1. Introduction

1.1. Motivation of feature selection in video classification

With the growth of applications in multimedia technology, video data has become a fundamental resource for modern databases. The problem of efficient video data management is an important issue [1].

One typical approach is to use existing image retrieval algorithms, starting from a good segmentation of the video into shots and then selecting certain images of the shots as key-frames [2]. But as Lew et. al. mentioned in [3], the gap between the high level query from the human and the low-level features persists because of a lack of a good understanding of the "meanings" of the video, of the "meaning" of a query, and of the way a result can incorporate the user's knowledge, personal preferences, and emotional tone. Machine learning methods such as classification [4] and boosting [5] are introduced in order to find features that associate image properties with user labels. But due to the high volume of video data, the time complexity of these methods has been prohibitively high. Researchers therefore have worked on speeding up their algorithms; one way has been by seeking efficient ways of reducing the dimensionality of the data prior to processing.

Vailaya [6] and Smeulders [7], among others, discuss this problem from the view of image processing and computer vision. They assume that some features, such as color histograms or texture energies, are more sensitive than others, based on the researchers' intuition. They provide theoretical analysis and empirical validations for their choices. However, there appears to be little work that supports efficient feature selection in video classification without some prior human specification of the underlying feature sets.

The heart of this proposal is a series of novel feature selection algorithms, which focuses on selecting representative features in the massive and complex dataset automatically. We do not pre-select features, as the relation between features and concepts in video data is unclear, even perhaps to the user. Instead, we induce their relationship, and find those subsets of features that most capture the trained semantic categories given by the user. This form of learning has received significant attention in the AI literature recently and has been applied to moderately large data sets in applications. Learning research is not often carried out in video indexing and retrieval--although Lew et. al. [10] used a feature selection method to refine features for stereo image matching.

1.2. Definition of feature selection

Feature selection focuses on seeking a feature subset that has the most discriminative information from the original feature space. The object of feature selection is three-fold: improving the prediction performance of the predictors, providing faster and more cost-effective predictors, and providing a better understanding of the underlying process that generated the data; see [11].

Although a precise mathematical statement of the feature selection problem is not widely agreed upon, there appears to be two major approaches to define feature selection problem. The first emphasizes the discovery of any relevant relationship between the features and the concept, whereas the second explicitly seeks a feature subset that minimizes prediction error of the concept [12]. The first is referred to as a filter method, and the second approach is referred to as a wrapper method. In general, wrapper methods attempt to optimize directly the classifier performance so that they can perform better than filter algorithms, but they require more computation time.

1.3. Feature selection algorithms design and evaluation

Feature selection methods are typically designed and evaluated with respect to the accuracy and cost of their three components: their search algorithm, their statistical relationship method (in the case of filter methods) or their induction algorithm (in the case of wrapper methods), and their evaluation metric (which is simply prediction error in the case of wrapper methods).

We illustrate these definitions in Figure 1. The original feature space has N features. The target feature space is a subset of original feature space, including k features selected from N features, k is the number between 1 and N. Since the number of possible feature subsets is the power set of N, search algorithm focus on how to search the feature subsets space to get the target feature subset as soon as possible. In wrapper model, feature subsets selected by search algorithm will pass a classifier to train and test on this classifier. This classifier is designed for evaluating the performance of selected feature subset, so we call it induction algorithm. The classification result from wrapper model is compared with the correct label of the data in the evaluation stage. Based on the prediction error, we will decide how to search next or stop search. Ordinarily, filter methods use simple statistics computed from the empirical feature distribution to select strongly relevant features and to filter out weakly relevant features before induction occurs.

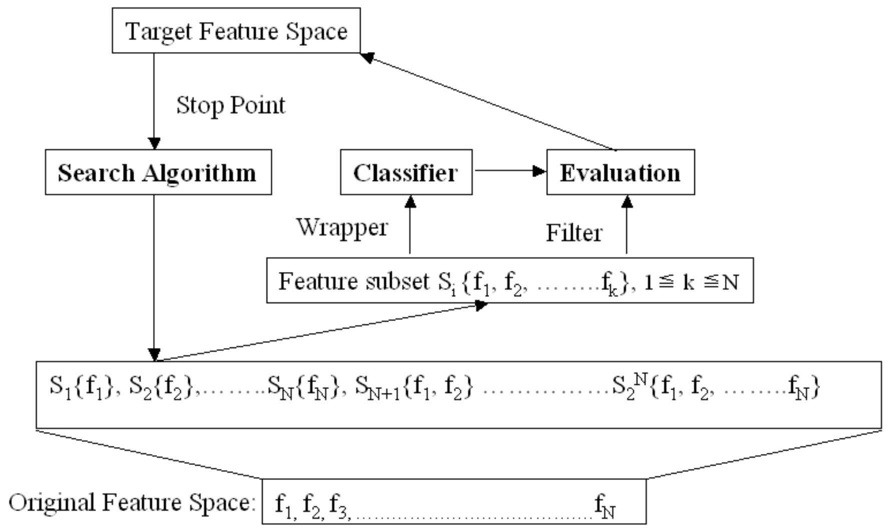

Target Feature Space
Stop Point
Search Algorithm
Classifier
Evaluation
Wrapper
Filter
Feature subset $S_i\{f_1, f_2, \ldots f_k\}$, $1 \leqq k \leqq N$
$S_1\{f_1\}$, $S_2\{f_2\}$, ……$S_N\{f_N\}$, $S_{N+1}\{f_1, f_2\}$ ……$S_{2^N}\{f_1, f_2, \ldots f_N\}$
Original Feature Space:
$f_1, f_2, f_3, \ldots f_N$

Figure 1. Three components of feature selection algorithm

1.3.1. Search algorithm of feature selection methods

The dominating cost of any method, however, is that of the search algorithm, since feature selection is fundamentally a question of choosing one specific subset of features from the power set of features. This is an exponentially hard problem, and intractable if the set of features is very large as it is with image data. A more realistic design is to look for an approximate search algorithm that achieves high performance; this is necessarily a heuristic approach.

So far, three general kinds of heuristic search algorithms have been used: forward selection (FS), backward elimination (BE), and genetic algorithms (GA). Forward selection [13] starts with the empty set of features and successively adds individual features, usually following a variant of a greedy algorithm, terminating when no improvement is possible. However, it can't remove any features, and therefore ends up making what amounts to local optimizations to the growing set. Backward elimination [14], which does the reverse, starts with the full set of features and heuristically subtracts individual features. It suffers from a similar problem of local optimization, as removal of a feature is irrevocable. A genetic algorithm [15], which permits both the addition and deletion of features to a surviving population of evolving subsets of limited cardinality, is more likely to seek a global optimum. But it is computationally costly, and requires a more elaborate definition of algorithm convergence.

1.3.2. Induction algorithm of feature selection methods

As mentioned above, the induction algorithm is actually a classifier. So common classification algorithms all can be used as induction algorithms, such as Bayes network in [13], Neural Network in [15], k-nearest neighbor and k-nearest unlike neighbor in [16], SVM in [17] and boosting algorithm in [18].

An algorithm that focuses on seeking a feature subset that is most efficient for a certain kind of classier is a called classifier-specific feature selection, such as [19]. In contrast, Abe et. al. proposed a classifier-independent feature selection algorithm based on non-parametric discriminate analysis in [16].

1.3.3. Evaluation metric of feature selection methods

For a wrapper method, prediction error rate, which is a very simple evaluation metric, is often used to evaluate the performance of feature subset. For a filter method, feature selection is mainly done by a different evaluation metric. A distance measure, such as Euclidean distance, is used in [20]. A dependence measure, such as correlation coefficient, evaluates the dependence relation between features or between features and concepts. A consistency measure is an inconsistency rate over the data set for a given feature set. Two patterns are considered inconsistent if they match all features but have different class labels; see [21]. An information measure, such as information gain, conditional information gain, mutual information, is the main metrics for filter methods [14].

1.4. Applications of feature selection

Although many efficient feature selection techniques have been proposed, applying them to applications adequately is another problem. It is related with the size of the feature space, the data type and data range of each feature, and the accuracy of certain classifier and complexity cost. Feature selection also has been applied to text categorization [8], gene micro-array [9], web mining [22], handwritten recognition [15] successfully.

In the video area, massive data, high dimensionality, and complex hypotheses present a similar need for feature selection compared with other applications. Moreover, video applications have a higher requirement of time cost than others. Unrealistic amount of computer time is one important obstacle to apply current feature selection algorithm to video classification. Jaimes and Chang apply a feature selection algorithm to baseball video retrieval and get the better performance [23].

2. Research progress

In this section, four feature selection algorithms will be introduced based on different video classification applications. The low time and space costs of our novel feature selection approaches allow us the practical implementation of fast video retrieval system. We show the accuracy improvement and efficiency improvement in the experimental part by comparing the result of our novel method against an imperfect but feasible method, random feature selection; see the work of Xing [9]. These experiments use the same data and same classifiers; the only difference is how the feature subset was chosen.

2.1. Basic Sort-Merge Tree (BSMT)

BSMT, proposed by Liu and Kender in [24], combines the advantages of forward selection, backward elimination, and genetic algorithms. To avoid irrevocable adding or subtracting, it always operates on some representation of the original feature space, so that at each step every feature has an opportunity to impact the selection. To avoid heuristic randomness, at each step a greedy algorithm is used to govern subset formation. Further, the recursive nature of our method enables the straightforward creation of a hierarchical family of feature subsets with little additional wrapper method.

2.1.1. Search algorithm of BSMT

BSMT can be divided into two parts: the creation of a tree of feature subsets, and the manipulation of the tree to create a feature subset of desired cardinality or accuracy. Each part uses a heuristic greedy method.

```
Initialize level = 0
        Create N singleton feature subsets.
While level < $\log_2 N$
        Induce on every feature subset.
        Sort subsets based on performance.
        Combine, pairwise, feature subsets.
```

Table 1. Setup the Basic Sort-Merge Tree

Table 1 shows the Sort-Merge feature selection basic algorithm. The method is straightforward. Initially, there are N singleton feature subsets. Their performance is evaluated on training data, and they are sorted in order of performance. Then, N/2 subsets of cardinality 2 are formed by merging, pair-wise and in order, the sorted singleton feature sets. After another round of training and sorting, a third level of N/4 subsets of cardinality 4 are formed, and the process continues until it attains a level or condition prespecified by the user. Figure 2 illustrates the algorithm with an initial set of features with cardinality N = 256. Table 2 shows the related algorithm to select exactly r features from the hierarchy of feature subsets.

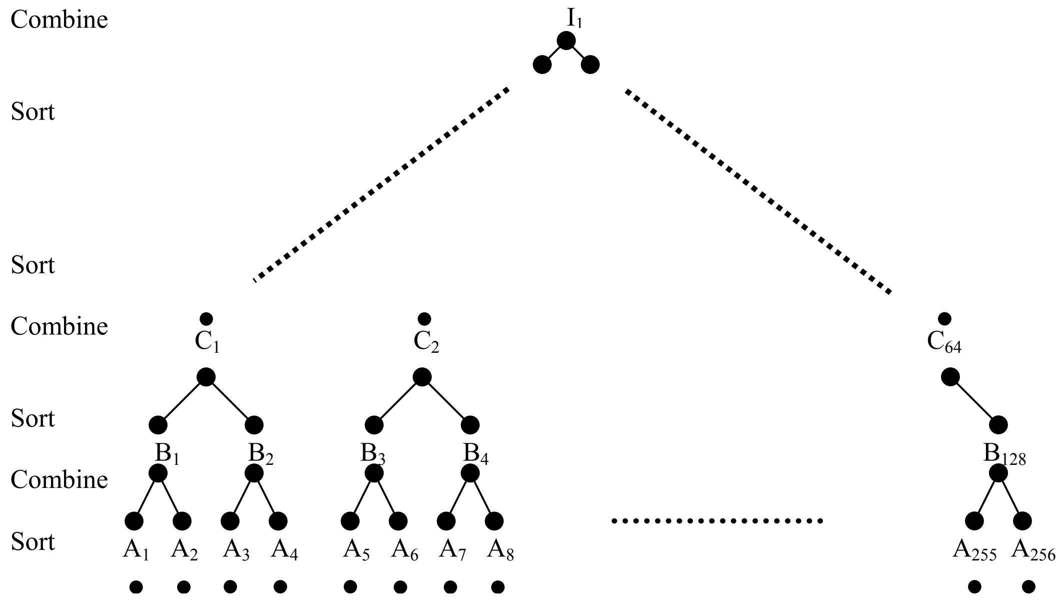


Figure 2. The Basic Sort-Merge Tree. Leaves correspond to singleton feature subsets. Interior notes are formed by the pair-wise merge of neighboring feature subsets that have been sorted according to their classification accuracy.

```
Select the leftmost branch of size $2^{\lceil \log 2r \rceil}$.
Initialize cutout = $2^{\lceil \log 2\ r \rceil}$ - r.
While cutout >0
    Let branch-size = $2^{\lfloor \log 2\ cutout \rfloor}$.
    For all remaining branches of this size, evaluate the induction
        result of removing those branches individually.
    Remove the branch with best result.
    Let cutout = cutout – branch-size.
```

Table 2. Algorithm to select exactly r features from the tree of feature subsets

2.1.2. Induction algorithm of BSMT

The performance of a wrapper feature selection algorithm not only depends on the search method, but also on the induction algorithm. For our induction method during the course of the learning we use a novel, low-cost, and scalable combination of Fastmap for dimensionality reduction with Mahalanobis maximum likelihood for classification.

The Fastmap method proposed in [25] approximates Principal Component Analysis (PCA), with only linear cost in the number of reduced dimensions sought, c, and in the number of features, N. The method heuristically replaces the computation of the PCA eigenvector of greatest eigenvalue, which represents the direction in the full feature space that has maximum variation, with a (linear) search for the two data elements that are maximally separated in the space. The vector between these two elements is taken as a substitute for the eigenvector of greatest eigenvalue, and the full space is then projected onto the subspace orthogonal to this substitute vector for the first eigen dimension. The process then repeats for a desired and usually small number of times. By the use of clever bookkeeping techniques, each additional new dimension and projection takes time approximately linear in the number of features.

In brief, as defined in statistical texts Duda et. al.. [26], or in the documentation of Matlab, the Mahalanobis distance computes the likelihood that a point belongs to a distribution that is modeled as a multidimensional Gaussian with arbitrary covariance. During training, each image frame in a training set for a video category is first mapped to a point in the space of reduced dimension c. Then the distribution of these mapped points is approximated by a c-dimensional Gaussian with a non-diagonal covariance matrix. Multiple categories and training sets are represented each with their own Gaussian distribution. The classification of a test image frame is obtained by mapping it, too, into the reduced c-dimensional space, and then calculating the most likely distribution to which it belongs. That is, the classification label assigned to it is the label of the training set center to which it has the minimum Mahalanobis distance.

2.1.3. Time cost analysis of BSMT

We now analyze the time cost of BSMT, which we show is linear in the number of features N. Using Fastmap-Mahalanobis, the induction step is also linear in the size of the training data, m. Therefore the method is well-suited to the high data volumes necessary in video retrieval applications.

We use the following definitions:
N: Number of dimensions of the original feature space
r: Number of dimensions of the reduced feature space
m: Cardinality of the training data set
c: Number of dimensions extracted using the Fastmap algorithm
l: level number of Sort-Merge feature selection tree
Tm: Time of induction using m training data in the Mahalanobis classifier
T_{basic}: Time of the basic Sort-Merge feature selection algorithm

We first show that $T_{basic} = O(NT_m) = O(Nmc^2)$. The cost of each level is proportional to: the number of subsets at that level, the cost of reducing the dimensionality of each subset, the cost of classifying the training data, the cost of evaluating the classifier on test data, and the cost of the final sort. (The merge cost is trivially linear in the number of subsets.) The number of subsets is $N/(2^l)$. The cost of the Fastmap per subset is O(mc), based on the proof given in [32], and the cost of the Mahalanobis classification is $O(mc^2)$, based on the proof given in [31]. Thus, the cost of the induction is a fixed $Tm = O(mc^2)$. The final sort is again dependent on the number of subsets at that level, and is $O((N/2^l)\log(N/2^l))$. The cost at a given level is therefore $O((N/2^l)mc^2) + O((N/2^l)\log(N/2^l))$. Given that the size of the training data generally must dominate the size of the feature set, it certainly dominates its logarithm, and therefore the cost at a level is $O((N/2^l)mc^2)$. Summing these costs up the levels of the tree yields a total cost of $O(Nmc^2)$. It is not hard to show in a similar manner that the space cost is also linear in N.

2.1.4. Apply feature selection algorithm to video classification

First, in our application and in general, the video may be down-sampled temporally, spatially, and/or spectrally. We temporally subsample by using only every other I frame (that is, one I frame per second). We spatially subsample by a factor of 16 in each direction by using only using the DC terms of each macro-block of the I frame (consisting of six terms, one from each block: four luminance DC terms and two chrominance DC terms); this subsampling is very popular in video retrieval [1]. This gives us, for each second of video, 300 macroblocks (15 by 20) of 6 bytes (4 plus 2) of data: 1800 initial features. For convenience of decoding, we consider the 6 DC terms from the same macro-block to be an un-decomposable vector, so our initial data consists more accurately of 300 six-dimensional features per second of video.

Second, one of feature selection algorithms such as BSMT is used as search algorithm to select feature subset. Fastmap is used to reduce the dimensionality of each feature subset to a pre-specified small number, c, of dimensions. Third, for each feature subset at this level, using the reduced dimensionality representation, the training sets of the video train the induction algorithm to classify the test sets of the video. Fourth, the feature subsets are sorted by accuracy. Pair-wise feature subsets are then merged. Fifth, the process repeats again, starting at the Fastmap step. Sixth, if needed, exactly r features are extracted from the tree of the feature subsets. Seventh, the frames of the learned category are classified from the video only using these r features.

2.1.5. Experimental result of BSMT

2.1.5.1. Test bed of BSMT

The first task is to classify one extended instructional video mentioned above, of 75 minutes duration, which has about 134,010 frames in MPEG-1 format, each with 240 by 320 pixels into four categories as illustrated in Figure 3: handwriting, announcement, demo, and discussion. For training data, we used 400 I-frames distributed over the video and across these four classes.

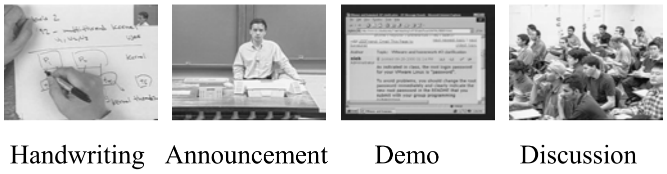
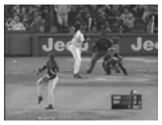


Figure 3. Instructional video frame classification

The second task is applying BSMT to the application of binary classification of sports video. As shown in Figure 4, we are interested in defining "pitching frames" in baseball video that look like Figure 4 (a); the rest of the video has many different competing image types, some of which are shown in Figure 4 (b). The data is sampled somewhat more finely, with every I-frame of extracted as one data frame, giving 3600 frames for the half-hour.



(a) **(b)**

Figure 4. Retrieve "pitching"(a) from an entire video with competing image types(b)

2.1.5.2. Accuracy improvement of BSMT in instructional video

Figure 5 compares the frame categorization error rate of instructional video using 30 macro-blocks. For random feature selection, we ran 100 experiments in which 30 features were selected randomly and calculated the mean of error rate, where "error rate" is defined as the pure ratio (not percentage) of misclassified frames to total frames. The classification error rate of the sort-merge method is not only less than that of random selection, but also appears to be very stable as the Fastmap dimension varies. As discussed later, this is expected to be a critical consideration for retrieval system designers.

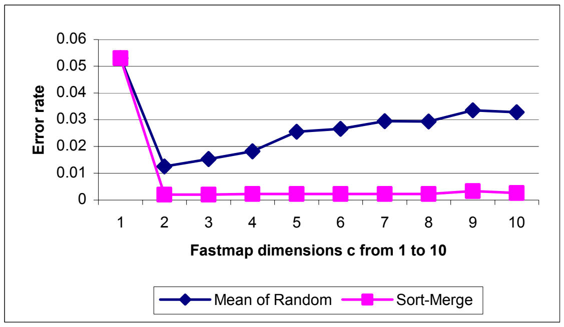


Figure 5. Instructional video frame classification error rate with same feature subset size (r=30) and different Fastmap dimensions c (from 1 to 10)

2.1.5.3.Accuracy improvement of BSMT in sports video

For baseball video retrieval, we ran two experiments. The first did not use any prior temporal segmentation or other pre-processing. Figure 6 shows the result, which fixes the Fastmap dimension c=2 and compares the classification error rate of random feature selection and Sort-Merge feature selection of different size of selected feature subsets. In general, the error rate is halved.

In the second experiment, the video has been segmented to 182 segments (roughly, "shots", except that commercials are considered one segment). We attempt to retrieve the 45 "pitching" segments. As mentioned by Lin and Hauptman in [27], simple accuracy is often an insufficient measure, so we compare the feature selection algorithms based on recall and precision in Figure 7. Precision is nearly perfect, and recall is better by a factor of 2, compared to random feature selection.

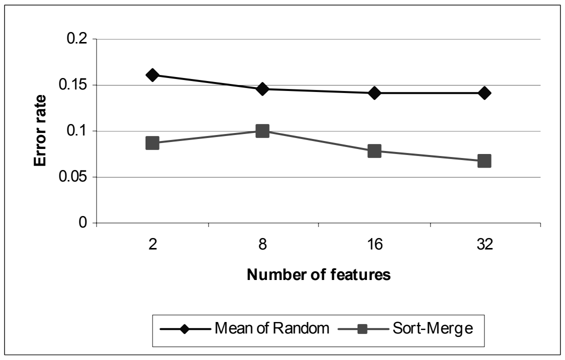


Figure 6. Baseball video retrieval error rate using with different feature subset sizes and same Fastmap dimension (c=2)

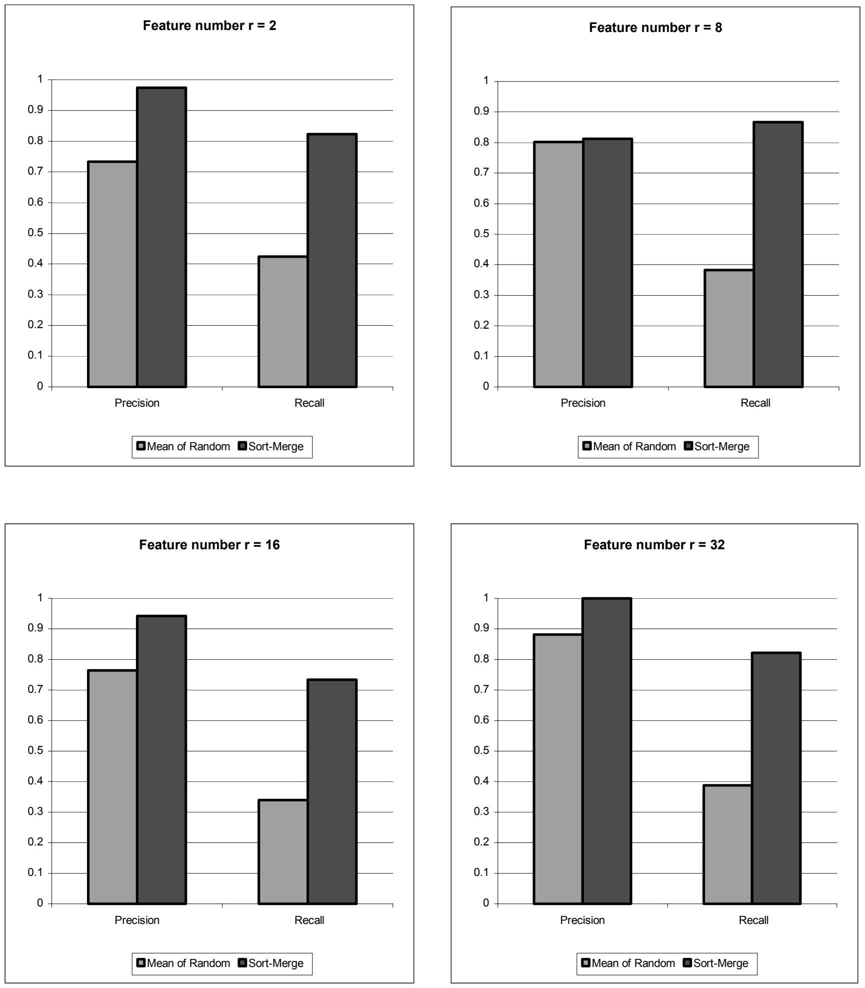


Figure 7. Baseball video retrieval Precision and recall using with different feature subset sizes.

2.2. Complement Sort-Merge Tree (CSMT)

2.2.1. Feature selection under sparse training data

As Xing et. al. mentioned in [9], the number of “replicates” (i.e. ground truth) in some experiments is often severely limited in genetic microarray applications; he gives a real world problem in which only 38 observation vectors exist, each one encoding the expression levels of 7130 features. Feature selection when there are so few observations on so many features is very different from the more general cases typical in the learning literature; it even renders some powerful algorithms ineffective. This is easy to see in Xing’s case: since there are only 38 observations, no matter what feature set has been chosen the prediction error is severely quantized to one of 39 levels. With 7130 features, on average we could expect about 183 features to produce each of these error levels; either a forward or backward wrapper method will be forced to choose randomly over this large set at each iteration. If ultimately we wish only a small set of about 50 features to avoid over-learning, too much randomness is introduced by these methods. Moreover, randomness accumulates, with the chose of each feature heavily influencing the choice of its successors.

The alternatives to wrapper methods are filter methods, which select feature subset independently of the actual induction algorithm. Koller and Sahami in [14] employ a cross-entropy measure, designed to find Markov blankets of features using a backward greedy algorithm. In theory going backward from the full set of features may capture interesting features more easily, especially under sparse training data. However, in Xing’s case this means that if we want a target feature space with 50 features, we have to remove 7080. To avoid this expensive time cost, Xing proposes to sort the 7130 features based on their individual information gain in classification, then uses only the best N features in a series of filter methods. Additionally, he selects N=360 manually. It is not clear how well such a technique generalizes or how effective it is, given its mixture of models.

2.2.2. Search algorithm of CSMT

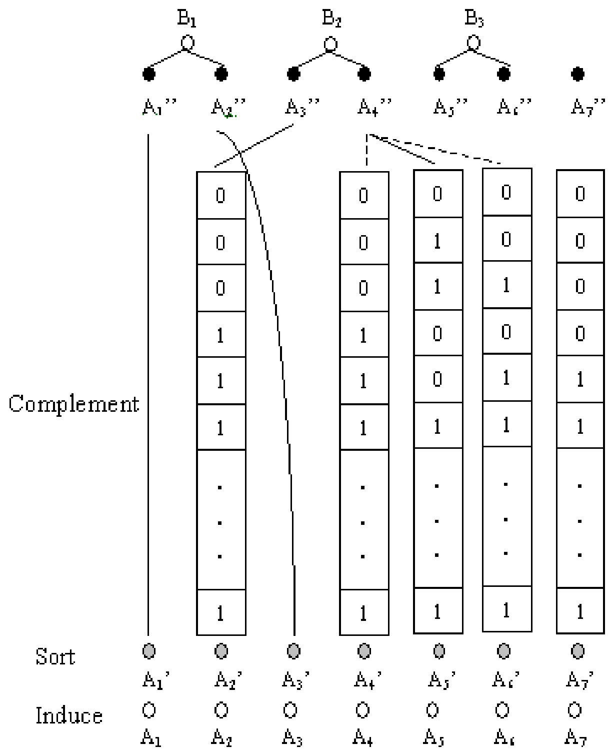
Our overall approach in [28] is to use an outer wrapper model for high accuracy, and an inner filter method for resolving the problem of random selection when the training set is relatively small and errors are quantized (as they inevitably are for video data).

Similar with BSMT, the CSMT algorithm also can be divided into two parts: the creation of the full tree of feature subsets, and subsequent manipulation of the tree (if necessary) to create a feature subset of desired cardinality or accuracy. The different between CSMT and BSMT is how to pair-wise the two feature subset. Instead of combine two neighbor feature subsets in the rank list in BSMT, CSMT choose the pair of feature subsets based on complement requirement.

Table 3 shows the CSMT basic algorithm. Figure 8 illustrates the complement test, which uses a filter method to inform the otherwise random selection of feature subsets. It employs a heuristic approximation to a Markov Blanket that attempts to maximize classification performance on the m training samples. An m-length performance vector records for each feature subset correct classifications with a 1 and failures with a 0. Any feature subset seeking a complementary feature subset will examine all unpaired feature subsets sharing identical error rates with it. It then selects from these that feature subset which maximizes the number of 1s in the OR of their two performance vectors. These complementary feature subsets are then merged. This step of the CSMT method is a greedy algorithm, but one that is more informed than random choice.

```
Initialize level = 0
        Create N singleton feature subsets.
While level < $\log_2 N$
        Induce on every feature subset.
        Sort subsets based on their classification accuracy.
        Choose pairs of feature subsets based on the complement requirement.
        Merge to new feature subsets.
```

Table 3. Complement Sort-Merge Tree basic algorithm



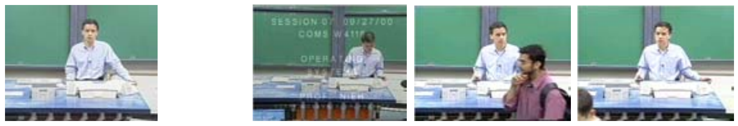
B1
B2
B3
A1'' A2'' A3'' A4'' A5'' A6'' A7''
Complement
Sort
A1' A2' A3' A4' A5' A6' A7'
Induce
A1 A2 A3 A4 A5 A6 A7

Figure 8. The complement requirement, illustrated for the A level of Figure 2. White nodes are unsorted feature subsets and gray nodes are the white nodes rank-ordered by performance. Black nodes are the pairwise mergers of gray nodes, with pairs formed under the complement requirement. The sorted singletons A_1' and A_3' have already been paired to form pair B_1. To find the best complementary feature subset for A_2', examine all sorted subsets (A_4,' A_5', A_6') with the same error rate on the m training samples. The bitwise OR of performance vectors of A_2' and A_5' maximizes performance coverage; A_5' complements A_2' for B_2.

2.2.3. Experimental result of CSMT

2.2.3.1. Test bed of CSMT

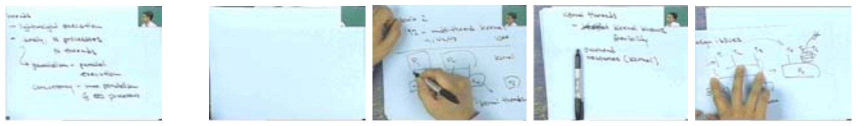
In the first experiment, we attempt to retrieve from the 4500 frames "announcement" frames, without any prior temporal segmentation or other pre-processing using the same instructional video discussed above and the same down-sampling method in section 2.1.5. Only 80 training frames are provided, and as shown in Figure 9, they include considerable noise.



Normal data Noise data

Figure 9. Training data also includes noise.

A second experiment learns and retrieves a more subtle and semantically-defined class. It is based on the intuition that some of handwriting frames in the video are probably of more use to the student observing them if they are accompanied by the instructor explaining what has just been written. These frames of "emphasis" are characterized by the lack of hand or pen regions obscuring the hand-drawn words and diagrams; such frames are essentially pre-formatted "clean" slides, as shown in Figure 20, although their content may vary greatly. A lengthy video segment of one hour of instructor handwriting was hand segmented into 69 teaching segments, where a segment was defined by a new sheet of paper, or by a lengthy verbal digression without visual activity. We want to find the "emphasis" frames within the 3600 frames comprising the hour. Only 200 training frames, taken from 10 different example sequences, were available for training.



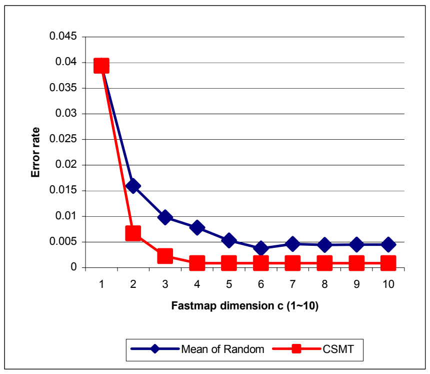
Emphasis part Non-emphasis part

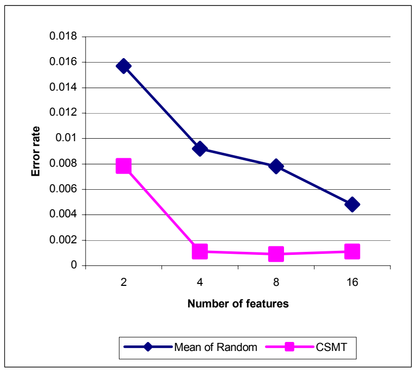
Figure 10. Examples of "emphasis" and "non-emphasis" data, with otherwise similar visual content.

2.2.3.2.Accuracy improvement of CSMT in instructional video

For the first experiment, Figure 11 (a) compares the performance of different Fastmap dimensions from 1 to 10 using the same number of features. Figure 11 (b) fixes the Fastmap dimension c=4 and compares the classification error rate of different numbers. The performance of CSMT is much better than that of random selection in all cases.

Figure 12 shows the error rate in retrieving "emphasis" frames using 16 features selected randomly; these experiments were repeated 100 times. The error bars are drawn at the mean error plus one standard deviation. In contract, asterisks show the error rate of retrieval using 16 features selected by the CSMT algorithm. The error rate of CSMT is much lower even on this more semantically defined and conceptually useful set, and it appears to be very stable as the Fastmap dimension varies. This stability is critical, as c must be fixed before hand.





(a) **(b)**

Figure 11. Instructional video frame classification error rate
(a): Same feature subset size (r=8) and different Fastmap dimensions c (from 1 to 10)
(b): different feature subset sizes and same Fastmap dimension (c=4)

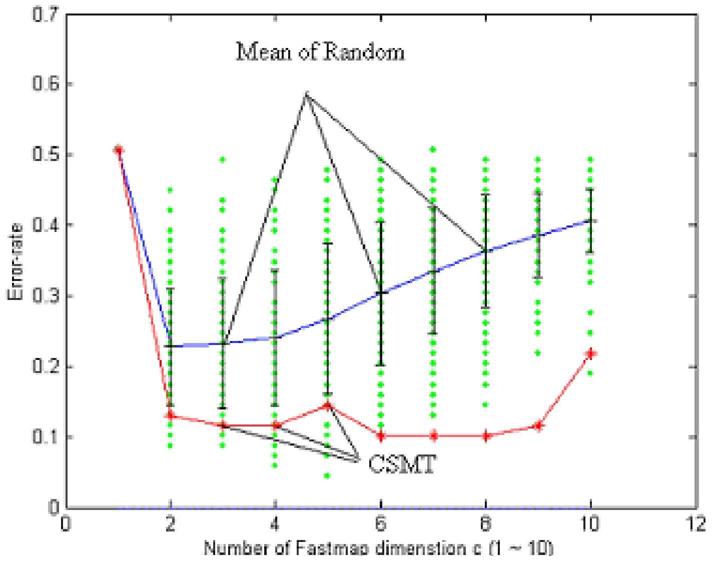


Figure 12 Instructional video frame classification error rate with same feature subset size (r=16) and different Fastmap dimensions c (from 1 to 10)

2.3. Fast-converging Sort-Merge Tree (FSMT)

2.3.1. Search algorithm of FSMT

As discussed above, one advantage of BSMT is that it naturally creates near-optimal feature subsets of any or all desired cardinalities or accuracies, with little additional work. However, if what is desired is a feature subset with a specific fixed cardinality, this may no longer be helpful. For example, if the original feature space has 256 dimensions as shown in Figure 2, and all that we desire is a feature subset with r=16 features, the induction and sorting of feature subsets with 32 features, 64 features and 128 features may be wasteful. This is particularly so since most of the quality features tend to lie in the left-most branches, and it would be better to do early pruning of the earlier merges to eliminate the less useful right-most branches.

As shown above, the times of induction dominates the total time cost of BSMT feature selection. Therefore, we provide a Fast-converging Sort-Merge Tree (FSMT) in [29] to reduce the time complexity of BSMT dynamically, by setting up selected parts of the feature selection tree, based on several heuristic parameters: the number of selected features r, a vector of tree-trimming convergence rates for each level V = (v1, v2, …v⌈ log2r ⌉+1), and an evaluation metric G. Table 4 shows the algorithm for setting up the FSMT. Figure 16 illustrates a tree with N=1800, r=16, a constant convergence rate of 0.5 above the lowest level, and information gain evaluation metric G. The number of inductions is only 682 using FSMT,The number of inductions is only 682 using FSMT, compared with 4095 using BSMT, an improvement of about 80%.

```
Initialize level = 0
        N singleton feature subsets.
        Calculate **R**: number of features of each level based on **V**
        and r.
While level <  $\log_2 r$ +1
        Induce on every feature subset.
        Sort subsets based on G(**C**, **F**).
        Prune the level based on **R**.
        Combine, pair-wise, feature subsets form those remaining.
```

Table 4. Fast-converging Sort-Merge Tree basic algorithm

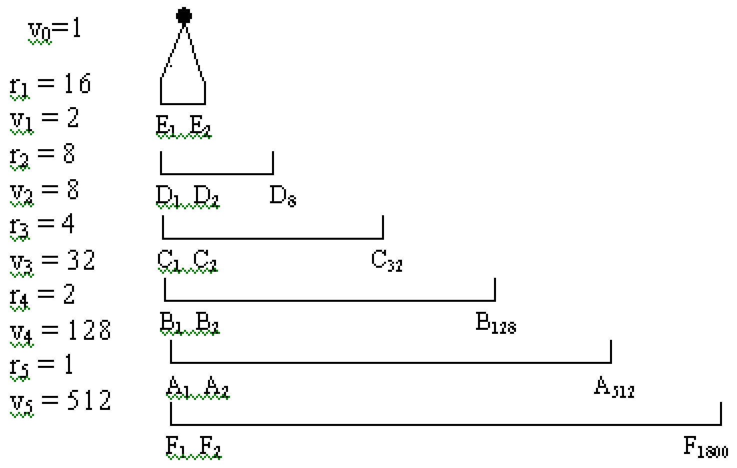


Figure 13. FSMT for N=1800,r=16, constant convergence of 0.5, and levels j (except the bottom level), from 5 down to 1. r_j is the size of the feature subset at level j, and v_j is the number of such features subsets.

Although FSMT also produces a hierarchy of feature subsets with increasing cardinality as one nears the root, in contrast to BSMT the root of the Fast-converging Sort-Merge Tree consists not of one feature subset with all N features, but of one feature subset with only the requested r features. FSMT does not therefore attempt to select feature subsets in full generality of cardinality, but is optimized for a specific user request. To do so, the vector V controls the convergence speed of FSMT, the rate at which each successive level of feature subsets is pruned. Here, this rate is constant and each successive level is a fixed fraction of the one below it (although other convergence rates are also possible). As shown in Figure 13, the root of FSMT is the feature subset with r features. The next level from the root will prune r1 = r / v1 features based on evaluation metric G. Define R as the number of features of each level in Fast-converging Sort-Merge Tree, we can calculate R by R = r / V.

Additionally, instead of simply using the prediction error of the resulting classifier as an evaluation metric of feature subset quality as in BSMT, a method that results in many ties, we use instead the information gain of the resulting classifier of the feature subsets at each level, a method that breaks ties in a useful way (although other tie-breakers are also possible). This metric, G(C, F), calculates the reduction of entropy in classifying C categories using the feature subset F. As shown in experiment part, this tie-breaker performs better than simple prediction error. In a sense, this is a further incorporation of a filter method technique within our overall wrapper scheme; information gain is commonly used in filter methods.

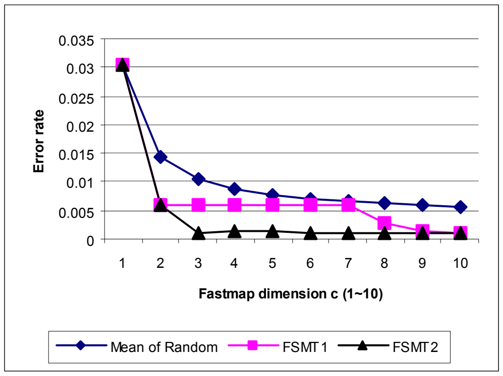
2.3.2. Experimental result of FSMT

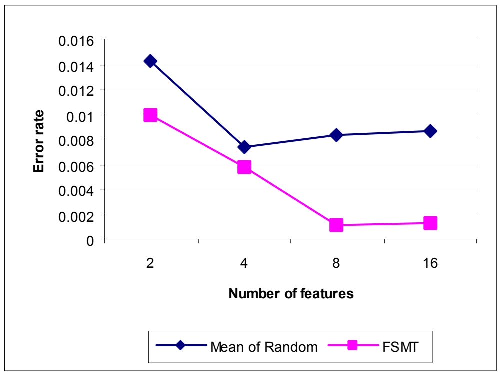
2.3.2.1. Test bed of FSMT

We apply the FSMT algorithm to retrieve "Announcement" frames in instructional video and "Pitching" segments in sports video. The first task use the same test bed with the first task in 2.2.1 and the second task uses the same test bed with the second task in section 2.1.5. The difference between these two experiments and the prior experiments is the original feature space is 1800-dimensional instead of 300 six-dimensional features.

2.3.2.2. Accuracy improvement of FSMT in instructional video

Figure 14 (a) compares the video retrieval error rate using 16 features' subset from 1800-dimensional original feature space. The difference of FSMT1 and FSMT2 is only in the evaluation metric and tie-breaker: FSMT1 uses error rate only (as does BSMT), while FSMT2 uses information gain. We show that FSMT2 results in a nearly perfect classification, with only a 0.001 error rate.





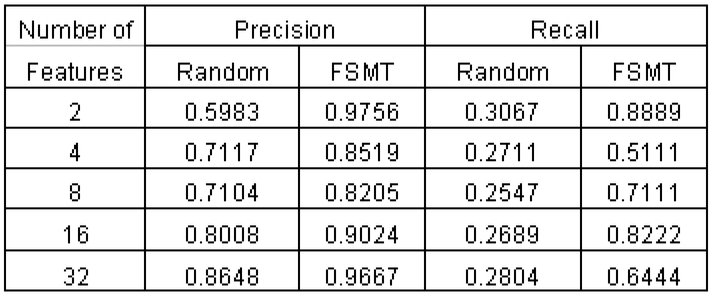
(a) **(b)**

Figure 14. Instructional video retrieval error rate
(a): Same feature subset size (r=16) and different Fastmap dimensions c (from 1 to 10)
(b): Different feature subset sizes and same Fastmap dimension (c=4)

In moving to FSMT, we sacrificed some of the flexibility of BSMT. We note that BSMT enables the straightforward creation of near-optimal feature subsets of all cardinalities or accuracies, with little additional work. Although FSMT is an application-driven algorithm directed at forming a single subset of cardinality r, it nevertheless still retains some of the advantages of BSMT, namely, that further subsets of the FSMT-selected subset are highly useful. In Figure 14 (b), a test fixes the Fastmap dimension at c=4, and compares the retrieval error rate of different values of r from the 16 features' subset. We conclude that although FSMT is not as good as BSMT if one demands fully flexible r, it is still quite powerful in selecting good features.

2.3.2.3. Efficiency improvement of FSMT in sports video

In Table 5, we compare the performance of retrieval using features selected by random selection and FSMT with information gain as tie-break with different number of r. FSMT performs much better than random feature selection, especially under sparse features in sports video retrieval. For example, only using a feature subset with 2 features from the 1800-dimensional original feature space, the precision is near perfect and the recall is near 0.9.



Number of	Precision		Recall	
Features	Random	FSMT	Random	FSMT
2	0.5983	0.9756	0.3067	0.8889
4	0.7117	0.8519	0.2711	0.5111
8	0.7104	0.8205	0.2547	0.7111
16	0.8008	0.9024	0.2689	0.8222
32	0.8648	0.9667	0.2804	0.6444

Table 5. Baseball video retrieval Precision and recall using with different feature subset sizes and same Fastmap dimension (c=2)

2.4. Fast video retrieval system

Our novel feature selection algorithms in [24] [28] [29], which can exploit several properties unique to video data to induce appropriate but small feature sets, therefore leads to new feasible approaches for rapid video segment boundary refinement and for lazy evaluation of on-line queries. Although the method is transparent to these particular video preprocessing transformations, we illustrate feature selection applications using the instructional video discussed in 2.1.5.

2.4.1. Multi-Level Feature Selection algorithm (MLFS)

We now show how the feature subset hierarchy can be exploited to efficiently refine the boundaries of contiguous video segments with differing classification labels. The hierarchy enables less work to be done on the segment interiors, and permits a multi-level refinement strategy using more accurate but more costly feature subsets at segment edges. This method was proposed by Liu and Kender in [30].

To illustrate, we select the best 2-feature subset from the 300 features using BSMT, and classify each frame of the video into different categories. This is shown as the uppermost line in Figure 15 as C_2, C_1, C_3, etc. The classification tends to have more errors at segment transitions, whether they are abrupt (cuts) or gradual (fades and dissolves) (see Koprinska and Carrato in [31]). So we devise a multi-level (coarse-to-fine) strategy to more carefully investigate the video wherever a neighborhood of frames shows a lack of consistency of labeling. Note that this will occasionally occur even within the interior of a well-defined segment.

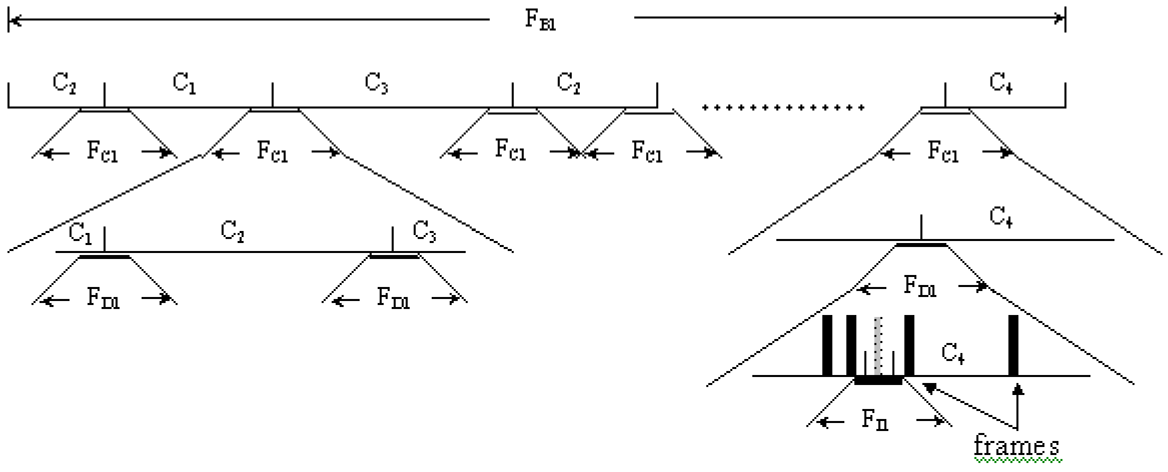


Figure 15. Video segment boundary refinement by Multi-Level Feature Selection (MLFS)

This strategy is governed by several parameters, which vary depending on the number of the successive iterations of refinement. We therefore define a feature subset size R_i, which increases with i and therefore increases the classification accuracy, and a neighborhood parameter L_i, which remains constant or decreases with i and therefore focuses the attention of the more costly classifier. Further, we define a decision threshold S_i, according to:

$$S_i = \Pr(C_j) - \sum \Pr(C_k) \qquad k = 1, 2 \ldots n \text{ and } k \neq j$$

where $\Pr(C_j)$ is the maximum Mahalanobis likelihood among all categories using this feature subset. This threshold ensures that classification is correct and unambiguous.
Figure 15 illustrates three typical cases. Most of the refinements result in the first case: a clarification of the location of the boundary developed by the initial classification of frames. However, in a second case, shown at the transition between C_1 and C_3, it is possible that an intervening segment of a completely different label is refined such as C_2. In the third case, refinement is forced to proceed to full use of all available features in order to resolve the labeling of an individual frame sufficiently confidently: this frame is often the exact center of a dissolve between two classes.

2.4.2. Lazy evaluation of unanticipated queries

This application allows the dynamic extension of video retrieval indices, which is proposed by Liu and Kender in [32]. An off-line part of the application classifies video segments into categories that users are often interested in, and constructs a main index with text tags used for retrieval. As shown in Figure 16, a user then inputs a textual query which is first matched with the main textual index, then with any dynamically created sub-index or aide-index, which are described below. If all miss, the lazy evaluation method has not found anything in its cache, and on-line computation is necessary.

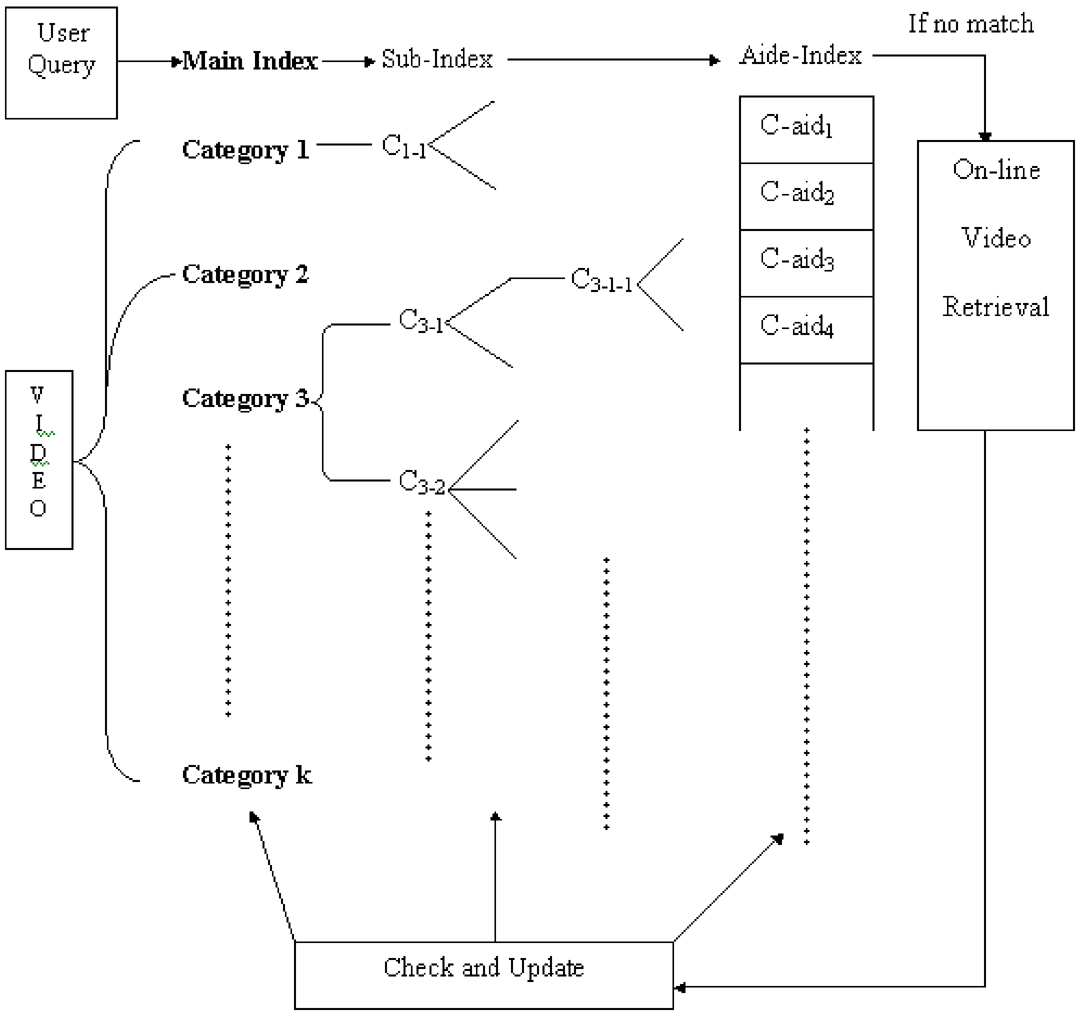


Figure 16. Lazy evaluation of unanticipated on-line queries

In the on-line evaluation and retrieval part, the user provides a short training video clip as a positive example of the frames of his textual query, together with a negative example clip. Using the multi-level feature selection algorithm, a feature subset is progressively sought that discriminates the two. Then, other clips are retrieved from the video, which have been labeled as being in the same category as the training clip, and the user is asked for iterative feedback. If necessary, the feature subset is progressively increased further until discrimination is satisfactory to the user. The resultant text query and its successful feature subset are then stored appropriately in the following way. If the clips match an existing labeled set of clips in the index, then the new label is stored as a synonym in the main index. If they are instead a proper subset of some main index clips, then both the text and the feature subset are stored in the sub-index. If the concept sought is neither a synonym nor a specialization, its text and feature subset are stored in a simple aide-index list (or a more elaborate data structure). The speed of the multi-level feature selection algorithms enables such lazy evaluations, and the indexing system becomes self-adaptive.

2.4.3. Experimental result of MLFS

In this part, we use the same instructional video and same down-sapling methods as section 2.1.5 and illustrate video segmentation using MLFS. Table 6 summarizes the results of the application of the method detailed in section 2.4.1 to the entire instructional video. The method begins by selecting the best 2-feature subset (R_1 = 2) for classification. Using it to classify all the frames of the video (fraction of video examined = 1), there remain 27 video segments that contain frames that did not attain the unambiguous level of likelihood determined by the value of S_i. These frame numbers are listed in the first column, where R_1 = 2; the value of c is also given for reference, and is discussed below.

We now proceed through four additional rounds of refinement. We keep the neighborhood of examination constant at L_i = 6, meaning that the 3 frames before and after any ambiguous frame are also re-examined and reclassified with the more costly but more accurate classifiers that use the larger feature subsets. For example, since frame 109 did not attain the required level of likelihood, we will examine frames 106 to 112 using the classifier with four features (R_2 = 4); likewise, since frames 2880 to 2887 were all determined to be ambiguous, we will examine frames 2877 to 2890 with the same classifier (R_2 = 4).

The table shows that even though the neighborhood adds 6 frames for each suspect frame or frame range, the first round of refinement at R_2 = 4 has re-examined only 7% of the video. The column labeled R_2 = 4 now lists the frame numbers that again failed to meet the likelihood threshold; this is a more refined threshold determined from the properties of the more refined 4-feature classifier. As a comparison of this column with its predecessor indicates, there are several different possible outcomes to this refinement. Some frame ranges are partially resolved, as in clip 5, where the range of ambiguous frames is reduced from 22 to only 1. Some frames remain ambiguous, as in clip 1. But sometimes the entire expanded neighborhood fails to meet the more stringent likelihood test, as in clip 3; this forces upward the fraction of the video that must be examined in the next round, as shown at the bottom of the R_3 = 8 column. The fourth outcome, the one most desired, appears at the next level of refinement at R_3 = 8, where all frames in many clips are classified with the required level of certainty.

In this experiment, we terminate the process at R_5 = 32, where we attain a classification error rate of 0.002. We stop here for comparison reasons, as we already know that this error rate is equivalent to the error rate attained by applying the more expensive 30-feature BSMT classifier of Section 2.1.5 above to the full video. However, the accumulated work of this boundary refinement approach has been much less, as the bulk of the processing has been done with simpler classifiers; on average, only 3.6 features are used per frame.

Clip	R_1=2, c=9	R_2=4, c=7	R_3=8, c=4	R_4=16, c=4	R_5=32, c=3
1	109	109	109	109	109
2	212	212	212	212	212
3	240	237-243	234-240	240	240
4	251	251	251	251	251
5	1389-1410	1410	1408-1411	1410	1407-1410
6	1532-1533	1532-1536	X	X	X
7	2566-2567	2563-2567	X	X	X
8	2571-2572	2571-2572	X	X	X
9	2577-2578	2577-2578	X	X	X
10	2630-2632	2630-2632	2629-2632	2629-2632	X
11	2763-2764	2762-2764	2762-2763	2763-2764	X
12	2880-2887	2880-2890	X	X	X
13	2895-2904	2892-2905	X	X	X
14	2942-2944	2942-2944	X	X	X
15	3103-3116	3103-3119	X	X	X
16	3138-3141	3138-3144	X	X	X
17	3165-3166	3163-3169	3164-3169	X	X
18	3174-3175	3171-3178	3170-3180	X	X
19	3184-3190	3181-3190	3181-3186	X	X
20	3249-3250	3249-3250	X	X	X
21	3271-3275	3268-3275	X	X	X
22	3287-3289	3287-3289	X	X	X
23	3304-3305	3301-3308	X	X	X
24	3366-3369	3364-3372	X	X	X
25	3380-3389	3377-3392	X	X	X
26	3401-3402	3398-3405	X	X	X
27	3408-3410	3406-3410	X	X	X
Fraction of video examined	**1**	**0.631**	**0.714**	**0.231**	**0.119**

Table 6. Segmentation of video clips in a coarse-fine manner using multi-level feature selection algorithm. At iteration i, R_i = size of feature subset, c = Fastmap dimension. Frames with uncertain classifications are indicated in the column of the level that failed to resolve them.

It is clear that the selection of the Fastmap dimension c decreases with increasing refinement level. Partly this is due to the very small size of the initial feature subsets, R1 = 2 and 4, representing features spaces with 12 and 24 components, respectively (as each feature records a macro-block's vector of 4 intensity and 2 chrominance DC terms). For feature subsets this small, Fastmap does show increased performance with increasing c. However, for feature subsets of more moderate size, the choice c becomes less significant, as one of the notable advantages of the BSMT is its empirically observed performance stability over a range of values of c. Nevertheless, the boundary refinement method is still cost-effective compared to a full application of a more elaborate classifier, even if the boundary refinement begins with a 2-feature subset at c = 9, and the full classifier uses a 30-feature subset at c = 2, since the cost of a single classification is $O(Nc+c^2)$, that is, it is dominated by N.

2.5. Summary of research progress

As shown in Figure 17, we propose a low time cost feature selection algorithm called BSMT [24] that is well-adaptive to video data [34]. The linear time cost of BSMT allows us the practical implementation of video frame categorization. MLFS, based on the hierarchical structure of BSMT, permits a multi-level refinement strategy using more accurate but more costly feature subsets at segment edges [30]. Addressing the problem of sparse and noisy training data in video retrieval, we provide CSMT, which combines the virtues of a wrapper model approach for better accuracy with those of a filter method approach for incrementally deriving the appropriate features quickly [28]. FSMT is able to speedup BSMT further by pruning the feature subset tree, based on the user's final requirement for a small, manageable subset of features [29]. We also presented a fast video retrieval system with these feature selection algorithms [32].

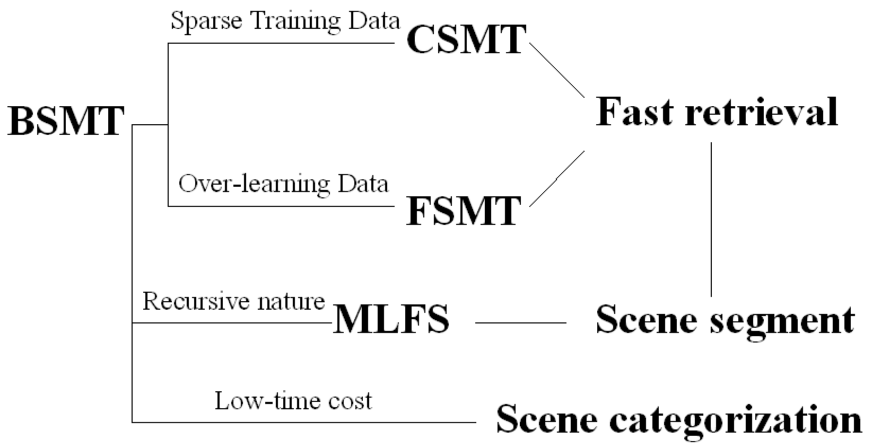


Figure 17. Summary of prior work

3. Proposed work

The issues to be explored in this research topic are multi-fold. We propose the further work from two aspects: how to improve the feature selection algorithms and how to apply feature selection algorithms to video applications better.

3.1. Improve current feature selection algorithm

As mentioned above, several search algorithms of feature selection methods have been provided successfully for video classification. BSMT, CSMT and FSMT all search the original feature space based on a sorting prediction error. One problem is that on the bottom of the Sort-Merge Tree, prediction error is calculated based on the classification result only using one feature. If no individual feature has distinguished discriminative information for classification, induction and sorting of such a weak feature subset is useless. Setting up the bottom of the Sort-Merge Tree more efficiently may increase the accuracy and efficiency of video classification further.

For the induction algorithm, we use a novel combination of low cost feature extraction method Fastmap together with the Mahalanobis likelihood, both well-suited to video data. But what happens when we choose other classifiers, such as kNN, Bayes, SVM, NN? We are especially interested in SVM, which is a very powerful classifier for binary classification and sparse training data. But the time cost of SVM is quadratic in the number of training data. HMM, which is often used in video classification and video clustering, will also be tested as an induction algorithm because of its good performance in temporal analysis.

Since we use wrapper methods mainly in our feature selection algorithm, prediction error rate is used as evaluation metric. We discussed how to combine a wrapper search method and a filter evaluation metric together to reach better performance in FSMT. We have already demonstrated in the FSMT method that filter evaluation metrics can be effectively used with a wrapper method.

3.2. Algorithm Evaluation

Although we propose and compare some feature selection algorithms in the prior sections, it is not enough to answer the questions just like which algorithm is better than others or which algorithm is better in video classification. Since there is no agreed upon metric to evaluate feature selection algorithms, we will explore this issue.

3.2.1. Accuracy evaluation of feature selection algorithms

The first question to be answered is: does the selected feature subset have a similar or better performance than the original feature space? Error rate, which is used as evaluation metric in wrapper method, also can be used here:

$$\text{Error Rate} = \frac{mis-classifiedData}{AllData}$$

When the number of data in different class is not same, error rate can only express the global performance. Balanced Error Rate (BER), which is used as main performance measure in Feature Selection Challenge 2004; see [33], considers the performance of each class. It can be calculated from confusion matrix below, where a, b, c and d represent the number of examples falling into each possible outcome:

Prediction

	Class -1	Class +1
Class -1	a	b
Class +1	c	d

In this case, BER = 0.5*(b/(a+b)+c/(c+d)).

Furthermore, Received Operating Characteristic Curve (ROC curve), a general method to evaluate the classifier also can be used to evaluate the sensitivity and specificity of the selected feature subset to certain class. We can define Hit rate and False alarm rate for Class +1:

$$\text{Hit rate} = \frac{d}{c+d} \qquad \text{False alarm rate} = \frac{a}{a+b}$$

As shown in Figure 18, the ROC curve is a two-dimension drawing relating different Hit rates and False alarm rates at the same time. Area Under Curve (AUC) is defined as the area under the ROC curve.

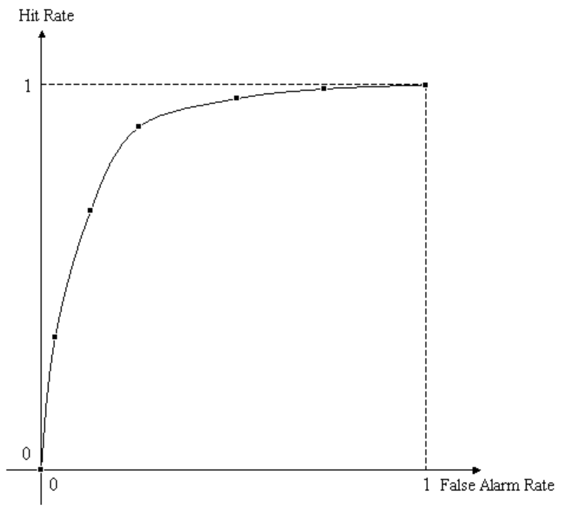


Figure 18. ROC curve and AUC

Precision and recall are similar definitions used in video retrieval often. Precision is the percentage of items classified as positive that actually are positive and recall is the percentage of positives that are classified as positive. Based on the case discussed above, they can be calculated as:

$$\text{Precision} = \frac{d}{b+d} \qquad \text{Recall} = \frac{d}{c+d}$$

In general case, we are interested in improving precision above a certain recall baseline.

F-measure is the harmonic average of precision and recall. It can be calculated by:

$$\text{F-Measure} = \frac{2 * \mathrm{Re}\,call * \mathrm{Pr}\,ecision}{\mathrm{Re}\,call + \mathrm{Pr}\,ecision}$$

There are two methods for averaging the F-measure over a collection of 2-class classification problems. One is the macro-averaged F-measure, which is the traditional arithmetic mean of the F-measure computed for each problem. Another is the micro-averaged F-measure, which is an averaged weighted by the class distribution. The former gives an equal weight to each problem and the later gives an equal weight to each data.

3.2.2. Efficiency evaluation of feature selection algorithms

3.2.2.1. Efficiency of selected feature subset

Efficiency of selected feature subset is often evaluated by Fraction of Features (FF), which is the ratio of the number of features used by the classifier to the total number of features in the dataset.

Best size of the feature subset is another interesting topic. Few papers provide the solution of this problem while the most ignore it. Xing in [9] estimates the possible best size of feature subsets, and tests them one by one using cross-validation. Beside the concern of high time cost, two things should be considered too. To evaluate prediction error, should we choose the base size with the lowest error rate? If we have 0.1 error rate with 1000 features and 0.11 error rate with 10 features, what is the best size? If the training data is not enough and also they have 'high quality', maybe we can reach a zero error rate only using very small fraction of features: but should we keep more feature to avoid overfitting? How many features should be added? What kind of features should be added? Should they be selected based on the same evaluation metric as the feature selection algorithm or on a new method based on the selected features?

3.2.2.2. Efficiency of feature selection algorithm

Efficiency of feature selection algorithm is often evaluated by time cost. The time cost of a search algorithm dominates the efficiency of feature selection algorithm. For a wrapper method, the time cost of the induction algorithm is also an important component. For example, if one classifier is powerful and has a high time cost, maybe we can only use it only in classification, and choose an alternative method, which may be not as powerful as the original one, but has a similar theoretical foundation but low time cost as the induction algorithm

The stopping point is another concern of efficiency of feature selection algorithm. In certain cases, it may be similar to the problem of choosing the best size of the features. But it may also be a totally different problem. For example, for a genetic search algorithm, it depends on how many generations are defined, and has nothing to do with the size of the feature subsets.

3.2.3. Dependent relation of feature selection algorithms

Forman et. al. present an empirical comparison of twelve filter feature selection methods in text categorization in [37]. The data comes from 299 text classification problem instances that were gathered from Reuters, TREC, OHSUMED, etc. Error rate, precision, recall and macro-average F-measure are used as performance measures. Compared with a variety of classifiers, including Naïve Bayes, C4.5, logistic regression and Linear SVM, Linear SVM was selected because of its outstanding performance. In contrast, Yang used two statistically different classifiers to reduce the possibility of classifier bias in the result: one is kNN, which is a non-parametric and non-linear classifier and another one is LLSF, based on a linear parametric model in [8].

Accuracy, precision, and recall are often used as the metrics to evaluate feature selection algorithm in video classification now. Different classifiers and different number of features are used to show the robustness of the feature selection algorithm. Since current work in evaluation is based on these general evaluation methods in video classification, we will look into additional evaluation strategies to show the three domains needed for a comparison of feature selection performance, which relates to (1) the choice of search algorithm, (2) the choice of induction algorithm and (3) the choice of the evaluation metric. We will explore most of the design space shown in Figure 19.

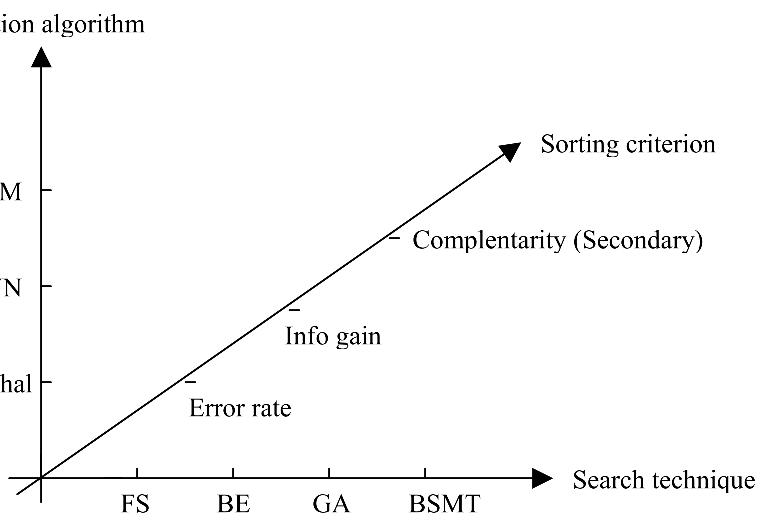


Figure 19. Comparison different feature selection algorithm

3.3. New applications

In our prior work, several feature selection algorithms have been successfully applied to the video classification problem. The original feature space we currently used is the DC term of each macro-block. It works well for instructional video and sports video for simple classification, but may be useless for more complex video clips and other semantic classification requirement. The first question is: should we start with a different original feature space? In current video classification applications, several kinds of features are often used:

- Low level features such as color, shape, edge, texture, and motion
- Audio information such as MFCC
- Objects such as face and text
- Closed-caption text
- Global features such as DCT parameters of the whole frame or other extracted features from low-level features
- Feature spaces transformed from several different kinds of features

Several feature space we are especially interested in. Some researchers work on how to use different kinds of features to get better performance. Two main methods are used: first is called decision fusion and the second is feature fusion. Decision fusion uses separate feature set to classify, and then combines their classification results. Feature fusion puts different kinds of features in one feature space and makes a single decision. How to put different features together, how to choose a classifier under this case, and how to select proper feature subset are all interesting topics.

Another feature space we are interested in consists of high-level features, such as objects, some semantic concepts, and some temporal-spatial information. For example, in scalable video coding, Wang et. al. select different coding methods based on different kinds of videos in [35]. The information in one Group Of Picture is considered as one single data so that temporal information is included. How to select features in such a dataset is an interesting problem; it also can be applied to content-based video compression and on-the-fly search.

Feature selection in video clustering is a similar problem for video classification. The main difference is that video classification is supervised learning, while video clustering is unsupervised learning. Xie et. al. use a forward wrapper method to select features and filter method to remove redundant ones in video clustering in [36]. They get good performance using three or two features from a total nine features. Our interest is in video clustering in a much large feature space.

3.4. Size of training data

Size of training dataset is an open issue in classification problem. We explore this problem from several aspects

- Sparse training data
- Massive training data
- Non-balanced training data
- One class training

In video classification, sparse and noisy training data is the general case. As mentioned above, we have made some progress in prior work, such as CSMT. We will: continue the work in two ways. The first way is seeking a better feature selection algorithm, such as finding a search algorithm that can avoid the quantization of prediction error, choosing an induction algorithm that is not sensitive to sparse training data, and selecting an evaluation metric that distinguishes the performance of feature subsets more accurately. Another way is how to make use of limited training data efficiently in cross-validation. In general, each training data is evaluated once in cross-validation in wrapper feature selection method. We can't increase the number of training data, but maybe we can increase the number of permutations of the training data and test data.

Feature selection with massive training data is the inverse problem. Some induction algorithms such as SVM, which are sensitive to the number of the training data, will increase the time cost further during the cross-validation. Randomly selecting one subset from the original training dataset for cross-validation is a general method. Actually, it is based on two assumptions:

- Feature subset performance stability: Let N be the number of training data. If we divide N into k groups and k is a reasonable number, any feature subset that has a better performance on the N data should also have better performance on N/k data, relative to the performance of other subsets.
- Training set independence: Let there be two training data subsets Train_1 and Train_2, each with N/k different training data. If one feature subset has better performance in Train_1, it will also have better performance in Train_2.

These assumptions appear reasonable, and they have been verified on some simple experiments that we have tested. More evidence or theoretical analysis is needed to support sampling training data in feature selection. For example, Liu et. al. are not satisfied with random data selection, and wants to find the training data subset that can replace the whole training dataset best [20]. They split the training data into a certain number of buckets based on data variance, and select one instance from each bucket. Likewise, we are interested in how to extract a representative data subset for feature selection.

Another related problem is feature selection in 'non-balanced' training data. For example, in feature selection for binary classification, we may have sparse training data for one label, and massive training data for another label. This is a general case in video retrieval in a large database but seldom discussed. For example, we are interested in one kind of video clip but we may have only a few positive clips. But all the others we are not interested in can be considered as the negative clips. The solution of feature selection in 'non-balance' may face more difficulties than solving the sparse training data or massive training data cases individually.

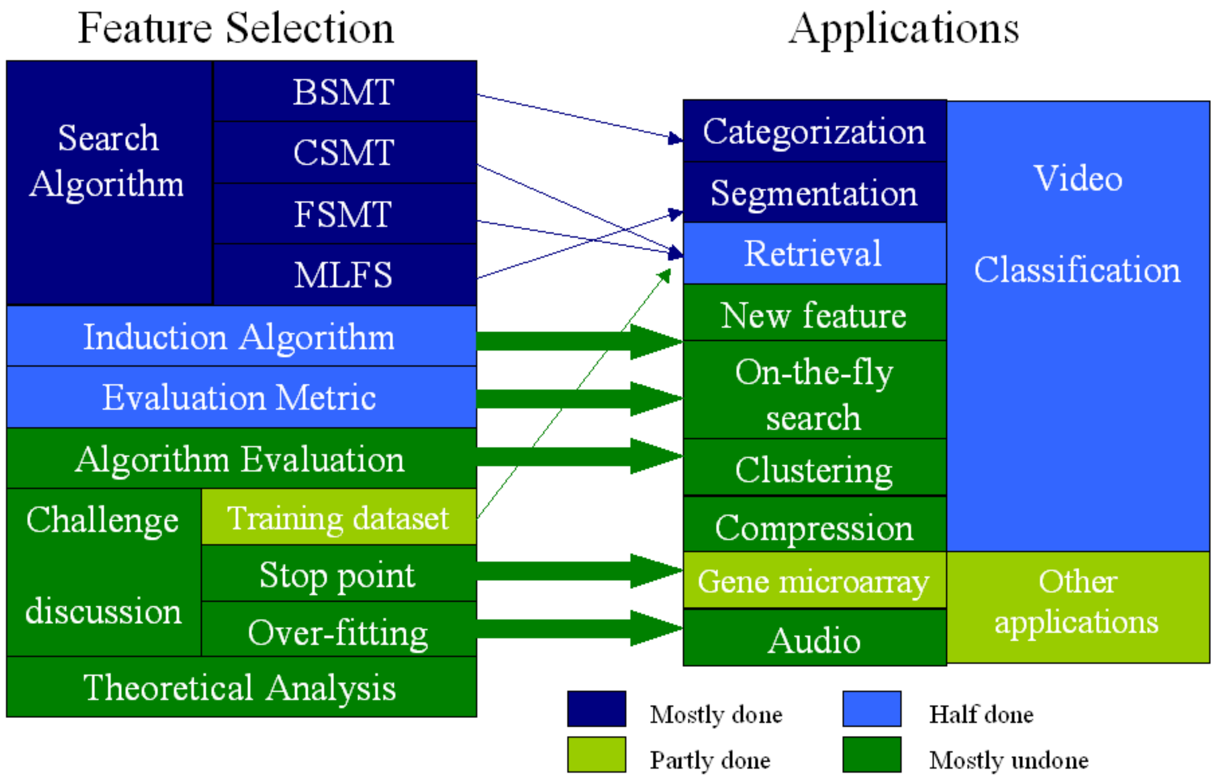


Figure 20. Finished work and further work

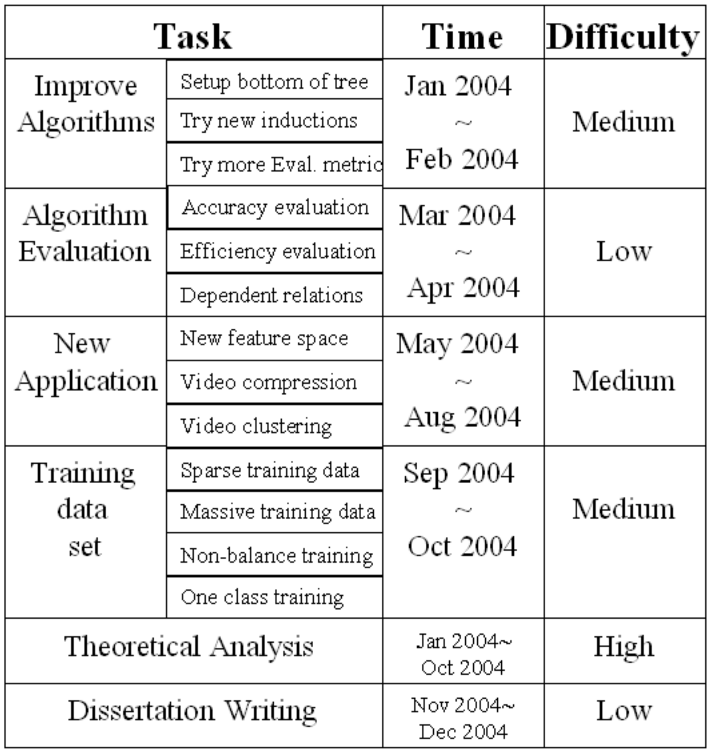
Another difficulty is in video retrieval applications, where we can't guarantee that any selected negative training data can replace the whole negative training data space. Under this case, which has better performance: using a partly sampled negative data set, or only using positive training data? We have explored this issue using data for masquerade detection in a network security problem in [38]. It has achieved similar accuracy as measured by ROC, using only one third of the features, selected by BSMT. How to apply feature selection to one class training data in general is also a very interesting problem.

3.5. Theoretical Analysis

Although most of the work will be empirically based, we hope to be able to extend some of the existing theoretic results available for the accuracy and cost of our heuristic algorithms. This should allow us to give order of magnitude estimates for the many validations on SMT techniques, such as style to what we reported in Section 2.1.3 (The cost analysis of BSMT).

4. Conclusion and schedule

High time complexity is always a bottle-neck in video segmentation, classification, and retrieval. Feature selection methods can improvement the accuracy and efficiency of video analysis. In Figure 20, we show the research progress till now and some further work discussed above. We provide a tentative work in Table 7.



Task		Time	Difficulty
Improve Algorithms	Setup bottom of tree	Jan 2004 ~ Feb 2004	Medium
	Try new inductions		
	Try more Eval. metric		
Algorithm Evaluation	Accuracy evaluation	Mar 2004 ~ Apr 2004	Low
	Efficiency evaluation		
	Dependent relations		
New Application	New feature space	May 2004 ~ Aug 2004	Medium
	Video compression		
	Video clustering		
Training data set	Sparse training data	Sep 2004 ~ Oct 2004	Medium
	Massive training data		
	Non-balance training		
	One class training		
Theoretical Analysis		Jan 2004~ Oct 2004	High
Dissertation Writing		Nov 2004~ Dec 2004	Low

Table 7. Task schedule: this table lists all the interesting topics mentioned in the proposal, but we anticipate that we not be able to fully explore all the topics in equal depth.

5. References

[1] Z. Lei and Y.T. Lin, "3D Shape Inferencing and Modeling for Semantic Video Retrieval", Proceedings of Multimedia Storage and Archiving Systems, SPIE's Photonics East 96 Symposium, Boston, MA, November 1996.

[2] Irena Koprinska and Sergio Carrato, "Temporal video segmentation: A survey", Signal processing: Image communication 16, 2001, pp.477-500.

[3] Michael S. Lew, Nicu Sebe, John P. Eakins, "Challenges of Image and Video Retrieval", International Conference on Image and Video Retrieval, Lecture Notes in Computer Science, vol. 2383, Springer 2002, pp.1-6.

[4] Wensheng Zhou, Asha Vellakial and C.-C. Jay Kuo, "Rule-based video classification system for basketball video indexing", ACM Multimedia, 2000.

[5] Pickering, M., Ruger, S., Sinclair, D., "Video Retrieval by Feature Learning in Key Frames", International Conference on Image and Video Retrieval, Lecture Notes in Computer Science, vol. 2383, Springer 2002, pp.316-324.

[6] A. Vailaya, M. Figueiredo, A. K. Jain, and H.-J. Zhang, "Image Classification for Contnet-Based Indexing", IEEE Transactions on Image Processing, vol. 10, no. 1, January, 2001, pp 117-130.

[7] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta and R. Jain, "Content-based image retrieval: the end of the early years", IEEE trans. PAMI, 22 - 12:1349 -- 1380, 2000.

[8] Yiming Yang and Jan O. Pedersen, "A comparative study on feature selection in text categorization", Proceedings of the Fourteenth International Conference on Machine Learning, 1997, pp.412-420.

[9] Eric P. Xing, Michael I. Jordan, Richard M. Karp, "Feature selection for high-dimensional genomic microarray data", Proceedings of the Eighteenth International Conference on Machine Learning, 2001.

[10] Michael S. Lew, Thomas S. Huang, Kam W. Wong, "Learning and Feature Selection in Stereo Matching", IEEE Transactions on Pattern Analysis and Machine Intelligence on Learning in Computer Vision, September, 1994, pp. 869-881.

[11] Isabelle Guyon and Andre Elisseeff, "An introduction to variables and feature selection", Journal of Machine Learning Research 3 (Mar): 1157 -- 1182, 2003.

[12] Avrim L. Blum and Pat Langley, "Selection of Relevant Features and Examples in Machine learning", Artificial Intelligence, 1997, pp.245-271.

[13] Singh, M. and Provan, G.M. "A comparison of induction algorithms for selective and non selective Bayesian Classifiers". In Machine Learning: Proceedings of the 12th International Conference. Morgan Kaufmann, 1995.

[14] D. Koller, R. M. Sahami. "Toward optimal feature selection". In Proceedings of the 13 International Conference on Machine Learning (1996) pp. 284-292.

[15] Luiz E. Soares de Oliveira, N. Benahmed, Robert Sabouin, Flavio Bortolozzi, Ching Y. Suen. "Feature subset selection using genetic algorithms for handwritten digit recognition". 14th Brazilian Symposium on Computer Graphics and Image Processing 2001.

[16] Naoto Abe, Mineichi Kudo and Masaru Shimbo. "Classifier-Independent Feature Selection Based on Non-parametric Discriminant Analysis". In Proceeding of Joint IAPR International Workshops (2002) pp. 470-479.

[17] Jinbo Bi, Kristin Bennett, Mark Embrechts, Curt Breneman, Minghu Song, "Dimensionality reduction vis sparse support machine", Journal of Machine Learning Research 3 (Mar): 1229 -- 1243, 2003.

[18] Sanmay Das. "Filters, Wrappers and a Boosting based hybrid for feature selection". In Proceedings of the Eighteenth International Conference on Machine Learning, pages 74--81, Williams College, June 2001.

[19] J. Weston, S. Mukherjee, O. Chapelle, M. Pontil, T. Poggio, and V. Vapnik. "Feature selection for SVMs". In Advances in Neural Information Processing Systems 13. MIT Press, 2000.

[20] H. Liu, H. Motoda, and L. Yu, "Feature Selection with Selective Sampling", Proceedings of the 19th International Conference on Machine Learning. July 8-12, 2002. Sydney, Australia.

[21] M. Dash, H. Liu and H. Motoda, "Consistency Based Feature Selection", pp 98 -- 109, PAKDD 2000, Kyoto, Japan. April, 2000. Springer.

[22] Frans M. Coetzee, Eric Glover, Ste ve Lawrence, C. Lee Giles “Feature selection in web applications by ROC inflections and powerset pruning”. Symposium on applications and the Iternet, SAINT 2001.

[23] Alejandro Jaimes and Shih-Fu Chang, "Automatic Selection of Visual Features and Classifiers", Storage and Retrieval for Media Databases 2000, IS&T/SPIE. San Jose, CA, January 2000.

[24] Yan Liu and John R. Kender. Video frame categorization using Sort-Merge feature selection.In Proceedings IEEE Workshop on Motion and Video Computing, pages 72--77, 2002.

[25] Christons Faloutsos and king-Ip (David) Lin“FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets”, Proceedings of ACM SIGMOD,1995, pp 163-174.

[26] Richard O. Duda, Peter E. Hart and David G. Stork, Pattern classification, Wiley, New York, 2000.

[27] Wei-Hao Lin and Alexander Hauptman, “News video classification using SVM-based multimodal classifiers and combination strategies”, Proceedings of ACM Multimedia 2002, Juan-les-Pins, France, December 1-6, 2002.

[28] Yan Liu and John R. Kender. Video retrieval under sparse training data. CIVR 2003, pp 406- 413.

[29] Yan Liu and John R. Kender. Video Feature Selection Using Fast-converging Sort-Merge Tree. Submitted to IEEE International Conference on Multimedia & Expo, 2004.

[30] Yan Liu and John R. Kender. Fast scene segmentation using multi-level feature selection algorithm. IEEE International Conference on Multimedia & Expo, 2003.

[31] Irena Koprinska and Sergio Carrato, “Temporal video segmentation: A survey”, Signal processing: Image communication 16, 2001, pp.477-500.

[32] Yan Liu and John R. Kender. Fast video segment retrieval by Sort-Merge feature selection, boundary refinement, and lazy evaluation. Computer Vision and Image Understanding , volume 92, Issues 2-3, November-December 2003, pp 147-175.

[33] Feature selection challenge 2004. http://www.nipsfsc.ecs.soton.ac.uk/.

[34] Yan Liu and John R. Kender. Feature selection for video data. Proceedings of the Third SIAM International Conference on Data Mining, San Francisco, CA, USA, May 1-3, 2003.

[35] Yong Wang, Tian-Tsong Ng, Mihaela van der Schaar, Shih-Fu Chang, "Predicting Optimal Operation of MC-3DSBC Multi-Dimensional Scalable Video Coding Using Subjective Quality Measurement", Proc. SPIE Video Communications and Image Processing (VCIP) 2004.

[36] L.Xie, S.F. Chang, A.Divakaran and H.Sun. Feature selection for unsupervised discovery of statistical teoporal structures in video. Accepted by ICIP 2003.

[37] George Forman. “An extensive empirical study of feature selection metrics for text classification”, Journal of Machine Learning Research 3 (Mar): 1289 -- 1305, 2003.

[38] Ke Wang, Salvatore J. Stolfo. "One Class Training for Masquerade Detection". ICDM Workshop on Data Mining for Computer Security (DMSEC). Melbourne, FL, Nov. 19, 2003.