Efficient Approximate Classification with Support Vector Machines and Index Structures in the Input Space *

Giuseppe Amato
ISTI-CNR
Via G. Moruzzi, 1, 56124,
Pisa, Italy
giuseppe.amato@isti.cnr.it

Paolo Bolettieri
ISTI-CNR
Via G. Moruzzi, 1, 56124,
Pisa, Italy
paolo.bolettieri@isti.cnr.it

Pasquale Savino
ISTI-CNR
Via G. Moruzzi, 1, 56124,
Pisa, Italy
pasquale.savino@isti.cnr.it

ABSTRACT

This paper proposes an approach to efficiently and effectively identify, in very large datasets, the best elements belonging to classes defined using Support Vector Machines (top-k classification). The proposed approach leverages on techniques of efficient similarity searching to identify a subset of candidate elements for a class, substantially smaller than the original dataset. Thus, the decision function, associated with a class, needs to be applied to the elements in the candidate set, rather than to all elements of the dataset, dramatically reducing the needed cost. Given that it might happen that some qualifying elements are not included in the candidate set, the result is an approximation of the exhaustive classification. We show that the proposed approach is order of magnitude faster than exhaustive classification, still providing an high degree of accuracy.

1. INTRODUCTION

There are applications where it is not necessary to exhaustively identify all objects belonging to a class, trough a process of automated classifications. Rather, it is more meaningful to be able to identify the best k objects belonging to that class. Consider for instance a huge image dataset consisting of billions of images (e.g. the images on the web, the images on Facebook, or those on Flickr). In this case a user might be interested in the best images (those with highest probability) that contain a castle, a pyramid, the eiffel tower, etc, rather than all images that contain them. We refer to this process of identifying the k best objects in a class as *top-k classification*.

In this paper we discuss how the top-k classification can be executed efficiently, leveraging on access methods for similarity search, to provide the user with the possibility of executing on-line interactive classification on very large data sets.

Efficient top-k classification techniques were proposed in [16, 15] by leveraging on the property that instances in the feature space lie on a hypersphere. This approach is able to use one index for various classes obtained using Support Vector Machines and built using the same kernel. In [14] the authors propose a method for efficient top-k classification based on boosting. In [11] the authors propose an efficient method for retrieving the instances closest to the separating hyperplane (the most ambiguous instances) to support active relevance feedback.

The approach that we propose is able to support efficient top-k classification in case of classifiers built using Support Vector Machines, using one single index in the input space and being able to support various classes built using different kernels and kernel parameters.

2. SVM: OVERVIEW

An SVM builds classifiers by learning from a training set that is composed of positive and negative examples. The training set is denoted as $T = \{(\mathbf{o_1}, y_1), ..., (\mathbf{o_n}, y_n)\}$ where $\mathbf{o_i}$ is a convenient representation, which we will discuss later, of the objects in the training set. Each y_i is 1 or -1 according to the fact that the training object i is a positive or negative sample of the class to learn.

The simplest case is that of the linear SVM, where the learning phase determines a hyper-plane that divides the space in two subspaces. One subspace corresponds to vector representation of objects belonging to the class, the other to objects that do not belong. In this case, omitting several theoretical details (see [10] for more information), the learning phase has to find a vector ω such that the decision function

$$f(o) =< \omega, \mathbf{o} > +b$$

is able to optimally classify most of the training set examples ($< \omega, \mathbf{o} >$ is the dot product between vectors ω and $\mathbf{o}$).

In most cases it is not possible to linearly separate negative from positive examples, so a linear SVM is not effective. However, linear separation might still be possible by mapping the objects in a higher dimensional vector space.

Suppose that the mapping function $\Phi(\cdot)$ maps vectors in a space of large (even infinite) dimensions where a linear separation can be found between positive and negative examples. In this case the decision function can be written as

$$f(o) =< \omega, \Phi(\mathbf{o}) > +b$$

We call *input space* the space where objects are defined, and *feature space*, the space where objects are mapped by $\Phi(\cdot)$.

SVM methods do not define the mapping function explicitly, but use the properties of the kernel functions to perform learning and classification.

*This work was partially supported by the MultiMatch and by the EFG projects, funded by the European Commission respectively under FP6 and FP7. We also wish to thank Michel Crucianu, from CEDRIC-CNAM Paris, for the interesting discussions on the topic of this paper and the useful suggestions.

A kernel function K, defined as $K(\mathbf{o}_i, \mathbf{o}_j) =< \Phi(\mathbf{o}_i)^T, \Phi(\mathbf{o}_j) >$, computes the dot-product of $\mathbf{o}_i$ and $\mathbf{o}_j$ in the feature space. There are simple kernel functions that easily compute the dot-product of objects mapped in very high or even infinite dimensional spaces without even knowing the actual mapping functions. As an example consider the Radial Basis Function (RBF) kernel, defined as $K(\mathbf{o}_1, \mathbf{o}_2) = e^{-\frac{\|\mathbf{o}_1 - \mathbf{o}_2\|^2}{2\sigma^2}}$, which computes the dot product of $\mathbf{o}_1$ and $\mathbf{o}_2$ in a space of infinite dimensionality.

It can be proven [10] that the kernel based decision function, defined above, can be also represented in the dual form as

$$f(\mathbf{o}) = \sum_{(\mathbf{o}_i, y_i) \in T} y_i \alpha_i K(\mathbf{o}, \mathbf{o}_i) + b$$

in terms of the training vectors $\mathbf{o}_i$. In this formulation, the learning phase consists in finding the parameters α_i, which basically determine the contribution of each example $\mathbf{o}_i$ of the training set to the solution of the learning problem, rather than the vector ω. Most of the α_i, obtained in the training phase, will be equal to 0. So, in order to compute the decision function f, we only need to maintain the training objects for which the α_i are greater than 0. These objects are the *support vectors*.

There are various algorithms that can be used to solve the classification problem. A frequently used one is the adatron, which was first introduced in [6] as a perceptron-like procedure to classify data and then a kernel-based version was proposed in [13]. The learning strategy of the adatron is to perform a gradient ascent to solve the margin-maximization problem between the positive and negative examples of the training set.

3. EFFICIENT APPROXIMATE TOP-K CLASSIFICATION

As stated in the introduction, our technique aims at efficiently finding the k best matches to a concept c, recognized by a classifier, in a very large test set.

Given a classifier for the class c represented by its decision function f_c, and a test set TS, the trivial procedure to find the best k matches is to sequentially scan the objects in TS, apply the decision function f_c to every object and maintain the current k best matches, until all objects in TS have been examined. Clearly, this procedure is not scalable, given that its complexity is linear with the size of the test set, and cannot be used interactively in case of huge test sets, as for instance, all the images in the web.

In the following we will discuss a strategy to significantly reduce the number of objects of the test set TS that have to be checked with the decision function, by efficiently identifying a small subset of promising candidates for the concept c.

As previously stated, the training set T_c for the concept c consists of positive and negative examples. Let us denote as PT_c and NT_c respectively the positive and negative training objects ($T_c = PT_c \bigcup NT_c$). The learning phase identifies the αs parameters, for the decision function, and implicitly the support vectors, by choosing the $\mathbf{o}_i$ whose α_i is strictly greater than 0. Let us denote as $PSV_c \subseteq PT_c$ and $NSV_c \subseteq NT_c$ respectively the positive and negative support vectors identified after the training of the classifier for a class c. The decision function can be rewritten as

$$f_c(\mathbf{o}) = \sum_{\mathbf{p} \in PSV_c} \alpha_{\mathbf{p}} K(\mathbf{o}, \mathbf{p}) - \sum_{\mathbf{n} \in NSV_c} \alpha_{\mathbf{n}} K(\mathbf{o}, \mathbf{n}) + b$$

Let us think of the kernel K as a similarity function. That is, K returns large values when the two compared objects are similar and small values when the two objects are not similar. In this case, from the definition of f_c, it is easy to see that the objects of the test set that are very similar, according to K, to several positive support vectors and are dissimilar to several negative support vectors, have higher chances to belong to the class c. This observation suggests a strategy to select a subset of promising candidates for c. In short, we can search the objects of the test set that are closer to each positive support vector; then we apply the decision function f_c only to the selected candidates to find the best k matches. More formally, the candidate set CS_c for concept c can be obtained as

$$CS_c = \bigcup_{\mathbf{p} \in PSV_c} NN_K(\mathbf{p}, k')$$

where $NN_K(\mathbf{p}, k')$, is a nearest neighbors query, which returns the k' most similar objects to $\mathbf{p}$ according to kernel K, for some k' much smaller than the size of TS. The selection of the k best objects matching the concept c can now be obtained by applying f_c only to CS_s, which is significantly smaller than TS.

This strategy clearly returns an approximation of the best k matches to c. In fact, there could be objects, not selected in the candidate set, for which the application of the decision function f_c returns a score higher than that of the k best matches found. This might be due to the (correct) effect of the negative support vectors in the decision function that might penalize objects similar to both positive and negative support vectors and, especially, might reward some objects that are dissimilar to both negative and positive support vectors. However, we can expect this effect not to be very dangerous, because of the way in which training sets are typically built: dissimilarity to a negative example is not a sign of positiveness, given that negative examples are used to identify what is *not* in c, rather than what is in c.

Several scalable and efficient access methods were proposed in the literature to process nearest neighbors queries. For a complete survey you can refer to [18, 17]. The obvious option would be to try using one of these techniques in the feature space so that objects similar to the positive support vector can be found efficiently.

However, indexing in the feature space is not the best option, because of some problems and some limitations of this strategy. Access methods for similarity search typically rely on the fact that the similarity function can be expressed in terms of a distance (dissimilarity) function, which should satisfy some specific properties, like for instance the metric postulates. While a distance function can always be expressed in the feature space, such function, in many cases, it is not suitable to be used to build an efficient index structure, for instance because of the underlying distribution of distances and the course of dimensionality.

In addition, if we succeed to find a suitable distance function and we create an index in the feature space, we are bound to a specific kernel and kernel parameters. This means that we cannot use the same index for different concepts, which might use different kernels and very probably different kernel parameters.

To go around these problems, in the next section we will see that in some relevant cases we can obtain the same result by building the index data structure and searching by similarity in the input space, remaining isolated and independent from the mapping in the feature space, so being able to use one single index for several kernels and concepts.

4. USING AN INDEX STRUCTURE IN THE INPUT SPACE

If we are able to use an index in the input space we can use just one single index to search for the k best matches to any concept

c independently of the kernel and kernel parameters used in the SVM. Clearly, nothing is given for free and we will see that some conditions have to be satisfied by the distance function used in the input space and by the kernels used in the SVM.

Suppose the input space is defined by $\mathcal{S} = (\mathcal{D}, d(\cdot,\cdot))$, where $\mathcal{D}$ is the domain of the objects in the input space (for instance vectors), and $d : \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$ is a distance function between objects of $\mathcal{D}$.

Suppose the kernels can be obtained in terms of the distance function in the input space as follows:

$$K(\mathbf{o}_1, \mathbf{o}_2) = g(d(\mathbf{o}_1, \mathbf{o}_2)) \quad (1)$$

For instance, if d is the Euclidean distance (or L_2), with $g(x) = e^{-\frac{x^2}{2\sigma^2}}$ we obtain the RBF kernel, with $g = -x^\beta$, for some $\beta > 0$, we obtain the power kernel; when d is the L_1 distance, with $g(x) = e^{-\gamma x}$ we obtain the Laplacian. Other examples exist.

Let $\overline{\mathbb{R}} = d(\mathcal{D}, \mathcal{D})$, that is $\overline{\mathbb{R}}$ is the set of possible values that d can return for any arbitrary pair $\mathbf{o}_1, \mathbf{o}_2 \in \mathcal{D}$. It is easy to see that if g is monotonously decreasing over $\overline{\mathbb{R}}$, then the k objects closest to $\mathbf{o}$ in the input space, with respect to d, are the k objects most similar to $\mathbf{o}$ in the feature space, with respect to K.

In other words, *given kernel K defined according to equation 1, with a monotonous decreasing g, the most similar objects to a support vector in the feature space induced by K, are exactly the objects closest to the same support vector, in the input space*.

Therefore, we can use just one single index structure, built in the input space, to search by similarity in the feature space induced by a large class of kernels. This, as discussed in Section 3, also gives us the possibility of identifying the subset of promising candidates just working in the input space, for the same class of kernels.

Note that for arbitrary d and g, before applying a SVN training algorithm, it must first be proven that the obtained kernel is positive definite [10] or conditionally positive definite [8]. For instance, [9] shows that when $g = e^{-tx}$, for all $t > 0$ K is positive definite iff d is negative definite and symmetric.

Note also that the gs, that produce the RBF, Laplacian, and power kernels, mentioned above, are all monotonously decreasing in $\mathbb{R}^+$. Therefore, in this case, any (possibly, negative definite and symmetric) d, which always returns positive values, allows this technique to be used.

5. EXPERIMENT SETTINGS

Tests of the proposed techniques were executed using the CoPhIR dataset [3, 7] created in the SAPIR project [4]. CoPhIR consists of 106 millions images, taken from Flickr [1], described by MPEG-7 [2] visual descriptors. In the tests we used one million images from CoPhIR.

The access method used to efficiently search for objects close to the support vectors, in the input space, is the MI-File [5] (Metric Inverted File). The MI-File is a disk maintained index, based on inverted files, which supports efficient and scalable approximate similarity search on data represented in metric spaces. It relies on a space transformation that uses the notion of perspective to decide about the similarity between two objects. More specifically, if two objects are close one to each other, also the view of the space from their position is similar. To realize this idea, the MI-File chooses a set of reference objects RO from the dataset to be indexed. When an object o has to be inserted in the index, we first search the k_i ($k_i \leq \#RO$, but typically $k_i \ll \#RO$) objects of RO closest to o. The ordered sequence of the k_i reference objects $(ro_1^o, \ldots, ro_{k_i}^o)$ is the representation of o in the transformed space. This representation can be conveniently maintained in an inverted file if we assume that each $ro \in RO$ is the entry of a posting list of the inverted file. The posting list associated with an entry $ro \in RO$ is a list of pairs (o, i), such that ro is the $i-th$ closest object to o among those in RO.

Suppose the user chooses q as a query image. In order to execute the query, first the $k_s (k_s \leq k_i)$ closest reference objects to q are retrieved. Then, the retrieved k_s reference objects are used to select the posting lists to be scanned, in order to incrementally compute the *Induced Spearman Footrule Distance* distance [12] between the query representation and the image in the index. The query execution can be optimized asking the system just to scan the most promising portion of the posting lists. The value of k_s and the fraction of the posting lists to be accessed can be chosen by the user.

The CoPhIR dataset was indexed using 4000 reference objects and representing each image with the 100 ($k_i = 100$) closest reference objects.

To define the kernel for the support vector machine we used $g(x) = e^{-\frac{x^2}{2\sigma^2}}$, and as d we used a combination of MPEG-7 distance functions. We trained the support machine to recognize 5 different concepts: churches, pyramides, seascapes, paintings, and temples. We used a standard kernel based adatron [13] with cross-validation, to learn these concepts.

The set of candidates CS_c for a concept c was obtained searching, for each support vector, the 100 closest data objects using the MI-FILE. The best k matches where computed for various values of k ranging from 10 up to 200 best matches. Various settings for the query representation in the MI-FILE were also used. Specifically the number of closest reference objects (k_s) to represent the query ranged from 10 up to 100.

6. RESULTS

To objectively evaluate the performance of the proposed approximate classification approach, we used the measures of recall and position error discussed in [18]. Specifically, given a concept c and a choice for k (the number of wanted best matches), recall is

$$R_k = \frac{\#(S_k \cap S_k^A)}{\#S_k} \quad (2)$$

and the position error is

$$EP_k = \frac{\sum_{o \in S_k^A} |OX(O) - S_k^A(O)|}{\#S_k^A \cdot \#X}, \quad (3)$$

where S_k and S_k^A are the ordering of the k best matches to c found respectively by the exact classification (exhaustive classification of the entire dataset) and by the proposed method. OX is the ordering of the entire dataset X with respect to the decision function f_c.

The recall gives an idea of the percentage of the missed objects. The error on the position measures the quality of the ranking obtained by the approximate method with respect to the exact one.

The cost was evaluated measuring the time required to execute the approximate classification and the size of the candidate set CS, for various settings of the experiments.

In addition we also performed a user evaluation, asking users to blindly judge the results obtained with the exact and approximate classification. To compare the two results we used the precision measure defined as follows:

$$P_k = \frac{\#(S_k \cap S^c)}{\#S_k} \quad (4)$$

where S_k is the result obtained by either the approximate or the exact classification method, and S^c is the set of images correctly

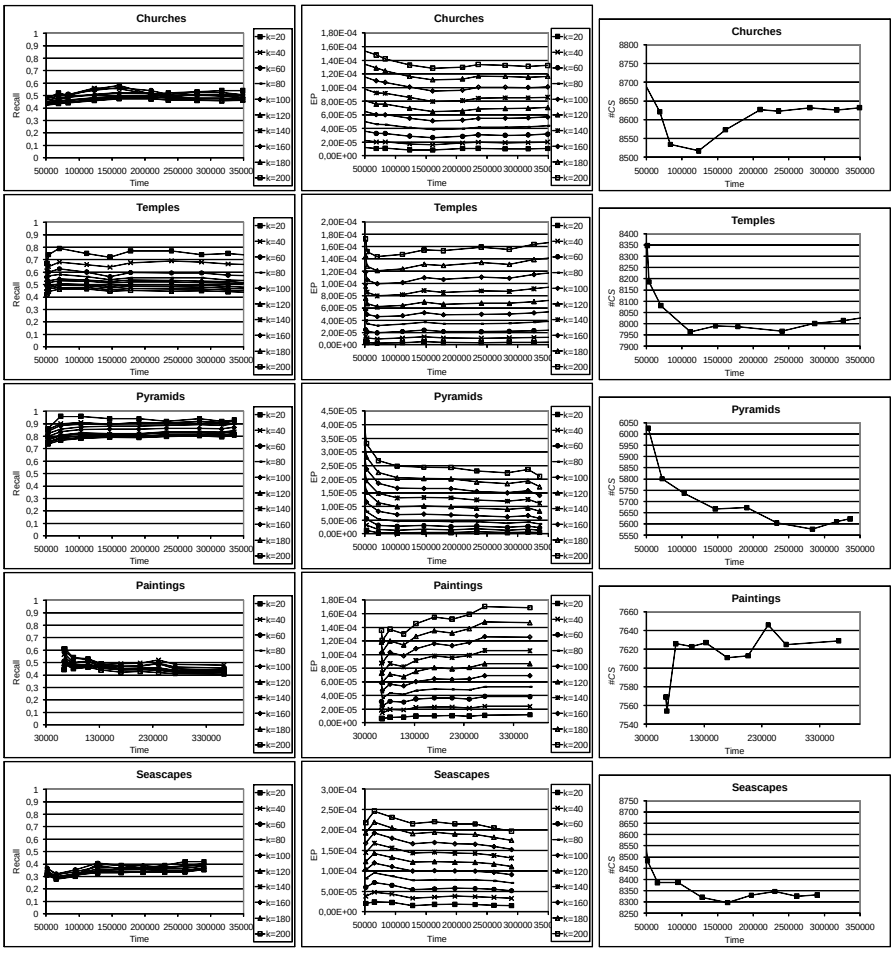

Figure 1: Approximate vs. exact classification

classified for c. $S_k \cap S^c$ was obtained by asking the users to select the correct results in S_k (blindly for exact and approximate classification).

6.1 Approximate vs exact classification

Results of the comparison between approximate and exact classification are depicted in Figure 1. For each of the considered concepts, churches, temples, pyramides, paintings, and seascapes, we plot the recall and the EP for various choices of the number of best matches k. For each concept we also show the total number of objects contained in the obtained candidate set. The graphs range over the time required to the approximate approach to determine the best k matches in accordance to various settings of the approximation parameters of the MI-FILE. Short processing time means high similarity search approximation and, viceversa, long processing time means low similarity search approximation.

A first comment on the results is that the proposed strategy seems not to be much affected by the similarity search approximation. In fact both the recall and EP values are in a very small interval for the various approximation parameters. This is due to the fact that when the approximation is high some objects might be missed in a query created with a support vector but might be found in another query corresponding to another support vector. In the end most of the needed objects occur in the candidate set anyway. In general, the recall seems not to be influenced by the choice of k, the number of best matches. This means that on average the approximate classification strategy is able to find always the same percentage of correct objects. On the other hand, we can see that the quality of the ranking, given by EP, decreases when k increases. This means that when k increases the missed correct objects are replaced by worse objects. The size of the candidate set is on average less than two order of magnitude smaller than the total size of the test set. In fact, as previously mentioned, the dataset contains one million objects, while the size of the candidate set is always smaller than 10,000 objects, which is *more than two orders of magnitude smaller than the entire dataset*.

The time required to perform exact classification of the entire dataset was on average 39 minutes. Good approximate classification of the same dataset can be obtained on average in 1.5 minutes, which is *more than two orders of magnitude faster than exact classification*.

6.2 User evaluation

Previous section was mainly dedicated to discuss the difference in performance of the approximate and exact classifier, by considering the result of the exact classifier as the ground truth. The experiments basically say what is the difference between the exact and approximate classification, under the hypothesis that any discrepancy of the approximate classifier, from the exact classifier, has to be considered an error. However, the exact classifier is not a per-

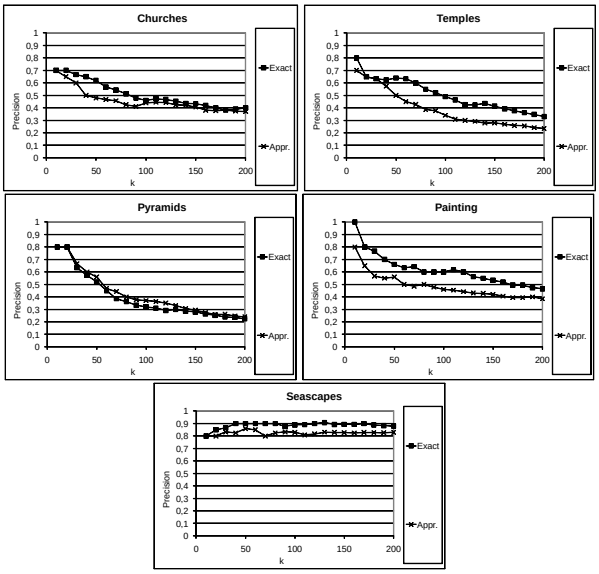

Figure 2: User evaluation of approximate and exact classification

fect classifier, in practice. It also performs some errors, and quality of a classifier is typically measured against a manually generated ground truth. An interesting test would be that of evaluating the difference of quality between the exact and the approximate classifier when tested against a ground truth. Unfortunately, the CoPhIR data set does not have a ground truth that can be used for that purpose. Furthermore, given its size it is not even realistic to have such ground truth. Therefore, in order to compare the quality of the approximate and the exact classifier we asked a number of users to blindly judge the results, obtained by the exact and approximate classifiers, by selecting the good and the wrong images. Based on this, we computed the average precision using Equation 4. Results are shown in Figure 2 for various choices of the number of best matches k.

It can be seen that generally there is not a significant difference between the precision of the exact and the approximate classifier, even thought the approximate classifier is more than one hundred times faster. It is worth mentioning that in one case of the tested classes, the approximate classifier performed even better than the exact one. In fact, it can be seen that for the Pyramids class, the curve of the approximate classifier is always higher than that of the exact one. This, we believe, is a further proof that no significant information is actually lost during the approximation: the lost information is mainly noisy information.

7. CONCLUSIONS

We have presented a technique for efficiently and effectively executing top-k classification tasks on very large datasets. The proposed technique is able to use one single index built in the input space to support top-k classification tasks on several classes defined using various kernels and kernel parameters. We discussed the properties that the kernel has to satisfy so that they can be used with the proposed technique and we have seen that many widely used kernels are in fact included. Experiments have shown that the proposed technique is some orders of magnitude faster with respect to exhaustive classification and that accuracy is very high.

8. REFERENCES

[1] http://www.flickr.com/.

[2] Mpeg-7. ISO/IEC JTC1/SC29/WG11N6828, October 2004. http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm.

[3] CoPhIR: Content-based Photo Image Retrieval Test-Collection. Project Web Site, 2009. http://cophir.isti.cnr.it/.

[4] SAPIR: Search In Audio Visual Content Using Peer-to-peer IR. Project Web Site, 2009. http://www.sapir.eu/.

[5] G. Amato and P. Savino. Approximate similarity search in metric spaces using inverted files. In *InfoScale '08: Proceedings of the 3rd international conference on Scalable information systems*, pages 1–10. ICST, 2008.

[6] J. Anlauf and M. Biehl. The adatron-an adaptive perceptron algorithm. *Europhysics Letters*, vol.10, 1989.

[7] P. Bolettieri, A. Esuli, F. Falchi, C. Lucchese, R. Perego, T. Piccioli, and F. Rabitti. CoPhIR: a test collection for content-based image retrieval. *CoRR*, abs/0905.4627v2, 2009.

[8] S. Boughorbel, J.-P. Sabri, and N. Boujemaa. Conditionally positive definite kernels for svm based image recognition. In *IEEE International Conference on Multimedia and Expo 2005*, pages 113–116. IEEE, July 2005.

[9] C. Cortes, P. Haffner, and M. Mohri. Rational kernels: Theory and algorithms. *J. Mach. Learn. Res.*, 5:1035–1062, 2004.

[10] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March 2000.

[11] M. Crucianu, D. Estevez, V. Oria, and J.-P. Tarel. Speeding up active relevance feedback with approximate knn retrieval for hyperplane queries. *International Journal of Imaging Systems and Technology*, 18(2-3):150–159, August 2008. http://perso.lcpc.fr/tarel.jean-philippe/publis/ijist08.html.

[12] P. Diaconis. *Group Representations in Probability and Statistics*, volume 11 of *IMS Lecture Notes - Monograph Series*. Institute of Mathematical Statistics, Hawyard Ca, 1988.

[13] T.-T. Frieß, N. Cristianini, and C. Campbell. The Kernel-Adatron algorithm: a fast and simple learning procedure for Support Vector machines. In *Proc. 15th International Conf. on Machine Learning*, pages 188–196. Morgan Kaufmann, San Francisco, CA, 1998.

[14] S. Litayem, A. Joly, and B. Nozha. Interactive objects retrieval with efficient boosting. In *Proccedings of ACM Multimedia 2009*.

[15] N. Panda and E. Y. Chang. Exploiting geometry for support vector machine indexing. In *roceedings of SIAM International Data Mining Conference*, 2005.

[16] A. Qamra and E. Y. Chang. Using pivots to index for support vector machine queries. In *CVDB '05: Proceedings of the 2nd international workshop on Computer vision meets databases*, pages 59–64, New York, NY, USA, 2005. ACM.

[17] H. Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[18] P. Zezula, G. Amato, V. Dohnal, and M. Batko. *Similarity Search - The Metric Space Approach*, volume 32 of *Advances in Database Systems*. Springer, 2006.