LIF TREC VIDEO 2009 High Level Feature Extraction Using Genetic Fusion

Stéphane Ayache
Laboratoire dInformatique Fondamentale de Marseille
UMR 6166 - CNRS, Université de la Méditerranée, Université de Provence
stephane.ayache@lif.univ-mrs.fr

October 26, 2009

Abstract

This paper describes our participation to the TRECVID 2009 challenge [5]. This year, we focused on an optimized fusion by means of Genetic Algorithm. We have implemented a classical approach for concept detection in video shots based on low-level and intermediate features extraction, supervised classifiers and fusion process. We compare the genetic fusion with usual late fusion (sum or weighted sum) in order to combine a lot of classifiers output. We show that, empirically, genetic fusion can achieve reasonable performance although it tends to overfit the annotation data.

1 Introduction

The High-Level semantic retrieval task concerns features or concepts such as "Indoor/Outdoor", "People", "Speech" etc., that occur in video databases. The TRECVid HLF task [6] contributes to work on a benchmark for evaluating the effectiveness of detection methods for semantic concepts. The task of high-level feature extraction is as follows: given the feature test collection composed of hundred of hours of videos, the common shot boundary reference for the feature extraction test collection, and the list of feature definitions, participants return for each feature the list of at most 2000 shots from the test collection, ranked according to the highest possibility of detecting the presence of the feature. Each feature is assumed to be binary, i.e., it is either present or absent in the given reference shot.

We describe our participation to the TRECVID'09 HLF task where we focused on an optimized fusion using Genetic Algorithm. We have implemented a classical approach for concept detection in video shots based on low-level and intermediate features extraction, supervised classifiers and fusion process. We compare the genetic fusion with usual late fusion (sum or weighted sum) in order to combine a lot of classifiers output. In next section we present the feature with used to represent video shots. Section 2.3 presents briefly the purpose of Genetic Algorithm. Section 3 describes our fusion approach to merge classifiers by GA. Then, in section 4, we show the runs we submitted and discuss about their relative performance.

2 Feature extraction

We performed visual analysis at several level of granularity from global to fine blocs analysis, as well as various semantic level. Our low-level feature extractors first split images on overlapped blocs to form a grid of N $\times$ M blocs. For our submissions, we chose N and M such as we obtained a satisfying trade-off between classification performance and time computing. The analysis first treats each keyframe to extract several feature vectors, secondly, for some of them, merge features by concatenation and normalization in order to form a single feature vector.

2.1 Low-level features

At global level, we consider classical color and texture features. Color is represented by a 3-dimensional histogram on RGB space. We discretize the color space to form a 4x4x4 bins histogram. Texture information is described with Gabor bank of filters, we used 8 orientations and 5 scales. Finally, global features are normalized and concatenated on a 104 dimensions vector.
We also extracted color and texture features at bloc levels, features obtained from each bloc are then concatenated to form a rich description of keyframes :

Color (1) : is represented by 3x3x3 3D histogram on a grid of 8x6 blocs. The overall local color feature vector has 1296 dimensions.

Color (2) : is represented by the two first moments on a grid of 8x6 blocs. This local color feature vector has 432 dimensions.

Edge Direction Histogram : is computed on a grid of 4x3 blocs. Each bin is defined as the sum of the magnitude gradients from 50 orientations. Thus, overall EDH feature has 600 dimensions. EDH feature is known to be invariant to scale and translation.

Local Binary Pattern : is computed on grid of 2x2 blocs, leading to a 1024 dimensional vector. The LBP operator labels the pixels of an image by thresholding the 3x3-neighborhood of each pixel with the center value and considering the result as a decimal number. LBP is known to be invariant to any monotonic change in gray level.

2.2 Intermediate features

We call intermediate features those able to abstract some low-level features in order to go further on semantic representation. Such a first kind of feature, called “Semantic” is based on a ”bag of concepts” approach: we consider each bloc from keyframes which are relevant for a concept, as relevant for this concept too. This is a very strong assumption but it could be reasonable depending of the concepts. Thus, we use existing concepts annotations (from a part of the learning set) at global level, to train SVM classifiers at blocs level, where blocs are represented with moments color and edge direction histogram features. Then blocs of keyframes are classified using models of all the concepts, leading to $nb\_blocs \times nb\_concets$ classification scores per keyframe. The final Semantic feature is defined by the sum of scores on nb_blocs for each concepts, leading to a $nb\_concepts$ dimensional feature.

A second intermediate feature, we call “Percept”, is based on outputs of classifiers trained at bloc level with various (currently 15) ”intermediate concepts” such as Sky, Greenery or Water. The Percept feature vector is formed by the concatenation of such $15 \times N \times M$ normalized scores. Our Percept approach is presented in [2].

2.3 Other Features

We also used other visual features (SIFT, PEF, SURF), audio features (MFCC) and motion features (Optical flow) provided by the IRIM group [1]. The full list of video features provided by the IRIM group is presented in the corresponding notebook paper.

3 Genetic Algorithms

In optimization, Genetic Algorithms are a way of solving problems by mimicking the same processes as the nature do in order to resolve a problem by using pseudo-random searches to locate optimal solutions [4]. Genetic algorithms are implemented in a computer simulation in which a population of abstract representations (called chromosomes) of candidate solutions (called individuals) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0 and 1, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated with respect to a predefined criteria. Multiple individuals are randomly selected based on the criteria, then combined by means of crossover and mutation to form a new population which . The new population is then used in the next iteration of the algorithm. In our implementation, the algorithm terminates when a fixed number of generations has been produced. One of the contestable point of GA is that at the end of the process, the algorithm may or may not have found the optimal solution.

The simplest form of genetic algorithm involves three types of operators: selection, crossover and mutation.

Selection selects chromosomes in the population for reproduction. Depending of a criteria, chromosomes are more likely to be selected to reproduce.

Crossover randomly chooses a point (a locus) in the chromosome and invert the subsequences before and after this point between two chromosomes, and create two offspring. For example, the chromosomes 10000100 and 11111111 could be crossed over after the third indice in each to produce the two offspring 10011111 and 11100100.

Mutation randomly switch some of the bits in a chromosome. For example, the chromosome 00000100 might be mutated in its second position to yield 01000100. Similarly to the nature, mutation can occur with some probability, usually very small.

4 Genetic Fusion

We conducted most experiments on Late Fusion to combine output of classifiers trained independently from various available feature vectors. Considering the fact that Early Fusion is highly time consuming in the presence of numerous features we conducted the based our fusion scheme on a Late Fusion.

Several methods can be used for the weighting of the classifiers. A classical choice consists in a weighting based on the individual performance of the classifier. Whereas such a weighting scheme doesn't need any optimization process, performance of classifiers is not necessarily fully correlated with the contribution of classifiers on the fusion. An other possibility is to optimize the weight for maximizing the performance evaluated by cross-validation. In all the cases, these methods can be optimized with respect to a global (all the concepts) performance, or optimized by concept each one independently. We can also notice that a uniform weighting could be also a good choice as it tends to avoid overfitting of the data.

We call Genetic Fusion the process which aim at combining numerous classifiers output by optimizing their relative contribution (ie: the weights of a linear combination) using a GA optimization. Hence, the purpose of the optimization is to find weights of linear combination which maximize the performance of fusion on development set. The score reflecting the presence of a given concept on keyframe i is then defined by the following formula:

$$score(i) = \sum_{c=0}^{N} w_c \times g(x_c(i))$$

Where N is the number features to combine, $g(x)$ is a real value returned by a classifier and $x_c(i)$ is the representation of keyframe i on feature c.

In the context on Genetic Algorithm, solutions of a given problem are encoded as chromosomes and are evaluated with respect to a given criteria for selecting *best* chromosomes for creating the population in next iteration. Consequently, the two main points to model Genetic Fusion concern the representation and the criteria:

- We used a standard way of representing our problem using an array of bits for coding the N weights of the Genetic Fusion. Coding each weight with B bits conducts to chromosome representation as a vector of $N \times B$ boolean. For instance, $B = 4$ leads to weight varying from 0 to 15 and are then normalized such as $\sum_{c=0}^{N} w_c = 1$.
- We simply based the criteria for Genetic Fusion on infAP performance measure calculated on training set and development set. This choice might be not the best as it might leads to overfit the data.

We describe, above, an overview of our algorithm for Genetic Fusion:

1. Generate randomly $Nb_population$ chromosomes for initialization, with $Nb\_population$ as power of 2,
2. Evaluate chromosomes criteria and select $1/K$ chromosomes which maximize criteria,
3. Constitute randomly couples of chromosome,
4. Determine randomly crossover point for each couple,
5. Generate K chromosomes (child) by couple on which apply c mutations, with c randomly selected in $[0..c_{max}]$,
6. Last operation conducted to $Nb\_population$ chromosomes. Iterate $Nb\_iteration$ from 2.

5 Submitted Runs

We used respectively dev and test parts of the TRECVID'07 video collection for training classifiers and optimizing fusion weights. Once this step finished, we retrained all of our classifiers on dev+test TRECVID'07 collection and used optimized weights for classification of TRECVID 2009 test collection. We have considered two kind of classifiers: KNN are provided by IRIM group and SVM classifiers come from libSVM [3].

As LIF, we submitted 6 runs, but some need to be compared with IRIM group runs. We describe above the detail of these runs with their corresponding infAP performance.

IRIM1	0.1194	Genetic fusion of runs IRIM3, IRIM4, IRIM5 and IRIM6 with Context
IRIM2	0.1189	Genetic fusion of runs IRIM3, IRIM4, IRIM5 and IRIM6
IRIM3	0.0992	Genetic fusion of KNN classifiers on numerous visual and audio features
IRIM4	0.1220	Genetic fusion of SVM and KNN classifiers on selected visual and audio features
IRIM5	0.1116	Genetic fusion of SVM classifiers on selected visual and audio features
IRIM6	0.1014	Genetic fusion of KNN classifiers on selected visual and audio features
LIF1	0.0998	Genetic fusion of SVM classifiers on LIF only visual features with Context
LIF2	0.0972	Genetic fusion of SVM classifiers on LIF only visual features
LIF3	**0.1317**	Late fusion of runs IRIM3, IRIM4, IRIM5, and IRIM6
LIF4	0.0929	Late fusion of SVM and KNN classifiers on visual and audio features with Context
LIF5	0.0924	Late fusion of SVM and KNN classifiers on visual and audio features
LIF6	0.0943	Late fusion of SVM classifiers on LIF only visual features

Our best run LIF3 is a simple late fusion (without optimization) of IRIM group runs, it should be compared with the run IRIM2 which perform poorer despite of the genetic optimization. On this case, it seems that optimization had conducted to poorer generalization due to overfitting during the optimization process, which can be explain by the two-layer kind of optimization in IRIM3. In an other situation, the run LIF5 should be compared with run IRIM4 as they are based on same features. Here, the results show that genetic fusion leads to better performance than usual late fusion. In the same way, runs LIF2 and LIF6 show that genetic fusion performed a bit better than late fusion for an other set of features.

We have modeled a notion of context by integrating scores of related concepts, calculated form coocurrence of concepts on training corpus. We do not develop the details of our context model because experimentation does not show significant improvement. However, considering the two couples of runs LIF1 / LIF2 and LIF4 / LIF5, which are based on same features, we observe a small improvement in infAP performance.

6 Conclusions

This year, our main focus is on the exploration of genetic algorithm for fusion of classifiers. Our experimented show some improvement using Genetic Fusion in the case of one-layer fusion. However, we suspect that our implementation actually have overfitted data and unfortunately was not able to generalize as we expected. We think that Genetic Fusion could show significant improvement is the criteria for selection of chromosomes at each iteration is more carefully chosen, especially to avoid overfitting.

References

[1] http://mrim.imag.fr/irim.

[2] S. Ayache and G. Quénot. Image and video indexing using networks of operators. *J. Image Video Process.*, 2007(4):1–13, 2007.

[3] C.-C. Chang and C.-J. Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

[4] M. Mitchell. *An introduction to genetic algorithms.* MIT Press, Cambridge, MA, USA, 1996.

[5] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *MIR '06: Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, New York, NY, USA, 2006. ACM Press.

[6] A. F. Smeaton, P. Over, and W. Kraaij. High-Level Feature Detection from Video in TRECVid: a 5-Year Retrospective of Achievements. In A. Divakaran, editor, *Multimedia Content Analysis, Theory and Applications*, pages 151–174. Springer Verlag, Berlin, 2009.