Speed Up Learning in a Test Feature Classifier Using Overlap Index Lists

Yoshikazu Matsuo
Hokkaido University
Information Science and Technology
Sapporo, Japan
matsuo@ssc.ssi.ist.hokudai.ac.jp

Takamichi Kobayashi
Nippon Steel Corporation
Futtsu, Japan
kobayashi.takamichi@nsc.co.jp

Hidenori Takauji
Muroran Institute of Technology
Robot Arena
Muroran, Japan
uji@mmm.muroran-it.ac.jp

Shun’ichi Kaneko
Hokkaido University
Information Science and Technology
Sapporo, Japan
kaneko@ssi.ist.hokudai.ac.jp

***Abstract*—This paper presents a novel low cost learning algorithm for a Test Feature Classifier using Overlap Index Lists (OILs). In general, pattern classifiers require a large amount of training data to attain high performance, which is expensive in terms of computation time. Our proposed algorithm uses OILs to efficiently find and check combinations of features starting with lower dimensions and working up-to higher ones. Our algorithm can solve classification problems in real industrial inspection lines with large reductions in computation time.**

Keywords-Test Feature Classifier; Overlap Index List; Speed Up Learning; Curse of Dimensionality.

I. INTRODUCTION

Recently, automatic tools for inspecting products have become increasingly important for develop flexible manufacturing lines. Examples can be found in visual inspection in production and precision work that is necessary for quality management. Classification is one of the most important techniques employed in automatic inspection systems. In general, classifiers require a large set of training samples for learning in order to attain a high level of performance. However, the labelling of samples, which makes supervised learning possible for a classifier, is typically, a very time-consuming task.

A current research subject in pattern recognition is reducing the cost of learning. for which several approaches have been proposed. One approach aims at reducing the size of the training data set [1] [2]. Another approach involves streamlining the learning algorithm [3] [4]. In this study, we discuss the latter approach and propose a high speed learning algorithm for a Test Feature Classifier (TFC), which is a pattern classifier. Real inspections of manufacturing-line quality have several problems including the following: 1) some results of labelling for single data are not matched, 2) the reliability of the labels must be validated, and 3) data with low reliability are not useful for learning. Research into efficient learning methods would contributes to solve these problems.

TFC’s learning process involves finding and recording PTFs which are basic sub-features, beginning with a search in lower dimensions. In this paper, we propose the method for efficient learning that exploits the fact that class overlaps in a high dimensional feature space require overlaps in lower dimensions. In addition, we propose an efficient learning technique for adding new features or dimensions. The detail of these methods shows in section 3 and 4. The verification experiments shows in section 5.

II. TEST FEATURE CLASSIFIER

Because the mathematical formalization of TFC has already been provided in [5], we briefly introduce classifiers through qualitative and semantic explanations. TFC consists of learning and discrimination procedures. In the learning procedure, a nonparametric and specific investigation divides the overall feature space into local subspaces of combinatorial features in which class overlap. Combination of features are called “test features”, (TFs) or “prime test features”, (PTFs), which are irreducible test features. In the discrimination procedure, an unclassified candidate input pattern is checked in each subspace using the corresponding PTF, and then, the pattern is classified according to the average voting scores in sub-spaces. Thus, the classifier aims to achieve high performance with a small training dataset by the partial discriminations contained in sub-spaces. Given a training dataset in which classes do not overlap, a TF is defined as any combination of features that does not classify every pattern with the selected features alone, that is, it satisfies the condition of having non-overlapping classes. In general, smaller combinations of features may allow class overlaps, but they are advantageous form the perspective of low-cost computing. Because it is provable that any combination of features that includes a TF is itself a TF,

Table I
EXSAMPLE OF TWO CLASSES PROBLEM.

		f_1	f_2	f_3	f_4
C_1	${}_1x_1$	2	5	9	7
	${}_1x_2$	5	6	10	5
	${}_1x_3$	4	7	7	6
	${}_1x_4$	5	8	8	9
	${}_1x_5$	3	9	7	6
C_2	${}_2x_1$	5	8	6	1
	${}_2x_2$	7	9	2	2
	${}_2x_3$	3	10	4	5
	${}_2x_4$	6	7	2	6
	${}_2x_5$	5	4	1	5

we should choose irreducible TFs, or PTFs, for discrimination. In [6], an efficient and effective learning method for successively adding training data by using weights for PTFs was proposed.

III. HIGH-SPEED LEARNING ALGORITHM USING OVERLAP INDEX LISTS

TFC's learning procedure involves identifying enough TFs so that the non-overlap condition is satisfied. In our proposed algorithm, the learning procedure finds NTFs, which are sub-spaces that have data with overlapping classes. The learning procedure begins by searching for TFs in low dimensional sub-spaces and adding feature dimensions. Data with overlapping classes in higher-dimensional sub-spaces will be overlapped in each lower dimensional subspace that is contained in a higher dimensional subspace. Therefore, we use Overlap Index Lists (OILs) to record classification overlaps in order to circumvent checking data that are already known to satisfy the non-overlap condition. In this algorithm, additional checking is necessary in lower dimensional sub-spaces where few data overlaps occur, and this checking provides additional overhead for the algorithm. We will briefly discuss the computational complexity of our high-speed learning algorithm using OILs.

A. Creating OILs for Single Feature Dimensions

TableI shows an example of a two class problem. It has a dataset with five data elements ${}_1x_1, {}_1x_2, \cdots, {}_2x_5$ in each class and four independent features f_1, f_2, f_3 and f_4 . The left subscripted letters indicate the class the elements belong to, whereas the right subscripted letters are indices. We will describe our high-speed learning algorithm by tracing the procedures that search PTFs. The first step of the algorithm is create OILs for one-dimensional features. In the case of f_1, ${}_1x_2, {}_1x_4, {}_2x_1$ and ${}_2x_5$ as well as ${}_1x_5$ and ${}_2x_3$ constitute an interclass overlap. Thus, feature f_1 is an NTF, and the index information below is added to the OIL $L(f_1)$.

$$\begin{aligned} L(f_1) &= \{i_1, i_2\} \\ i_1 &= \{\{2,4\},\{1,5\}\} \\ i_2 &= \{\{5\},\{3\}\} \end{aligned}$$

The set i_1 shows the interclass overlap consisting of ${}_1x_2, {}_1x_4, {}_2x_1$, and ${}_2x_5$. The set $\{2,4\}$ represents the indices of ${}_1x_2$ and ${}_1x_4$ in class 1, and the set $\{1,5\}$ represents the indices of ${}_2x_1$ and ${}_2x_5$ in class 2. The set i_2 shows the overlap involving ${}_1x_5$ and ${}_2x_3$, 5 and 3 represent the indices of ${}_1x_5$ in class 1 and ${}_2x_3$ in class 2, respectively. We create OILs for f_2, f_3 and f_4 in a similar manner, obtaining.

$$\begin{aligned} L(f_2) &= \{i_1, i_2, i_3\} \\ i_1 &= \{\{3\},\{4\}\} \\ i_2 &= \{\{4\},\{1\}\} \\ i_3 &= \{\{5\},\{2\}\} \end{aligned}$$

$$\begin{aligned} L(f_4) &= \{i_1, i_2\} \\ i_1 &= \{\{2\},\{3,5\}\} \\ i_2 &= \{\{3,5\},\{4\}\} \end{aligned}$$

There is no interclass overlap with respect to f_3. Thus, f_3 is a TF, for which an OIL is not needed.

B. Updating OILs for Higher Feature Dimensions

In the next step, we search TFs and update the OILs for two dimensions using the OILs for single dimensions. Searching all data for interclass overlaps is necessary while checking for TFs in higher dimensions. Thus, the result of checking for TFs does not depend on the order in which features are added. In the sub-feature f_1f_2, we must decide whether to add information about features f_1 to the OIL for f_2 or vice versa. To perform a more efficient check, we use the OIL with the smallest number of overlapping combination. The number of data elements that must be searched while checking for a TF is $n_1 \times n_2$, where n_1 and n_2 are the number of data elements in class 1 and class 2, respectively. Assume $s(f_1)$ is the number of data elements that must be searched. It is calculated by multiplying the number of $L(f_1)$'s i_1 and i_2 elements of class 1 by the number of elements of class 2:

$$\begin{aligned} s(f_1) &= (2 \times 2) + (1 \times 1) \\ &= 5 \end{aligned}$$

Similarly, $s(f_2) = (1 \times 1) + (1 \times 1) + (1 \times 1) = 3$. Thus, we choose to update L_2 because $s(f_2) < s(f_1)$. In the case of the feature f_1f_2, the three pair of data $({}_1x_3, {}_2x_4), ({}_1x_4, {}_2x_1)$, and $({}_1x_5, {}_2x_2)$. TableI shows that there is a class overlap with respect to these features in the case of ${}_1x_4$ and ${}_2x_1$. Thus, f_1f_2 is not a TF and the OIL is updated:

$$\begin{aligned} L(f_1f_2) &= \{i_1\} \\ i_1 &= \{\{4\},\{1\}\} \end{aligned}$$

We then check for TFs among the five sets of features $f_1f_3, f_1f_4, f_2f_3, f_2f_4$, and f_3f_4. Features f_1f_3, f_2f_3, and

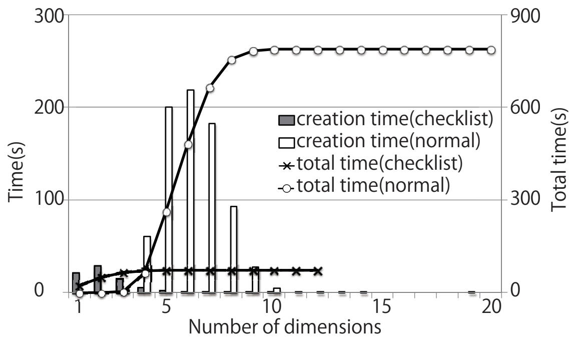

Figure 1. Comparison of time costs for searching PTFs (Inspection Dataset).

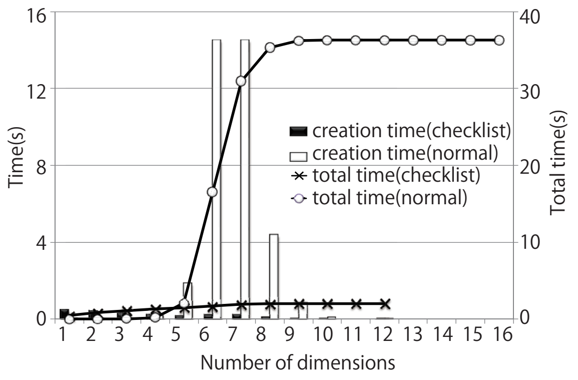

Figure 2. Comparison of time costs for searching for PTFs (Letter Dataset).

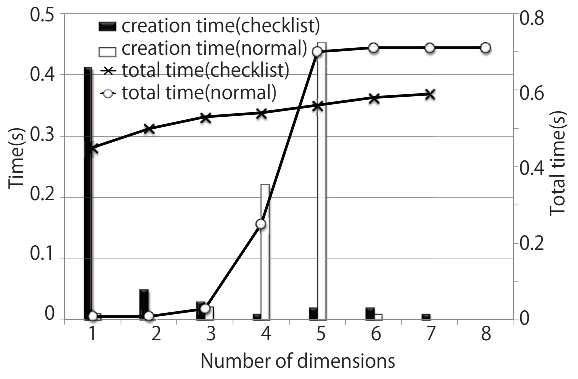

Figure 3. Comparison of time costs for searching for PTFs (Abalone Dataset).

f_3f_4 are TFs because f_3 is a TF. Thus, we only need to check the two sets of features f_1f_4 and f_2f_4. For f_1f_4, we use f_4's OIL because $s(f_1) = 5 > s(f_4) = 4$. Checking the data $({}_1x_2, {}_2x_3)$, $({}_1x_2, {}_2x_5)$, $({}_1x_3, {}_2x_4)$ and $({}_1x_5, {}_2x_4)$ from the list $L(f_4)$ revealed that there is a class overlap in the case of ${}_1x_2$ and ${}_2x_3$. Thus, f_1f_4 is an NTF and the OIL is updated with $L(f_1f_4) = \{\{2\}, \{5\}\}$. For f_2f_4, we use f_2's list because $s(f_2) < s(f4)$. Similarly, f_2f_4 is an NTF because there is an overlap involving ${}_1x_3$ and ${}_2x_4$. This completes the process of checking for TFs in two dimensions.

Our next step is to check for TFs in three dimensions. The process is almost the same as that for two dimensions. The only feature combination that we need to check is $f_1f_2f_4$. We can use any of the lists $L(f_1f_2)$, $L(f_1f_4)$, or $L(f_2f_4)$ because $s(f_1f_2) = s(f_1f_4) = s(f_2f_4)$. In this case, we will use $L(f_1f_2)$. Based on this list, we only need to check the set $({}_1x_4, {}_2x_1)$, where there is no overlap. Thus, the combination $f_1f_2f_4$ is a TF. The process of creating TFC is complete because there are no NTFs in the three dimensional case.

IV. Efficient Learning for Adding Dimensions

After defining a set of PTFs, i.e., training the initial TFC with a specified data for which feature dimensions are provided, TFC must be able to up-date or modify itself for augmented feature spaces. We propose an algorithm for modifying a set of PTFs on the basis of new features. When a new feature is presented and new subspaces are added, the original sub-spaces are included in the new set of sub-spaces, but bit vice versa. Thus, the modification procedure will only check for TFs in the new sub-spaces that are not known to be TFs on the basis of previously identified PTFs.

V. Experimental Results

A. High Speed Learning Algorithm using OILs

We used three data sets for our experiments. The first data set is used in a real industrial inspection line (Inspection Dataset). It has two classes and 20 dimensions. Class 1 and 2 contain 2033 and 3799 data elements, respectively. The second dataset is the Letter Data-set from UCI database [7]; it has two classes, class D and class P, and 16 features. Class D and P contain 805 and 803 data elements, respectively. The third dataset is the Abalone Data-set from UCI database [7] that is used for distinguishing between male and female abalones. The male and female classes contain 1528 and 1307, respectively, and there are 8 features. We used these three data sets as training data. The respective times for checking for TFs in each dimension for the three datasets are shown in Fig. 1, 2, and 3. The bar graphs show the time needed for checking for TFs in each dimension, and the line graphs show the total time for each case. These results demonstrate the efficiency of the algorithm for the Inspection and Letter Datasets. However, the method dose not show a clear advantage in the case of the Abalone Dataset because of the overhead for lower dimensions. We believe that one factor may be that the total amount of time needed to create the initial TFC was too small to allow a significant change.

A PTF is defined as a prime TF that dose not include other TFs. In TFC and sTFC, the learning procedure involves searching and recording PTFs. As an example, Fig. 4 shows the number of PTFs and TFs for the Inspection Data-set.

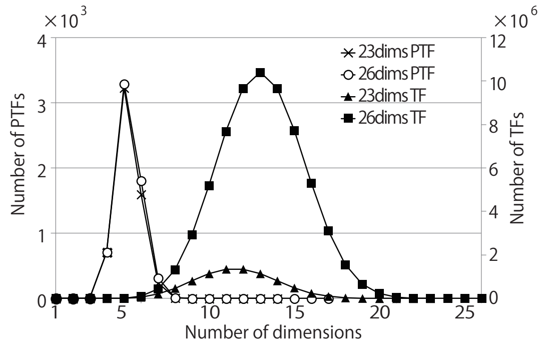

Figure 4. Number of PTFs with increasing number of feature dimensions.

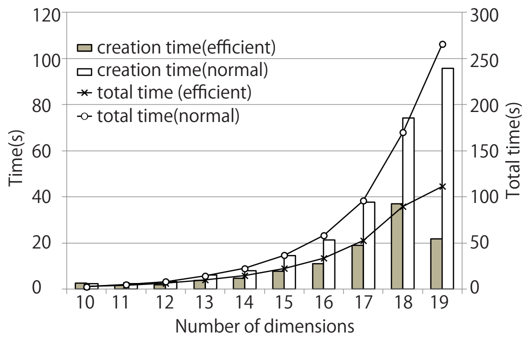

Figure 5. Comparison of time costs for adding feature dimensions (Inspection Dataset).

The figure 4 shows two cases: 23 and 26 features. The highest numbers of dimensions of PTFs in two cases are 8 and 9. We observed similar characteristics for the other two datasets, and we assume that this is a n instance of the curse of dimensionality [8] [9]. In general, higher-dimensional features need sufficient amount of data. Features with very high dimensions can negatively impact training performance. Thus, it is necessary to maintain a certain density when adding features. The highest dimensions that PTFs can have depend on the data set. If we can determine the highest dimensions in advance, TFC can learn more quickly. This will be one of our future research projects.

B. Efficient Learning for Adding Feature Dimensions

In this section, we present the result of an experiment that examined the efficiency of our learning procedure for adding feature dimensions. In the experiment, we used the Inspection Data set with 10 features as training data for the initial learning and up to 19 features were successively added. Fig. 5 shows the times needed for creating TFCs for each dimension. The bar graph shows the time needed for checking for TFs in each dimension, and the line graph shows the total time for creating TFCs in each dimension. The average time reduction for each dimension was approximately 50%.

VI. Conclusion and Future Work

We have proposed a new method for efficient learning using OILs that can eliminate unnecessary checks and efficient learning method for adding feature dimensions using preliminary information. We compared the performance of our proposed methods with that of conventional TFC methods. Our proposed methods can greatly reduce the time needed for creating TFCs.

We found that the highest number of dimensions in the created PTFs was stable, although the highest number of dimensions depends on the dataset. One topic for our future research will be determining the highest number of dimensions.

Acknowledgments

We are deeply indebted to associate Professor Tanaka, whose comments and suggestions were invaluable for this research.

References

[1] U. Bhattacharya, S. Vajda, A. Mallick, B.B. Chaudhuri, and A. Belaid.: "On the choice of training set, architecture andcombination rule of multiple mlp classifiers for multireso-lution recognition of handwritten characters" In IWFHR, pp. 419-424, 2004

[2] J. Wang, P. Neskovic and L.N. Cooper： "Training Data Selection for Support Vector Machines" LNCS, vol.3610, pp. 554-564, 2005

[3] D. Thanh-Nghi, N. Van-Hoa and P. Francois: "Speed Up SVM Algorithm for Massive Classification Tasks" Advanced Data Mining and Applications, vol.5139, pp. 147-157, 2008

[4] J.X. Dong, A. Krzyzak, and C.Y. Suen: "Fast SVM training algorithm with decomposition on very large datasets" IEEE Transaction Pattern Analysis and Machine Intelligence, vol.27, pp. 603-618, 2005.

[5] V. Lashkia, S. Kaneko and S. Aleshin： "Distance-based Test Feature Classifiers and its Applications" IEICE Trans. Inf. and Syst., vol.E83-D, no.4, pp. 904-913, 2000.

[6] Y. Sakata, S. Kaneko, Y. Takagi, H. Okuda： "Successive pattern classification based on test feature classifier and its application to defect image classification" Pattern Recognition, vol.38, no.11, pp. 1847-1856, 2005

[7] P.M. Murphy and D.W. Aha: "UCI Repository of Machine Learning Databases" University of California-Irvine, 1994(anoymous http://archive.ics.uci.edu/ml/machine-learning-databases/letter-recognition/) [retrieved: May, 2012].

[8] Bellman R.E.: "Adaptive Control Processes" Princeton University Press, Princeton, NJ. 1961

[9] Brown. M, Bossley.K.M, Mills.D.J, Harris.C.J: "High dimensional neurofuzzy systems: overcoming the curse of dimensionality" International Joint Conference of the Fourth IEEE International Conference on Fuzzy Systems, vol.4, pp. 2139 - 2146, 1995