiments, we used the Reuters Corpus [17] as it is a well-known test bench for text categorization experiments. It also has a hierarchical organization with four major groups which is well suited to test the classification performance of a parallel architecture. We used the Reuters Corpus headlines for our experiments as they provide a concise summary of each news article. Each Reuters headline consists of one line of text with about 3 – 12 words. Some examples of Reuters Headlines are:

*"Estonian president faces reelection challenge."*
*"Guatemalan sides to sign truce in Norway report."*

The topic codes in the Reuters Corpus represent the subject areas of each news story. They are organized into four hierarchical groups, with four top-level nodes: Corporate/Industrial (CCAT), Economics (ECAT), Government/Social (GCAT) and Markets (MCAT). Ten thousand headlines along with their topic codes were extracted from the Reuters Corpus. These headlines were chosen so that there was no overlap at the first level categorization. Each headline belonged to only one level 1 category. At the second level, since most headlines had multiple level 2 subtopic categorizations, the first subtopic was taken as the assigned subtopic. Thus, each headline had two labels associated with it – the main topic (Level 1) label and the subtopic (Level 2) label. Headlines were then preprocessed to separate hyphenated words. Dictionaries with term frequencies were generated based on the TMG toolbox [18] and were then used to generate the Full Significance Vector [14], the Conditional Significance Vector [14] and the tf-idf [19] representation for each document. The datasets were then randomized and divided into a training set of 9000 documents and a test set of 1000 documents.

The WEKA machine learning workbench [20] provided various learning algorithms which we used as base classifiers to test our parallel architecture. Six algorithms were used as base classifiers in parallel classifier representations to examine the performance of various classification algorithms. Classification accuracy, training time and testing time were recorded for each experiment run. The average of ten runs for each representation was used to compare the various classifiers.

## 3 Data Processing for Experiments

### 3.1 Text Data Processing

Ten thousand Reuters headlines were used in these experiments. The Level 1 categorization of the Reuters Corpus divides the data into four main topics. These main topics and their distribution in the data along with the number of subtopics of each main topic in this data set are given in Table 1. Level 2 categorization further

divides these main topics into subtopics. Here we took the direct (first level nesting) subtopics of each main topic occurring in the 10,000 headlines. A total of 50 subtopics were included in these experiments. Since all the headlines had multiple subtopic assignments, e.g. C11/C15/C18, only the first subtopic e.g. C11 was taken as the assigned subtopic. Our assumption here is that the first subtopic used to tag a particular Reuters news item is the one which is most relevant to it.

**Table 1. Reuters Level 1 Topics**

| No. | Main Topic | Description | Number Present | Number of Subtopics |
|---|---|---|---|---|
| 1. | CCAT | Corporate/Industrial | 4600 | 18 |
| 2. | ECAT | Economics | 900 | 8 |
| 3. | GCAT | Government/Social | 1900 | 20 |
| 4. | MCAT | Markets | 1600 | 4 |

### 3.2 Semantic Significance Vector Generation

We used a vector representation which represents the significance of the data and weighs different words according to their significance for different topics. Significance Vectors [15], [16] were determined based on the frequency of a word in different semantic categories. A modification of the significance vector called the semantic vector uses normalized frequencies where each word *w* is represented with a vector $(c_1,c_2,..,c_n)$ where $c_i$ represents a certain semantic category and *n* is the total number of categories. A value $v(w, c_i)$ is calculated for each element of the semantic word vector as follows:

$$v(w, c_i) = \frac{\text{Normalized Frequency of } w \text{ in } c_i}{\sum_{k=1}^{n} \text{Normalized Frequency of } w \text{ in } c_k}$$

For each document, the document semantic vector is obtained by summing the semantic vectors for each word in the document and dividing by the total number of words in the document. Henceforth it is simply referred to as the *significance vector*.

The TMG Toolbox [18] was used to generate the term frequencies for each word in each news document. Word vectors were generated for the main and subtopic levels separately and then concatenated. The final word vector consisted of 54 columns (for 4 main topics and 50 subtopics) for the Reuters Corpus. While calculating the *significance vector* entries for each word, its occurrence in all subtopics of all main topics was taken into account. This was called the *Full Significance Vector* [14]. We also generated the *Conditional Significance Vector* [14] where a word's occurrence in all subtopics *of only a particular main topic* is taken into account while calculating the word significance vector entries.

For each document, the document significance vector was obtained by summing the significance vectors for each word in the document and dividing this sum by the total number of words in the document.

### 3.3 Data Vector Sets Generation

Three different vector representations (Full Significance Vector, Conditional Significance Vector and tf-idf) were generated for our data. The Conditional Significance Vectors were processed further to generate four main category-wise data vector sets.

#### 3.3.1 Full Significance Vector

Here, the document vectors were generated by using the full significance word vectors as explained in section 3.2. For each Reuters Full Significance document vector the first four columns, representing four main topics – CCAT, ECAT, GCAT & MCAT, were ignored leaving a vector with 50 columns representing 50 subtopics. The order of the data vectors was then randomised and divided into two sets – a training set of 9000 vectors and a test set of 1000 vectors.

#### 3.3.2 Category-based Conditional Significance Vectors

Here, the conditional significance word vectors were used to generate the document vectors. The order of the 10,000 Reuters Conditional Significance document vectors was randomised and divided into two sets – a training set of 9000 vectors and a test set of 1000 vectors. The training set was then divided into 4 sets according to the main topic labels. For each of these sets, only the relevant subtopic vector entries (e.g. C11, C12, etc for CCAT; E11, E12, etc for ECAT) for each main topic were retained. Thus, the CCAT category training data set had 18 columns for the 18 subtopics of CCAT. Similarly the ECAT training data set had 8 columns, the GCAT training data set had 20 columns and the MCAT training data set had 4 columns. These 4 training sets were then used to train the 4 separate base classifiers of the Reuters parallel classifier. The main category of a test data vector was determined by the maximum significance vector entry for the first four columns representing the four main categories. After this, the entries corresponding to the

subtopics of this predicted main topic were extracted along with the *actual* subtopic label and given to the classifier trained for this predicted main category.

For the Reuters Corpus, the accuracy of choosing the correct main topic by selecting the maximum significance level 1 entry was 96.80% for the 1000 test vectors, i.e. 968 vectors were assigned to the correct trained classifiers whereas 3.20% or 32 vectors were assigned to a wrong classifier – resulting in a wrong classification decision for all these 32 vectors. Hence the upper limit for classification accuracy was 96.80% for our parallel classifier for the Reuters Corpus.

#### 3.3.3 TF-IDF Vector generation

The tf-idf weight (Term Frequency–Inverse Document Frequency) measures how important a word is to a document in a data set. This importance increases with the number of times a word appears in the document but is reduced by the frequency of the word in the data set. Words which occur in almost all the documents have very little discriminatory power and hence are given very low weight. The TMG toolbox [18] was used to generate the tf-idf vectors for our experiments. The tf-idf vector datasets were then randomized and divided into 9000 training /1000 test vectors.

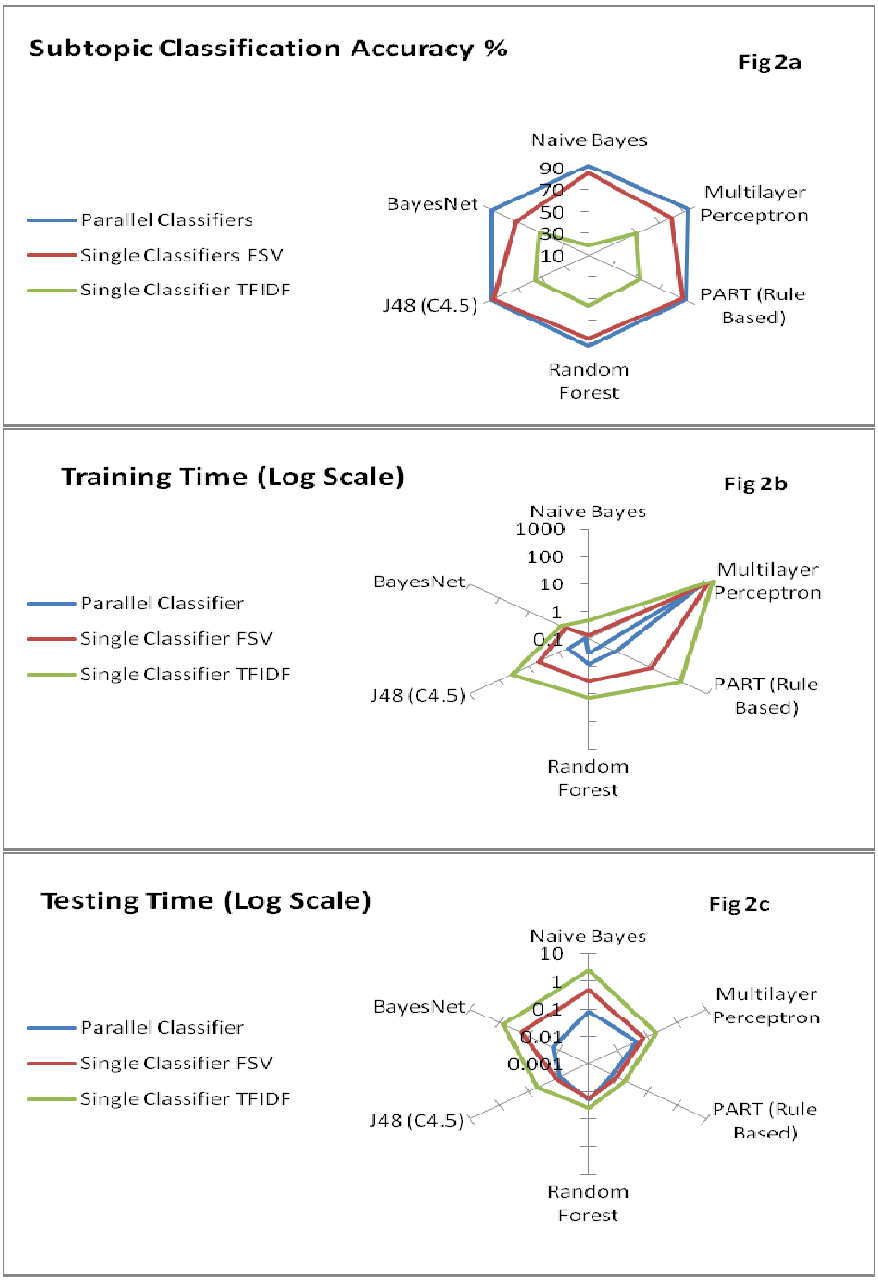
### 3.4 Classification Algorithms

Six classification algorithms were tested with our data sets namely Random Forest, J48(C4.5), the Multilayer Perceptron, Naïve Bayes, BayesNet, and PART. Random Forests [21] are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently. C4.5 [22] is an inductive tree algorithm with two pruning methods: subtree replacement and subtree raising. The Multilayer Perceptron [23] is a neural network which uses backpropagation for training. Naive Bayes [24] is the simplest form of Bayesian network, in which all attributes are independent given the value of the class variable. BayesNet [25] implements Bayes Network learning using various search algorithms and quality measures. A PART [26] decision list uses C4.5 decision trees to generate rules.

## 4 Results & Analysis

We tested our parallel classifier architecture using six different types of base classifiers. In the parallel classifier using Naïve Bayes, four different Naïve Bayes classifiers were trained on the four subspaces of the Reuters Corpus namely CCAT, ECAT, GCAT and MCAT. Similarly for the parallel classifier using Multilayer Perceptrons, four different Multilayer Perceptron classifiers were trained on the four subspaces of the Reuters Corpus and so on. The performance of each single classifier on the full data was compared with the performance of the parallel classifier combina-

**Fig. 2: Parallel Classifier Performance Metrics**



tion in which this particular classifier was used as a base classifier. For the baseline single classifier experiments, the Full Significance Vector and the tf-idf vector representations were used whereas for the parallel classifier experiments, the category-wise separated Conditional Significance Vector representation was used.

In all comparisons, it was observed that the parallel classifier combination performed better than the single basic classifier. The classification accuracy was improved (Friedman test, p=0.0025), the training times were reduced (Friedman test, p=0.0025) and the testing times were reduced (Friedman test, p=0.0094). The baseline using Full Significance Vectors (FSV) performed better than the baseline using tf-idf. Fig 2 shows the subtopic classification accuracy, training time and testing time for the parallel classifiers along with the baselines. Fig 2a shows that the maximum improvement in subtopic classification accuracy is achieved by the Naïve Bayes Classifier while the other classifiers also show a substantial improvement.

**Fig. 3: Parallel Classifier Speed-up**

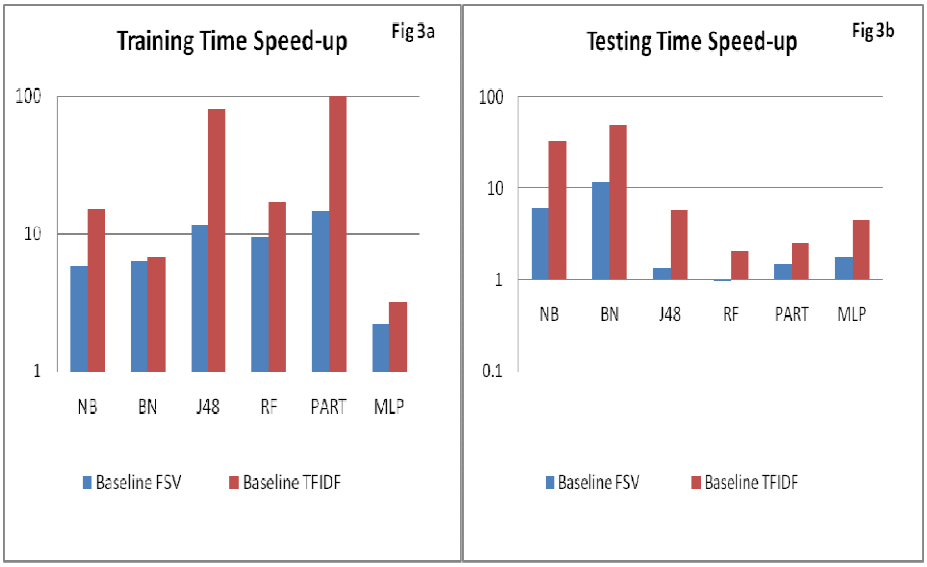
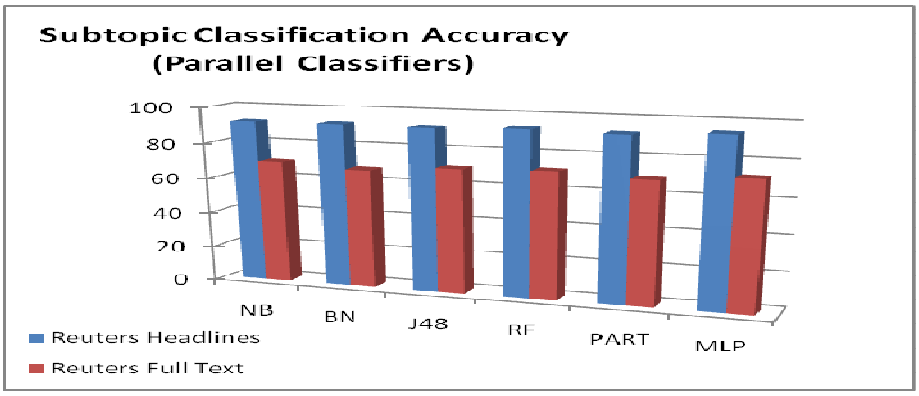



Fig 3 shows the speed-up of the parallel classifiers with respect to both baselines. Speed-up is calculated by dividing the baseline time by the corresponding parallel classifier time. The timing diagrams in Fig. 2 and the speed-up diagrams in Fig. 3 are shown on a log scale to accommodate a wide range of values. The maximum training speed-up was achieved by the rule-based classifier PART (14.4 with reference to the FSV baseline and 149 with reference to the tf-idf baseline) which was followed by the tree-based classifier J48(C4.5) at speed-up 11.76 with reference to the FSV baseline

and 79.5 with reference to the tf-idf baseline. The testing time speed-up was maximum for the Bayesian classifiers. Naïve Bayes achieved a speed-up of 6 with respect to FSV and 32.8 with respect to tf-idf while BayesNet achieved a speed-up of 11.75 and 48.75 with the corresponding baselines. Naïve Bayes achieved significant speed-up in both training and as well as testing (Train/Test speed-up of 5.8/6.0 and 15.1/32.8 for FSV and tf-idf respectively).

We also ran the parallel classifier experiments on 10,000 Reuters Full Text news items (containing headlines and body text). It was observed that the subtopic classification accuracy of Reuters news items was better with Reuters Headlines than with Reuters Full Text (Wilcoxon Signed Rank test, p=0.031). A possible explanation for this can be that the extra text present in Reuters Full Text acts as noise which degrades classifier performances. Fig 4 shows the corresponding subtopic classification accuracies.

**Fig. 4: Comparison of Reuters Headlines and Reuters Full Text**




Classifier Index:

| | |
|---|---|
| NB – Naïve Bayes | BN – BayesNet |
| J48 – C4.5 Tree | RF – Random Forest |
| PART – Rule Based Classifier | MLP – Multilayer Perceptron |

## 5 Conclusion

Our results show that combining classifiers of the same type in parallel improves the classification accuracy of the concerned basic classifier where the underlying data has distinct semantic categories. They also show that Reuters Headlines perform better than Reuters Full Text for the purpose of news categorization. These results

show further that a parallel combination of classifiers results in a very sharp reduction in training and testing times. The speed-up achieved is very significant in all cases. Naïve Bayes achieved a significant speed-up in *both* training and test timings along with the maximum improvement in classification accuracy. Since Naïve Bayes is already a fast classifier, further speedup can be put to good use especially in search technology. The experiments confirm the fact that the Maximum Significance Value is very effective in detecting the relevant subspace of a test document and that training separate classifiers on different subsets of the original data enhances overall classification accuracy and significantly reduces training/testing times.

## 6 References


1. Friedman, J.H.: On Bias, Variance, 0/1—Loss, and the Curse-of-Dimensionality. Issue 1, 1997, Data Mining and Knowledge Discovery, Vol. 1, pp. 55 – 77.
2. Parsons, L., Haque, E. and Liu, H.: Subspace Clustering for High Dimensional Data : A Review. ACM SIGKDD Explorations Newsletter. 2004, Vol. 6, Issue 1, pp. 90 – 105.
3. Varshney, K.R. and Willsky, A.S. : Learning dimensionality-reduced classifiers for information fusion. Proceedings of the 12th International Conference on Information Fusion. Vol. July 2009, pp. 1881–1888.
4. Fradkin, D. and Madigan, D.: Experiments with Random Projections for Machine Learning. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining., 2003, pp. 517-522.
5. Ho, Tin Kam. : The random subspace method for constructing decision forests. 1998. IEEE Transactions on Pattern Analysis and Machine Intelligence. Vols. Volume 20, Issue 8 (Aug 1998), pp. 832-844.
6. Yaslan, Y. and Cataltepe, Z.: Co-training with relevant random subspaces, Elsevier, 2010, Neurocomputing 73 (2010), pp 1652-1661.
7. Garcia-Pedrajas, N. and Ortiz-Boyer, D.: Boosting Random Subspace Method, Vol 21(2008), pp 1344-1362.
8. Kotsiantis, S.B.: Local Random Subspace Method for constructing multiple decision stumps. International Conference on Information and Financial Engineering, 2009. pp. 125-129.
9. Breiman, L.: Bagging predictors. Issue 2, 1996, Machine Learning, Vol 24, pp 123-140.
10. Schapire, R.E. : The boosting approach to machine learning: An overview. New York : Springer, 2003, Nonlinear Estimation and Classification. Lecture Notes in Statist., Vol. 171, Springer, New York, pp. 149-171.
11. Al-Kofahi, K., et al.: Combining multiple classifiers for text categorization. Proceedings of the tenth international conference on Information and knowledge management , CIKM 2001. pp. 97-104.

12. Ruiz, M.G. and Srinivasan, P.: Hierarchical Neural Networks for Text Categorization. SIGIR 1999.
13. Estabrooks, A. and Japkowicz, N. : A mixture-of-experts framework for text classification. Proceedings of the 2001 Workshop on Computational Natural Language Learning. Vol. 7, pp. 1-8, July 06 - 07, 2001, Toulouse, France.
14. Tripathi, N., et al.: Semantic Subspace Learning with Conditional Significance Vectors. Barcelona, July 2010 : s.n., 2010. Proceedings of the IEEE International Joint Conference on Neural Networks. Vol. July 2010, pp. 3670-3677.
15. Wermter, S., Panchev, C. and Arevian, G.: Hybrid Neural Plausibility Networks for News Agents. 1999. Proceedings of the Sixteenth National Conference on Artificial Intelligence. pp. 93-98.
16. Wermter, S.: Hybrid Connectionist Natural Language Processing : Chapman and Hall, 1995, 1995.
17. Rose, T., Stevenson, M. and Whitehead, M.: The Reuters Corpus Volume 1 - from Yesterday's News to Tomorrow's Language Resources. 2002. Proceedings of the Third International Conference on Language Resources and Evaluation (LREC- 02),2002. pp. 827–833.
18. Zeimpekis, D. and Gallopoulos, E. : TMG : A MATLAB Toolbox for Generating Term Document Matrices from Text Collections. [ed.] J. Kogan and C. Nicholas. Book Chapter in Grouping Multidimensional Data: Recent Advances in Clustering.: Springer, 2005.
19. Manning, C., Raghavan, P. and Schutze, H.: Introduction to Information Retrieval. Cambridge University Press, 2008.
20. Hall, M., et al.: The WEKA Data Mining Software: An Update. ACM SIGKDD Explorations Newsletter. July 2009, Vol. 11, Issue 1, pp. 10-18.
21. Breiman, L.: Random Forests. Issue 1, October 2001, October 2001, Machine Learning, Vol. 45, pp. pp 5-32.
22. Quinlan, J.R.: C4.5 : Programs for Machine Learning. San Mateo, CA. : Morgan Kaufmann Publishers, 1993.
23. Verma, B. : Fast training of multilayer perceptrons. 1997. IEEE Transactions on Neural Networks. Vol 8, Issue 6, Nov 1997, pp. 1314-1320.
24. Zhang, H. and Su, J. : Naive Bayes for Optimal ranking. Issue 2, June 2008, Journal of Experimental and Theoretical Artificial Intelligence, Vol 20, pp. 79-93.
25. Pernkopf, F.: Discriminative learning of Bayesian network classifiers. 2007. Proceedings of the 25th IASTED International Multi-Conference: artificial intelligence and applications. pp. 422-427.
26. Frank, E. and Witten, I.H.: Generating Accurate Rule Sets Without Global Optimization. [ed.] J. Shavlik. Machine Learning: Proceedings of the Fifteenth International Conference. : Morgan Kaufmann Publishers, 1998.