Fast Video Segment Retrieval by Sort-Merge Feature Selection, Boundary Refinement, and Lazy Evaluation

Yan Liu and John R. Kender

Department of Computer Science

Columbia University

New York, NY 10027

{liuyan, jrk}@cs.columbia.edu

Abstract

We present a fast video retrieval system with three novel characteristics. First, it exploits the methods of machine learning to construct automatically a hierarchy of small subsets of features that are progressively more useful for indexing. These subsets are induced by a new heuristic method called Sort-Merge feature selection, which exploits a novel combination of Fastmap for dimensionality reduction and Mahalanobis distance for likelihood determination. Second, because these induced feature sets form a hierarchy with increasing classification accuracy, video segments can be segmented and categorized simultaneously in a coarse-fine manner that efficiently and progressively detects and refines their temporal boundaries. Third, the feature set hierarchy enables an efficient implementation of query systems by the approach of lazy evaluation, in which new queries are used to refine the retrieval index in real-time. We analyze the performance of these methods, and demonstrate them in the domain of a 75-minute instructional video and a 30-minute baseball video.

Keywords

Fast video retrieval, Sort-Merge feature selection, Boundary refinement, Lazy Evaluation

1. Introduction

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

Methods focused on efficiency take several forms. Audio information analysis [2] [3] [4] and text extraction and recognition [5] [6] are often used, as are some approaches based on segmentation of video shots and key-frame selection [7] [8] [9]. Some specialized applications, particularly sports videos [10], use object recognition [11] and object tracking [12] for efficient compression. Some systems combine several methods together [13], and some can work directly on compressed domain [14]. However, there appears to be little work that supports efficient on-line video retrieval without some prior human specification of the underlying feature sets.

A second concern is semantic video indexing and retrieval. As Lew et al mentioned in [15], the main challenge in image and video retrieval is bridging the semantic gap between the high level query from the human and the low-level features that can be easily measured and computed. This gap 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.

This paper addresses these two problems, and presents a method of efficient semantic video retrieval, based on automatically learned feature selection. 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 which most capture the trained semantic categories given by the user. By imposing a natural hierarchical structure on these subsets, we can use these learned features to efficiently segment and label the video without loss of accuracy, and to adaptably improve response of a query system by efficiently caching dynamic results. We also show that these methods can be applied to either the original video, or its compressed form.

The heart of our method is a novel approach to feature selection. This form of learning has received significant attention in the AI literature, and has been applied to moderately large data sets in applications like text categorization and genomic microarray analysis. Learning research is not often carried out in video indexing and retrieval--although Lew et al [16] used a feature selection method to refine features for stereo image matching. This is because the sheer magnitude of video data has limited the choice and application of existing feature selection algorithm, which have been designed for smaller databases and which run inordinately long even on those. One emphasis of this paper is the low time cost of our heuristic method, which can exploit several properties unique to video data to induce appropriate but small feature sets.

This paper is organized as follows. Some related work in video retrieval and in feature selection is introduced in section 2. Section 3 proposes the novel algorithm, and gives some proofs of some of its essential features and of its time complexity. The data structure output by the feature selection supports two additional methods, boundary refinement and lazy query evaluation, which are presented in section 4. Section 5 provides empirical validation. We close the paper in section 6.

2. Related work

2.1. Video retrieval

We are interested in how to retrieve video sequences based on metadata: video segmentations, indices, annotations, and summaries, derived primarily from the visual (not audio) steam. There has been a great deal of work in this area, and we cannot review it all here.

One typical way is to make use of 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. Among other approaches, Pickering et al in [7] consider the key-frames of each shot as a single image, and use the machine learning approach of Boosting Image Retrieval of Tieu and Viola in [8] to

retrieve matching video sequences. One of several semantic-based key-frame approaches is that of Schaffalitzky and Zisserman in [9], where they consider video shots to happen in the same 3D real world scene by representing their key-frames using invariant descriptors and by pruning out false matches using a spatial neighborhood consensus.

Semantic video retrieval in some special applications such as sports or TV news, can be based on object recognition and object tracking. For example, by tracking a basketball and related objects in Kim et al [28], useful semantics about a class of shots are extracted. In another specialized domain [30], Wei and Sethi use the presence of skin-tone pixels coupled with shape, edge patterns, and face-specific features to detect faces for image and video retrieval.

Among more general approaches is that of Naphade et al in [12], who propose a framework for video indexing and retrieval using semantic unit "multijects" (i.e., "multiple objects"). Different feature sets such as color, texture, edges, shape, and motion are extracted from each frame and pass through different classifiers that check the multijects individually and combine the results to get the final decision. Similarly, Smith et al in [19] propose a problem of feature fusion when integrating features, models, and semantics for TREC video retrieval. They represent and retain the results from different feature sets in GMMs (Gaussian mixture models), instead of combining different feature sets to one. However, they no longer can index only once into one uniform feature space because of the large dimensionality of their feature space model.

This paper follows the spirit of the more uniformed approach, where frames are represented by a large feature set, but one that is not selected based on any preconceived ideas of the appropriateness of a feature. Instead, we solve this problem using a novel feature selection algorithm that learns the most appropriate feature subset.

2.2. Feature selection

2.2.1. Definition of feature selection

A precise mathematical statement of the feature selection problem is not widely agreed upon, partly because there has been substantial independent work on feature selection in several fields: machine learning, pattern recognition, statistics, information theory, and the philosophy of science. Each area has formalized the definition from its own viewpoint, and each definition has been colored by the intended application. However, there appears to be two major approaches.

The first approach emphasizes the discovery of any relevant relationship between features and concept. This is referred to as a filter method, and it finds a feature subset independently of the actual induction algorithm that will use this subset for classification. This is formalized by Blum and Langley in [20]:

Definition 1 (Relevance to the target)

A feature x_i is relevant to a target concept c if there exist a pair of examples A and B in the instance space such that A and B differ only in their assignment to x_i and $c(A) \neq c(B)$.

The second approach explicitly seeks a feature subset that minimizes prediction error. This is referred to as a a wrapper method, and it searches the space of feature subsets, using cross-validation to compare the performance of a trained classifier on each tested subset, and directly optimizes the induction algorithm that uses the subset for classification. This is formalized by Caruana and Freitag in [21].

Definition 2 (Incremental usefulness)

Given a sample of data S, a learning algorithm L, and a feature set A, feature x_i is incrementally useful to L with respect to A if the accuracy of the hypothesis that L produces using the feature set $\{ x_i \} \cup A$ is better than the accuracy achieved using just the feature set A.

2.2.2. Filter methods

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; see

Blum and Langley [20]. Greedy set-cover algorithms are the simplest filter methods that are often used for classifiers, particularly binary concept classifiers. They begin with zero chosen features, and assume that every data instance is in a single category. By incrementally adding to the evolving feature set that next best feature which can discriminate those data belonging to another category, it constructs the final subset of features using this greedy approach.

A better and more typical filter method is the Focus algorithm proposed by Almuallim and Dietterich in [22]. This method begins by looking at each feature in isolation, then turns to pairs of features, then triples, and so forth, halting only when it finds a combination that generates pure partitions of the training set, using a decision-tree induction. Similar algorithms are proposed by Cardie in [23] using KNN induction, and Kubat et al in [24] using native Bayesian induction. Jebara and Jaakkola in [25] follow a similar approach, but employ better metrics based on information theory, using SVM induction.

2.2.3. Wrapper methods

Wrapper methods assess the quality of feature subsets according to their prediction error. The typical wrapper algorithm searches in the feature subset space. It evaluates alternative subsets by running an induction algorithm on training data, and uses the estimated accuracy of the resulting classifier as its metric to find an optimal subset of features.

As Xing et al. state in [26], wrapper methods attempt to optimize directly the predictor performance so that they can perform better than filter algorithms, but they require more computation time. This cost has led some researchers to invent ingenious techniques for speeding the evaluation process. For example, Langley and Sage's OBLIVION algorithm [27] carries out a backward greedy search decision-tree induction; nevertheless the cost is still substantial.

Alternatively, Singh and Provan use information-theoretic metrics in [28] based on the forward greedy algorithm in a Bayesian network they proposed in [29]. Likewise, Koller and Sahami in [30] employ a cross-entropy measure, designed to find Markov blankets of features using a backward

greedy algorithm; this algorithm has been successfully applied to the classification of Genomic Microarray data by Xing et al in [26]. The approach we propose is also a wrapper method.

3. Sort-Merge feature selection algorithm

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). 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.

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, the accuracy of certain classifier and complexity cost. Lew et al [16] used a filter model of feature selection to refine the feature sets in stereo matching and found good performance in image matching. They selected one of several traditional feature selection search methods to reduce the original feature space, which consisted of the gray levels of every pixel in the image, in order to more tractably evaluate their optimality criterion, defined as the average distance between elements of all classes. This is an interesting beginning, but the nature of video data, such as massive data, high dimensionality, and complex hypotheses, together with the unrealistic amounts of computer time involved even in these more limited domains (on the order of weeks), limits the choice and application of existing feature selection algorithms for video retrieval. This paper proposes a

novel Sort-Merge feature selection method to select features for video retrieval with low time and space cost.

3.1. Sort-Merge search algorithm for feature selection

There are 2^N possible subsets for a feature space with N features. Exhaustive testing of each subset is impossible even when N is moderate. Necessarily heuristic in approach, current feature selection algorithms often work well when there are a straightforward logical relationships (in the sense of conjunctions or disjunctions) between features and categories. Categorization of entire video frames, however, does not appear to be either straightforward or logical, and is further complicated by the redundancy of neighboring pixels.

So far, three general kinds of heuristic search algorithms have been used: forward selection, backward elimination, and genetic algorithms. Forward selection starts with the empty set and successively adds individual features, usually following a variant of a greedy algorithm, terminating when no improvement is possible. However, it can not remove any features, and therefore ends up making what amounts to local optimizations to the growing set. Backward elimination, 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 the removal of a feature is irrevocable. A genetic algorithm, 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.

Our Sort-Merge feature selection algorithm combines the features 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 work.

The Sort-Merge algorithm 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.

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 1 illustrates the algorithm with an initial set of features with cardinality N = 256.

```
Initialize level = 0
        Create N singleton feature subsets.
While level <  log2 N
        Induce on every feature subset.
        Sort subsets based on performance.
        Combine, pairwise, feature subsets.
```

Table 1. Sort-Merge feature selection basic algorithm

Table 2 shows the related algorithm to select exactly r features from the hierarchy of feature subsets, if such a subset is desired, and Figure 2 illustrates this process when r = 20. Since r is between 2^4 and 2^5, the leftmost and therefore most accurate sub-tree with 2^5 nodes is extracted from the full tree. This gives a sub tree with 12 nodes in excess of the desired amount, thus, the value of "cutout" is 12. The algorithm then, as part of its recursive greedy approach, looks for further subtrees with "branch-size" of size 8 to next remove; the branch that impacts performance the least is chosen for removal. The tree is now 4 nodes in excess, and the algorithm repeats once again (and then

terminates) by finding which subtree of this branch-size has the least impact on this wrapper-based classification performance.

Select the leftmost branch of size $2^{\lceil \log 2 r \rceil}$.
Initialize cutout = $2^{\lceil \log 2\ r \rceil}$ - r.
While cutout >0
    Let branch-size = $2^{\lfloor \log 2\ \text{cutout} \rfloor}$.
    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.

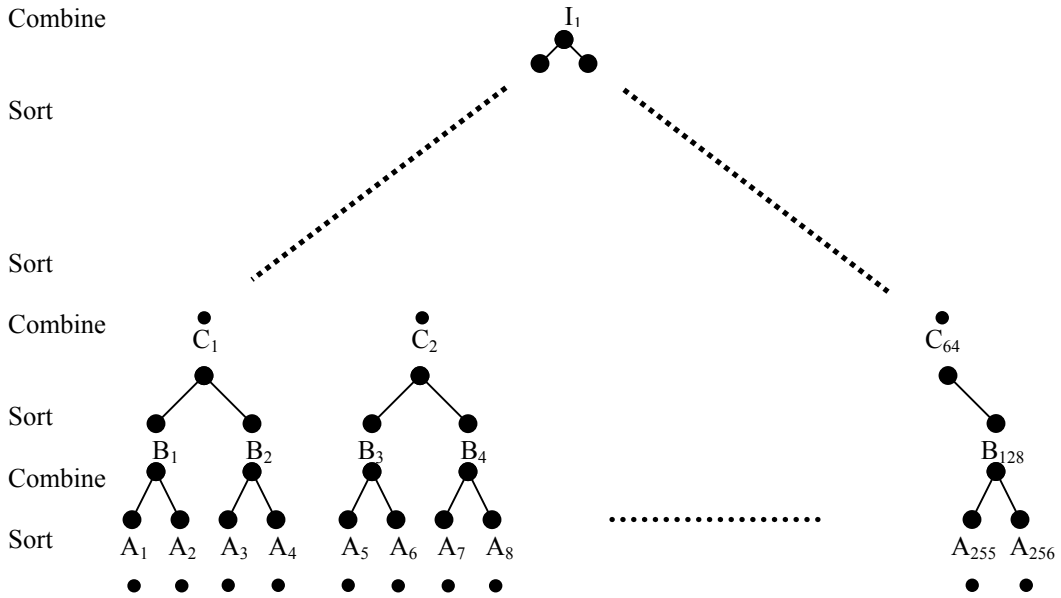
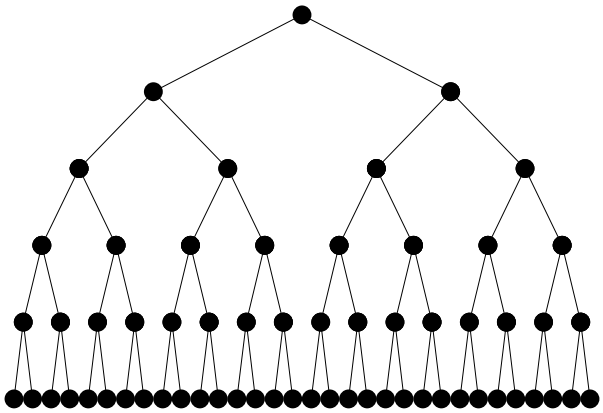


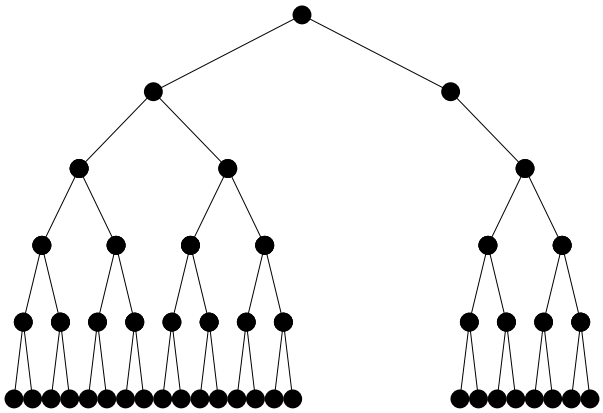
Figure 1. The Sort-Merge feature selection algorithm. Leaves correspond to singleton feature subsets. Interior notes are formed by the pair-wise merge of neighboring feature subsets that have been sorted according to their classification accuracy.

Although it should be apparent that the Sort-Merge method should select relevant features (Definition 1) due to the nature of the sort, it is not clear that it also gives priority to features that are incrementally useful (Definition 2) due to the nature of the merge. Although we have no rigorous proof that this is so, we illustrate what we have found to be its reliable occurance in Figure 3. Some

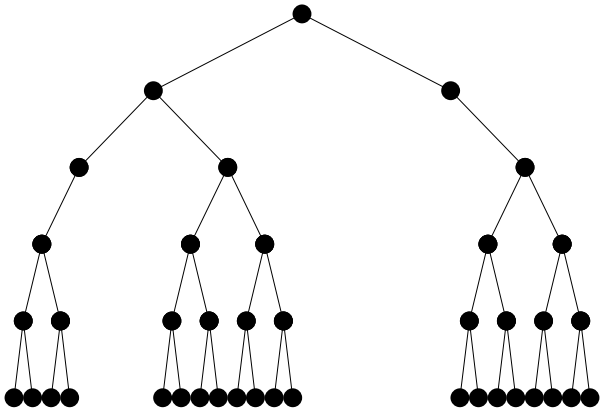
examples of how Sort-Merge feature selection addresses the problem of redundant features are discussed in section 5.1.2.



(a) Select the leftmost sub-tree



(b) Cut the most errorful thick branch from the sub-tree



(c) Cut the most errorful thin branch from the sub-tree

Figure2. Selecting a feature subset of a pre-specified size from the full Sort-Merge feature selection tree

Figure 3 is a more detailed view of the lowest two levels of the tree in Figure 1. In the original feature space, there are a total of N=256 features. We first divide them into 256 singleton feature subsets. Next we induce using each feature subset in the given dataset, and evaluate the performance of the feature subsets by their accuracy. We sort feature subsets based on their prediction error; feature subset A_1 has the lowest prediction error. Then we combine two neighboring singleton sets in rank order into new feature subsets and induce again, and again evaluate the performance of the 128 feature pairs and sort

them based on their prediction error. As Figure 3 illustrates, feature subsets A_1 and A_2, which are ranked first and second after the first sort, may be ranked (as a pair) third after the second sort; similarly feature subsets A_5 and A_6 which are ranked fifth and sixth after the first sort, are ranked second after the second sort. Since the merge step always combines neighbors with very similar performance, any decrease in the rank order of a new pairing cannot be ascribed to one or other of the individual neighbors being "bad". What makes a pair (A_1 and A_2) worse is that their combined performance did not increase as much as those of other pairs (A_5 and A_6). That is, their two elements tended to be redundant in their classification failures whereas other pairs had complementary strengths.

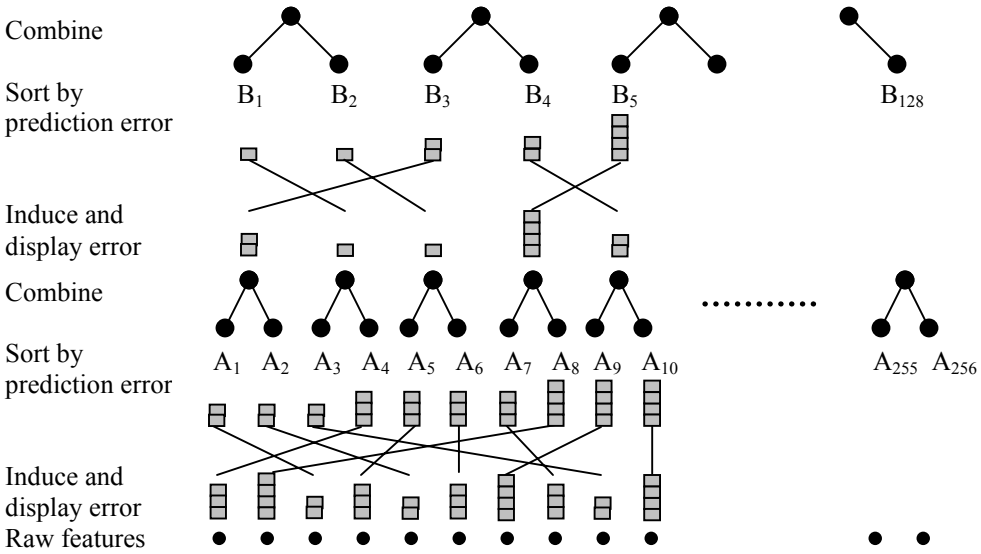


Figure 3. A more detailed view of how the Sort-Merge algorithm addresses the problem of redundant features. Even though features of a level are combined only by performance, redundant features (A_1, A_2) don't increase performance as much as non-redundant features (A_5, A_6) after merge.

3.2. Induction algorithm for feature selection

The performance of a wrapper feature selection algorithm not only depends on the search method, but also on the induction algorithm. Some feature selection methods have high computational cost only because the induction algorithm is time-consuming and does not scale well to large feature spaces. For our induction method during the course of the learning, we use the novel, low-cost, and scalable combination of Fastmap for dimensionality reduction, with Mahalanobis maximum

likelihood for classification. We refer readers to the literature for a detailed explanation of these two component methods, but we summarize their significance here.

In brief, as defined in statistical texts Duda et al. [31], 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.

The Mahalanobis metric has good performance in classifying data with multiple dimensions, even if each dimension has a different range of feature values. However, it is necessary that the cardinality of the training set be much larger than the number of dimensions; the usual lower bound given by Devijver and Kittler in [34] for this cardinality is N(N-1)/2, where N is the number of dimensions.

Principal Component Analysis (PCA) is the usual method of choice for dimensionality reduction, but carries high computational complexity. Instead, the Fastmap method proposed in [32] approximates 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. This linearity of the cost of Fastmap is a critical advantage, and permits its use for high-dimensional feature sets: the method scales well.

In summary, then, our full Sort-Merge method consists of the repeated application of the following processing steps at each level of the tree. For each feature subset at a level, the dimensionality of its feature space (which is always a power of 2) is reduced by using the Fastmap algorithm to a small number of dimensions (which is a parameter set by the user; we report on performance of this dimension c, for c = 1 to 10). Then, within this reduced space, induction occurs by modeling the classifications by their likelihoods given by the Mahalanobis distance to classification centers. The scores of each subset's test classification accuracy are then rank ordered, and new subsets of features are formed by merging pairwise those old subsets that are adjacent in the rank ordering.

It is appropriate to comment on why, given its virtues, we do not apply Fastmap to the entire feature space directly, and simply use the first c dimensions of the resulting reduced space as the set of features. The answer is that each of these c eigen-like vectors is a linear sum of the full feature space, and the classification of any video frame would require the accessing and transformation of every original feature, at substantial cost. By first determining a good subset of the features, the cost of any subsequent classification, whether by Fastmap-Mahalanobis or otherwise, is greated reduced.

3.3. Analysis of Sort-Merge feature selection algorithm

Generally speaking, the accuracy of a classifier based on the Mahalanobis distance increases as the dimensionality of the feature space increases. In our method, this means that accuracy tends to increase as the feature subsets are merged to form the hierarchy. We now analyze the time and space cost of doing so, 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

T_m: Time of induction using m training data in the Mahalanobis classifier

T_{basic}: Time of the basic Sort-Merge feature selection algorithm

T_{select}: Time of algorithm to select r features from the tree

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 $T_m = O(mc^2)$. 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.

Similarly, one can show that the additional cost of $T_{select} = O(rT_m) = O(rmc^2)$; this is dominated by T_{basic}. The argument is based again on the sum of a geometrically decreasing series of costs, with each cost proportional to the effects of pruning ever smaller numbers of subtrees.

4. Fast off-line video indexing and on-line video retrieval system

The linear time and space costs of our novel feature selection approach allows us the practical implementation of three related retrieval applications, two of which are novel in their own right. Other retrieval tasks would now also appear to be feasible, and it is likely that the feature selection method can also be applied to other data from video sequences, such as audio data or higher level features such as shape descriptions. Here we focus on three purely visual tasks of: fast video frame classification and retrieval, video segment boundary refinement, and lazy evaluation of unanticipated on-line queries. Our inputs are the compressed frames of MPEG-1 instructional videos, generally without prior temporal segmentation into shots.

4.1. Fast video frame classification and retrieval

Although the method is transparent to these particular video preprocessing transformations, in our examples we illustrate its application after down-sampling a 75 minute long MPEG-1 video both temporally and spatially. We use only every other I frame (that is, one I frame per second), and we spatially subsample by only using the DC terms of each macroblock of the I frame (consisting of six terms: four luminance DC terms, one from each block, and two chrominance DC terms). We therefore do not have to decompress the video. We did not further spectrally subsample, but the method again would be transparent to this. This gives us, for each second of video, 300 macroblocks (15 by 20) of 6 bytes (4 plus 2) of data: 1800 initial features. This is a much larger feature set than virtually all examples in the machine learning literature. For convenience of accessing and decoding, we generally consider the 6 DC terms from the same macro-block to be an undecomposable vector, so our initial application most often consists more accurately of 300 features per second of video. Each six-dimensional feature is first placed into its own subset to initialize the Sort-Merge process. So, in our application, we start with 300 such feature subsets, and each feature subset has cardinality 1.

Next, using Fastmap, the dimensionality of each feature subset is reduced to a pre-specified small number, c, of dimensions. (For this stage, we do in fact look inside the vectors and use each component as a dimension.) In our applications, we ran experiments in which c varied from 1 to 10. The value of c is not fixed and is related with the original size of feature space, which will be discussed in the experiment section.

Then, for each feature subset at this level, using the reduced dimensionality representation, the training frames of the video train the induction algorithm to classify the test frames of the video. In our application, this means that each training set was represented by a c-dimensional Gaussian according to the Mahalanobis classifier, although other learning methods can be trained on the reduced representation. In our application, in the context of instructional video, we had four class labels: the instructor is writing an overhead slide, the instructor is announcing, the instructor is displaying a computer demo, and the class is discussing.

Next, the classification accuracy of each feature subset is measured. If any subset achieves the user's pre-specified desired accuracy, or if the cardinality of each subset achieves the user's pre-specified desired cardinality, the process stops, and that subset is the desired feature subset. Otherwise, the feature subsets are sorted by accuracy, and the next level of the feature subset hierarchy is formed by merging these subsets pair-wise and in order (see Figure 1).

Lastly, the process repeats again, starting at the Fastmap step. It is clear that at most O(log N) iterations of this Sort-Merge algorithm are necessary. The resulting feature subset is then used for classification in the usual way: the given features are taken from any other video frame, reduced to c dimensions, and classified by their maximum likelihood in the Mahalanobis sense.

4.2. Video segments boundary refinement

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.

To illustrate, we select the best 2-feature subset from the 300 features using Sort-Merge feature selection algorithm, and classify each frame of the video into the four different categories mentioned above. This is shown as the uppermost line in figure 4 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 [33]). 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.

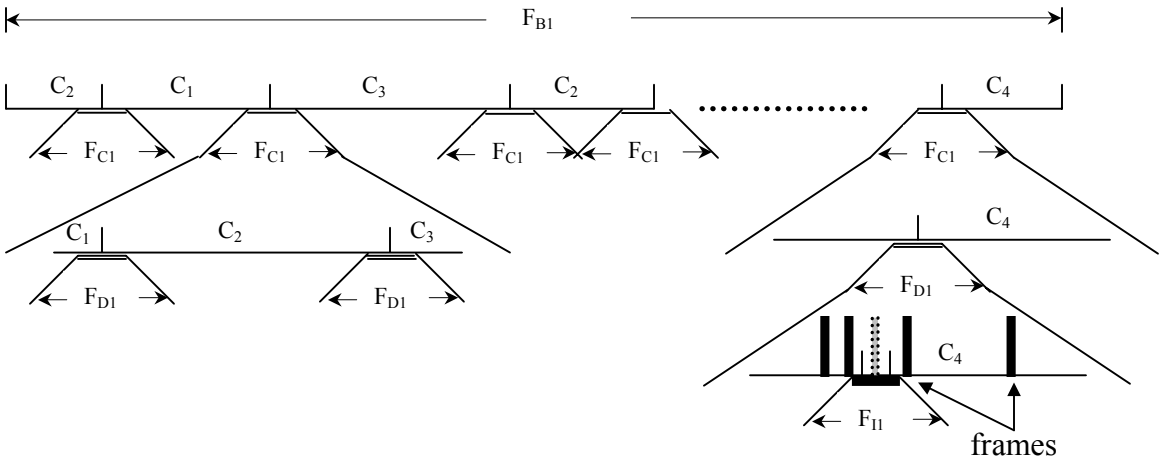


Figure 4. Video segment boundary refinement by multi-level feature selection.

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:

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

where $Pr_{mahal}(C_j)$ is the maximum Mahalanobis likelihood among all categories using this feature subset. This threshold ensures that classification is correct and unambiguous.

Figure 4 illustrates three typical cases. Most of the refinements result in the first case: a clarification of the location of the boundary developed by the intial 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.

4.3. Lazy evaluation of unanticipated on-line queries

This application allows the dynamic extension of video retrieval indices. 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 5, 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.

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 retrived 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 enable such lazy evaluations, and the indexing system becomes self-adaptive.

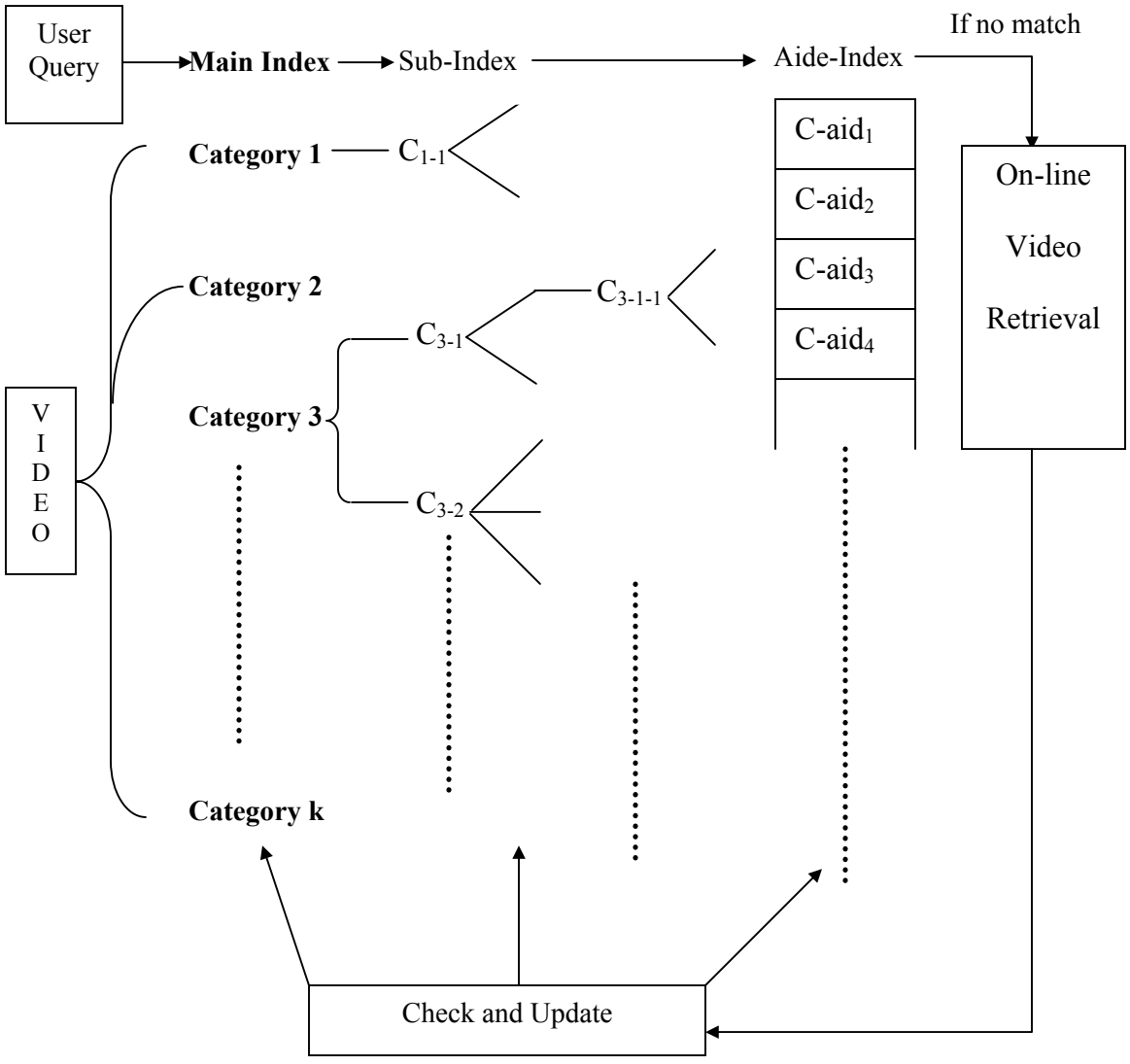


Figure 5. Lazy evaluation of unanticipated on-line queries.

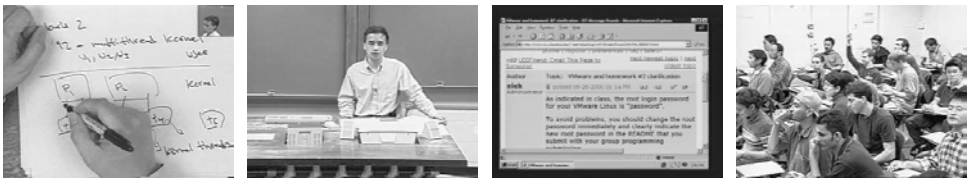
5. Experiments

5.1. Fast video frame classification and retrieval

The first experiment evaluates the effectiveness of the Sort-Merge feature selection algorithm on standard video frame classification and retrieval.

5.1.1. The basic classification task

Our goal 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 6: handwriting, announcement, demo, and discussion. For training data, we used 400 I-frames distributed over the video and across these four classes.



Handwriting Announcement Demo Discussion

Figure 6. Main-index of video

As mentioned above, the time cost of feature selection algorithms is decided by the cost of the search algorithm and the induction algorithm together. The primary contribution of the Sort-Merge feature selection algorithm is the low time complexity of these two components. Existing feature selection methods, which typically have been reported to run for several days on features sets of cardinality of at least one decimal order of magnitude smaller, as Koller and Sahami note [30], are intractable on this dataset. Therefore, we first compared the indexing accuracy of our new method against two imperfect but feasible benchmarks: random feature selection, and hand feature selection. These application experiments all used the same data and same induction methods; the only difference is how the feature subsets are chosen. Secondly, we fixed the induction algorithm to highlight the efficiency of the search algorithm in Sort-Merge feature selection, and then compared its performance and time cost with reasonable implementations of the forward selection, backward elimination, and genetic algorithm approaches. To simplify our presentation, only the comparison

experiments that selected 30 features from the 300 possible feature are displayed here, although we do display the effect of varying the value of c.

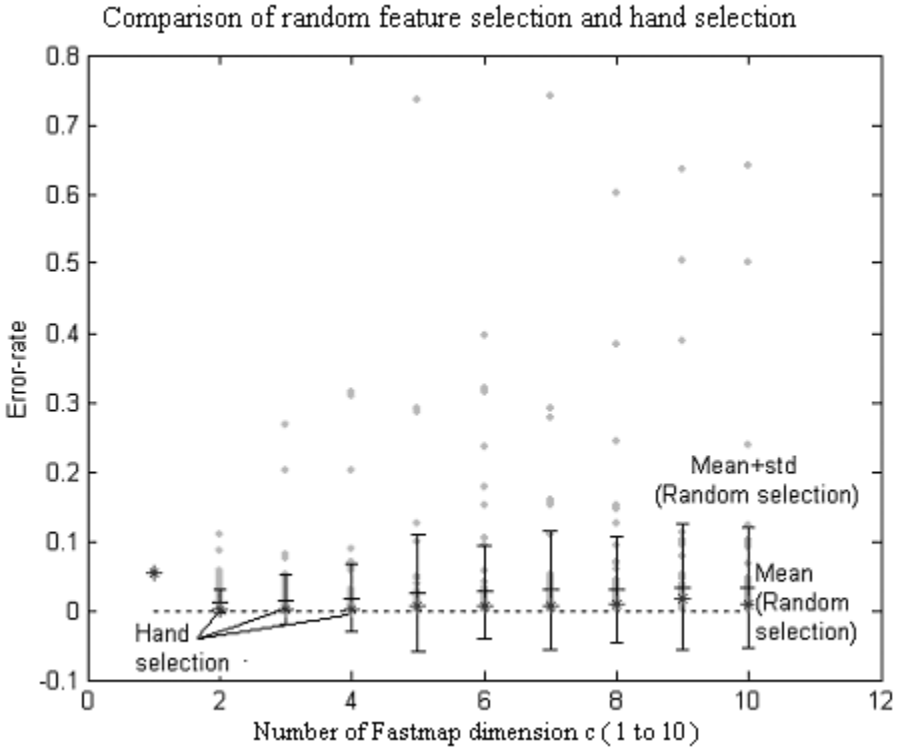


Figure 7. Classification results for random feature selection and hand feature selection. Error bar is shown for one standard derivation, but all points have positive error rates.

For random feature selection, we ran 100 experiments in which 30 features were selected randomly. Points in Figure 7 show the error rate of frame categorization under different Fastmap reduced dimensions of c, from 1 to 10, where "error rate" is defined as the pure ratio (not percentage) of misclassified frames to total frames. The error bars are drawn at the mean error plus one standard deviation. The dashed-line shows the base of error rate of zero. Superimposed on the graph are asterisks representing the error rate of hand selection, typically about 0.002 (9 misclassifications out of 4467). For precision, Table 3 lists the exact mean, maximum, and minimum of the classification error rate for these different values of c. Figure 8 gives more detail still, as it depicts the error rate of scene categorization under one of these cases, the case of c=2, with each run of the random experiments explicitly illustrated. As expected, the rate of error of random selection is highly variable, with the standard deviation being larger than the mean.

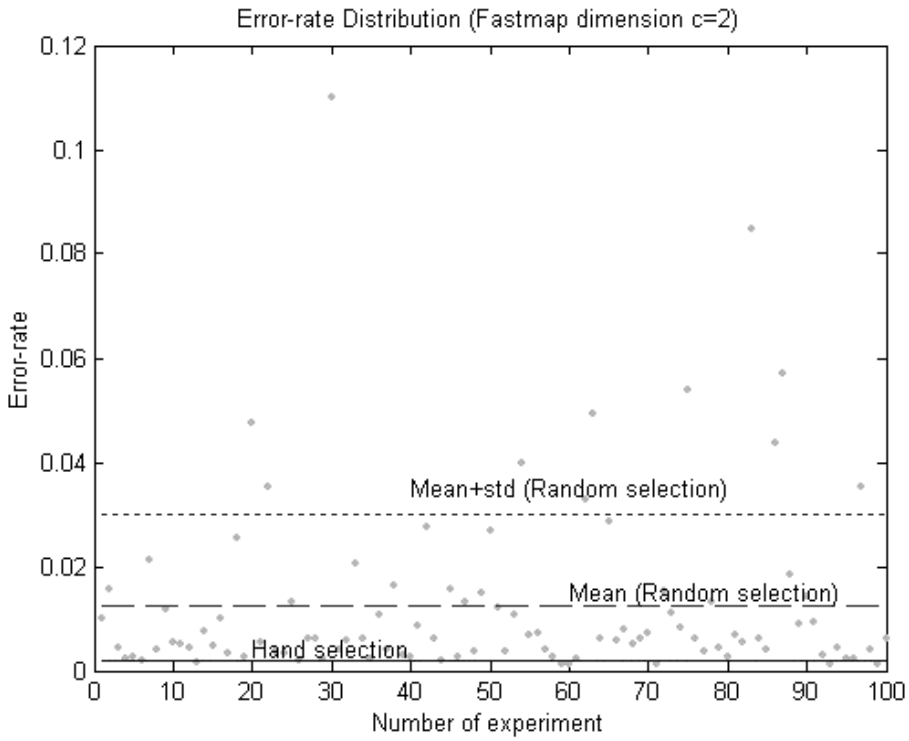


Figure 8. Error rates for classification results with Fastmap dimension c=2.

C	1	2	3	4	5	6	7	8	9	10
Mean	0.0531	0.0125	0.0153	0.0182	0.0255	0.0266	0.0294	0.0293	0.0335	0.0328
Max	0.0531	0.1101	0.2677	0.3141	0.7347	0.3971	0.7405	0.6013	0.6367	0.6409
Min	0.0531	0.0011	0.0009	0.0011	0.0011	0.0011	0.0011	0.0009	0.0016	0.0016

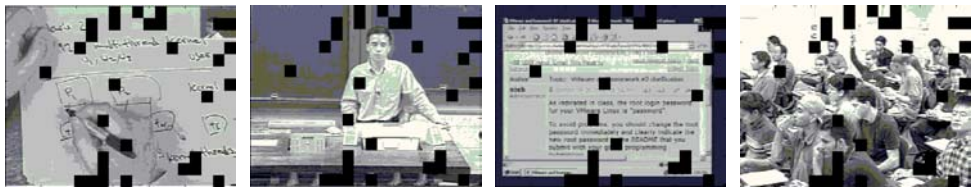
Table 3. Error-rate of classification using 30 randomly selected macroblocks.

For hand feature selection, we show as black boxes in Figure 9 those 30 features (i.e., macro-blocks) selected by hand, based solely on the experimenter's intuition and with some effort taken to provide pixels sensitive to the positions of instructor, desk, paper, and frame border. The error rate of the classification that results is also shown in figure 7 as asterisks and in figure 8 as a solid line, which is obviously better than the random selection result.

Handwriting Announcement Demo Discussion

Figure 9. Hand-selected macro-blocks.

For the Sort-Merge feature selection, we show in Figure 10 the 30 features (i.e., macro-blocks) selected by the Sort-Merge method; surprisingly, the method favors border macro-blocks, with 20 of the 30 chosen on, or just one macro-block away from, the image border. This is possibly because these pixels tend to be the most stable over time and little human movement appears to affect them.



Handwriting Announcement Demo Discussion

Figure 10. Macro-blocks selected by Sort-Merge method.

Figure 11 is a grand summary comparing these three feature selections (random, hand, Sort-Merge). The classification error rate of the Sort-Merge method is not only less than that of hand 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.

Table 4 summarizes the results of our second comparison, that of the accuracy and efficiency of four different search algorithms for feature selection. Although we used the same low time cost induction algorithm, we found that the forward selection, the backward elimination, and the genetic algorithm approaches have at least a decimal order of magnitude higher time cost than Sort-Merge feature selection, solely attributable to the different search strategies. Backward elimination, in fact, did not terminate.

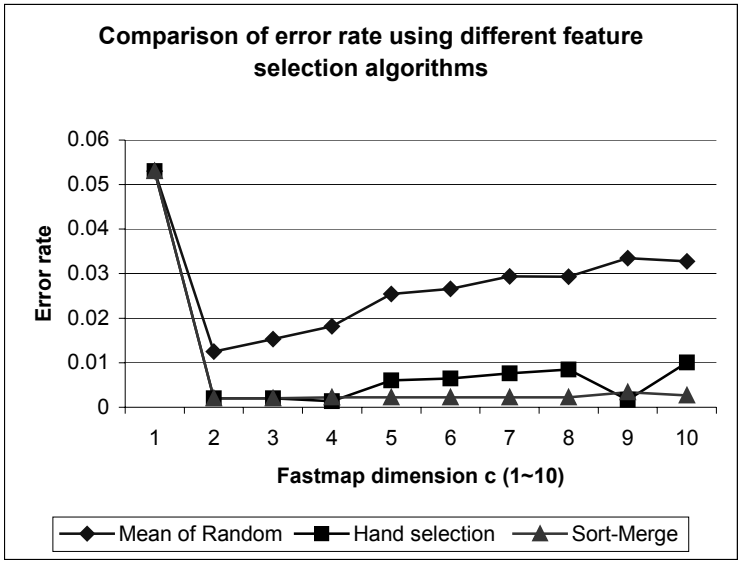


Figure 11. Off-line video indexing result of different feature selection methods

Search algorithm	Forward Selection	Backward Elim.	Genetic Alg.	Sort-Merge
Error rate	0.007	---------	0.007	0.002
Running time	691.32 minutes	Several weeks(?)	306.95mins	31.01mins

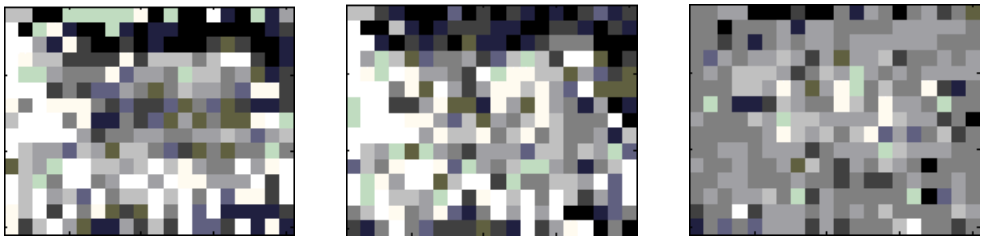
Table 4. Classification performance of different search algorithms

5.1.2. Further analysis and discussion

In this section, we further demonstrate that the Sort-Merge method tends to avoid selecting redundant features that add little further discriminating power to a developing feature subset.

In Figure 12(a) we have encoded the accuracy score of the 300 individual macro-blocks, where darker blocks are those with lower error rate, that is, dark means "good". This also simultaneously illustrates the results of the first step of the algorithm: the darker blocks represent those more accurate singleton feature subsets that would appear in the leftmost leaves of the feature selection tree. Figure 12(b) encodes the results of the next step of the algorithm, after singleton sets have been merged into 150 feature pairs and their performance evaluated. Again, a darker block represents a macro-block

that is a member of a highly performing pair that would appear in the leftmost part of the feature selection tree at exactly one level above the leaves.



(a) **(b)** **(c)**

Figure 12. Visualization of the accuracy of features: (a) singleton sets, (b) pairs, (c) difference.

Figure 12(c) displays the difference of (a) and (b). That is, it shows the change in rank order of performance, where again dark means "good"; dark blocks are those features whose performance as a member of pair increased in rank order, compared with their placement in the rank order of singleton performance. Careful comparison of Figure 12(c) against Figure 8 shows that the right central area of the frame (roughly, the area of the instructor's body and outstretched arm in the "announcement" example) is encoded in Figure 12(c) with lighter colors, indicating redundancy, and is missing any selected macro-blocks in Figure 8, indicating relatively high error rate. In a second example, we illustrate this more exactly in Figures 13 and 14, by tracing the performance of an evolving feature subset. Figure 13 shows a 2-feature subset, consisting of macro-block 16 (in the top row at right, as each row has 20 blocks) and macro-block 44 (in the third row at left). They have become a 2-feature subset because as singleton features they were ranked by performance in neighboring positions 34 and 33, respectively, out of 300 singletons. However, their performance together as a pair improved, and was ranked in position 5 out of 150 2-feature subsets.

It is not hard to see why: the pair is spatially separated, and therefore less likely to detect redundant information. For example, in the discussion example in Figure 13, macro-block 16 always views the bright classroom wall even when the camera pans, whereas macro-block 44 intercepts the colored clothing of the students. Similarly, in the demo example in Figure 13, macro-block 16 always views the black border of the computer screen, whereas macro-block 44 views varying content over

time. (Both blocks, however, see essentially the same information in the handwritting and announcement examples.)

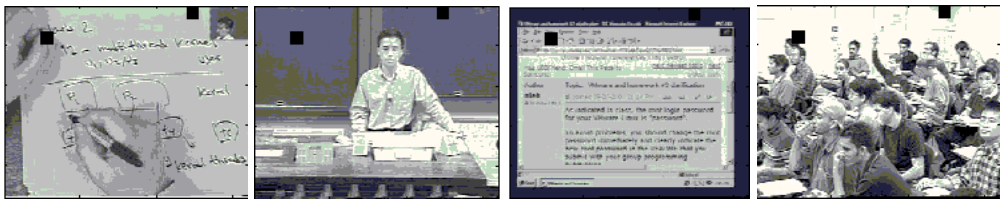


Figure 13. The 2-feature subset consisting of macro-block 16 (top row) and macro-block 44 (third row); it performs well.

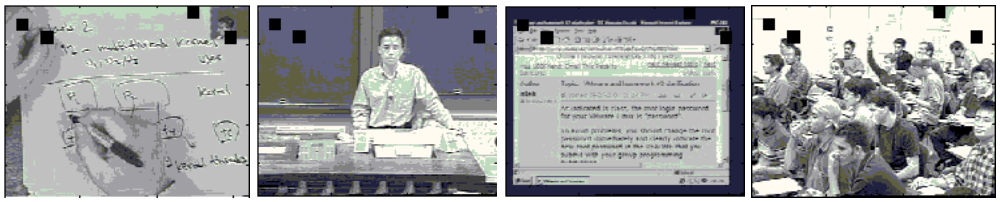


Figure 14. The 4-feature subset consisting of the 2-feature subset in Figure 12, together with its successor in rank order, consisting of macro-blocks 22 and 59; it performs poorly.

Nevertheless, at the next level of the Sort-Merge, as shown in Figure 14, this fifth-best pair consisting of blocks 16 and 44, is merged with the neighboring sixth-best pair consisting of blocks 22 (in the second row at extreme left) and 59 (in the third row at extreme right), to form a feature subset of size 4. The performance suffers dramatically, however, and is ranked only 53 of 75 four-element subsets. Again, a glance at their spatial distribution shows why. The pairs cover very similar areas of the frame; they are redundant, rather than complementary.

It is necessary to point out that Sort-Merge feature selection is necessarily heuristic, and is more likely to be successful in precisely those domains for which it was designed: those with very many features. Like all greedy methods, it can and does occasionally suffer from the inability to more intelligently look ahead in its merging choices, and will occasionally fail to pick complementary feature subsets to merge. For example, if the best two singleton feature subsets were spatially redundant but not perfect in their classification, and the third best singleton feature was spatially complementary to the first two and again not perfect, the method as currently developed would

greedily merge the first two singletons, even though a merge of either of those two with the third is more likely to increase performance. Once merged, this pair can never be undone, potentially "wasting" one or other of its component features.

This problem becomes more acute as the number of features becomes smaller and there is less likelihood that (many) other merges will happen to be complementary. But if the number of features becomes very small, more traditional methods then become more tractable and more appropriate. Likewise, as the training set becomes smaller, performance scores become more coarsely quantized, and more feature subsets appear to be redundant. However, under this circumstance, other feature selection algorithms suffer similar degradations. We currently are exploring these issues, by incorporating additional inexpensive merge heuristics that are sensitive to where in the training set a feature's error occurs, and to the spatial relationships among the merged feature subsets' macroblocks.

5.2. Fast video retrieval using multi-level feature selection

The second experiment uses multi-level feature selection to demonstrate its improvement of the efficiency of video segment boundary detection.

5.2.1. The basic refinement task

Table 5 summarizes the results of the application of the method detailed in section 4.2 to the entire instructional video. The method begins by seleting 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.

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 5. Classification 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.

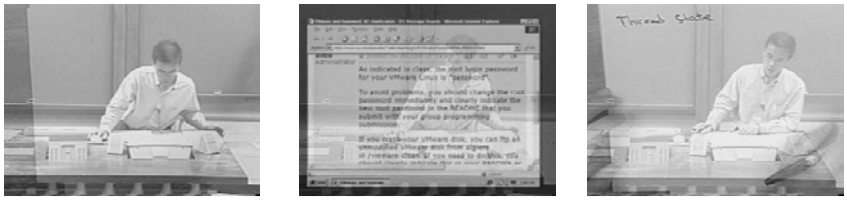
We now proceed through four additional rounds of refinement. We keep the neighborhood of examination constant at Li = 6, meaning that the 3 frames before and after any ambiguous frame are also re-examined and reclassified with the more costly but more accurate classifiers that use the larger feature 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 comparision 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 Sort-Merge classifier of Section 5.1 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.

5.2.2. Further analysis and discussion

Figure 15 shows three frames that remain (properly) ambiguous even after refinement ends using a 32-feature classifier. Table 6 shows their Mahalanobis likelihood for each category. These three frames are the exact midpoints of three dissolves, and their Mahalanobis likelihoods are properly much lower than those of more usual frames.



Data 240 Data 251 Data 1410

Figure 15. Dissolves in uncertain segmentation areas, using 32 macro-blocks

Data number (every other I-frame number)	Handwriting	Announcement	Demo	Discussion
240	0.0000	0.0000	0.0000	0.0444
251	0.0000	0.0152	0.0000	0.0000
1410	0.6013	0.0000	0.0000	0.0227

Table 6. Mahalanobis likelihood (in units of 10^{-6}) of dissolves belonging to the four classes.

Revisiting Table 5, 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, R_1 = 2 and 4, representing features spaces with 12 and 24 components, respectively (as each feature records a macroblock'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 advantanges of the Sort-Merge feature selection algorithm 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.

5.3. Lazy evaluation of unanticipated on-line queries

The third experiment demonstrates the construction of a feature subset using the Sort-Merge feature selection algorithm, for a category that has not been previously determined and computed off-line. We use two very different sources of video data.

5.3.1. On-line video retrieval of instructional video

Our first task is to discriminate between two subcategorizes of the handwriting category of the instructional video. The experiment is based on the intuition that handwriting frames are probably of more use to the student 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.

We noted that there were a total of 69 segments of "emphasis" and "non-emphasis" handwriting in the video, totaling about 1 hour. We simulated the user query by selecting and labeling 10 positive example sequences of ten frames each, and an equal number of negative ones. For classification, we limited the feature subset to 16 macroblocks.

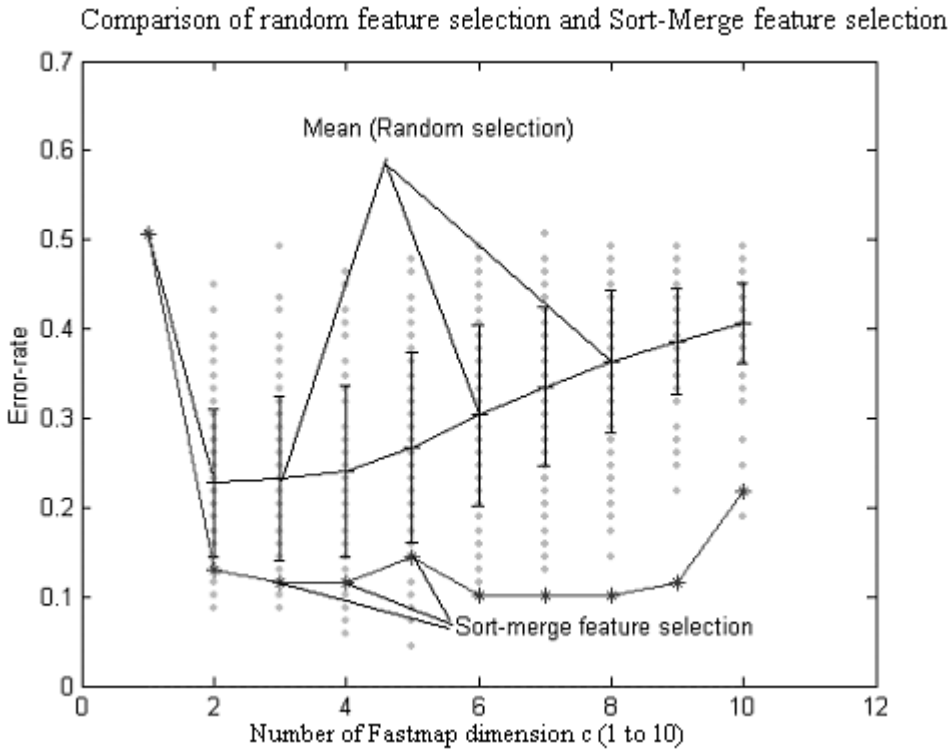


Figure 16. Retrieval results for random feature selection and Sort-Merge feature selection.

Figure 16 shows the error rate of labeling the 69 segments based on a classifier using 16 features selected randomly; this was repeated 100 times. The error bars are drawn at the mean error plus one standard deviation. In contrast, asterisks show the error rate of retrieval using 16 features selected by

the Sort-Merge feature selection algorithm. This query requirement is based on the lazy evaluation of a semantic query defined by example, but this retrieval is possible at low cost, and is again stable with respect to the choice of dimensionality reduction parameter, c.

The second example illustrates a somewhat different exercise. We show that lazy evaluation can proceed by the method of boundary refinement, that is, these two applications can be combined. In this task we attempt to retrieve frames in the category of announcement using only a very limited feature subset, and only a very limited example set, of a size comparable to what a user can be reasonably expected to provide: 80 frames, 40 positive and 40 negative examples each. Figure 17 shows the retrieval result for feature subsets of size 2, 4, and 8 (with Fastmap dimension c held equal to 4). However, it is important to note that the Sort-Merge method is progressive, and than only 0.02 of the frames were need to be re-examined by the 4-feature classifier, and only 0.01 needed the full power of the 8-feature classifier.

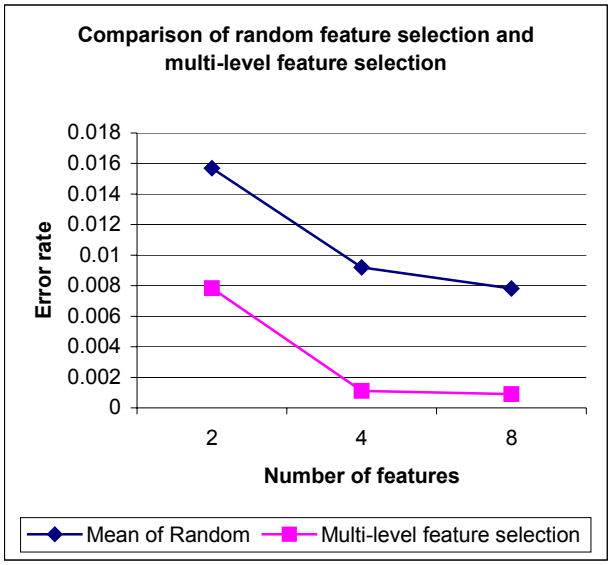


Figure 17. Retrieval results for random feature selection and multi-level feature selection

5.3.2. On-line video retrieval of baseball video

Our second example of on-line video retrieval applies the Sort-Merge feature selection algorithm to sports videos. As shown in Figure 18, we are interested in defining "pitching frames" that look like

Figure 18(a); the rest of the video has many different competing image types, some of which are shown in Figure 18(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. We classify frames as to their binary membership in the category "pitching". We ran two experiments. The first did not use any prior temporal segmentation or other pre-processing. Figure 19 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. In general, the error rate is halved. Figure 20 shows as black boxes those macro-blocks selected by the Sort-Merge feature selection under several choices for the number of features.

(a) **(b)**

Figure 18. Task: retrieve "pitching"(a) from an entire video with competing image types(b).

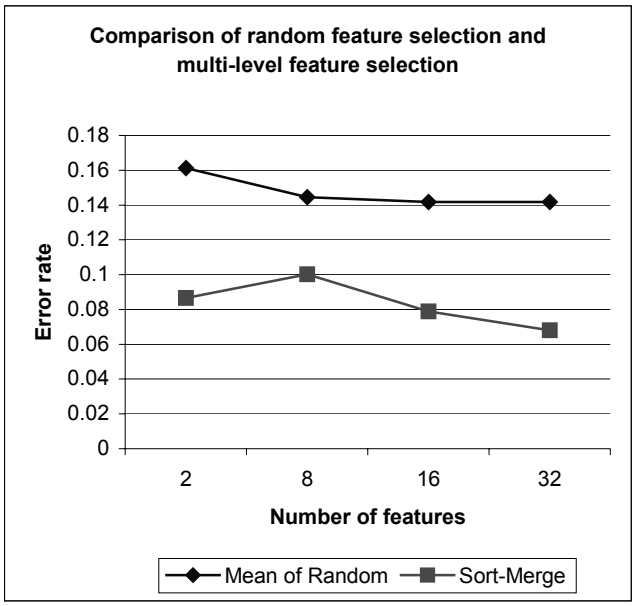


Figure 19. Retrieval results for random feature selection and multi-level feature selection

Figure 20. Selected features (macroblocks) using Sort-Merge feature selection.

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 [35], simple accuracy is often an insufficient measure, so we compare the feature selection algorithms based on recall and precision in Figure 21. Precision is nearly perfect, and recall is better by a factor of 2, compared to random feature selection.

6. Conclusion

We have presented a novel feature selection algorithm that is well-suited to the difficult domain of video frame classification. It relies on three underlying algorithms that are well-adapted to this large and continuous-valued domain and work together in linear time: Fastmap for dimensionality reduction, Mahalanobis distance for classification likelihood, and the Sort-Merge approach to combining relevant and non-redundant feature subsets into more accurate ones. Together, they

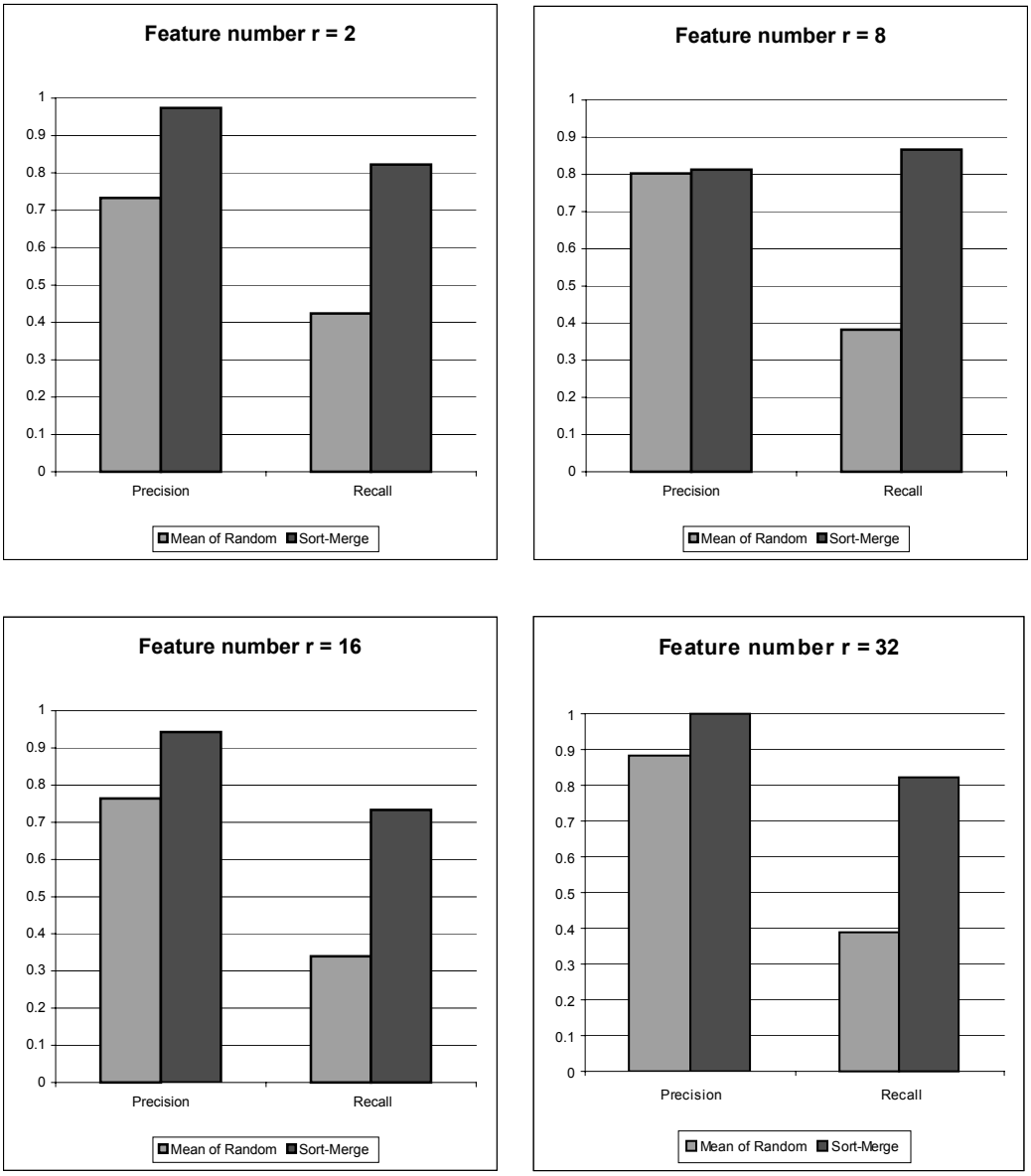


Figure 21. Precision and recall of random feature selection vs. multi-level feature selection.

combine the performance guarantees of wrapper methods with the speed and logical organization of filter methods. The method is shown to be linear in the number of features and in the size of the training set, and it constructs a complete hierarchy of increasingly accurate classifiers. It therefore leads to new feasible approaches for rapid video segment boundary refinement and for lazy evaluation of on-line queries. We have illustrated its performance using two long videos from

different genres. We plan to investigate its utility both across a library of videos of these genres, and also on other genres such as situation comedies, which share a similar recurring category structure.

Acknowledgements

The authors gratefully acknowledge fruitful discussions with Prof. Tony Jebara about computational costs in feature selection, and thank Dongqing Zhang for the digitized sports video. We thank one of the referees for pointing out the example used at the end of section 5.1.2. This research was supported in part by NSF grant EIA-00-71954.

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] R. Brunelli, O. Mich, C. M. Modena, "A Survey on Video Indexing", *J. of Visual Communication and Image Representation 10, pp.78-112, 1999.*

[3] Bakker, E., Lew, M., "Semantic Video Retrieval Using Audio Analysis", *International Conference on Image and Video Retrieval, Lecture Notes in Computer Science, vol. 2383, Springer 2002,* pp.260-267.

[4] A. G. Hauptmann, M. A. Smith, "Text, *Speech and Vision for Video Segmentation: The Informedia Project*", *In AAAI-95 Fall Symposium on Computational Models for Integrating Language and Vision, November, 1995.*

[5] Rainer Lienhart and Wolfgang Effelsberg, "Automatic Text Recognition for Video Indexing", *Proceedings of ACM Multimedia 96, Bosten, MA, Nov. 1996.*

[6] T. Sato, T. Kanade, E. Hughes, M. Smith, S. Satoh, "Video OCR: Indexing Digital News Libraries by Recognition of Superimposed and Caption ", *ACM Multimedia Systems Special Issue on Video Libraries, February, 1998.*

[7] 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.

[8] K. Tieu and P. Viola, "Boosting Image Retrieval", *Proceedings of IEEE Conference of Computer Vision and Patter Recognition, 2000.*

[9] F. Schaffalitzky and A. Zisserman, "Automated Scene Matching in Movies", *International Conference on Image and Video Retrieval, Lecture Notes in Computer Science, vol. 2383, Springer 2002,* pp.186-197.

[10] Kyungsu Kim, Junho Choi, Namjung Kim, and Pankoo Kim, "Extracting Semantic Information from Basketball Video Based on Audio-Visual Features", *International Conference on Image and Video Retrieval, Lecture Notes in Computer Science, vol. 2383, Springer 2002,* pp.278-288.

[11] Gang Wei and Ishwar K. Sethi, "Omni-face detection for video/image content description", *ACM Multimedia Workshops*, *2000*, pp.185-189.

[12] M. R. Naphade, T. Kristjansson, B. Fery and T. S. Huang, "Probabilistic Multimedia Objects Multijects: A novel Approach to Indexing and Retrieval in Multimedia Systems", *Proceedings of IEEE International Conference on Image Processing, vol. 3,* pp.536-540.

[13] H. Zhang, C. Y. Low and S. W. Smoliar, "Video Parsing and Browsing Using Compressed Data", *Multimedia Tools and Applications, vol. 1, 1995*, pp.89-111.

[14] J. Lee and B. W. Dickinson, "Multiresolution video indexing for subband coded video databases", *SPIE Proceedings: Image and Video Processing, vol. 2185,1994,* pp. 162-173.

[15] 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.

[16] 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.

[17] Chih-Chin Liu and Arbee L. P. Chen, "3D-List: A Data Structure for Efficient Video Query", *IEEE Transactions on Knowledge and Data Engineering, vol. 14, 2002*, pp.106-122.

[18] Gulrukh Ahanger, Dan Benson, and T.D.C. Little, "Video Query Formulation", *SPIE Proceedings: Image and Video Processing, vol. 2420,1995,* pp. 280-291.

[19] John R. Smith, Savitha Srinivasan, Arnon Amir, Sanker Basu, Giri Iyengar, Ching-Yung Lin, Milind Naphade, Dulce Ponceleon, Belle Tseng, "Integrating Features, Models, and Semantics for TREC Video Retrieval", *NIST TREC-10 Text Retrieval Conference, Gaithersburg, Maryland, November, 2002.*

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

[21] Caruana, R.A., & Freitag, D, "How useful is relevance", *Working notes of the AAAI Fall Symposium on Relevance, 1994,* pp 25-29.

[22] Almuallim & Dietterich, T. G., "Learning with many irrelevant features", *Proceedings of Ninth National Conference on Artificial Intelligence, 1991,* pp. 547-552.

[23] Cardie, C., "Using decision trees to improve case-based learning", *Proceedings of the tenth International Conference on Machine Learning, 1993,* pp 25-32.

[24] Kubat, M., Flotzinger, D., & Pfurtscheller, "Discovering patterns in EEG signals: Comparative study of a few methods", *Proceedings of the 1993 European Conference on Machine Learning,* pp. 367-371.

[25] Tony Jebara & Tommi Jaakkola, "Feature selection and dualities in maximum entropy discrimination. Uncertainty in Artificial Intelligence", *Proceedings of the Sixteenth Conference (UAI-2000),* pp. 291-300.

[26] 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.*

[27] Langley, P., & Sage, S., "Oblivious decision trees and abstract cases", *Working Notes of the AAAI 94 Workshops on Case-Based Reasoning,* pp. 113-117.

[28] Singh, M., & Provan, G. M., "Efficient learning of selective Bayesian network classifiers", *Proceedings of the Thirteenth International Conference on Machine Learning 1996*.

[29] Singh, M., & Provan, G. M., "A comparison of induction algorithms for selective and non-selective Bayesian classifiers", *Proceedings of the Twelfth International Conference on Machine Learning, 1995,* pp. 497-505.

[30] Koller, D. & Sahami,M., "Toward optimal feature selection", *Proceedings of the Thirteenth International Conference on Machine Learning 1996.*

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

[32] 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.

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

[34] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall International, Englewood Cliffs, NJ, 1980.

[35] 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.