HAL
open science

A factorized model for multiple SVM and multi-label classification for large scale multimedia indexing

Bahjat Safadi, Georges Quénot

To cite this version:

Bahjat Safadi, Georges Quénot. A factorized model for multiple SVM and multi-label classification for large scale multimedia indexing. 13th International Workshop on Content-Based Multimedia Indexing (CBMI), Jun 2015, Prague, Czech Republic. pp.1-6, 10.1109/CBMI.2015.7153610 . hal-01230720

HAL Id: hal-01230720

https://hal.science/hal-01230720v1

Submitted on 18 Nov 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Factorized Model for Multiple SVM and Multi-Label Classification for Large Scale Multimedia Indexing

Bahjat Safadi[1,2] Georges Quénot[1,2]
[1]Univ. Grenoble Alpes, LIG, F-38000 Grenoble, France
[2]CNRS, LIG, F-38000 Grenoble, France
{Bahjat.Safadi ,Georges.Quenot}@imag.fr

***Abstract*—This paper presents a set of improvements for SVM-based large scale multimedia indexing. The proposed method is particularly suited for the detection of many target concepts at once and for highly imbalanced classes (very infrequent concepts). The method is based on the use of multiple SVMs (MSVM) for dealing with the class imbalance and on some adaptations of this approach in order to allow for an efficient implementation using optimized linear algebra routines. The implementation also involves hashed structures allowing the factorization of computations between the multiple SVMs and the multiple target concepts, and is denoted as Factorized-MSVM. Experiments were conducted on a large-scale dataset, namely TRECVid 2012 semantic indexing task. Results show that the Factorized-MSVM performs as well as the original MSVM, but it is significantly much faster. Speed-ups by factors of several hundreds were obtained for the simultaneous classification of 346 concepts, when compared to the original MSVM implementation using the popular libSVM implementation.**

keywords : Multimedia indexing and retrieval, support vector machine, multi-learner approach, class imbalance.

I. Introduction

The aim of multimedia indexing is to automatically detect visual concepts in massive, continuously growing image/ video collections. In general, the indexing is achieved by supervised learning approaches for each target concept (binary classification). In which, a classifier is trained on positive and negative samples of the target concept to generate a classification model. This model is then used to predict the likeliness of new samples to contain the learnt concept.

Support Vector Machine (SVM) classifiers with RBF kernels [1], [2] are very popular and efficient for supervised classification, specially, in the case where the learned classes are not linearly separable. But they are not very well suited for highly imbalanced datasets. They process multiple concepts independently for training and for predicting. Ensemble learning extensions like multi-SVM approach (MSVM) [3] efficiently handles the class-imbalance problem that usually appears when indexing large-scale multimedia datasets. The main idea of MSVM is to split the training set into m balanced subsets and train a SVM classifier on each subset, thus generates m different models. For each sample in the test set, these models are used to predict its likelihood to contain the target concept separately, and a fusion function is applied on the m predicted scores to generate its final score. However, a drawback limitations of MSVM are crucial as they need huge processing times, especially at the prediction phase. In this paper, we propose some improvements to the MSVM approach, which makes it faster mainly at the prediction phase and usable almost in a real time.

The availability of specific and highly optimized mathematical packages like the Netlib BLAS (Basic Linear Algebra Subprograms [4]) or the Intel VML (Vector Math Library) and/or of specialized parallel architectures like GPU (Graphical Processing Units) boards can bring huge improvements in computing speed and partly solve the related scalability issues. Such speed-up is generally obtained at the price of a more complex implementation and often requires a significant redesign effort. Considering their specificities (e.g. some are more sequential or iterative), not all methods can easily take advantage of these possibilities.

The proposed work is in the context of methods based on small to intermediate size descriptors combined with classifiers using non-linear kernels. It is also more interesting in the case in which several concepts are annotated for each training sample; the annotation does not have to be dense however. This work also addresses the imbalanced data problem and is designed for taking advantage of optimized mathematical libraries (BLAS and VML). Though not tested yet, it should work well too with GPU implementations of such libraries.

We propose the following improvements to the original MSVM approach in the context of multiple concept classification:

- speed-up of the SVM prediction function using matrix multiplication;
- fusing of SVM models corresponding to a single target concept and to different splits into a single model, with a probability normalization (Platt [5]) extension;
- multi-label indexing, based on fusing of SVM models corresponding to different target concepts into a single and global model.

All these improvements permit an efficient use of BLAS1/2/3 and VML at many levels yielding a very significant speed-up.

The remainder of this paper is organized as follows: section II discusses related work; section III presents the proposed methods for improving the MSVM implementation; section IV presents experimental evaluation of the proposed contributions; and section V draws some conclusions and discusses perspectives.

II. Related work

A. Optimizing the prediction of SVM

In the last decade, the optimization methods for SVM training has been intensively studied, including kernel approximation approaches [6], [7], greedy basis selection [8], and online SVM solvers [9]. Recently, Jose et al. [10] proposed an approximate solver to reduce the prediction time of SVM, by reformulating the problem or by using derivatives of the SVM. However, in this paper we focus on reducing the prediction time of the SVM approaches based on RBF-kernel. We propose an algorithm for speeding up the SVM prediction function and apply it with the MSVM approach, as it is mostly used as a solution for indexing imbalanced datasets [3].

B. Class-imbalance problem

The most widely used approaches for handling the class-imbalance problem are the re-sampling methods, which are used for balancing the datasets before SVM training. Re-sampling methods have been successfully applied for training SVMs with class-imbalanced datasets in different domains [11], [12], [13], [14], [3].

The alternative approach presented in [12], which mainly works by selecting randomly a subset from the negative (i.e. majority) samples with a size comparable to that of all positive (i.e. minority) ones, then train a classifier on this subset. Support cluster machines (SCMs) method [11] can be viewed as another focused re-sampling method for SVMs. This method first partitions the negative examples into disjoint clusters by using the kernel-k-means clustering method. Then it trains an initial SVM model using the positive examples and the representatives of the negative clusters (i.e. the data examples representing the cluster centers). These approaches may lead to loss of information, especially when the dataset is of large-scale and the learnt concepts are sparse, since many informative majority class samples may be ignored. Thus, it is possible to balance this loss by making several selections on the majority class set, and merging the outputs of different classifiers built from these subsets.

Ensemble learning has also been applied as a solution for training classifiers with imbalanced datasets [13], [14], [3]. Among the state-of-the-art methods in this scenario, we are interested in the multi-learner approach based on SVM (denoted as MSVM) presented in [3]. The authors have showed the ability of MSVM to increase the classification performance of the video indexing system. They gave a full evaluation and comparison of their approach versus several single- and multi-learner approaches (e.g. logistic regression and SVM) for the concept indexing task in video documents. They have compared their method with different approaches, such as [12], [14], and concluded that multi-learner approach based on SVM is designed to address best the problem of sparse concepts in large-scale multimedia datasets.

Most of the reviewed multi-learner approaches are based on fusing the classification scores; this needs a prediction step of all test samples with each learnt model. In large-scale data sets, this is crucial since the prediction time of the all samples may be larger than the time for learning the models themselves. In this paper, we propose an efficient multi-learner approach based on a factorization of SVM models.

III. The proposed method

A. Accelerating the SVM prediction function

When using SVM for multimedia indexing with large-scale data sets, the prediction step is computationally the most expensive step. In this section, an implementation of SVM prediction for n test samples ($Z = (z_j)_{(1 \leq j \leq n)}$) against a SVM model with m SVs ($X = (x_i)_{(1 \leq i \leq m)}$) is proposed. Let the test set length to be $n \times d$, and the SVM-model length for a learnt concept to be $m \times d$, where d is the size of the used descriptor. The prediction function requires the computation of the full ($m \times n$) distance matrix between all descriptors and SVs. Then, a merging function is applied on the m distances of each sample to assign a prediction score or a probability for the sample to contain the learnt concept. The computation of the distance matrix is the most expensive computational step in the SVM prediction function.

Basically, the square Euclidean distance, between two vectors used in the RBF kernel, is computed as follows: $\|x_i - z_j\|^2 = \sum_{k=1}^{k=d}(x_{ik} - z_{jk})^2$. This can be rewritten as:

$$\|x_i - z_j\|^2 = \left(\sum_{k=1}^{k=d} x_{ik}^2\right) + \left(\sum_{k=1}^{k=d} z_{jk}^2\right) - 2\left(\sum_{k=1}^{k=d} x_{ik} z_{jk}\right) \quad (1)$$

The advantage of this form of the Euclidean distance is that it decomposes the computation of a distance matrix between sets of vectors into smaller steps which are faster to compute. The first two sums have to be computed respectively m and n times only for the $m \times n$ distances and this can be done efficiently by calling the BLAS1 scalar product routine. The second sum has to be computed $m \times n$ but all the values can be computed at once and very efficiently using the BLAS3 matrix-matrix multiplication routine. These are simply the elements of the $X.Z^T$ matrix. Once the set of distances are computed and multiplied by the γ value (i.e. the free parameter of the Gaussian radial basis function), the kernel $K(x_i, z_j)$ values can be obtained as a K_{ij} matrix by applying a vectorized exponentiation operator (such as the one available in the Intel VML library) on its $m \times n$ elements.

The computation of the set of scalar products $x_i.z_j$, using matrix multiplication, can also be used when computing linear and polynomial kernels. The computation of the first two sums is not necessary in this case, and vectorized operators can still be used for the final kernel computations in the polynomial case. Unfortunately, the RBF kernel with χ^2 or L_1 distances still cannot benefit from the matrix multiplication trick. However, the work presented in [15] concluded that applying a power transformation on low-level descriptors makes the Euclidean distance as effective as or even better than χ^2 and L_1.

Once the matrix of $K(x_i, z_j)$ values are computed, the decision function values for the set of vectors can be obtained, by multiplying the transposed K_{ij} matrix by the vector a ($y_i \alpha_i$) of the m values (y_i, α_i are the target and the contribution of the ith training example to the model, respectively), which can be

efficiently done using the BLAS2 vector-matrix multiplication, and adding the bias term ρ:

$$(f(z_j))_{(1 \leq j \leq n)} = f = \rho.1_n + K^T.a \quad (2)$$

with 1_n being a vector of n elements all equal to 1.

Finally, if probability normalization is required, the application of the sigmoid function (on the set of z_j vectors) can also be accelerated using vector operators.

Though all the mentioned linear algebra and vector operation optimizations lead to some performance improvement, the most important and significant one is, by far, the use of matrix multiplication (BLAS3) to accelerate kernel computations. It is particularly efficient when the dimensions of the considered matrices are large (number of components in the descriptor, number of support vectors and number of test samples). For maximum efficiency, a minimum number of test vectors must be processed at one. If the number of test vectors is very large, these can be processed by blocks of this minimum size. In practice, blocks of $1K$ test vectors are large enough to optimally benefit of BLAS3 speed-up. Beyond this, the computation is not significantly faster but it requires significantly more memory. In the case in which test vectors have to be processed one by one (this may happen during the training phase or if prediction is needed for a single test sample), BLAS2 matrix-vector operations can be used instead of BLAS3 matrix-matrix ones, leading to a less substantial but still significant speed-up.

Though the structure of the SVM training algorithm does not permit a simple use of the matrix multiplication trick directly within it as for the SVM prediction, optimizing the prediction function can lead to some acceleration of the training phase as well since, during it, the SVM algorithms apply the prediction function for learning the weights of the SVs (i.e. the α vector), as well as for tuning the Platt's parameters by cross-validation.

B. The factorized MSVM Model (FMSVM)

Multiple-SVM [3] approach handles the class imbalance by building a number of balanced elementary classifiers, and by fusing their predictions. The elementary classifiers do not actually need to be perfectly balanced. SVM classifiers do tolerate a moderate amount of imbalance, and a compromise can be found between the number of elementary classifiers and the amount of imbalance left to them.

In the original MSVM approach, the authors fuse the scores after their normalization as probabilities using a sigmoid function (Platt's method). They consider five possible fusion functions of the score of the elementary classifiers: the minimum, the harmonic mean, the geometric mean, the arithmetic mean and the maximum. We found out that all of them were roughly equivalent, except the maximum one that performed significantly less well (see the experiments section).

The MSVM considers score fusion after probability normalization for practical reasons (these were directly given by the `libsvm` tool [16] for each elementary classifier, but they could have considered as well as the fusion of scores directly produced by the initial decision functions and then perform a probability normalization. This would have been more complicated, however, since the computation of the global B and A coefficients (i.e. the symmetry point and the slope at this point in a sigmoid function respectively) by cross-validation would have require to manage cross-validation sub-problems' splits jointly with the multiple classifiers' splits. We propose here a simple method for computing global A and B sigmoid coefficients directly from the A_k and B_k sigmoid coefficients computed separately for the elementary classifiers by cross-validation.

There is no a priori reason for the score fusion within the MSVM approach is better before and after sigmoid normalization and experiments show that the overall performance is also equivalent. This is probably due to the fact that the SVM approach gives similar distributions to all elementary classifiers. Less choice is given for the fusion function since harmonic and geometric means are not suited for values that can be negative. The minimum and maximum values could also be used but would be less natural. Finally, only the arithmetic mean seems meaningful in this case. It is also the only one that can lead to a simplification of the computation of the global decision function.

Sigmoid normalization applied after score fusion has no effect on the ranking of the test samples since the sigmoid function is always increasing. It does have effect however in the case of further processing, for instance if the obtained global scores have to be fused again, considering results coming from different descriptors or from other learning methods also normalized as probabilities. We again found out that the method that we propose for the fusion of sigmoid parameters performed as well as the original one in the case of further descriptor and classifier fusion.

The fusion method that we propose is defined as follows. The elementary classifier decision functions are defined as:

$$f_k(z) = \rho_k + \sum_{x_i \in S_k} y_i \alpha_{ki} K(x_i, z) \quad (3)$$

and the associated probability functions are defined as:

$$p_k(z) = \frac{1}{1 + e^{A_k . f_k(z) + B_k}} \quad (4)$$

where $k \in [1, l]$ is the elementary classifier index and S_k the set of support vectors associated to the kth elementary classifier. Without loss of generality, we can consider the global set of support vectors ($S = \bigcup_{k=1}^{l} S_k$) and rewrite equations 3 as:

$$f_k(z) = \rho_k + \sum_{x_i \in S} y_i \alpha_{ki} K(x_i, z) \quad (5)$$

taking all α_{ki} for x_i in $S \setminus S_k$ equal to zero (this requires a global re-numbering of the support vectors). The arithmetic mean of the elementary decision functions can then be rewritten as:

$$f(z) = \rho + \sum_{x_i \in S} y_i \alpha_i K(x_i, z) \quad (6)$$

with ρ and the α_i being respectively the arithmetic mean of the ρ_k and of the α_{ki} for $k \in [1, l]$. This leads to a decision function, which is exactly of the same type as the one computed using a single and larger SVM. Though of the

same type, the MSVM fused function is different and though it does not benefit from a global convex optimization, it performs better in the case of highly imbalanced classes.

Considering the probability normalization, if needed, the elementary sigmoid functions can also be fused into a global one. The elementary sigmoid functions are defined by two parameters that can be taken as the abscissa of their symmetry point which is equal to B_k/A_k and the slope at this point which is equal to A_k. We simply propose to take the average of the abscissa of the symmetry points and of the slope at these points for the global sigmoid function. As the data are normalized before (by the SVM implementation) and after (by Platt's method), the elementary sigmoid are already quite "close" in general and fusing them is not only correct but is also likely to reduce the noise in their estimation. Finally, the global probability function is defined as:

$$p(z) = \frac{1}{1 + e^{A.f(z)+B}} \tag{7}$$

with A and B defined so that A and B/A are respectively the arithmetic mean of the A_k and of the B_k/A_k for $k \in [1, l]$. Once again, this leads to a formulation which is exactly of the same type as the one computed using a single and larger SVM.

Many computations are factorized in this approach. This is why we call this method *factorized* MSVM (FMSVM). The overall form is well suited for an efficient computation using BLAS routines. In particular, $K(x_i, z)$ is computed only once for all the x_i elements of the global set of support vectors while these may appear several times in the sets of support vectors of the different elementary classifiers. The factorization of the elementary classifiers is implemented via an accumulation in a hashed structure of the S_k, ρ_k, α_{ki}, A_k and B_k/A_k elements.

C. Factorized Model for Multi-Label Classification

The algorithm presented in the previous section can be extended for computing prediction values for a set of q predefined and learnt concepts. With slight modifications, the method used for factorizing computations for multiple elementary learners can also be applied for factorizing computations for multiple target concepts.

The multi-label classification method that we propose is defined as follows. The per concept classifier decision functions are learnt separately for each concept c, and defined as:

$$f_c(z) = \rho_c + \sum_{x_i \in S_c} y_{ci}\alpha_{ci}K(x_i, z) \tag{8}$$

where $c \in [1, q]$ is the concept index and S_c the set of support vectors associated to the classifier trained for the cth concept. For each concept, the $f_c(z)$ values can come, either directly from a classical SVM or from a FMSVM as presented in the previous section. The main difference with equation 3 is that a c index has to be added to the y labels but this has no practical implication in the computations. The associated probability functions are defined as:

$$p_c(z) = \frac{1}{1 + e^{A_c.f_c(z)+B_c}} \tag{9}$$

Again, without loss of generality, we can consider the global set of support vectors: ($S = \bigcup_{c=1}^{q} S_c$) and rewrite equations 3 as:

$$f_c(z) = \rho_c + \sum_{x_i \in S} y_{ci}\alpha_{ci}K(x_i, z) \tag{10}$$

taking all α_{ci} for x_i in $S \setminus S_c$ equal to zero. So far, the method is the same as for the MSVM fusion but we do not perform any fusion operation here. Instead we keep all the per concept results in parallel and we rewrite equation 2 in a vectorized form considering the set of concepts as:

$$(f_c(z_j))_{(1 \le j \le n)(1 \le c \le q)} = f = \rho^T.1_n + K^T.a \tag{11}$$

where ρ is the bias vector ρ_c of the q terms and a is the $q \times m$ matrix of the $y_{ci}\alpha_{ci}$ values. The BLAS3 matrix multiplication routine can be used once again for an efficient computation of the set of $f_c(z_j)$ values.

Considering the probability normalization, if needed, no fusion has to be performed and the sigmoid computation has to be done for each $f_c(z_j)$ value. These can be efficiently computed however using VML functions.

As in the MSVM fusion, many computations are factorized in this approach and the overall form is well suited for an efficient computation using BLAS routines. Once again, the implementation involves a hashed structure. The main difference is that the accumulation of values is replaced by a vectorization of them. We also refer to this method as *factorized* MSVM (FMSVM). The F in FMSVM may refer either to the factorization of multiple elementary classifiers for a single concept or to the factorization of multiple classifiers corresponding to multiple concepts or to the combination of both.

The proposed optimization requires that the γ parameter is the same for all concepts while it could in principle be tuned by cross-validation separately for each one. This is not a problem since the classification performance is not very sensitive to the exact value of this parameter and the optimal value is more related to the descriptor type and collection contents than to the target concepts. Furthermore, the estimation of this parameter can be noisy in the case of highly imbalanced classes and taking a single value for all concepts is a good way to reduce such noise. In the case one would want to use a separate γ_c parameter for each concept, the proposed method could still bring significant improvement: the factorization of the Euclidean distance computations could still be used while the second factorization corresponding to equation 11 could not. The first one is however the most important for speeding up the prediction, especially when applied with high dimensionality descriptors.

Practical implementation may face a problem of memory space, especially when indexing a large number of concepts (e.g. when indexing thousands of concepts). However, this problem can be solved by processing the data by blocks over the test examples and the concepts to index.

IV. Experiments

Experiments were conducted in the context of the TRECVid 2012 semantic indexing task [17]. The data collection

TABLE I. THE MAP VALUES ON THE TRECVID 2012 VALIDATION SET, USING THE DIFFERENT MSVM METHODS.

Descriptor	Size	PCA	Original MSVM	Factorized MSVM
hg	104	104	0.1155	0.1153
bov2_dsiftSC	8192	512	0.1701	0.1699
labm1x3x1024	3072	384	0.1328	0.1327
vlat_hog6s8_dict64	4096	512	0.1637	0.1603
dense_sift_k1024	1024	256	0.1290	0.1289
vlad	32768	640	0.1720	0.1717
mlhmslbp_spyr	26624	768	0.1461	0.1466
superpixel_color_sift	1064	256	0.1413	0.1409

consists of two large sets: the development and the test sets. The development set contains $400,289$ annotated video shots, while the test set contains $145,634$ shots. The evaluations were conducted on 346 concepts, which were provided by NIST and Quaero [1]. We have divided the development set into two parts: the training set (`train`, $197,185$ video shots) and the validation set (`val`, $203,104$ video shots). In this paper, we evaluate and compare the proposed factorized model and the original MSVM approach. For comparison with other classifiers, we refer to the work presented in [3], which compares between different classifiers and shows the effectiveness of the MSVM approach to handle the class-imbalance problem in large-scale multimedia datasets.

A. Video description

We have used several descriptors of different types and sizes, which have been produced and shared by various partners of the IRIM project of GDR-ISIS [18]. Most of the selected descriptors are based on the color histograms or on the bag of words approaches. In practice, we have used eight descriptors whose size and performance are given in table I. The performance is evaluated using the inferred Mean Average Precision on the 346 evaluated concepts.

All the used descriptors were optimized by applying a power law normalization and a dimensionality reduction using the principle component analysis (PCA) followed by a second power law normalization. The use of this optimization is motivated, as it makes the Euclidean distance outperform the χ^2 distance with RBF kernels [15]. The optimization of the hyper-parameter, the α exponents for the power laws and the k values for PCA, were tuned by cross-validation separately for each descriptor.

B. Effectiveness of the Factorized MSVM

In this section, we compare the performance of the indexing system using the original MSVM and the new proposed method (Factorized-MSVM). Both methods were applied with the Euclidean distance and the RBF-kernel. The performances of the systems were measured as the MAP value over the 346 concepts of TRECVid 2012, calculated on the validation set. Table I shows the system performance with the MSVM and Factorized-MSVM with the eight descriptors considered in this work. As the table shows, for each descriptor, the MAP values are not significantly different between both methods. We believe that the small difference comes from the floating point rounding (single precision was used) when fusing the scores in the original approach. Final scores for the test samples using the MSVM were produced by applying the arithmetic mean as a fusion function on the m predicted scores while in the Factorized-MSVM, the scores were computed only once.

This is a very interesting result. It validates that the proposed Factorized-MSVM method performs as well as or very close to the original MSVM. Thus, factorizing the models leads to similar results than fusing predicted scores.

C. The efficiency of Factorized-MSVM

All the experiments were done on a machine which has two quad-core processors running at 2.66 GHz and 32 Gbytes of Ram. The execution time depends upon the descriptor size. In table II, we report and compare the training and prediction times of the original MSVM approach (based on `libsvm` prediction) and of the proposed Factorized-MSVM approach. The table shows the processing times in hours for the semantic indexing task of 346 concepts on the validation set using the considered descriptors. Furthermore, it presents the processing times of the *multi-class* prediction function of the 346 concepts with each of the considered descriptors. A very significant improvement in efficiency of the proposed prediction method has been observed.

TABLE II. PROCESSING TIMES (IN HOURS FOR THE 346 CONCEPTS), ON THE `val` SET OF TRECVID 2012. G INDICATES THE SPEEDUP RATES OF FMSVM OVER MSVM.

Descriptor	MSVM		FMSVM		multi-class
	train	predict	train (G_t)	predict (G_p)	predict (G_m)
hg	13.3	42.1	05.3 (2.5)	1.6 (26)	0.10 (421)
bov2_dsift*	53.7	255.1	17.0 (3)	2.9 (88)	0.17 (1500)
labm*	76.1	274.1	26.0 (3.0)	1.9 (141)	0.16 (1713)
vlat_hog*	55.9	232.3	18.3 (3)	3.1 (75)	0.16 (1451)
dense_sift*	20.1	96.5	07.9 (2.5)	1.7 (54)	0.18 (536)
vlad	39.8	222.5	22.0 (1.8)	2.3 (99)	0.12 (1854)
mlhmslbp*	101.0	390.7	27.3 (3.7)	3.2 (124)	0.15 (2604)
superpixel*	31.4	95.5	11.4 (3)	1.5 (64)	0.11 (868)

With the original MSVM, the training times for all the concepts are smaller than the prediction times, with each descriptor. The training times range from 13 hours to 101 hours, while the predicting times it go from 42 hours up to 390 hours. This depends upon the descriptor size and upon the number of SVs found for each concept. However, the learning process using the Factorized-MSVM (which applies the new prediction algorithm), are between 5.3 to 27.3 hours, which are almost 3 times faster than the original MSVM. The improvement of the training process are due to the prediction that the `libsvm` does while training, since the training algorithm applies a cross-validation method (prediction included) while training. Furthermore, the prediction times are less than learning times and much faster than the prediction times with the original MSVM, between 1.5 and 3.2 hours. Finally, using the multi-class prediction (applied once for the 346 concepts), the prediction time decreases to only a few minutes, which is almost negligible when compared to the original prediction times and also to the descriptors' computation time.

The prediction times are also reduced to a single SVM. For instance, in the case of indexing the data set for a concept *Person*, the data is almost balanced. Thus, the system required only one learner according to the MSVM approach. It has been observed that FMSVM significantly reduced the processing times with all descriptors. The learning time was reduced about 3 to 5 times, depending on the descriptor type. The prediction times was reduced about 32 to 245 times, and became less than two minutes for each descriptor.

[1] http://www.quaero.org/

D. Performance on the TRECVid 2012 test set

The same experiments were conducted on the full collection of TRECVid 2012, in which we used the full development set for training set and made predictions on all the samples of the test set. These data sets are more challenging; they are larger, and thus they need more processing time. The improvement in efficiency of the proposed method is shown in table III. The reported results validate the efficiency of the proposed method when indexing large-scale collections, such as TRECVid 2012.

TABLE III. PROCESSING TIMES (IN HOURS FOR THE 346 CONCEPTS) ON THE TEST SET OF TRECVID 2012. G INDICATES THE SPEEDUP RATES OF FMSVM OVER MSVM.

	MSVM		FMSVM		multi-class
Descriptor	train	predict	train (G_t)	predict (G_p)	predict (G_m)
hg	48.4	58.1	21.2 (2.3)	1.5 (39)	0.13 (445)
bov2*	195.4	289.8	67.6 (2.9)	2.5 (116)	0.17(1705)
labm*	274.0	395.5	90.5 (3.0)	3.3 (122)	0.25 (1643)
vlat*	311.7	364.1	73.0 (4.7)	3.3 (109)	0.24 (1518)
dense_sift*	87.5	145.6	30.3 (2.9)	1.5 (94)	0.14 (1051)
vlad	173.7	231.0	77.5 (2.2)	3.6 (64)	0.28 (805)
mlhmslbp*	388.2	569.7	122.0 (3.2)	3.9 (145)	0.18 (3074)
superpixel*	119.3	134.3	42.3 (2.8)	2.0 (67)	0.15 (861)

V. CONCLUSION AND FUTURE WORK

We have proposed and evaluated a new algorithm for content-based multimedia indexing on large-scale dataset. The algorithm is an extension to the multi-learner based on SVM approach and is denoted as Factorized-MSVM. The approach is designed for taking advantage of optimized mathematical libraries (BLAS and VML). Moreover, the approach involves hashed structures allowing the factorization of computations between the multiple SVMs and the multiple target concepts. The method has been validated and evaluated using several descriptors in the context of the TRECVid 2012 semantic indexing task. Experimental results show that the Factorized-MSVM performs as well as the original MSVM, but it is much faster. It is about $39 - 145$ times faster than the prediction made using the original `libsvm` separately for each concept. By applying the multi-class prediction method, the prediction times over all the concepts reduces to a few minutes instead of several hours with the original MSVM. The learning time was accelerated as well. Furthermore, the method was validated in the single-learner case.

A released software package of our proposed Factorized-MSVM implementation can be found on 'http://mrim.imag.fr/FMSVM'. In future work, we plan to compare the Factorized-MSVM with the methods based on fisher-vector with linear classifier [19] and the Deep Convolutional Neural Networks approaches [20], [21].

ACKNOWLEDGEMENTS

This work was partly realized as part of the Quaero Program funded by OSEO, French State agency for innovation.

REFERENCES

[1] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. [Online]. Available: citeseer.ist.psu.edu/cortes95supportvector.html

[2] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[3] B. Safadi and G. Quénot, "Evaluations of multi-learners approaches for concepts indexing in video documents," in *RIAO*, Paris, France, Apr 2010, pp. 88–91.

[4] C. L. Lawson, R. J. Hanson, D. R. Kincaid, and F. T. Krogh, "Basic linear algebra subprograms for fortran usage," *ACM Trans. Math. Softw.*, vol. 5, no. 3, pp. 308–323, Sep. 1979.

[5] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," in *Advances in Large Margin Classifiers*, 1999, pp. 61–74.

[6] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[7] Q. Le, T. Sarlós, and A. Smola, "Fastfood?approximating kernel expansions in loglinear time," in *Proceedings of the international conference on machine learning*, 2013.

[8] S. S. Keerthi, O. Chapelle, and D. DeCoste, "Building support vector machines with reduced classifier complexity," *The Journal of Machine Learning Research*, vol. 7, pp. 1493–1515, 2006.

[9] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *The Journal of Machine Learning Research*, vol. 6, pp. 1579–1619, 2005.

[10] C. Jose, P. Goyal, P. Aggrwal, and M. Varma, "Local deep kernel learning for efficient non-linear svm prediction," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, 2013, pp. 486–494.

[11] J. Yuan, J. Li, and B. Zhang, "Learning concepts from large scale imbalanced data sets using support cluster machines," in *Proceedings of the 14th annual ACM international conference on Multimedia*. ACM, 2006, pp. 441–450.

[12] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, October 2007.

[13] X.-Y. Liu, J. Wu, and Z.-H. Zhou, "Exploratory undersampling for class-imbalance learning," *Trans. Sys. Man Cyber. Part B*, vol. 39, no. 2, pp. 539–550, 2009.

[14] M. A. Tahir, J. Kittler, K. Mikolajczyk, and F. Yan, "A multiple expert approach to the class imbalance problem using inverse random under sampling," in *MCS '09: Proceedings of the 8th International Workshop on Multiple Classifier Systems*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 82–91.

[15] B. Safadi and G. Quénot, "Descriptor Optimization for Multimedia Indexing and Retrieval," in *CBMI 2013, 11th International Workshop on Content-Based Multimedia Indexing*, Veszprem, HUNGARY, jun 2013.

[16] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[17] P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, B. Shaw, W. Kraaij, A. F. Smeaton, and G. Quénot, "Trecvid 2012 – an overview of the goals, tasks, data, evaluation mechanisms and metrics," in *Proceedings of TRECVID 2012*. NIST, USA, 2013.

[18] N. Ballas, B. Labb, H. Le Borgne, P. Gosselin, M. Redi, B. Merialdo, R. Vieux, B. Mansencal, J. Benois-Pineau, S. Ayache, A. Hamadi, B. Safadi, T.-T.-T. Vuong, D. Han, N. Derbas, G. Qunot, B. Gao, C. Zhu, Y. tang, E. Dellandrea, C.-E. Bichot, L. Chen, A. Benot, P. Lambert, and T. Strat, "IRIM at TRECVID 2013: Semantic Indexing and Instance Search," in *Proc. TRECVID Workshop*, Gaithersburg, MD, USA, nov 2013.

[19] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid, "Towards Good Practice in Large-Scale Learning for Image Classification," in *CVPR 2012 - IEEE Computer Vision and Pattern Recognition*. Providence (RI), United States: IEEE, Jun. 2012, pp. 3482–3489.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks." in *NIPS*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 1106–1114.

[21] C. Farabet, C. Couprie, L. Najman, and Y. Lecun, "Learning hierarchical features for scene labeling," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 35, no. 8, pp. 1915–1929, Aug 2013.