STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LVII**, Number 4, 2012

HOW THE KERNELS CAN INFLUENCE IMAGE CLASSIFICATION PERFORMANCE

LAURA DIOŞAN[(1)] AND ALEXANDRINA ROGOZAN[(2)]

Abstract. Support Vector Machines deliver state-of-the-art performance in real-world applications and are now established as one of the standard tools for machine learning and data mining. A key problem of these methods is how to choose an optimal kernel and how to optimise its parameters. Selection of the most appropriate kernel highly depends on the problem at hand and fine tuning its parameters can easily become a tiresome and awkward task. Our purpose is to investigate how the used kernels and their parameters influence the learning performance in the context of a particular classification task: object recognition. The numerical results indicate that the best kernel function depends on the problem to be solved.

1. Introduction

The performance of a classification algorithm is strongly influenced by two ingredients: first, a suitable representation of the objects to be categorized and second a powerful decision maker algorithm on top of this representation. Even if the first aspect has been deeply discussed and analysed in the community of computer vision and the second one inside the machine learning community, the combination of them (how to combine the image representation with a learning algorithm) is still a highly challenging problem.

One of the most important issues in computer vision is how to extract relevant image features. The features that characterise an image can be classified from many points of view. An important criterion is the area from that the feature is extracted (global features and local feature). Another important criterion is the complexity of extraction (low-level features or high level features).

Received by the editors: November 25, 2012.

2000 *Mathematics Subject Classification.* 68T05,91E45.

1998 *CR Categories and Descriptors.* I.2.6 [**Artificial Intelligence**]: Learning – *Concept learning*.

Key words and phrases. Object recognition, Kernel descriptors, Support Vector Machines, Kernel selection.

Furthermore that to extract these features from an image, it is very important to store all these characteristics in an adequate representation such that a learning algorithm can work with them in order to label the corresponding images by a particular class. The most efficient representations are: bag of words [21] and kernels of local features [8].

The second aspect that must be considered when the problem of object recognition has to be solved is the classification algorithm. Since the classification must be performed in an automatically manner, a machine learning algorithm can be utilised. The general problem of machine learning is to search a, usually very large, space of potential hypotheses to determine the one that will best fit the data and any prior knowledge. In supervised image classification, we are given a training set of images and their corresponding labels. The goal is to learn (based on the training set) a classifier to label unseen images.

There are many learning algorithms today and their performances (estimated by different measures, e.g. classification accuracy, solution correctness, solution quality or speed of performance) are related not only to the problem to be solved, but also to their parameters. Therefore, the best results can be achieved only by identifying the optimal values of these parameters. Although this is a very complex task, different optimisation methods have been developed in order to optimise the parameters of Machine Learning algorithms.

In this paper we present a short survey of the most important image features and how they can be involved in a particular representation suitable for an automatically learning process based on kernel methods, since the kernel-based methods have been proved to reach good efficiency in solving such type of problems (with a particular emphasis of Support Vector Machines (SVMs) [23, 24]). Furthermore, we will compare the classification performance by taking into account different image representations and different kernels involved in SVM learning. In fact, we will develop a process of kernel selection at two levels:

- image processing level - at this stage we try to identify which is the most suitable kernel patch descriptors and its kernel in order to transform an image into an efficient and representative vector of features;
- learning level - here we try to identify the kernel involved in the decision process with the highest performance; the kernel is utilised to map the input of the classification algorithm from a non-linear separable space into a linear separable one.

The paper is organized as follows: Section 2 outlines the most important image features and representations, while Section 3 outlines the theory behind SVM classifiers giving a particular emphasis to the kernel functions. This is

followed by Section 4 where our study case and the results of the experiments are presented and discussed. Finally, Section 5 concludes the paper.

2. Image representation

A highly challenging problem in computer vision is how to extract relevant image features. The features that characterise an image can be classified from many points of view. An important criterion is the area from that the feature is extracted: if the entire image is used, then some global features are computed, while if one or more image regions (patches) are utilised, then local features are determined. Another important criterion is the complexity of extraction. Image features can be extracted from scratch that means the features are extracted directly from the image (and in this case we discuss about low-level features) or can be computed based on some previously extracted features (in this case high-level features are obtained).

Examples of low-level features or image descriptors include: characteristics extracted from an image (global feature) or from one or more patches of an image (local features) around salient interest points or regular grids, characteristics regarding the edges, corners or blobs from the image/patches. The most popular (due to their success) low-level image descriptors are:

- orientation histograms such as Scale-Invariant Feature Transform (SIFT) [15] - based on a rectangular grid [12];
- Gradient Location and Orientation Histogram (GLOH) [17] - based on a log-polar grid [12];
- Histogram of Oriented Gradients (HOG) [6] - based on a radial grid [12];
- Speeded-Up Robust Features (SURF) and Haar [12];

while one of the best high-level descriptor is the kernel view of orientation histograms [1]. The results presented by L. Bo in [1] indicate that the performance of an image classifier based on the family of kernel descriptors surpasses the performance of a classifier that works only with low-level features (for instance a classifier based on SIFT features or only on HOG). In their work, the authors have indicated that the kernel descriptors are able to convert the pixel attributes into compact patch-level features. For pixel attributes the authors have considered the orientation, the position and the colour of each pixel (from a patch). Furthermore, the authors have introduced three types of match kernels to measure similarities between image patches. They have introduced these new similarity measures since the previous ones (based on HOG) suffer from discretisation (HOG can be considered a special case of linear kernels, but with a restrictive set of values – 1 or 0). The investigated match kernels from [1] are:

- the gradient match kernel (able to capture image variations) based on a kernel of magnitudes, an orientation kernel and a position kernel;
- the colour kernel (able to describe image appearance) based on a colour kernel and a position kernel;
- the local binary pattern kernel (able to capture local shape more effectively) based on a kernel of standard deviations of neighbour pixels, a kernel of binarized pixel value differences in a local window and a position kernel.

Furthermore that to extract these features from an image, it is very important to store all these characteristics in an adequate representation such that a learning algorithm can work with them in order to label the corresponding images by a particular class. These features extracted from an image can be used directly or indirectly by a classification algorithm. In the first case, the image features are simply concatenated into vectors that are utilised as input data by the classifier. The main drawback of such approach is the length of input vectors (sometimes it can be huge and the computational cost of the learning process is very large). Therefore, other efficient representations of image features have been identified and two of them are:

- Bag of words [21] - each local feature is represented with the closest visual word (from a predefined visual vocabulary) and a histogram is computed by counting the occurrence frequencies of words in the entire image (global representation). This histogram is actually used as image descriptor by a learning/recognition algorithm.
- Kernels over local features [8] – a kernel function is required in order to compare two images by using the previously extracted features (local descriptors). The local features are mapped into a low dimensional space by using kernel functions on sets such as:
  - sum match kernel [11] - adds all kernels corresponding to all combinations of local features extracted from two images;
  - neighbourhood match kernel [18] - similar to the previous one, but take into account the spatial location of local features also;
  - pyramid match kernels [9, 13, 14] - the local features are transformed into a multi-resolution histogram;
  - efficient match kernels [2] - set-level features are constructed by averaging the resulting feature vectors.

3. Learning algorithm

The general problem of Machine Learning is to search a, usually very large, space of potential hypotheses to determine the one that will best fit the data and any prior knowledge. In 1995, SVMs marked the beginning of a

new era in the paradigm of learning from examples. Rooted to the Statistical Learning Theory and the Structural Risk Minimization principle developed by Vladimir Vapnik at AT&T in 1963 [23, 24], SVMs gained quickly attention from the Machine Learning community due to a number of theoretical and computational merits.

SVMs are a group of supervised learning methods that can be applied to classification or regression. SVMs arose from statistical learning theory; the aim being to solve only the problem of interest without solving a more difficult problem as an intermediate step. SVMs are based on the structural risk minimisation principle, closely related to regularisation theory. This principle incorporates capacity control to prevent over-fitting and thus is a partial solution to the bias-variance trade-off dilemma.

One issue with SVMs is finding an appropriate positive definite kernel (and its parameters) for the given data. A wide choice of kernels already exists. Many data or applications may still benefit from the design of particular kernels, adapted specifically to a given task (i.e. kernels for vectors, kernels for strings, kernels for graphs, Fisher kernels or rational kernels). There are only some hints for working with one or another of these classic kernels, because there is no rigorous methodology to choose *a priori* the appropriate one between them. Moreover, the kernel parameters influence the performance of the SVM algorithm. The selection of the penalty error for an SVM (that controls the trade-off between maximizing the margin and classifying without error) is also critical in order to obtain good performances. Therefore, one has to optimise the kernel function, the kernel parameters and the penalty error of the SVM algorithm in order to guarantee the robustness and the accuracy of an SVM algorithm. Chapelle [4] has proposed to denote the kernel and SVM parameters as hyper-parameters.

3.1. **Generalities of SVM.** Initially, SVM algorithm has been proposed in order to solve binary classification problems [23]. Later, these algorithms have been generalized for multi-classes problems. Consequently, we will explain the theory behind SVM only on binary-labelled data.

Suppose the training data has the following form: $D = (x_i, y_i)_{i=\overline{1,m}}$, where $x_i \in \Re^d$ represents an input vector and each y_i, $y_i \in \{-1, +1\}$, the output label associated to the item x_i. SVM algorithm maps the input vectors to a higher dimensional space where a maximal separating hyper-plane is constructed [23]. Learning the SVM means to minimize the norm of the weight vector (w in Eq. (1)) under the constraint that the training items of different classes belong to opposite sides of the separating hyper-plane. Since $y_i \in \{-1, +1\}$ we can formulate this constraint as:

$$y_i(w^T x_i + b) \geq 1, \ \forall i \in \{1, 2, \ldots, m\}, \tag{1}$$

where the primal decision variables w and b define the separating hyper-plane and v^T represents the transpose of v.

The items that satisfy Eq. (1) with equality are called support vectors since they define the resulting maximum-margin hyper-planes. To account for misclassification, e.g. items that do not satisfy Eq. (1), the soft margin formulation of SVM has introduced some slack variables $\xi_i \in \Re$ [5].

Moreover, the separation surface has to be nonlinear in many classification problems. SVM was extended to handle nonlinear separation surfaces by using a feature function $\phi(x)$. The SVM extension to nonlinear datasets is based on mapping the input variables into a feature space $\mathcal{F}$ of a higher dimension and then performing a linear classification in that higher dimensional space. The important property of this new space is that the data set mapped by ϕ might become linearly separable if an appropriate feature function is used, even when that data set is not linearly separable in the original space.

Hence, to construct a maximal margin classifier one has to solve the convex quadratic programming problem encoded by Eq. (2), which is the primal formulation of it:

$$\begin{array}{ll} \text{minimise}_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i & \\ \text{subject to:} & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, \forall i \in \{1, 2, \ldots, m\}. \end{array} \tag{2}$$

The coefficient C (usually called *penalty error* or *regularization parameter*) is a tuning parameter that controls the trade off between maximizing the margin and classifying without error. Larger values of C might lead to linear functions with smaller margin, allowing to classify more examples correctly with strong confidence. A proper choice of this parameter is crucial for SVM to achieve good classification performance.

Instead of solving Eq. (2) directly, it is a common practice to solve its dual problem, which is described by Eq. (3):

$$\begin{array}{ll} \text{maximise}_{a \in \Re^m} \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{i,j=1}^{m} a_i a_j y_i y_j \phi(x_i)^T \phi(x_j) & \\ \text{subject to} & \sum_{i=1}^{m} a_i y_i = 0, \\ & 0 \leq a_i \leq C, \forall i \in \{1, 2, \ldots, m\}. \end{array} \tag{3}$$

In Eq. (3), a_i denotes the Lagrange variable for the i^{th} constraint of Eq. (2).

The optimal separating hyper-plane $f(x) = w \cdot \phi(x) + b$, where w and b are determined by Eq. (2) or Eq. (3) is used to classify the un-labelled input data x_k:

$$y_k = sign \left(\sum_{x_i \in S} a_i \phi(x_i)^T \phi(x_k) + b \right), \tag{4}$$

where S represents the set of support vector items x_i.

We will see in the next section that is more convenient to use a kernel function $K(x, z)$ instead of the dot product $\phi(x)^T\phi(z)$.

3.2. **Kernel formalism.** The original optimal hyper-plane algorithm proposed by Vapnik in 1963 was a linear classifier [23]. However, in 1992, Boser, Guyon and Vapnik [3] have suggested a way to create non-linear classifiers by applying the *kernel trick*. Kernel methods work by mapping the data items into a high-dimensional vector space $\mathcal{F}$, called feature space, where the separating hyper-plane has to be found [3]. This mapping is implicitly defined by specifying an inner product for the feature space via a positive semi-definite kernel function: $K(x, z) = \phi(x)^T\phi(z)$, where $\phi(x)$ and $\phi(z)$ are the transformed data items x and z [20]. Note that all we required is the result of such an inner product. Therefore we do even not need to have an explicit representation of the mapping ϕ, neither to know the nature of the feature space. The only requirement is to be able to evaluate the kernel function on all the pairs of data items, which is much easier than computing the coordinates of those items in the feature space.

The kernels that correspond to a space embedded with a dot product belong to the class of positive definite kernels. This has far-reaching consequences. The positive definite and symmetric kernels verify the Mercer's theorem [16] - a condition that guarantees the convergence of training for discriminant classification algorithms such as SVMs. The kernels of this kind can be evaluated efficiently even though they correspond to dot products in infinite dimensional dot product spaces. In such cases, the substitution of the dot product with the kernel function is called the *kernel trick* [3].

In order to obtain an SVM classifier with kernels one has to solve the following optimization problem:

$$\begin{array}{ll} \text{maximise}_{a\in\Re^m} \sum_{i=1}^m a_i - \frac{1}{2}\sum_{i,j=1}^m a_i a_j y_i y_j K(x_i, x_j) \\ \text{subject to} \quad \sum_{i=1}^m a_i y_i = 0, \\ \qquad\qquad 0 \le a_i \le C, \forall i \in \{1, 2, \ldots, m\}. \end{array} \tag{5}$$

In this case, Eq. (4) becomes:

$$y_k = sign\left(\sum_{x_i\in S} a_i K(x_i, x_k) + b\right), \tag{6}$$

where S represents the set of support vector items x_i.

There are a wide choice for a positive definite and symmetric kernel K from Eq. (6). The selection of a kernel has to be guided by the problem that must be solved. Choosing a suitable kernel function for SVMs is a very important step for the learning process. There are few if any systematic techniques to

assist in this choice. Until now, different kernels for vectors have been proposed [22]; the most utilised of them by an SVM algorithm are listed in Table 1.

TABLE 1. The expression of several classic kernels.

Name	**Expression**	**Type**
Linear	$K_{Lin}\ \ (x,z) = x^T \cdot z$	projective
Polynomial	$K_{Pol}\ \ (x,z) = (x^T \cdot z + coef)^d$	projective
Normalised Polynomial	$K_{NPol}(x,z) = \frac{K_{Pol}(x,z)}{\sqrt{K_{Pol}(x,x)K_{Pol}(z,z)}}$	projective
Laplacian	$K_{Lapl}(x,z) = exp(-\frac{\lvert x-z\rvert}{\sigma})$	radial
Exponentail	$K_{Exp}(x,z) = exp(-\frac{\lvert x-z\rvert}{2\sigma^2})$	radial
Gaussian	$K_{Gauss}(x,z) = exp(-\frac{\lvert x-z\rvert^2}{2\sigma^2})$	radial
Euclidean	$K_{Euclid}(x,z) = \frac{\lvert x-z\rvert^2}{2\sigma^2}$	radial

While one of the first feelings about SVM algorithms is that they can solve a learning task automatically, it actually remains challenging to apply SVMs in a fully automatic manner. Questions regarding the choice of the kernel function and the hyper-parameters values remain largely empirical in real-world applications. While default setting and parameters are generally useful as a starting point, major improvements can result from careful choosing of an optimal kernel. There are many types of kernels for vectors and several criteria could be used for classifying them.

While SVM classifiers intrinsically account for a trade off between model complexity and classification accuracy [24], the generalization performance is still highly dependent on appropriate selection of the penalty error C and kernel parameters. Thus, several methods could be used to optimise the hyper-parameters of an SVM classifier.

Ideally, we would like to choose the value of the kernel parameters that minimise the true risk of the SVM classifier. Unfortunately, since this quantity is not accessible, one has to build estimates or bounds for it.

Cross-validation is a popular technique for estimating the generalization error and there are several interpretations [25]. In k-fold cross-validation, the training data is randomly split into k mutually exclusive subsets (or folds) of approximately equal size. The SVM decision rule is obtained by using $k-1$ subsets on training data and then tested on the subset left out. This procedure is repeated k times and in this manner each subset is used for testing once. Averaging the test error over the k trials gives a better estimate of the expected generalization error.

4. Study case

4.1. **Proposed framework.** Our aim is to investigate how the kernel function influences the performance of learning process. Therefore, we considered the framework proposed by L. Bo [1] for image classification and we test different kernel functions. We already establish that the selection of the kernel function is very important for SVM (see [7]). This time, our investigation about how kernel function affects classification process is developed at two levels:

- at the level of image descriptor and
- at the level of learning process.

In the first case, based on the available code of Kernel descriptors developed by Xiaofeng Ren (*http* : *//www.cs.washington.edu/ai/Mobile_Robotics /projects/kdes/*), we have tested different kernels when the local features are extracted from an image. Because we work only with gray images, we investigate only the kernels descriptors able to capture image variations (gradient match kernels [1]). As we already presented in Section 2, the Bo's gradient match kernel is composed by three kernels: a kernel of magnitudes, an orientation and a position kernel.

The magnitude kernel is a linear one and its role is to measure the similarity of gradient magnitudes of two pixels. The magnitude kernel type cannot be changes since it must be an equivalent of histogram of gradients in the feature map (a pixel has associated a feature vector obtained by multiplying the magnitude and the orientation of a pixel over all considered orientation bins).

The other two kernels involved in Ren's computation of the gradient match kernel, the orientation kernel (for computing the similarity of gradient orientations) and the position kernel (for measuring how close two pixels are spatially), are actually Gaussian kernels. Therefore, we have changes the implementation and we have involved in the feature extraction process more possible orientation and position kernels (Exponential, Laplacian, Euclidean). The expression of these kernels is given in Table 1.

In the second case, that of learning, the classifier utilised in our experiments is a kernel-based SVM. Again, we have tried to use several kernels with different parameters during the learning process in order to identify the best one. These kernels are the linear kernel, the Polynomial kernel, the Gaussian kernel and the Normalised Polynomial kernel. For the Polynomial kernel several exponents have been tested (2, 3), for parameter $\frac{1}{2\sigma^2}$ of Gaussian kernel the following values have been checked: 0.1, 0.01, 0.001, 0.0001 and for Normalised Polynomial kernel the exponent was 2.

The dual version of the optimisation problem which arises during the training of support vector machines was solved by Sequential minimal optimization (SMO) algorithm [19], since it is able to quickly solved the quadratic programming optimisation problem of SVM. We have chosen this formulation of SVM since the duality theory provides a convenient way to deal with the constraints and, in this form, the optimisation problem can be solved in terms of dot products, that allows using the kernel trick. Furthermore, SMO requires an amount of memory that increases only linear with the training set size, being able to handle very large training sets - as in the image classification case. These aspects are different to L. Bo's framework that is based on primal formulation of SVM and on conjugate gradient optimisation methods (in fact, Newton optimisation).

4.2. **Numerical experiments.** Several numerical experiments about how kernel selection influences the classification process in the case of image recognition task are presented. A benchmark (http://www.cs.unc.edu/ lazebnik) was considered in our study-case. More details about these data can be found in [14].

For all datasets a binary classification problem was actually solved: D1 corresponds to a classification between bedroom and kitchen images, D2 corresponds to a decision coast images *vs.* forest images, while in D3 we have to delimitate industrial scenes to suburban scenes. In all the cases, 50 images are utilised (the decision model is trained on 2/3 of them, while 1/3 of images are used for testing). All the experiments are performed by using a cross-validation technique of 3 folds and they are performed by using the Weka tool [10].

In order to measure the classification performance, the accuracy rate was actually computed. The accuracy rate represents the number of correctly classified items over the total number of items from a given data set.

In Tables 2, 3, 4 are presented the average accuracy rates for each dataset by considering different image descriptor kernels (when the SVM input vectors are actually constructed) and different SVM kernel functions for the first dataset (D1), the second dataset (D2) and the third dataset (D3), respectively.

Several remarks cab ne done based on the results from Tables 2, 3, 4. Regarding the kernel descriptors, the Gaussian kernel is the best one in 2 cases (D2 and D3), but for the first data the Exponential kernel perform better. Furthermore, if we consider all combinations, the Exponential and the Gaussian kernel have won 9 times each in terms of accuracy rate. Taking into account the computation time required for evaluating the kernel expression, we promote to use the exponential kernel (since involve just a simple norm, not a square one – see Table 1).

SVM kernel *vs.* Kernel Descriptor	Exponential	Gaussian	Laplacian	Euclid
Lin	74%	69%	65%	45%
Poly(2)	70%	67%	62%	50%
Poly(3)	59%	67%	55%	53%
Gaussian(0.1)	51%	55%	48%	46%
Gaussian(0.01)	50%	52%	43%	49%
Gaussian(0.001)	54%	54%	40%	42%
Gaussian(0.0001)	49%	49%	38%	53%
NormPoly(2)	49%	51%	45%	54%

TABLE 2. Accuracy rates obtained for dataset D1 by SVM algorithm with different kernel functions on images represented by different kernel descriptors.

SVM kernel *vs.* Kernel Descriptor	Exponential	Gaussian	Laplacian	Euclid
Lin	97%	99%	98%	66%
Poly(2)	94%	96%	96%	67%
Poly(3)	80%	80%	95%	63%
Gaussian(0.1)	83%	83%	85%	59%
Gaussian(0.01)	80%	80%	72%	60%
Gaussian(0.001)	57%	57%	72%	63%
Gaussian(0.0001)	56%	56%	75%	51%
NormPoly(2)	62%	62%	64%	49%

TABLE 3. Accuracy rates obtained for dataset D2 by SVM algorithm with different kernel functions on images represented by different kernel descriptors.

SVM kernel *vs.* Kernel Descriptor	Exponential	Gaussian	Laplacian	Euclid
Lin	79%	88%	80%	59%
Poly(2)	83%	80%	77%	57%
Poly(3)	68%	65%	66%	54%
Gaussian(0.1)	65%	63%	68%	44%
Gaussian(0.01)	60%	63%	56%	52%
Gaussian(0.001)	65%	51%	60%	54%
Gaussian(0.0001)	67%	57%	63%	49%
NormPoly(2)	57%	49%	57%	53%

TABLE 4. Accuracy rates obtained for dataset D3 by SVM algorithm with different kernel functions on images represented by different kernel descriptors.

Regarding the SVM kernels, the best results are obtained for all datasets by using a linear kernel (that means the data can be simple separate by a hyper plane, without requiring hyper spheres or conjunction of feature as in the case of Gaussian and Polynomial kernel, respectively).

5. Conclusions

An important problem was investigated in this paper: how kernel functions can affect the performance of object recognition process. In fact, the kernel functions are involved at two levels: that of extraction of image features and that of learning method. The most promising kernel function involved in the classification algorithm seems to be the simple linear one, while in the case of kernel descriptors (that extract image features) the results indicate that we cannot identify a best kernel. Each problem seems to be solved better with other kernel type. Therefore, we plan to investigate how we can automatically adapt the kernel descriptor (and its parameters) to a given set of images in order to increase the classification performance, but without reducing the generality and the extrapolation power of the method. Furthermore, other match kernels (colour and shape kernels) can be considered.

References

[1] Bo, L., Ren, X., and Fox, D. Kernel descriptors for visual recognition. In *NIPS* (2010), J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, Eds., Curran Associates, Inc, pp. 244–252.

[2] Bo, L., and Sminchisescu, C. Efficient match kernel between sets of features for visual recognition. In *NIPS* (2009), Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds., Curran Associates, Inc, pp. 135–143.

[3] Boser, B. E., Guyon, I., and Vapnik, V. A training algorithm for optimal margin classifiers. In *COLT* (1992), D. Haussler, Ed., ACM Press, pp. 144–152.

[4] Chapelle, O., Vapnik, V., Bousquet, O., and Mukherjee, S. Choosing multiple parameters for Support Vector Machines. *Machine Learning 46*, 1/3 (2002), 131–159.

[5] Cortes, C., and Vapnik, V. Support-Vector Networks. *Machine Learning 20*, 3 (1995), 273–297.

[6] Dalal, N., and Triggs, B. Histograms of Oriented Gradients for human detection. In *CVPR* (2005), C. Schmid, S. Soatto, and C. Tomasi, Eds., vol. 2, pp. 886–893.

[7] Diosan, L., Rogozan, A., and Pecuchet, J.-P. Improving classification performance of Support Vector Machine by genetically optimisation of kernel shape and hyper-parameters. *Applied Intelligence 2*, 36 (2012), 280–294.

[8] Eichhorn, J., and Chapelle, O. Object categorization with svm: Kernels for local features. Tech. rep., In Advances in Neural Information Processing Systems, 2004.

[9] Grauman, K., and Darrell, T. J. The pyramid match kernel: Discriminative classification with sets of image features. In *ICCV* (2005), pp. II: 1458–1465.

[10] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *SIGKDD Explorations 11*, 1 (2009), 10–18.

[11] Haussler, D. Convolution kernels on discrete structure. Tech. Rep. UCSC-CRL-99-10, University of California at Santa Cruz, Santa Cruz, CA, USA, July 1999.
[12] Johnson, M. Generalized descriptor compression for storage and matching. In *BMVC* (2010), F. Labrosse, R. Zwiggelaar, Y. Liu, and B. Tiddeman, Eds., British Machine Vision Association, pp. 1–11.
[13] Kumar, A., and Sminchisescu, C. Support kernel machines for object recognition. In *ICCV* (2007), pp. 1–8.
[14] Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR* (2006), pp. II: 2169–2178.
[15] Lowe, D. G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.
[16] Mercer, J. Functions of positive and negative type and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society 83*, 559 (1909), 415–446.
[17] Mikolajczyk, K., and Schmid, C. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell 27*, 10 (2005), 1615–1630.
[18] Parsana, M., Bhattacharya, S., Bhattacharyya, C., and Ramakrishnan, K. R. Kernels on attributed pointsets with applications. In *NIPS* (2007), J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, Eds., Curran Associates, Inc, pp. 1129–1136.
[19] Platt, J. Fast training of Support Vector Machines using Sequential Minimal Optimization. In *Advances in Kernel Methods — Support Vector Learning* (Cambridge, MA, 1999), B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds., MIT Press, pp. 185–208.
[20] Schölkopf, B. The kernel trick for distances. In *NIPS* (Cambridge, MA, 2000), T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., MIT Press, pp. 301–307.
[21] Sivic, J., and Zisserman, A. Video google: A text retrieval approach to object matching in videos. In *ICCV* (2003), pp. 1470–1477.
[22] Taylor, J. S., and Cristianini, N. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.
[23] Vapnik, V. *The Nature of Statistical Learning Theory*. Springer, 1995.
[24] Vapnik, V. *Statistical Learning Theory*. Wiley, 1998.
[25] Wahba, G., Lin, Y., and Zhang, H. GACV for Support Vector Machines. In *Advances in Large Margin Classifiers*, B. Smola and S. SchRolkopf, Eds. MIT Press, Cambridge, MA, 1999.

(1) Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: lauras@cs.ubbcluj.ro

(2) LITIS, EA - 4108, INSA, Rouen, France
E-mail address: arogozan@insa-rouen.fr