



Boosted kernel for image categorization

Alexis Lechervy, Philippe-Henri Gosselin, Frédéric Precioso

► To cite this version:

Alexis Lechervy, Philippe-Henri Gosselin, Frédéric Precioso. Boosted kernel for image categorization. Multimedia Tools and Applications, 2013. hal-00799068

HAL Id: hal-00799068

<https://hal.science/hal-00799068>

Submitted on 11 Mar 2013

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.

Boosted Kernel for Image Categorization

Alexis Lechervy · Philippe-Henri
Gosselin · Frédéric Precioso

Received: date / Accepted: date

Abstract Recent machine learning techniques have demonstrated their capability for identifying image categories using image features. Among these techniques, Support Vector Machines (SVM) present the best results, particularly when they are associated with a kernel function. However, nowadays image categorization task is very challenging owing to the sizes of benchmark datasets and the number of categories to be classified. In such a context, lot of effort has to be put in the design of the kernel functions and underlying high-level features.

In this paper, we propose a framework to learn an effective kernel function using the Boosting paradigm to linearly combine weak kernels. We then use a SVM with this kernel to categorize image databases. More specifically, this method create embedding functions to map images in a Hilbert space where they are better classified. Furthermore, our algorithm benefits from boosting process to learn this kernel with a complexity linear with the size of the training set.

Experiments are carried out on popular benchmarks and databases to show the properties and behavior of the proposed method. On the PASCAL VOC2006 database, we compare our method to simple early fusion, and on the Oxford Flowers databases we show that our method outperforms the best MKL techniques of the literature.

Keywords Image categorization · Kernel Machines · Boosting

Alexis Lechervy · Philippe-Henri Gosselin
ETIS - CNRS - ENSEA - Univ of Cergy-Pontoise, BP44 F95014 Cergy-Pontoise - France
email: {alexis.lechervy,philippe-henri.gosselin}@ensea.fr

Frédéric Precioso
I3S - UMR7271 - UNS CNRS, F06903 Sophia Antipolis - France
email: precioso@polytech.unice.fr

1 Context

Recent years have seen exponential growth of image and multimedia data. Recent machine learning techniques have demonstrated their capability for identifying image categories from image features, like SVM or Boosting. On the one hand, SVM classifiers can be successfully applied to all datasets requiring few training samples, thanks to kernel trick [19]. However, the performance of classification depends highly on kernel design. On the other hand, the boosting classifiers have good performance [20] and they can be tuned more easily by using weak classifiers based on simple features but they require to be trained on large training sets.

Nowadays, the challenges of Image categorization task lie in the size of the test databases and the large number of categories to be considered. The SVM has been originally proposed for binary classification contexts and its extension to multi-class problems is not straight forward. In most cases, one-against-all or one-versus-ones strategies are considered. In other words, multi-class learning is derived to multiple binary classification problems. Several adaptations have been recently proposed for multi-class [23, 15, 13, 17, 9].

One of the main issues when designing a multi-class method is to achieve both a good inter-class separability while being robust to intra-class variations. It is obvious that not any unique descriptor can have the same discriminative power for all classes. In this paper we propose an intrinsically multi-class method to learn a major kernel function as a combination of a set of diverse and complementary minor kernels. There are several ways to learn and combine kernels such as linear combinations [1, 3], geometric combinations [18] or even more general combinations [22]. When working on the Gram matrix, values of the kernel function itself can be learnt directly [11] or indirectly with metric learning [6].

In this paper, we propose a method to learn a combination of kernels using the Boosting approach. In the remaining of this section, we present techniques for linear combination of kernels and the involved objects like Alignment and centering. Then, we present the Boosting in this specific context. In Section 2, we present the approach we propose. It differs from other approaches thanks to choice of rank one weak kernels, and more specifically to the design of the learners target used in the Boosting process. In Section 3, we propose a detailed algorithm to implement the learning method. We also show, assuming that the number of categories is negligible against the training set size, that the complexity of this method is linear according to the size of the training set. Finally, in the last section we present comparisons to others methods on image databases.

1.1 Learning kernels as linear combinations

We want to address large multi-category databases thus memory and computational complexity will drive many of our choices in this paper. We thus

focus on building a major kernel K_T as a linear combination of minor kernel functions k_t :

$$K_T(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^T \beta_t k_t(\mathbf{x}, \mathbf{y}), \quad (1)$$

where $\mathbf{x}$ and $\mathbf{y}$ are two images and $\beta_t \geq 0$.

First methods to learn linear combinations are based on a joint optimization of the coefficients in a conic combination of kernel matrices and the coefficients of a discriminative classifier, usually Support Vector Machines [1]. In this context, combined kernels k_t are called *base kernels*, and they are related to high level features. For instance, in the Multimedia Retrieval community each base kernels correspond to high level features like Bag-of-Words (BoW) [14].

Recently, the one-stage approach is criticized, since achieving a performance superior to that of the uniform combination of kernels has proven to be surprisingly difficult [2]. As a result, Cortes et al. propose a two-stage procedure: in the first stage, the method finds the optimal weights to linearly combine the kernels, and in the second stage a SVM learns the optimal hyper-plane using the learn kernel [3]. For these reasons, we also use the two-stage approach in this method proposed in this paper.

In order to use the two-stage approach, a criterion is required to evaluate the kernel combination, like the Kernel Alignment [5]:

$$\mathcal{A}(\mathbf{K}, \mathbf{K}^*) = \frac{\langle \mathbf{K}, \mathbf{K}^* \rangle}{\|\mathbf{K}\| \|\mathbf{K}^*\|} \quad (2)$$

where $\mathbf{K}$ is the $n \times n$ kernel matrix to evaluate, $\mathbf{K}^*$ is the $n \times n$ target kernel matrix, $\langle \mathbf{K}, \mathbf{K}^* \rangle$ is the Frobenius inner product, $\|\mathbf{K}^*\|$ is the Frobenius norm, and n is the number of images in the training set.

Let us note that this criterion evaluates a kernel function K using the values of its Gram matrix $\mathbf{K}$ on a specific training set $\mathbf{X}$. In the following we will use bold symbol to represent all values of a function on a training set, for instance $\mathbf{K}_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$ with $\mathbf{X} = (\mathbf{x}_i)_{i \in [1, n]}$.

For example the target kernel matrix $\mathbf{K}^*$ can be defined from the $n \times c'$ matrix $\mathbf{L}$ of labels on the training set, $\mathbf{K}^* = \mathbf{L}\mathbf{L}^\top$, with c' the number of categories and $L_{ij} = 1$ if image i is in category j , 0 otherwise.

In order to deal with the unbalance of training data, an interesting approach we will use in this paper is the Centered Kernel Alignment [14]:

$$\mathcal{A}_H(\mathbf{K}, \mathbf{K}^*) = \frac{\langle \bar{\mathbf{K}}, \bar{\mathbf{K}}^* \rangle}{\|\bar{\mathbf{K}}\| \|\bar{\mathbf{K}}^*\|} \quad (3)$$

where $\bar{\mathbf{K}}$ denote a centered matrix $\mathbf{K}$:

$$\bar{\mathbf{K}} = \mathbf{H}\mathbf{K}\mathbf{H}$$

with $\mathbf{H} = \mathbf{I} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top$, $\mathbf{I}$ being the $n \times n$ identity matrix and $\mathbf{1}$ a $n \times 1$ vector full of ones. Note that Centered Kernel Alignment invariant to scale: $\mathcal{A}_H(\lambda\mathbf{K}, \mathbf{K}^*) = \mathcal{A}_H(\mathbf{K}, \mathbf{K}^*)$ and shift: $\mathcal{A}_H(\mathbf{K} + \lambda\mathbf{1}\mathbf{1}^\top, \mathbf{K}^*) = \mathcal{A}_H(\mathbf{K}, \mathbf{K}^*)$.

1.2 Boosting: a way for combining a large number of kernels

First MKL techniques were used to combine high-level kernels. In Multimedia Retrieval context, this approach allows to combine different visual descriptors like colors, textures, keypoints, histogram of gradients, etc. Considering or not their performance when compared to uniform combinations, this approach requires already effective kernels and features. In other words, a lot of research design is required to build the base kernels. Furthermore, it does not really allow the learning of completely new kernel functions.

In order to be able to get a wider range of learning is to consider a large number of simple base kernels. To illustrate this idea, we can see these simple base kernels as words, and a kernel combination as a sentence or a text. Then, finding a kernel combination can be interpreted as the best description of a database using relevant words. An example of this approach is proposed in [12], where each simple base kernels is related to a visual word, and the kernel combination is then related to a visual dictionary. A more mathematical view of this idea is to consider a family of kernel functions span by a set of "atomic" simple kernel functions.

Before considering these simple kernel functions, we have to consider the method that will combine them. Actually, usual MKL techniques are not able to process this case because of their computational complexity according to the number of base kernel to combine, which is in most cases quadratic. Furthermore, they are made to combine a finite set of base kernel.

In order to handle this, a strategy we propose to follow in this paper is the Boosting approach [4]. In this context, the base kernels used to be called *weak kernel* and the kernel combination the *strong kernel*. This approach iteratively selects weak kernel functions k_t and their corresponding weight β_t during a Boosting round t . There is several way to selects weak kernels. A first one is to consider a finite set of weak kernel candidates, and select the one that maximize some evaluation criterion. A second one is to ask a *weak trainer* for a relevant weak kernel. The weak trainer is a procedure that creates a new weak kernel that maximize the evaluation criterion. As a result, the possible kernel combinations (or strong kernels) that can be built using a Boosting approach is very large, allowing infinite possibilities in most cases.

Furthermore, the complexity linearly depends on the number T of selected weak kernels, and not on the total number of possible weak kernels. The method is not really sensitive to the setting of this hyperparameter T since, as stated in very recent paper, "*Boosting has demonstrated a remarkable (and perhaps unique) resistance to overfitting, although it is affected by the presence of outliers or high levels of noise*" [7].

2 Proposed Approach

2.1 Rank one weak kernels

In order to learn our mapping, we propose to considerer rank one weak kernel functions $k_t(\mathbf{x}, \mathbf{y}) = f_t(\mathbf{x})f_t(\mathbf{y})$, thus leading to the following strong kernel expression:

$$K_T(\mathbf{x}, \mathbf{y}) = \sum_{t=1}^T \beta_t f_t(\mathbf{x})f_t(\mathbf{y}) \quad (4)$$

Since they are related to weak kernel functions k_t , we call the functions f_t *weak functions*.

Eq. (1) becomes equivalent to:

$$K_T(\mathbf{x}, \mathbf{y}) = \langle F_T(\mathbf{x}), F_T(\mathbf{y}) \rangle = F_T(\mathbf{x})^\top F_T(\mathbf{y}), \quad (5)$$

where

$$F_T(\mathbf{x}) = \beta^{\frac{1}{2}} \odot \left(f_1(\mathbf{x}) \ f_2(\mathbf{x}) \ \dots \ f_T(\mathbf{x}) \right) \quad (6)$$

with $\odot$ the Hadamard product.

As we need to evaluate functions on a training set $\mathbf{X} \subset \mathcal{X}$, we consider their values on this set using bold symbols. For instance, values of weak functions $f_t(\cdot)$ on training set $\mathbf{X}$ are denoted $\mathbf{f}_t \in \mathbb{R}^n$ defined by, with n the size of the training set:

$$\forall \mathbf{x}_i \in \mathbf{X} \ (\mathbf{f}_t)_i = f_t(\mathbf{x}_i) \quad (7)$$

Similarly, values of the strong kernel can be expressed on the training set as:

$$\mathbf{K}_T = \sum_{t=1}^T \beta_t \mathbf{f}_t \mathbf{f}_t^\top = \mathbf{F}_T \mathbf{F}_T^\top. \quad (8)$$

2.2 Boosting approach

We propose to build the major kernel $\mathbf{K}_T$ using the Boosting approach. In the literature, there is several definitions and interpretations for Boosting. In this paper, we consider the one that attempts to minimize, or make small, a continuous bound on the classification error (see for instance [8] and the references therein).

This approach iteratively selects weak functions f_t (or equivalently weak kernels k_t):

$$F_t = \left(F_{t-1} \ \sqrt{\beta_t} f_t \right) \Leftrightarrow K_t = K_{t-1} + \beta_t k_t. \quad (9)$$

For each boosting round t , we have to find the weak kernel k_t or function f_t that increases the most the alignment on training set:

$$\{\beta_t, f_t\} = \arg \max_{\beta > 0, f} \mathcal{A}_{\mathbf{H}}(\mathbf{K}_{t-1} + \beta \mathbf{f} \mathbf{f}^\top, \mathbf{K}^*) \quad (10)$$

We perform this task in two stages. During the first stage, the basic idea is to require from the *weak learners* to build a set $\mathcal{K}_t$ of *weak kernel candidates* k . We expect that the weak learners provide weak kernels which increase the overall alignment. As usually in boosting processes, the alignment increment must be a *weak* increment, in order to get good generalization properties.

Actually, as we focus on rank one weak kernels $k(\mathbf{x}, \mathbf{y}) = f(\mathbf{x})f(\mathbf{y})$, we consider *weak learners* defining weak functions f . Hence, for each boosting round t , we build a set $\mathcal{F}_t$ of *weak function candidates*. We also expect that the weak learners not only provide weak functions but also increase the overall alignment. However, as they are learners for functions rather than kernels, we have to define a target function f^* which leads to a (weak) increment of the overall alignment. We will present a solution for this problem in section 2.3, but for now we assume that the weak learners provided us with a set $\mathcal{F}_t$ of weak function candidates.

Thanks to this set $\mathcal{F}_t$, we can run the second stage of resolution of Eq. (10). For each weak function candidate $f \in \mathcal{F}_t$, we compute the weight $\beta^*(f)$ maximizing the alignment:

$$\beta^*(f) = \arg \max_{\beta > 0} \mathcal{A}_{\mathbf{H}}(\mathbf{K}_{t-1} + \beta \mathbf{f} \mathbf{f}^\top, \mathbf{K}^*). \quad (11)$$

We end up with a set of candidate weak functions and their corresponding weights. We then select the pair (function, weight) maximizing the alignment. An interesting point about Eq. (11) is that it can be solved analytically [3].

2.3 Learner targets

Rather than using usual target kernels $\mathbf{K}^*$ like $\mathbf{L}\mathbf{L}^\top$, with $\mathbf{L}$ the label matrix, we propose to consider the matrix $\mathbf{Q}$ of the *QR* decomposition of $\mathbf{H}\mathbf{L}$. We center the matrix $\mathbf{L}$ with $\mathbf{H}$ to deal with unbalanced classes [14]. We only select columns of $\mathbf{Q}$ whose corresponding diagonal element in $\mathbf{R}$ are not zero (a.k.a. economic *QR*). Thus $\mathbf{Q}$ is a $n \times c$ full rank matrix. Our target Gram matrix is then defined as $\mathbf{K}^* = \mathbf{Q}\mathbf{Q}^\top$. This choice of kernel target matrix is to overcome the correlation between classes, thanks to the orthogonalization performed by the *QR* decomposition.

Considering the weak function target f^* , we propose a solution based on equidistance of class centers, in order to deal with multi-class problems. In the case of two-class context, this function can be defined by $f^*(\mathbf{x}_i) = 1$ if the sample $\mathbf{x}_i$ is in the first class, -1 otherwise. However, in the case of multi-class context, building f^* is not as straightforward since we need to spread each class data around equidistant class centers [21, 23, 10]. The idea of our method is then to move each center to make it equidistant from the others.

We propose to consider the barycenters of (orthonormalized) classes in the space induced by the current combination kernel $\mathbf{K}_t = \mathbf{F}_t \mathbf{F}_t^\top$:

$$\mathbf{G}_t = \mathbf{Q}^\top \mathbf{F}_t. \quad (12)$$

Each row of matrix $\mathbf{G}_t$ is the barycenter of one class in the space induced by $\mathbf{K}_t$. If we want equidistant class centers, it is sufficient to get matrix $\mathbf{G}_t^*$ such that $\mathbf{G}_t^*(\mathbf{G}_t^*)^\top$ is proportional to the identity matrix.

Based on this idea, we propose to work on the eigenvalues of $\mathbf{G}_t\mathbf{G}_t^\top$. More specifically, we propose a strategy that will turn all eigenvalues of $\mathbf{G}_t\mathbf{G}_t^\top$ to 1, which is equivalent to get the identity matrix, and thus equidistant class centers. This can be achieved by working on the lowest eigenvalue of $\mathbf{G}_t\mathbf{G}_t^\top$:

$$\varphi(\mathbf{G}_t) = \frac{\|\bar{\mathbf{K}}^*\|}{\|\bar{\mathbf{K}}_t\|} \arg \min_{\lambda} \{\lambda \in \text{spec}(\mathbf{G}_t\mathbf{G}_t^\top)\} \quad (13)$$

Note that it can be shown that $\varphi(\mathbf{G}_t) \in [0, 1]$.

As we proceed iteratively, we can only focus on the new column $\mathbf{g}_t^*$ to be added to $\mathbf{G}_t^* = [\mathbf{G}_{t-1}, \mathbf{g}_t^*]$. In order to increase $\varphi(\mathbf{G}_t^*)$, we propose to consider a vector collinear to the eigenvector $\mathbf{v}$ associated to $\varphi(\mathbf{G}_{t-1})$:

$$\mathbf{g}_t^* = \sqrt{1 - \varphi(\mathbf{G}_{t-1})} \mathbf{v} \quad (14)$$

The normalization term $\sqrt{1 - \varphi(\mathbf{G}_{t-1})}$ ensures that $\mathbf{G}_t\mathbf{G}_t^\top = \mathbf{I}$. Thanks to this result, we can define the target function $\mathbf{f}_t^*$ for weak learners. We propose to use $\mathbf{f}_t^* = \arg \min_{\mathbf{f}} \|\mathbf{Q}^\top \mathbf{f} - \mathbf{g}_t^*\|_2^2$ subject to minimizing $\|\mathbf{f}\|_2$. The solution is $\mathbf{f}_t^* = \mathbf{Q}\mathbf{g}_t^*$.

2.4 Learning semantic features

As $F_T(\cdot)$ will be the result of a training on high-level concepts, values $f_t(\mathbf{x})$ of function f_t on an image $\mathbf{x}$ are interpreted as *semantic features*.

In this article we propose a new supervised method to learn semantic features for image categorization task. Since this new feature space is created using supervised data, i.e. image labels on a training set, we call this space the *semantic space*. The idea is to learn an explicit embedding of the most relevant features to better represent each image category.

If we denote by $\mathcal{X}$ the low-level feature space (color, texture, keypoints...), and $\mathcal{S}$ the semantic space, the mapping function F is:

$$\begin{aligned} F_T : \mathcal{X} &\rightarrow \mathcal{S} \\ \mathbf{x} &\mapsto F_T(\mathbf{x}) \end{aligned} \quad (15)$$

Let us note that the function F_T is independent from the learning process that created it. Furthermore, it can be used to map any image inside or outside the training set.

In this paper, we consider linear combination of kernels. The semantic space $\mathcal{S}$ is thus a Hilbert space, i.e. a vector space with a dot product $\langle \cdot, \cdot \rangle$. More specifically, the dot product of $\mathcal{S}$ is the major kernel K_T we aim to learn:

$$K_T(\mathbf{x}, \mathbf{y}) = \langle F_T(\mathbf{x}), F_T(\mathbf{y}) \rangle \quad (16)$$

As usual in kernel machines, two formulations represent the same object: the vectorial (or primal) view F_T of the semantic $\mathcal{S}$, and the kernel (or dual) view K_T .

Algorithm 1. Learn boosted kernels

Input: Low-level feature spaces $\mathcal{X}_s$, Orthogonal $n \times c$ label matrix $\mathbf{Q}$, threshold θ , maximum number T of weak functions

Compute initial weak functions $(f_1(\mathbf{x}) \ f_2(\mathbf{x}) \dots f_c(\mathbf{x}))$ using Algorithm 2

For rounds $t=c+1, \dots, T$:

- [1] Compute barycenters matrix $\mathbf{G}_{t-1} = \mathbf{Q}^\top \mathbf{F}_{t-1}$
- [2] Compute eigendecomposition $\mathbf{G}_{t-1} \mathbf{G}_{t-1}^\top = \mathbf{V} \mathbf{\Lambda} \mathbf{V}^\top$
Sort eigenvalues $\mathbf{\Lambda}$ in ascending order
For $(\lambda, \mathbf{v}) \in \mathbf{\Lambda} \times \mathbf{V}$:
 - [3] Compute $\mathbf{g}^* = \sqrt{1 - \lambda \frac{\|\bar{\mathbf{K}}^*\|}{\|\bar{\mathbf{K}}_{t-1}\|}} \mathbf{v}$
 - [4] Compute weak target $\mathbf{f}^* = \mathbf{Q} \mathbf{g}^*$
Foreach low-level feature space $\mathcal{X}_s$:
Run weak learner on training set $\mathbf{X} \subset \mathcal{X}_s$:
 - [5] $f = \arg \min_{\mathbf{w} \in \mathcal{X}_s} \|\mathbf{X} \mathbf{w} - \mathbf{f}^*\|^2$ s.t. $\|\mathbf{w}\|^2 = 1$
 - [6] Compute $\beta^*(f)$ using Eq. (11)
 - [7] **If** $\mathcal{A}_{\mathbf{H}}(\mathbf{K}_{t-1} + \beta^* \mathbf{f} \mathbf{f}^\top, \mathbf{K}^*) > \mathcal{A}_{\mathbf{H}}(\mathbf{K}_{t-1}, \mathbf{K}^*) + \theta$
Update $F_t = (F_{t-1} \ \sqrt{\beta_t} f_t)$
Proceed to next round $t + 1$
- Endif**
- Endforeach**
- Endfor**
- If** no weak kernel increases Alignment
T = t
Stop
- Endif**

Endfor

Output: Strong kernel $K_T(\mathbf{x}, \mathbf{y}) = \langle F_T(\mathbf{x}), F_T(\mathbf{y}) \rangle$

Algorithm 2. Initialization of the proposed algorithm

Input: Low-level feature spaces $\mathcal{X}_s$, Orthogonal $n \times c$ label matrix $\mathbf{Q}$

For $t=1, \dots, c$:

- Compute $\mathbf{f}_t^* = \mathbf{Q}(:, t)$
- Foreach** low-level feature space $\mathcal{X}_s$:
Compute f_s using weak learner on training set $\mathbf{X} \subset \mathcal{X}_s$
If $\mathcal{A}_{\mathbf{H}}(\mathbf{f}_s \mathbf{f}_s^\top, \mathbf{K}^*) > 0$
Update $F_t = (F_{t-1} \ f_s)$
Proceed to next round $t + 1$
- Endif**
- Endforeach**

Output: Weak functions $(f_1(\mathbf{x}) \ f_2(\mathbf{x}) \dots f_c(\mathbf{x}))$

3 Algorithm

3.1 Initialization

At the beginning no class is learnt, therefore we initialize our method accounting for each class at least once. The first c target functions $\mathbf{f}_t^*$ are then the columns of the matrix, i.e. $\forall t \in [1, c] \ \mathbf{f}_t^* = \mathbf{Q}(:, t)$. These target functions are

used independently to train weak learners. Weak learners is a free component of our method that depends on the low-level features. In this paper, we used Least Mean Square (LMS), but other functions can be used.

We present this initialization in Algorithm 2. Note that weak functions can be created thanks to each available low-level feature (color, texture, key-points, ...) but also to different descriptor parameters (feature size, metric, ...). Consequently, the low-level feature spaces $\mathcal{X}_s$ in Algorithm 2 denotes different features with different parameters. Then, for each class $t \in [1, c]$, we select the first one that lead to a positive Alignment.

The result of initialization is c weak functions: $F_c = \begin{pmatrix} f_1 & f_2 & \dots & f_c \end{pmatrix}$, with c the width of matrix $\mathbf{Q}$.

3.2 Boosting rounds

Once the width of $\mathbf{F}$ equals the rank of $\mathbf{Q}$, we can start the main learning, as presented in Algorithm 1. For each round t , we first compute the barycenters $\mathbf{G}_t$, and the eigendecomposition of $\mathbf{G}_t \mathbf{G}_t^\top$. Then, we try each eigenvalue from the smallest to highest. For each eigenvalue λ and its corresponding eigenvector $\mathbf{v}$, we compute a target function $\mathbf{f}^*$ (line [4] in Algo. 1). Once we have a target function, we run a weak learner in order to get a weak function f for each low-level feature space $\mathcal{X}_s$. In this paper, we present this algorithm using LMS (line [5] in Algo. 1), but other methods can be used.

Note that, as we are subject to the low-level features, weak learners will not necessarily create weak functions f that match the weak target $\mathbf{f}^*$. For the same reasons, each weak function f will not necessarily leads to a significant increasing of Alignment. As a result, if we do not find a relevant weak function, we proceed to the next low-level feature space $\mathcal{X}_s$, until we find one. After several rounds, the algorithm may not find a new weak function to add, and the algorithm stops. Thus, the algorithm can select less weak functions that the number T a user can ask for.

In order to help the creation of a strong kernel with a better generalization capacity, we sort the different low-level feature spaces $\mathcal{X}_s$ according to their expected generalization capacity. Thus, the algorithm first tries feature spaces with the highest generalization capacity. In case of selection, we obtain the most interesting weak function. This is usually the case during the first rounds, but in the latter ones it is more and more difficult to select a weak function with high generalization capacity.

To get low-level feature spaces with different level of generalization, one can perform a PCA for each low-level feature, and then create new features depending on the number of principal components. We assume that features with fewer components less overfit than features with more components. This example of algorithm based on PCA to arrange features according to their hypothetical overfitting can certainly be improved in many ways.

3.3 Linear complexity

In this section, we show that each round t of Algorithm 1 has a computational complexity of $O(nt)$, with n the size of the training sample.

Algorithm starts with the computation of the eigendecomposition of $c \times c$ matrix $\mathbf{G}_t \mathbf{G}_t^T$ (lines [1] and [2]). The full eigendecomposition of this matrix has a computational complexity of $O(c^3)$, but since we only consider first eigenvalues, a $O(c^2)$ algorithm can be used. Concerning the computation of the matrix $\mathbf{G}_t \mathbf{G}_t^T$, we proceed iteratively:

$$\mathbf{G}_t \mathbf{G}_t^T = \mathbf{G}_{t-1} \mathbf{G}_{t-1}^T + \beta_t^* \mathbf{g}_t \mathbf{g}_t^T$$

Consequently, the main computations are the computations of $\mathbf{g}_t \mathbf{g}_t^T$, which can be performed in two steps. Firstly, we compute $\mathbf{g}_t = \mathbf{Q}^T \mathbf{f}_t$, which complexity is $O(nc)$ since $\mathbf{f}$ is a n vector and $\mathbf{Q}$ a $n \times c$ matrix. Secondly, we compute $\mathbf{g}_t \mathbf{g}_t^T$ which complexity is $O(c^2)$.

Similarly to $\mathbf{g}_t$, $\mathbf{f}^* = \mathbf{Q} \mathbf{g}^*$ is computed in $\mathcal{O}(nc)$ operations (line [4]).

In the remaining of the algorithm, computations mainly depend on variables $\mathbf{g}^*$ (line [3]), β^* (line [6]) and $\mathcal{A}_H(\mathbf{K}_{t-1} + \beta^* \mathbf{f} \mathbf{f}^T, \mathbf{K}^*)$ (line [7]).

To compute these terms, we only need the following values: $\|\overline{\mathbf{K}^*}\|$, $\|\overline{\mathbf{f} \mathbf{f}^T}\|$, $\langle \overline{\mathbf{f} \mathbf{f}^T}, \overline{\mathbf{K}_{t-1}} \rangle$, $\langle \overline{\mathbf{f} \mathbf{f}^T}, \overline{\mathbf{K}^*} \rangle$, $\langle \overline{\mathbf{K}_{t-1}}, \overline{\mathbf{K}^*} \rangle$ and $\|\overline{\mathbf{K}_{t-1}}\|$.

In fact, we have:

$$\mathbf{g}^* = \sqrt{1 - \lambda \frac{\|\overline{\mathbf{K}^*}\|}{\|\overline{\mathbf{K}_{t-1}}\|}} \mathbf{v}.$$

The solution of Eq. (11) is

$$\beta^* = \frac{\|\overline{\mathbf{K}_{t-1}}\|^2 \langle \overline{\mathbf{k}}, \overline{\mathbf{K}^*} \rangle - \langle \overline{\mathbf{K}_{t-1}}, \overline{\mathbf{k}} \rangle \langle \overline{\mathbf{K}_{t-1}}, \overline{\mathbf{K}^*} \rangle}{\|\overline{\mathbf{k}}\|^2 \langle \overline{\mathbf{K}_{t-1}}, \overline{\mathbf{K}^*} \rangle - \langle \overline{\mathbf{K}_{t-1}}, \overline{\mathbf{k}} \rangle \langle \overline{\mathbf{k}}, \overline{\mathbf{K}^*} \rangle}.$$

And by definition, we have:

$$\begin{aligned} \mathcal{A}_H(\mathbf{K}_{t-1} + \beta^* \mathbf{k}, \mathbf{K}^*) &= \frac{\langle \overline{\mathbf{K}_{t-1}} + \beta^* \overline{\mathbf{k}}, \overline{\mathbf{K}^*} \rangle}{\|\overline{\mathbf{K}^*}\| \|\overline{\mathbf{K}_t}\|} \\ &= \frac{\langle \overline{\mathbf{K}_{t-1}}, \overline{\mathbf{K}^*} \rangle + \beta^* \langle \overline{\mathbf{k}}, \overline{\mathbf{K}^*} \rangle}{\|\overline{\mathbf{K}^*}\| \|\overline{\mathbf{K}_t}\|} \end{aligned}$$

Firstly, in our case, $\|\overline{\mathbf{K}^*}\| = \sqrt{c}$ since $\mathbf{Q}$ is a full rank $n \times c$ orthonormal matrix:

$$\begin{aligned} \|\overline{\mathbf{K}^*}\|^2 &= \text{Tr}(\mathbf{K}^* \mathbf{K}^{*\top}) \\ &= \text{Tr}(\mathbf{Q} \mathbf{Q}^\top \mathbf{Q} \mathbf{Q}^\top) \\ &= \text{Tr}(\mathbf{Q} \mathbf{Q}^\top) \\ &= \text{Tr}(\mathbf{Q}^\top \mathbf{Q}) \\ &= \text{Tr}(\text{Id}_c) \\ &= c. \end{aligned}$$

Secondly, computation of value $\|\overline{\mathbf{f}\mathbf{f}^\top}\|$ is $O(n)$ since $\|\overline{\mathbf{f}\mathbf{f}^\top}\| = \|\mathbf{H}\mathbf{f}\|^2$ and $\mathbf{H}\mathbf{f} = \mathbf{f} - \frac{1}{n}\mathbf{1}\mathbf{1}^\top\mathbf{f} = \mathbf{f} - \text{mean}(\mathbf{f}) \in \mathbb{R}^n$.

Thirdly, values $\langle \overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}^\star} \rangle$ and $\langle \overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}_{t-1}} \rangle$ can be computed using the following property:

If $\mathbf{v}$ is a n vector and $\mathbf{M}$ is a $n \times c$ matrix, then $\langle \mathbf{v}\mathbf{v}^\top, \mathbf{M}\mathbf{M}^\top \rangle = \|\mathbf{v}^\top\mathbf{M}\|^2$, $\mathbf{v}^\top\mathbf{M} \in \mathbb{R}^n$ and the computation complexity is $O(nc)$.

Consequently computation of value $\langle \overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}^\star} \rangle = \|\mathbf{f}^\top\mathbf{Q}\|^2$ is $O(nc)$, and $\langle \overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}_{t-1}} \rangle = \|(\mathbf{H}\mathbf{f})^\top\mathbf{F}_{t-1}\|^2$ is $O(nt)$.

Finally for values $\|\overline{\mathbf{K}_{t-1}}\|$ and $\langle \overline{\mathbf{K}_{t-1}}, \overline{\mathbf{K}^\star} \rangle$ we propose an iterative formulation, which complexity is also linear using the same properties as previously:

$$\|\overline{\mathbf{K}_t}\|^2 = \|\overline{\mathbf{K}_{t-1}}\|^2 + \|\overline{\mathbf{f}\mathbf{f}^\top}\|^2 + 2\beta^\star \langle \overline{\mathbf{K}_{t-1}}, \overline{\mathbf{f}\mathbf{f}^\top} \rangle,$$

and

$$\langle \overline{\mathbf{K}_t}, \overline{\mathbf{K}^\star} \rangle = \langle \overline{\mathbf{K}_{t-1}}, \overline{\mathbf{K}^\star} \rangle + \beta^\star \langle \overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}^\star} \rangle.$$

To conclude about the complexity of the proposed algorithm, each round t has a computational complexity of $O(c^2 + nc + nt) = O(nt)$, since $c < n$ and $c < t$.

Note that we did not considered the complexity of weak learners. This is a free parameter the user of our method have to choose. Depending on the choice of this parameter, one can get very different levels of complexity. In this paper, we use LMS which training complexity is $O(nd)$, with d the dimension of visual features. In order to respect Boosting paradigm, one should choose *weak* learner, corresponding thus to visual features with low dimensions. Furthermore, in this case the computation of $\langle \overline{\mathbf{f}\mathbf{f}^\top}, \overline{\mathbf{K}_{t-1}} \rangle$ can be performed in $O(dt)$ operations. Consequently, the complexity of the whole learning process in the following experiments is linear with the size of the training set.

4 Experiments

In this section, we show the advantages of our method through various experiments. We have carried out two sets of experiments on real data. The first experiments compare our method to a simple feature concatenation method, and the second experiments compare it to MKL techniques proposed in the literature.

4.1 Comparison to feature concatenation

In this experiments, we compare our method to a simple method using concatenation of features. We evaluate the performance on Visual Object Challenge (VOC) 2006 dataset. This database contains 5304 images provided by Microsoft Research Cambridge and Flickr. The VOC2006 database contains 10 categories, example images are shown in *Figure 4.1*. Each image can belong to several categories. There are two distinct sets, one for training (2618 images) and one for testing (2686 images) with 9507 annotations (4754 object in train and 4753 object in test).

We first carried out experiments using histogram visual features based on $CIEL^*a^*b^*$ colors (lab) and textures using quaternionic wavelets filters (qw). Visual descriptors are projected on visual dictionaries of various size (from 16 et 128 visual words) to create different histogram features. Then, for each feature and for each category, we train a linear SVM to classify the test set of the database. The result of these runs are presented in the first rows of Fig. 2. We also present in the ever last row the result using a concatenation of all these features. In the latter case, the mean Average Precision (mAP) is 45.3%.

We also carried out an experiments using the proposed method. The process is almost the same, except that we used the method to combine the histogram features to create an embedding function $F_T(\mathbf{x})$ using the supervised data on the train set. This function is then used to create a new feature $\mathbf{x}' = F_T(\mathbf{x})$ for any image $\mathbf{x}$. This can be compared to the concatenation method, which is another way to combine the histogram features. The new feature is then used to train a linear SVM and to classify the test set. As one can see in Fig. 2, we obtain a significant increment of 3% when compared to a concatenation method. Note that we present here result with few visual features, but the method can also be used with a very large number of visual features. In fact, our method will produce better semantic features, or in the worst case the performance will not change. This has to be compared to the concatenation method whose performance can drop if too much visual features are used.

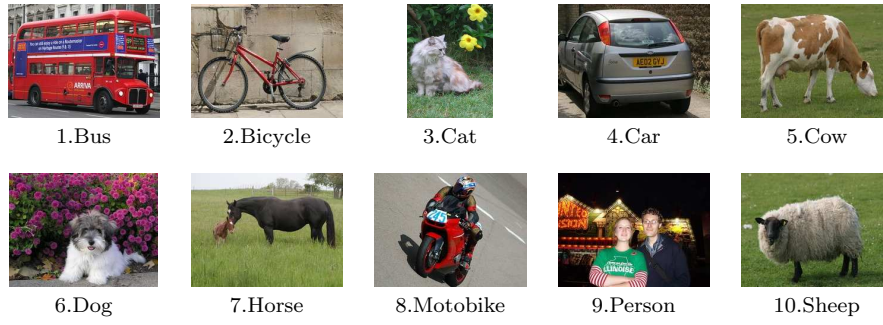


Fig. 1 Categories from PASCAL Visual Object Challenge (VOC) database, 2006.

Feature	1	2	3	4	5	6	7	8	9	10	Mean
lab16	12.4	9.4	24.6	16.0	9.4	11.2	9.0	8.0	27.2	10.9	14.0
lab32	10.7	8.1	45.7	27.1	34.2	15.9	10.0	9.1	27.0	25.6	22.8
lab64	10.0	12.5	47.9	28.5	37.5	19.1	9.9	16.4	31.7	50.3	26.3
lab128	18.7	24.7	46.6	28.8	34.1	20.0	16.0	16.5	33.2	50.0	28.7
qw16	14.0	46.8	55.5	15.1	7.5	14.6	8.3	21.0	25.6	11.2	22.0
qw32	38.5	52.2	60.2	22.2	7.7	15.9	8.9	36.0	36.9	25.6	30.4
qw64	43.0	53.1	63.4	22.0	14.2	18.8	13.2	43.3	37.5	36.2	34.4
qw128	47.9	57.4	65.6	25.8	14.8	20.3	21.6	45.5	33.2	48.6	37.6
Concatenation	52.1	58.2	72.2	37.4	38.5	27.1	26.7	44.4	39.5	56.1	45.3
Proposed	52.2	63.1	75.9	43.8	41.6	27.6	27.2	52.7	41.9	56.6	48.3

Fig. 2 Average Precision (%) on VOC2006 database, Image categorization using linear SVMs and different features. Each column presents results from one of the 10 categories, except for the last one which presents result mean Average Precision. Correspondence between category number and category name can be found in Fig. 4.1. The ever last row present the result using a concatenation of all lab and qw features. The last row presents the result using the proposed method to combine lab and qw features.

Rounds	16	32	64	128	256	512	1024
T=50	78.24	78.24	78.24	78.24	78.24	78.24	78.24
T=100	83.73	83.73	83.73	83.73	83.73	83.73	83.73
T=200	84.12	84.80	85.78	85.88	85.88	85.88	85.88
T=300	.	84.80	86.27	85.88	86.47	86.47	86.47
T=400	.	.	86.67	86.27	87.55	87.55	87.55
T=500	.	.	.	86.47	87.94	88.33	88.33

Fig. 3 Classification rates on Flower 17 according to the maximum dimension of features (from 16 to 1024), and to the maximum number T of rounds in the proposed algorithm (from 50 to 500).

Method	Result
MKL	87.2 ± 2.7
NLP- β	87.9 ± 1.8
NLP- ν MC	87.8 ± 2.1
NLP-B	87.3 ± 2.7
MKL-prod	85.5 ± 1.2
MKL-avg (l_∞)	84.9 ± 1.9
CF (l_∞) / AKM	86.7 ± 2.7
CG-Boost	84.8 ± 2.2
MKL (SILP or Simple)	85.2 ± 1.5
LP- β	85.5 ± 3.0
LP-B	85.4 ± 2.4
MKL-FDA (l_p)	86.7 ± 1.2
Proposed	88.3 ± 1.1

Fig. 4 Classification rate on Flowers 17. Except for our proposition, results are reported from [16]

4.2 Comparison to Multiple Kernel Learning (MKL)

In this section, we compare our method to state-of-the art Multiple Kernel Learning (MKL) techniques. We use the Oxford databases [17], a usual database in communities to compare kernel combination techniques. In these experiments, we use the χ^2 distance matrices $\mathbf{D}$ provided by the authors [17]. For each distance matrix, we compute a new matrix $\mathbf{X} = e^{-\mu_f \mathbf{D}}$, where μ_f is set to be 1 over the mean value of the χ^2 distances over all the training images (like stated in [17]). Then we use the method based on a PCA to generate many features with different hypothetical generalization capacities (cf section. 3.2). Our method then produces a new feature thanks to the learned embedding function $F_T(\mathbf{x})$, which is further used to train a SVM with a Gaussian ℓ_2 kernel. We conduct the experiments with the same protocol as in [17], we use the same features, the same data split and the same performance metric (average accuracy).

4.2.1 Flowers 17

Flowers 17 is a database of 17 categories of common flower in England. Each category have 80 flower images. The dataset is made of 3 predefined random splits. Each split is composed of 3 sets: training (40 images per class), valida-

tion (20 images per class) and test (20 images per class). We consider the visual features given by the authors of this base¹. These authors provide 7 χ^2 distance matrices to describe different properties of the flowers: colour, shape and texture histogram, HSV histogram, SIFT inside the foreground region, SIFT on the foreground region boundary and Histogram of Oriented Gradients.

We run our method with different settings. In section 3.2 we discussed the importance of the granularity of feature for a good generalization. In fact, we build our classifier with features of different granularity. Initially, the coarsest features (i.e in our case the feature with low dimensions) are tried first. If the boosting algorithm fails to find a good kernel from these, then we consider more refined features with more information and detail. In practice, after PCA we keep features of 16, 32, 64, 128, 256, 512 and 1024 dimensions and we start our selection with low dimensions features.

The results of this strategy are reported in Fig. 3. This table presents the classification rate for different maximum values of feature dimensions and rounds T (see Algorithm 1). For example, in the case where T is 300 and the maximum feature dimension 128, the algorithm is allowed to select up to 300 weak kernels using features of dimension 16, 32, 64 or 128. In some cases the algorithm does not select as much weak kernels as requested, for instance with a maximum feature dimension of 16 (first column of results) and with T values of 300, 400 and 500. In those cases, we did not report a result because it is the same as for $T = 200$ since the algorithm did not select more than 200 weak kernels.

One can also see that results are the same in the first row of results ($T = 50$). Whatever is the maximum size of low-level feature dimensions, the accuracy is always 78.24%. That means that, during the 50 first rounds of the algorithm, only features of dimension 16 are selected. The same can be observed in the second row of results ($T = 100$), and we have to wait for $T = 200$ rounds to see the first selection of features of dimension 32. This behavior is repeated with the increasing of rounds, showing the ability of the method to selected more and more complex features as the number of rounds increases.

We reported results from the literature [16] in Fig. 4. Our method outperforms these reference results, and gets lower standard deviation. This standard deviation is computed thanks to the 3 data split predefined by the authors of the dataset.

4.2.2 Flowers 102

Flower 102 database contains 8189 images divided into 102 flowers classes (cf. Fig. 5). Each class consists in about 40 up to 250 images and the distribution of the images between classes is unbalanced. The dataset is divided into three sets: training, validation and test set. The training and validation sets consist

¹ <http://www.robots.ox.ac.uk/~vgg/data/flowers/17/index.html>

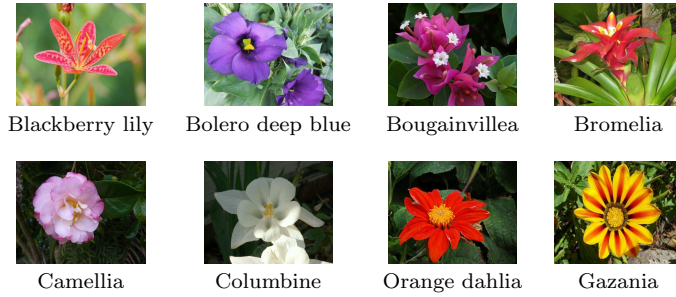


Fig. 5 Examples of categories from the database Flowers 102

Method	Result
MKL [17]	72.8
MKL [16]	73.4
NLP- β	75.7
NLP- ν MC	73.4
NLP-B	73.6
MKL-prod	73.8
MKL-avg	73.4
Proposed	77.8

Fig. 6 Classification rate on Flower 102. Except for our proposition, results are reported from [16]

in 10 images per class (1030 images) while the test set has 6129 images with at least 20 images per class.

Like [17] we use four different χ^2 distance matrices² to describe different properties of the flowers: HSV histogram, SIFT inside the foreground region, SIFT on the foreground region boundary and Histogram of Oriented Gradients.

In Fig. 6., we compare our method with the methods from the literature and mainly to the reference method of the state of the art [16]. It can be seen that our method provides a significant gain. Note that, since there is only one single data split in the original benchmark, no standard deviation can be computed. Changing the split process would bias the protocol which also explains why no standard deviation is reported in any other paper considering this benchmark.

5 Conclusion

This paper presents a new method to learn a kernel function for image categorization in large databases. This method has several advantages in this specific context, both during the training step, where the category models are learnt, and during the categorization step, where images are categorized.

² <http://www.robots.ox.ac.uk/~vgg/data/flowers/102/index.html>

Considering the training step, we follow a two-stage approach. We first learn a kernel function, and then use it to train a SVM classifier for each category. The kernel function is built using a Boosting approach, which allows a combination from a very large number of weak kernels, actually infinite thanks to well-chosen weak learners. Furthermore, we propose an algorithm to perform this learning with a computational complexity linear with the size of the training set. Note that we learn a single kernel function for all categories, using all the labels from the categories, thanks to novel learners target in the Boosting process. This property allows the method to take advantage of any kind of relationship between categories, for instance hierarchy. From the computational point of view, this is also an interesting property for dealing with current datasets, which number of categories grows more and more.

Considering the categorization step, the method we proposed creates an embedding function that maps any image into a Hilbert space. The categorization of any image is then the evaluation of this single function, followed by the projection on the SVM hyperplanes (one for each category). Since the dimension of the Hilbert space where images are embedded is small, the computations for the projection on hyperplanes is negligible. Most of the computation is then the embedding, that depends on the choice of the weak learners and underlying visual features. However, since we expect from these learners to be weak, a relevant choice is to use simple projection models and low dimension features. For instance, in this paper, we used Least Mean Squares with features from 16 to 1024 dimensions, and only with 128 dimensions we are still close to the best results. As a result, the categorization step using our approach, with a relevant tuning, leads to a fast processing of large databases.

The method we proposed also offers a wide range of new techniques, especially if we consider the weak learners. In this paper, we used a very simple weak learner (LMS). However, many other choices can be done, such as linear learners or k-Nearest-Neighbors... We have evaluated our method on real data. The experiments show that the final centroids are representative of category clusters. We compare our method with recent State-of-the-Art methods considering different types of features. Our method outperforms all other reference methods, thereby showing the relevance of our approach.

We are currently working on a straight forward generalization of our method to collaborative learning. Indeed, the same algorithm can easily target a kernel matrix for collaborative learning context by considering several retrieval session result as initial annotations of the classes.

References

1. Francis R. Bach and Gert R. G. Lanckriet. Multiple kernel learning, conic duality, and the smo algorithm. In *International Conference on Machine Learning*, 2004.
2. C. Cortes. Invited talk: Can learning kernels help performance? In *International Conference on Machine Learning*, page 161, 2009.
3. C. Cortes, M. Mohri, and A. Rostamizadeh. Two-stage learning kernel algorithms. In *International Conference on Machine Learning*, 2010.

4. K. Crammer, J. Keshet, and Y. Singer. Kernel design using boosting. In *Advances in Neural Information Processing Systems*, pages 537–544. MIT Press, 2003.
5. N. Cristianini, J. Shawe-Taylor, A. Elisseeff, and J. Kandola. On kernel target alignment. In *Advances in Neural Information Processing Systems*, pages 367–373, Vancouver, Canada, December 2001.
6. J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *International Conference on Machine Learning*, volume 227, Corvallis, Oregon, 2007.
7. A.R. Figueiras-Vidal and L. Rokach. An exploration of research directions in machine ensemble theory and applications. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2012.
8. J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 2:337–374, April 2000.
9. P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *IEEE International Conference on Computer Vision*, pages 221–228, 2009.
10. P.H. Gosselin and M. Cord. Feature based approach to semi-supervised similarity learning. *Pattern Recognition, Special Issue on Similarity-Based Pattern Recognition*, 39:1839–1851, 2006.
11. P.H. Gosselin, M. Cord, and S. Philipp-Foliguet. Combining visual dictionary, kernel-based similarity and learning strategy for image category retrieval. *Computer Vision and Image Understanding, Special Issue on Similarity Matching in Computer Vision and Multimedia*, 110(3):403–417, 2008.
12. P.H. Gosselin, F. Precioso, and S. Philipp-Foliguet. Incremental kernel learning for active image retrieval without global dictionaries. *Pattern Recognition*, DOI: 10.1016/j.patcog.2010.12.006, 2010.
13. T.J. Hazen. Multi-class svm optimization using mce training with application to topic identification. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 5350–5353, 2010.
14. M. Kawanabe, S. Nakajima, and A. Binder. A procedure of adaptive kernel combination with kernel-target alignment for object classification. In *ACM International Conference on Image and Video Retrieval*, 2009.
15. G. Madzarov, D. Gjorgjevikj, and I. Chorbev. A multi-class svm classifier utilizing binary decision tree. *Informatika*, 33(1):233–242, 2008.
16. Krystian Mikolajczyk Muhammad Awais, Fei Yan and Josef Kittler. Augmented kernel matrix vs classifier fusion for object recognition. In *British Machine Vision Conference*, pages 60.1–60.11. BMVA Press, 2011. <http://dx.doi.org/10.5244/C.25.60>.
17. M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.
18. David Picard, Nicolas Thome, Matthieu Cord, and Alain Rakotomamonjy. Learning geometric combinations of gaussian kernels with alternating quasi-newton algorithm. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2012.
19. J. Shawe-Taylor and N. Cristianini. *Kernel methods for Pattern Analysis*. Cambridge University Press, ISBN 0-521-81397-2, 2004.
20. Kinh Tieu and Paul Viola. Boosting image retrieval. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 228–235, 2000.
21. V. Vapnick. *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, 1982.
22. Manik Varma and Bodla Rakesh Babu. More generality in efficient multiple kernel learning. In *International Conference on Machine Learning*, 2009.
23. R. Vert. Designing a m-svm kernel for protein secondary structure prediction. Master’s thesis, DEA informatique de Lorraine, 2002.