Support Vector Machines et Classification d' Images

Olivier Chapelle

Juin-Août 1998

Stage de deuxième année de magistère d'informatique de l'Ecole Normale Supérieure de Lyon effectué sous la direction de Patrick Haffner et Vladimir Vapnik au sein du departement Image Processing Reseach, AT&T, Redbank, NJ, USA

Support Vector Machines for Image Classification

Olivier Chapelle

1 Introduction

Support Vectors Machines (SVM) have recently shown their ability in pattern recognition and classification [Vapnik, 1995]. The aim of this paper is to evaluate the potentiality of SVM on image recognition and image classification tasks.

Intuitively, given a set of points which belong to either of two classes, a linear SVM finds the hyperplane leaving the largest possible fraction of points of the same class on the same side, while maximizing the distance of either class from the hyperplane. According to [Vapnik, 1995], this hyperplane minimizes the risk of misclassifying examples of the test set.

The potential of the SVM is illustrated on a 3D object recognition task using the Coil database and on a image classification task using the Corel database. The images are either representated by a matrix of their pixel values (bitmap representation) or by a color histogram. In both cases, the proposed system does not require feature extraction and performs recognition on images regarded as points of a space of high dimension. We also purpose an extension of the basic color histogram which keeps more about the information contained in the images.

The remarkable recognition rates achieved in our experiments indicate that Support Vector Machines are well-suited for aspect-based recognition and color-based classification.

The report is organized as follows. In section 2, we briefly review the theory of SVM and present new facts about generalization performance and run-time complexity of SVM. Section 3 describes the main features of our classification system and discusses about the choices involved in this system. Section 4 describes the databases Coil and Corel used all along our experiments whose results are illustrated in section 5.

2 Support Vector Machines

2.1 Optimal separating hyperplane

Let $(\mathbf{x}_i, y_i)_{1 \leq i \leq N}$ be a set of training examples, $\mathbf{x}_i \in \mathbb{R}^n$ and belongs to class labeled by $y_i \in \{-1, 1\}$. The aim is to carry out the equation of an hyperplane which divides the set of examples such that all the points with the same label are on the same side of the hyperplane. This means find $\mathbf{w}$ and b such that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) > 0, \ i = 1, \ldots, N \tag{1}$$

If there exists an hyperplane satisfying eq (1), the set is said to be *linearly separable*. In this case, it is always possible to rescale $\mathbf{w}$ and b such that

$$\min_{1 \leq i \leq N} y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1, \ i = 1, \ldots, N$$

i.e. such that the closest point to the hyperplane has a distance of $1/||\mathbf{w}||$. Then, eq (1) becomes

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 \tag{2}$$

Among the separating hyperplanes, the one for which the distance to the closest point is maximal is called *optimal separating hyperplane* (OSH). Since the distance to the closest point is $1/||\mathbf{w}||$, finding the OSH amounts to solve the following problem :

minimize

$$\frac{1}{2}\mathbf{w} \cdot \mathbf{w} \tag{3}$$

under constraints (2)

The quantity $2/||\mathbf{w}||$ is called the margin and thus, the OSH is the separating hyperplane which maximizes the margin. The margin can be seen as a measure of difficulty of the problem : the smaller the margin is, the more difficult the problem is. See theorem 2, page 6 for details : the larger the margin is, the better the generalization is expected to be (see figure 1)

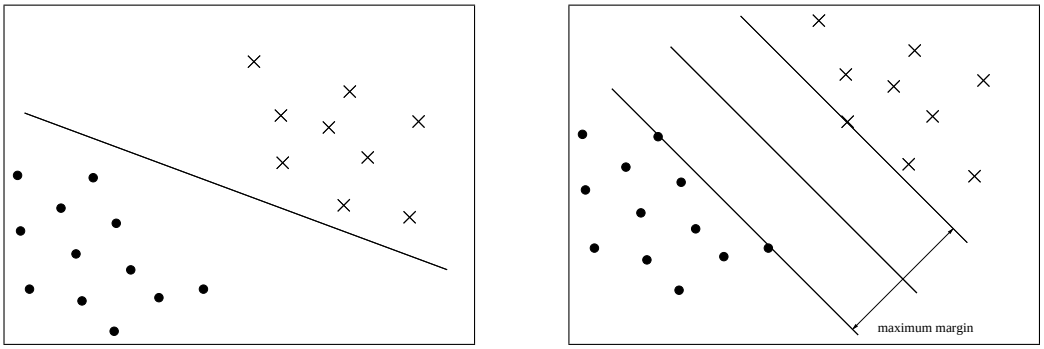

Figure 1: *Both hyperplanes separate correctly the training examples. But the Optimal Separating Hyperplane on the right hand side has a larger margin and is expected to give better generalization*

Since $\mathbf{w}^2$ is convex, minimizing eq (3) under linear constraints (2) can be achieved by the use of Lagrange multipliers. Let us denote the N non negative Lagrange multipliers associated with constraints (2) by $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$. Minimizing eq (3) amounts to find the saddle point of the Lagrange function :

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} - \sum_{i=1}^{N} \alpha_i [y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1] \tag{4}$$

To find this point, one has to minimize this function over $\mathbf{w}$ and b and to maximize it over the Lagrange multipliers $\alpha_i \geq 0$

The saddle point must satisfy the following conditions :

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{b} = \sum_{i=1}^{N} y_i \alpha_i = 0 \tag{5}$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\mathbf{w}} = \mathbf{w} - \sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i = 0 \tag{6}$$

Substituting equations (5) and (6) into (4), our optimization problem amounts to maximize

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{7}$$

with $\alpha_i \geq 0$ and under constraint (5). This can be achieved by the use of standard quadratic programming methods [Bazaraa and Shetty, 1979].

Once the vector $\boldsymbol{\alpha}^{\mathbf{0}} = (\alpha_1^0, \ldots, \alpha_N^0)$ solution of the maximization problem (7) has been found, taking (6) into account, the OSH $(\mathbf{w}_0, b_0)$ has the following expansion

$$\mathbf{w}_0 = \sum_{i=1}^{N} \alpha_i^0 y_i \mathbf{x}_i \tag{8}$$

while b_0 can be determined from the Kuhn-Tucker conditions

$$\alpha_i^0[y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) - 1] = 0 \tag{9}$$

Note that from equation (9), the points for which $\alpha_i^0 > 0$ satisfy inequation (2) with equality. Geometrically, it means that they are the closest points to the optimal hyperplane (see figure 1). These points play a crucial role, since they are the only points needed in the expression of the OSH (see equation (8)). They are called *support vectors* to point out the fact that they "support" the expansion of $\mathbf{w}_0$.

Given a support vector $\mathbf{x}_i$, the parameter b_0 can be obtained from the corresponding Kuhn-Tucker condition as

$$b_0 = y_i - \mathbf{w}_0 \cdot \mathbf{x}_i$$

The problem of classifying a new point $\mathbf{x}$ is solved by by looking at the sign of

$$\mathbf{w}_0 \cdot \mathbf{x} + b_0$$

Considering the expansion (8) of $\mathbf{w}_0$, the hyperplane decision function can thus be written as

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{N} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b\right) \tag{10}$$

2.2 Linearly non-separable case

If the data are not linearly separable, the problem of finding the OSH becomes meaningless. To allow the possibility of examples violating (2), one can introduce slack variables $(\xi_1, \ldots, \xi_N)$ with $\xi_i \geq 0$ [Cortes and Vapnik, 1995] such that

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1 - \xi_i, \; i = 1, \ldots, N \tag{11}$$

The purpose of the variables ξ_i is to allow misclassified points. Points which are misclassified have their corresponding $\xi_i > 1$, so $\sum \xi_i$ is an upper bound on the number of training errors. The generalized OSH is then regarded as the solution of the following problem : minimize

$$\frac{1}{2}\mathbf{w} \cdot \mathbf{w} + C\sum_{i=1}^{N} \xi_i \tag{12}$$

subject to constraints (11) and $\xi_i \geq 0$. The first term is minimized to control the capacity learning as in the separable case; the purpose of the second term is to keep under control the number of misclassified points. The parameter C is chosen by the user, a larger C corresponding to assigning a higher penalty to errors.

In analogy with what was done for the separable case, the use of the Lagrange multipliers leads do the following optimization problem : maximize

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

subject to : $\sum_{i=1}^{N} \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$. The only difference from the separable case is that now the α_i have an upper bound of C.

2.3 Nonlinear Support Vector Machines

The idea of Support Vector Machines is to map the input data into a high dimensional *feature space* through some non linear mapping chosen a priori [Boser et al., 1992]. In the space the OSH constructed (see fig 2).

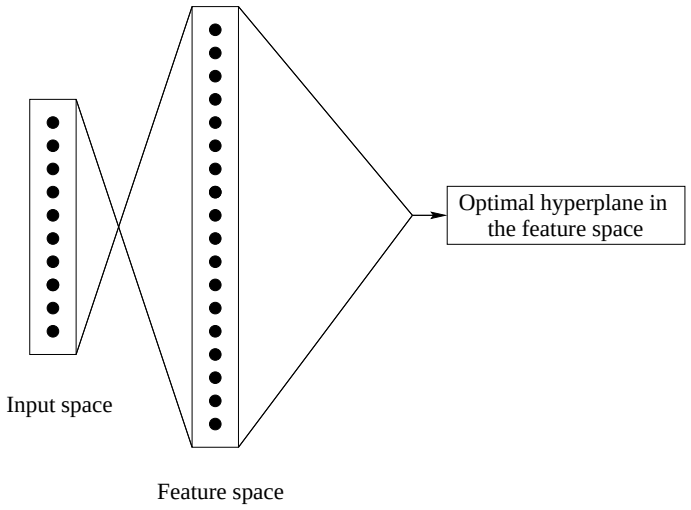

Figure 2: *The SV machine maps the input space into a high-dimensional feature space and then constructs an optimal hyperplane in the feature space*

Example : To construct a decision surface corresponding to a polynomial of degree 2, the following feature space of $\frac{n(n+3)}{2}$ can be created :

$$z_i = x_i,\ 1 \leq i \leq n$$

$$z_{n+i} = (x_i)^2,\ 1 \leq i \leq n$$

$$z_{2n+1} = x_1 x_2, \ldots, z_N = x_n x_{n-1}$$

where $\mathbf{x} = (x_1, \ldots, x_n)$ is the input vector and $\mathbf{z} = (z_1, \ldots, z_N) = \Phi(\mathbf{x})$ is the image of $\mathbf{x}$ through the mapping Φ. The separating hyperplane constructed in the feature space is a second-degree polynomial in the input space.

One computational problem arises : the dimension of the feature space can be very high and how construct a separating hyperplane in this high dimensional space ?

The answer to this problem comes from the fact that to construct the optimal separating hyperplane in the feature space, the mapping $\mathbf{z} = \Phi(\mathbf{x})$ does not need to be explicited. Indeed, if we replace $\mathbf{x}$ by $\Phi(\mathbf{x})$, eq (7) becomes :

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

Then of course the training algorithm would only depend on the data through dot products in the feature space, i.e. on functions of the form $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Now suppose we have a symmetric function K such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$. Then only K is needed in the training algorithm and the mapping Φ is never explicitely used.

Given a mapping Φ, the kernel K is obviously $K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$, but conversely, given a kernel K, which are the conditions for a mapping to exist ?

The answer is given by Mercer's conditions [Vapnik, 1995] :

Theorem 1 *Let $K(\mathbf{x}, \mathbf{y})$ be a continuous symmetric function in $L_2(C)$. Then, there exists a mapping Φ and an expansion*

$$K(\mathbf{x}, \mathbf{y}) = \sum_{i=1}^{\infty} \Phi(\mathbf{x})_i \cdot \Phi(\mathbf{y})_i$$

if and only if, for any $g \in L_2(C)$,

$$\int_{C \times C} K(\mathbf{x}, \mathbf{y}) g(\mathbf{x}) g(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0 \tag{13}$$

Note that for specific cases, it may not be easy to check whether Mercer's condition is satisfied, since eq (13) must hold for any $g \in L_2(C)$. However, it easy to prove that the condition is satisfied for polynomial kernel $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^p$

Let us take an example. Suppose our input data lie in $\mathbb{R}^2$ and we choose $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y})^2$, the following mapping is valid :

$$\Phi(\mathbf{x}) = \begin{pmatrix} x_1^2 \\ \sqrt{2} x_1 x_2 \\ x_2^2 \end{pmatrix}$$

In this case, the feature space is $\mathbb{R}^3$.

Once a kernel K satisfying the Mercer's condition has been chosen, the training algorithm consists of minimizing

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \tag{14}$$

and the decision function becomes

$$f(\mathbf{x}) = sgn \left(\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b \right) \tag{15}$$

The first kernels investigated for the pattern recognition problem were the following

- $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^p$
- $K(\mathbf{x}, \mathbf{y}) = e^{-||\mathbf{x}-\mathbf{y}||^2/2\sigma^2}$
- $K(\mathbf{x}, \mathbf{y}) = \tanh(a\mathbf{x} \cdot \mathbf{y} - b)$

The first one results in a classifier which has a polynomial decision function. The second one gives a Gaussian radial basis function classifier (RBF) and the last one gives a particular kind of two-layer sigmoidal network. For the RBF case, the number of centers (number of support vectors), the center themselves (the support vectors), the weights (α_i) and the threshold (b) are all produced automatically by the SVM training and give excellent results compared to classical RBF [Schölkopf et al., 1996]. In the same way, for the neural network case, the architecture (number of hidden units) is determined by SVM training.

Note, however, that the hyperbolic tangent kernel only satisfies Mercer's condition for certain values of the parameters a and b.

2.4 Generalization ability

We give here some theoretical results which describe the generalization ability of the Support Vector Machines.

2.4.1 Actual riusk, empirical risk and VC Dimension

We introduce here some classic definitions in learning theory (see [Vapnik, 1995]). Suppose we are given N observations $(\mathbf{x}_i, y_i)_{1 \leq i \leq N}$, $y_i \in \{-1, 1\}$ is the label of the example $\mathbf{x}_i$. It is assumed that there exists some unknown probability distribution $P(\mathbf{x}, y)$ from which these data are drawn.

Now suppose we have a machine whose task is to learn the mapping $\mathbf{x}_i \rightarrow y_i$. The machine is actually defined by a set of possible mappings $\mathbf{x} \rightarrow f(\mathbf{x}, \alpha)$, where the functions $f(\mathbf{x}, \alpha)$ themselves are labeled by the adjustables parameters α. A particular choice of α generates a "trained machine". Thus, for example, a neural network with a fixed architecture, with α corresponding to the weights and biases, is a learning machine in this sense.

The expectation of the test error for a trained machine is therefore :

$$R(\alpha) = \int \frac{1}{2}|y - f(\mathbf{x}, \alpha)| dP(\mathbf{x}, y)$$

The quantity $R(\alpha)$ is called the expected risk, or just the risk. Here we will call it the actual risk to emphasize it is the quantity we are ultimately interested in. The "empirical risk" $R_{emp}(\alpha)$ is defined to be just the measured mean error rate on the training set :

$$R_{emp}(\alpha) = \frac{1}{2N} \sum_{i=1}^{N} |y_i - f(\mathbf{x}_i, \alpha)|$$

The quantity $Q((\mathbf{x}_i, y_i), \alpha) = \frac{1}{2}|y_i - f(\mathbf{x}_i, \alpha)|$ is called the loss. For the case described here, it can only take the values 0 and 1.

Now choose some η such that $0 \leq \eta \leq 1$. Then, with probability at least $1 - \eta$, the following bounds hold [Vapnik, 1995] :

$$R(\alpha) \leq R_emp(\alpha) + \sqrt{\frac{h(\log(2l/h) + 1) - \log(\eta/4)}{l}}$$

where h is a non-negative integer called the Vapnik Chervonenkis (VC) dimension and is a measure of the capacity of the learning machine (see definition below). The second term on the right hand side of this inequation is called the "VC confidence". Note that this term is all the smaller as the VC dimension is small. Thus one way to control the generalization capacity of a learning machine is to control its VC dimension.

Let us define the VC dimension of a set of functions $\{f(\alpha)\}$. If a given set of N points can be labeled in all possible 2^N ways, and for each labeling, a member of the set $\{f(\alpha)\}$ can be found which correctly assigns those labels, we say that this set of points is *shatterd* by that set of functions. The VC dimension for the set of functions $\{f(\alpha)\}$ is defined as the maximum number of training points which can be shattered by $\{f(\alpha)\}$.

2.4.2 The VC Dimension of Support Vector Machines

First, we introduce a theorem giving a bound on the VC dimension of separating hyperplanes

Theorem 2 *Let $X \subset \mathbb{R}^n$ a set of vectors, $\forall \mathbf{x} \in X$, $||\mathbf{x}||_2 \leq R$. A subset S of hyperplanes such that $\forall (\mathbf{w}, b) \in S$,*

$$\inf_{\mathbf{x} \in X} |\mathbf{w} \cdot \mathbf{x} + b| = 1$$

$$|\mathbf{w}| \leq A$$

has its VC dimension bounded by

$$VC_{dim} \leq \min([R^2 A^2], n) + 1$$

Thus by minimizing $\mathbf{w}^2$, we minimize the bound on the VC dimension of separating hyperplane and a better generalization is expected.

Note that in the case of nonlinear SVM, this theorem needs to be applied in the feature space, and thus the capacity generalization is under control even if the feature space is of infinite dimension.

2.4.3 Leave-One-Out procedure

One way to predict the generalization performance of a SVM is to estimate the VC dimension by computing $R^2\mathbf{w}^2$ (see theorem 2). Another way is to use the *leave-one-out* estimator.

Given a sample of $N+1$ training examples, the leave-one-out procedure consists of the following steps : $\forall 1 \leq i \leq N+1$,

- Remove the example $\mathbf{x}_i$ from the training set
- Train the learning machine on this new training set in order to get parameters α_i.
- Test if $\mathbf{x}_i$ is correctly classified, i.e. if $Q((\mathbf{x}_i, y_i), \alpha_i) = 0$.

The number of errors made by the leave-one-out procedure is denoted by $\mathcal{L}_{N+1}$. By definition,

$$\mathcal{L}_{N+1} = \sum_{i=1}^{N+1} Q((\mathbf{x}_i, y_i), \alpha_i)$$

The quantity $\frac{\mathcal{L}_{N+1}}{N+1}$ is an estimation of the generalization error.

Indeed, the following theorem is valid [Vapnik, 1998]

Theorem 3 ([Luntz and Brailovsky, 1969]) *The leave-one-out estimator is unbiased; that is*

$$E\left(\frac{\mathcal{L}_{N+1}}{N+1}\right) = E(R_N)$$

The expectation on the left hand-side is taken on the training samples of size $N+1$ and $E(R_N)$ is the expectation of the actual risk (i.e. the expectation of the probability of error) for optimal hyperplanes constructed on the basis of training samples of size N.

Thus to control the generalization ability, one can try to minimize the number of errors made by the leave-one-out procedure.

2.4.4 Bounds on the leave-one-out estimator

This section presents bounds on the leave-one-out estimator for Support Vector Machines. The first one is for the separable case ($C = \infty$) and the second one is for the non separable case (C finite)

Theorem 4 *For optimal separating hyperplanes, the following inequality is valid*

$$\mathcal{L}_{N+1} \leq 4R^2\mathbf{w}^2$$

where $\mathcal{L}_{N+1}$ is the number of errors made by the leave-one-out procedure, R is the radius of the smallest sphere containing the support vectors and $\mathbf{w}$ the normal vector defining the optimal separating hyperplane

Proof : See appendix A

Theorem 5 *For optimal separating hyperplanes, under the condition* $C \geq \frac{1}{R^2}$ *the following inequality is valid*

$$\mathcal{L}_{N+1} \leq 4R^2 \sum_{i,\alpha_i<C} \alpha_i + Card\{i, \alpha_i = C\}$$

where $\mathcal{L}_{N+1}$ *is the number of errors made by the leave-one-out procedure,* R *is the radius of the smallest sphere containing the support vectors and* (α_i) *are given by the SVM training*

Proof : Similar to the proof of theorem 4

2.5 Reduced Support Vector Set

2.5.1 Introduction

We address the problem of reducing the set of support vectors, in order to increase the speed of a SVM during the test phase. Indeed, in spite of the fact that SVM yield very good performances compared to classical classifiers, the computational cost of this classifier can be very high if there are a lot of support vectors [Y. LeCun and Haffner, 1997]

After training a SVM, the result is the equation of an hyperplane which can be expressed by :

$$\mathbf{w} = \sum_{i=1}^{i=N} \alpha_i y_i \mathbf{x}_i \tag{16}$$

$\mathbf{x}_i$ are the support vectors and belong to the training set. N is the number of support vectors and the complexity of test is $O(N)$.

So, in order to increase the test speed, we have to reduce N. The general approach is to train the SVM and to replace eq (16) by :

$$\mathbf{w}' = \sum_{i=1}^{i=N'} \gamma_i z_i \tag{17}$$

with $N' << N$ and $||\mathbf{w} - \mathbf{w}'||$ as small as possible.

Two different approaches have been proposed. In the first one [Burges, 1996] N' is fixed and a gradient based algorithm compute the vectors z_i and the associated coefficients γ_i so that $||\mathbf{w} - \mathbf{w}'||$ is minimum. z_i are not called support vectors anymore as they do not belong to the training set.

In the second one [Osuna and Girosi, 1998] ϵ is fixed and a regression method using a SVM compute $\mathbf{w}'$ so that $||\mathbf{w} - \mathbf{w}'|| < \epsilon$. In this case, $\{z_i\}$ is a subset of the initial support vector set, and it can be proven that N' becomes smaller as ϵ increases.

2.5.2 Algorithm

Our approach comes from the following observation : denote by

$$\mathbf{w}(p, \gamma_i) = \sum_{\substack{i=1 \\ i \neq p}}^{i=N} \gamma_i \mathbf{x}_i$$

the hyperplane obtained by removing the support vector $\mathbf{x}_p$ and associating the coefficient γ_i to each support vector $\mathbf{x}_i$, for $i \neq p$. Then, we can compute analytically p and γ_i such that :

$$(p, \gamma_i) = \arg\min_{(p,\gamma_i)} ||\mathbf{w} - \mathbf{w}(p, \gamma_i)|| \tag{18}$$

In this way, after the training, we remove the support vector so that the distance between the original hyperplane and the new one is minimal. The processus is reiterated, removing the support

vectors one by one, until a stop condition is reached. This condition can either be on the number of points, as in the Reduced Support Vector Set of Burges [Burges, 1996] or be $||\mathbf{w} - \mathbf{w}'|| > \epsilon$ as in the regression method of Osuna [Osuna and Girosi, 1998]

Let us show how to compute p and γ_i in eq (18) in order to minimize $||\mathbf{w} - \mathbf{w}(p, \gamma_i)||$.
Let K be the matrix of dot products :

$$K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$$

First, let us consider the case where K is non invertible. It implies that one row of K is a linear combination of the others, i.e. there exists p and (β_i) such that :

$$\forall i, K(\mathbf{x}_i, \mathbf{x}_p) = \sum_{j \neq p} \beta_j K(\mathbf{x}_i, \mathbf{x}_j)$$

If we set,

$$\gamma_i = y_i \alpha_i + y_p \alpha_p \beta_i, \ \forall i \neq p$$

it comes

$$||\mathbf{w} - \mathbf{w}(p, \gamma_i)|| = 0$$

.

The support vector $\mathbf{x}_p$ can therefore be removed with any change to the equation of the separating hyperplane.

In the general case, let be (β_i) and (ϵ_i) such that :

$$\forall i, \sum_j \beta_j K(\mathbf{x}_i, \mathbf{x}_j) = \epsilon_i \tag{19}$$

Note that when K is non invertible, $\beta \in Ker(K)$ and $\epsilon = 0$ satisfy (19).

Let p be fixed such that $\beta_p > 0$. By setting,

$$\gamma_i = y_i \alpha_i - y_p \frac{\alpha_p}{\beta_p} \beta_i \tag{20}$$

we obtain :

$$||\mathbf{w} - \mathbf{w}(p, \gamma_i)||^2 = \frac{1}{\beta_p^2} \sum_i \beta_i \epsilon_i \tag{21}$$

Replacing (19) into (21),

$$||\mathbf{w} - \mathbf{w}(p, \gamma_i)||^2 = \frac{1}{\beta_p^2} \ {}^T \beta K \beta \tag{22}$$

We can prove that the minimum of eq (22) is reached when β is an eigenvector of K.
Since K is symmetric, let us write

$$K = {}^T A D A$$

with A orthonormal and D diagonal. The rows of A are eigenvectors of K
Consider i and p such that

$$\frac{D_i}{A_{ip}^2}$$

is minimum.

By taking $\beta_j = A_{ij}$, if we remove the support vector $\mathbf{x}_p$ and update the coefficients γ_i thanks to equation (20), the changes on the hyperplane will be minimal.

2.5.3 Experimental results

Table 3 compares the results of the regression method of Osuna and our incremental approach for different kind of experiments.

Note that for both methods the generalization error does not change significantly.

Database	# Examples	Kernel	C	#SV	regression		incremental	
					$\frac{\|\|\mathbf{w}-\mathbf{w}'\|\|}{\|\|\mathbf{w}\|\|}$	# SV	$\frac{\|\|\mathbf{w}-\mathbf{w}'\|\|}{\|\|\mathbf{w}\|\|}$	# SV
skin	1600	Poly 2	100	587	1.4×10^{-3}	11	0	6
skin	1600	Poly 5	1000	227	1.2×10^{-3}	11	8.0×10^{-5}	6
electrons	2000	Poly 2	100	611	6.1×10^{-3}	40	6.0×10^{-3}	37
electrons	2000	RBF	100	554	6.6×10^{-3}	297	6.6×10^{-3}	268
ripley	250	Linear	100	89	2.6×10^{-5}	3	1.0×10^{-4}	1
ripley	250	RBF	100	14	4.6×10^{-3}	12	4.2×10^{-3}	6

Figure 3: *Comparison between the results obtained using the SV regression and our incremental approach using various databases and kerneks*

This method provides an improvement, but requires a matrix diagonalisation per support vector removed. It could be avoided, if it were possible to compute directly the eigenvalues and eigenvectors of the matrix K at the step n thanks the ones of K at step $n-1$

3 SVM for image classification

Our goal is to evaluate the accuracy of a performant classifier such as Support Vector Machine on object recognition and image classification

3.1 Introduction

The general framework to measure the accuracy of a SVM on a given database is composed of the following stages :

- Preprocessing of the images in the database
- Separation of the database in training and test sets
- Choice of the representation of the input data
- Choice of the way of training, which includes :
  - Method of multi-class training
  - Value of the penalty term C
  - Choice of the kernel
- Training
- Test and evaluation of the performance

The choice of the representation of the input data and of the kernel are of the utmost importance and is discussed in section 3.4 and 3.6.

3.2 Multi class learning

Support Vector Machines are designed for binary classification. When dealing with several classes, as in object recognition and image classification, one needs an appropriate method. Different possibilities are :

- Modify the design of the SVM, as in [Weston and Watkins, 1998] in order to incorporate the multi-class learning directly in the quadratic solving
- Combine several binary classifiers :
  - "One against one" [Pontil and Verri, 1996]
  - "One against the others" [Blanz et al., 1996]

According to a comparison study [Weston and Watkins, 1998], the accuracy of these methods is almost the same, so that we choose the one with the lowest complexity, which is "one against the others".

In the "one against the others" algorithm, n hyperplanes are constructed, where n is the number of classes. Each hyperplane separates one class from the others classes. In this way, we get n decision functions $(f_k)_{1\leq k\leq n}$ of the form (10). The class of a new point x is given by $\arg\max_k f_k(x)$, i.e. the class with the largest output of the decision function.

3.3 Penalty term

A SVM requires to fix C in eq (12), the penalty term for misclassification. When training data which are not separable, this constant has to be chosen carefully. However, when dealing with images, most of time, the dimension of the input space is large (≥ 1000) compared to the size of the training set, so that the data are generally linearly separable. Consequently, the value of C do not matter and is fixed to an arbitrary large one.

In our differents experiments, unless mentioned, we did not come across training errors.

3.4 Choice of the input representation

3.4.1 Bitmap representation

The most simple way to represent an image is to consider its bitmap representation. Assuming the size of the images in database is fixed to h for the height and w for the width, then the input data for the SVM are vector of size $h \times w$ for grey-level images and $3 \times h \times w$ for color images. Each component of the vector is associated to a pixel in the image.

The main drawback of this representation is its lack of invariance with respect to translations. Nevertheless, in the case of object recognition and when the objects are centered as for the Coil database (see section 4.1), this representation can be efficient.

3.4.2 Basic luminance and color histograms

When trying to classify images, it is very difficult to have a representation which takes into account the intrasec features of a class rather than the specific features of its objects. In spite of the fact the color histogram technique is a very simple and low level method, it has shown good results in practice [Swain and Ballard, 1991]. As color is a discriminative component in image classification we keep this information, contrary to section 4.1 where the luminance component provides enough information to recognize objects. A color is represented by a 3 dimensional vector corresponding to the position of the color in the HSV space.

The HSV space (Hue-Saturation-Value) is in bijection with the classic RGB space, but is more suitable since it is less sensitive to illumination changes.

Thus, the color histogram is a 3 dimensional histogram. The number of bins per color component has been fixed to 16, and thus the dimension of this multidimensional histogram is $16^3 = 4096$. Some experiments with a smaller number of bins have been carried out, but the best results have been reached with 16 bins. Besides, increasing the number of bin yields computational issues. All those reasons lead us to consider 3 dimensional histograms with 16 bins per color component.

3.4.3 Improved histograms

As often in image processing, it might be interesting to consider not only the image itself, but also its derivatives. Thus, a first improvement is to compute the horizontal and vertical derivatives of the image and to construct for each one a similar histogram as the one used for the pixel values. In this way, the input data for the SVM is not one histogram anymore, but three.

The problem generally encountered with histograms of derivatives is their lack of robustness, specially when images are noisy. To bypass this difficulty, images can be smoothed, with Gaussian filters, for example.

We present here a new kind of histogram, called transition histogram to bypass this difficulty. As the histogram of derivative, it is meant to count the jumps in the signal.

Definition 1 *The transition histogram of the discrete signal* $(x_1, \cdots, x_N)$ *is an histogram* H *such that:*

$$H_p = Card\{i \in [1..N-1],\ p \in [x_i, x_{i+1}[\}$$

As shown in figure 4, the bin p in transition histogram counts the number of times the signal pass through the value p.

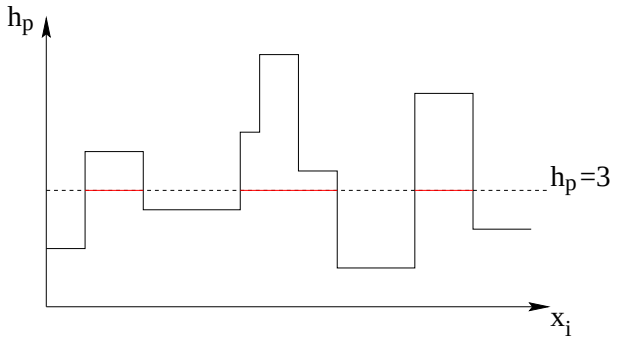

Figure 4: *Transition histogram*

It is quite straight forward to implement. Here is the core of the histogram computation :

```
for (i=0; i<N-1; i++)
  for (p=x[i]; p<x[i+1]; p++)
     h[p]++;
```

Note that only the positive transitions contribute to the histogram ($x_{i+1} > x_i$). It is also possible to take into consideration the negative transitions, but the histograms of positive and negative transitions are almost the same. Indeed, let H^+ and H^- be these histograms. Then it is obvious to see that : $\forall p, |H_p^+ - H_p^-| <= 1$. Thus, to avoid redundancy, only positive transitions are taken into account.

The characteristics of the transition histogram are the following ones :

- Translation invariant : this characteristic is common to all the histograms, but is essential in image classification.

- Scale invariant : consider the two signals of figure 5. On the left side the original signal and on the right side after subsampling and antialising. In both cases, the transition histograms are identical. Note that the derivative histogram is not scale invariant since the value of the derivative is proportional to the scale.

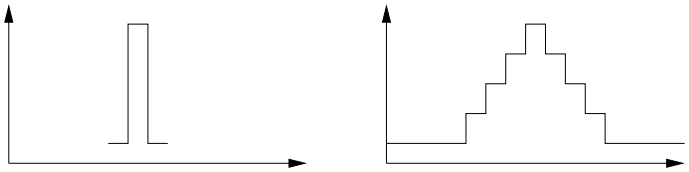

Figure 5: *The transition histogram is scale invariant : left hand- side the original signal and right-hand side the same after subsampling and antialiasing. In both cases, the transition histograms are identical*

The easiest way to cope with a n-dimensional signal, is to compute n transition histogram, one for each axe. For images, this leads to a horizontal transition histogram and a vertical one. To be rotation invariant, the two histograms are added. Strictly speaking, to have a real invariance to rotation, we should better compute the transition histogram of the norm of the gradient.

3.4.4 Multiscale analysis

The transition histogram takes only into account the local transitions in the image. Indeed, it considers only pixels which are side by side. It can be interesting to investigate into transitions between regions of an image. This can be achieved by subsampling the image at different scales and computing the transition histograms for each subsampled image.

3.5 Dimension reduction : PCA

One of the main issue in image learning is the curse of dimensionality. Indeed, if the number of training examples is not big enough in front of the number of dimensions of the input data, the learning machine tends to learn by heart and has difficulty to generalize.

For this reason, a dimension reduction if often needed. However, SVM are known to have very good performances even if the number of dimensions is very high. For non linear SVM, the number of dimension of the hidden space can even be infinite. This good performance in high dimensional space comes from the margin maximization which keep under control the generalization capacity.

We tried all the same to perform a dimensionality reduction. For this purpose, we used the easiest technic of dimensionality reduction, which is Principle Component Analysis (PCA). This "black box" has been implemented in a straight forward way on the various input data : bitmap images and color histograms.

The results are presented in sec 5.1.3

3.6 Choice of the kernel

Except the constance C, the kernel is the only adjustable parameter of a SVM. Nevertheless, it has to be chosen carefully since an inappropriate kernel can lead to poor performances.

The differents kernels we tried have been largely used in SVM literature and have been presented in section 2.3

- Linear

- Polynomial
- Radial Basis Function (RBF)
- Neural Network

When our input data are bitmap images, the RBF kernel

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{||\mathbf{x}-\mathbf{y}||_2^2}{2\sigma^2}}$$

provides good performance (see section 5.1). This result led us to consider more general kernels of the form

$$K(\mathbf{x}, \mathbf{y}) = e^{-\frac{d(\mathbf{x},\mathbf{y})}{\sigma^2}} \text{ with } d(\mathbf{x}, \mathbf{y}) = 0 \Longleftrightarrow \mathbf{x} = \mathbf{y} \text{ and } d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$$

$d(\mathbf{x}, \mathbf{y})$ can be chosen to be a relevant distance in the input space. In the case of images as input, the L_2 norm seems to be quite meaningful and that is the reason why RBF kernels provide good results. But for histogram classification, more suitable comparison functions exist, especially the χ^2 function [Schiele and Crowley, 1996].

- $\chi^2 : d(\mathbf{x}, \mathbf{y}) = \sum_i \frac{(x_i - y_i)^2}{x_i + y_i}$
- rotalink : $d(\mathbf{x}, \mathbf{y}) = \sum_i (\sqrt{x_i} - \sqrt{y_i})^2$
- $L_1 : d(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$

In the theory of SVM, the kernel K must satisfy the Mercer conditions (13) It has been proven that the kernel with the L_1 norm satisfy these conditions [Vapnik, 1998]. For the χ^2 and rotalink-based kernels, this theoretical question remains unsolved since checking the Mercer conditions is a difficult problem in general. Nevertheless, in practice, these kernels give excellent results (see section 5).

4 Databases used for experiments

All along these experiments, 2 databases have been used. The Coil database is meant for assessing 3D-object recognition accuracy, while the Corel database is dedicated to image classification testing.

4.1 The COIL database

The Coil (Columbia Object Image Library) [Murase and Nayar, 1995] been used by several authors, including [Schmid and Mohr, 1996, Schiele and Crowley, 1996, Rao and Ballard, 1995] for object recognition tasks. This database consists of 100 differents objects, seen from 72 different view angles, separated by 5 degrees rotations. Therefore, the database contains 7200 color images. Appendix B shows a selection of the objects in the database, as well as the same object under different points of view.

Apparently, images have been resampled so that the larger of the two dimensions fits the image size. Consequently, the apparent size of an object may change considerably between two images (see figure 20 for example).

The preprocessing consists of the following stage :

- Grey-level conversion
- Subsampling

In object recognition, the luminance provides enough information to be able to distinguish two different objects. Thus, the first stage of our preprocessing is a grey-level conversion. The formula used to get the luminance of a (R,G,B) pixel is :

$$L = 0.31R + 0.59G + 0.10B$$

The original resolution of the images is 128 x 128 pixels. Since a such accuracy is not needed, the images are subsampled, hence a gain in training and classification speed. The resolution is reduce to 32 x 32 pixels by averaging 4 x 4 pixels patches.

The database is divided between training set and test is divided in the following way : the training set is made up so that, for each object, the angle between images belonging to this set is $\Delta\alpha$. Typically, $\Delta\alpha = 20^o$, which means that one image over four is in training test and the remaining form the test set. Experiments have been yield with $\Delta\alpha = 10^o, 20^o, 40^o, 90^o$

4.2 The COREL database

The Corel database is meant for image classification. It consists of a set of photos divided in about 200 categories, each with 100 images. As shown in appendix C, categories and images are very diversified.

Two series of experiments have been conducted with this database. In the first one, we kept the categories imposed by the Corel labeling For the sake of the comparison, we chose the same subset of categories as [Carson et al., 1998], which are : Air Shows, Bears, Elephants, Tigers, Arabian Horses, Polar Bears, African Specialty Animals, Cheetahs-Leopards- Jaguars, Bald Eagles, Mountains, Fields, Deserts, Sunrises-Sunsets, Night Scenes. However, in [Carson et al., 1998], some images which are visually quite inconsistent with the rest of their category were removed.

Each of these 14 categories is made of 100 images, hence a database of 1400 images. We divided each category in training and test sets, containing 2/3 and 1/3 of the images respectively.

In another serie of experiments, we designed our own categories. Figure 6 displays the label of these categories and the number of images for each one.

Category	Nb of images
Airplanes	386
Birds	501
Boats	200
Buildings	625
Fish	300
People	358
Vehicle	300

Figure 6: *Hand-labeled categories used with the COREL database*

In this labeling, every category is a regroupement of original categories. For example, airplanes category includes images of air-shows, aviation-photography, fighter-jets and ww-II-planes. Thus, these categories are more general than the ones provided by corel and therefore more difficult to classify.

5 Experimental results

5.1 Results for the Coil database

5.1.1 Influence of the kernel with bitmap representation

As already mentioned, one image over p is in the training set and the remaining ones are in the test set. Thus, for each object, the training set contains $72/p$ images, with an angle of $\Delta\alpha = p \times 5^o$. To assess the quality of bitmap representation, we vary $\Delta\alpha$ between 10^o (1/2 image in the training set) and 90^o (1/9 image in the training set). We also tried the the following kernel :

- Linear
- Polynomial degree 2
- Polynomial degree 5
- RBF

Figure 7 shows the rate of misclassification on the test set for $\Delta\alpha = 10^o, 20^o, 40^o, 90^o$ and different kernels. Note that the training set has always been separated without error.

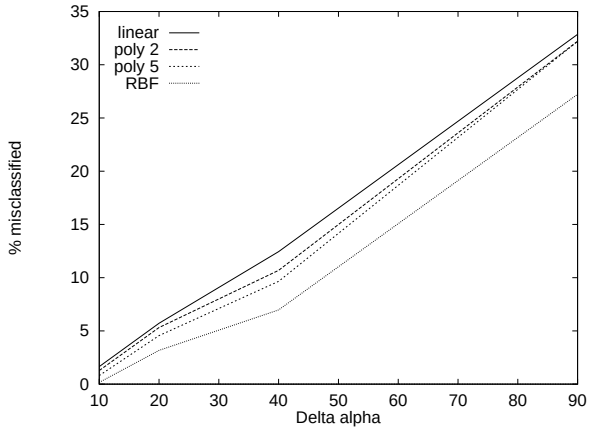

Figure 7: *Accuracy of the recognition in function of $\Delta\alpha$ and for different kernels*

The result of this experiment points out the good performance of RBF Kernels. But, it is well known that this performance highly relies on the choice of the parameter σ. Figure 8 shows the performance of the SVM using a RBF kernel versus σ on the first 32 objects of the base.

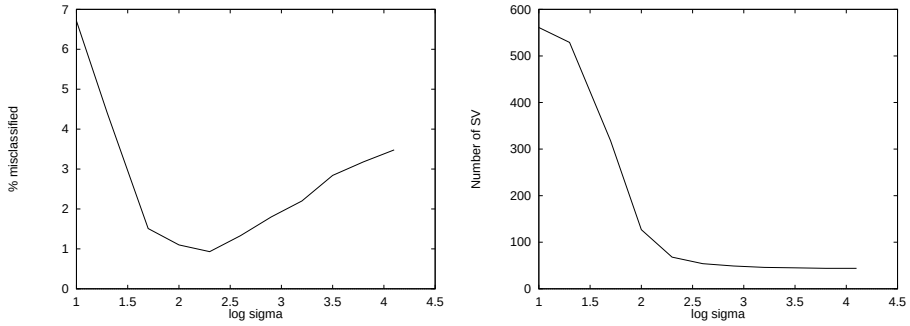

Figure 8: *Influence of the parameter σ in a RBF kernel on the accuracy of the recognition and on the number of SV*

It is interesting to point out the behavior of a SVM with a RBF kernel when $\sigma \to 0$ and when $\sigma \to +\infty$

- When $\sigma \to 0$, $K(\mathbf{x}, \mathbf{y}) = 1$ if $\mathbf{x} = \mathbf{y}$ and $K(\mathbf{x}, \mathbf{y}) = 0$ otherwise

  The functional (14) to be minimized is

  $$W(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2}\alpha_i^2$$

  and the solution is $\forall i, \alpha_i = 1$. In other words, all the training examples are support vectors. This means that the SVM learns by heart but after is unable to generalize. This is emphased by the right hand side of figure 8 where the number of training examples is 576.

- When $\sigma \to +\infty$, $K(\mathbf{x}, \mathbf{y}) \sim 1 - \frac{||\mathbf{x}-\mathbf{y}||_2^2}{2\sigma^2}$ and the decision function becomes

  $$\begin{aligned} f(\mathbf{x}) &= \sum_i \alpha_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \\ &\sim \sum_i \alpha_i y_i \left(1 - \frac{||\mathbf{x} - \mathbf{x}_i||_2^2}{21\sigma^2}\right) + b \\ &= \mathbf{x} \cdot \frac{\sum_i \alpha_i y_i \mathbf{x}_i}{\sigma^2} + \left(b - \frac{\sum_i \alpha_i y_i ||\mathbf{x}_i||^2}{2\sigma^2}\right) \end{aligned}$$

  since $\sum_i \alpha_i y_i = 0$

  Thus the decision function is an hyperplane. This means that when $\sigma \to +\infty$, the RBF kernel is equivalent to the linear kernel (see table 9 for illustration)

	$\sigma = 1.10^4$	$\sigma = 5.10^4$	$\sigma = 1.10^6$	Linear
% misclassified	3.48	3.65	3.82	3.82
Nb of SV	44	44	44	44

Figure 9: *Comparison between the performance of a linear kernel and of a RBF kernel when $\sigma \to +\infty$*

5.1.2 Influence of the data representation

The following data representations are compared :

- Bitmap
- Luminance histogram
- Luminance + Derivative histogram, ie histogram built on the derivative of the image
- Luminance + Transition histogram

We fix a polynomial kernel of degree 2. Figure 10 shows the results of those experiments which have been lead with only the first 32 objects of the database. The number of bins in the histograms is 64.

	Bitmap	Luminance histo	Lum + Derivative histo	Lum + Transition histo
Poly 2	3.24	9.55	1.89	0.89
RBF	1.1	6	1.22	0.6
χ^2	1.39	1.56	0.4	0.17

Figure 10: *Misclassification rate for various input data representations*

Three main conclusions can be drawn of these experiments :

- As often in image processing, the derivative provides more informations than the luminance itself
- The transition histogram offers excellent results and is more reliable than the histogram of the derivative.
- The χ^2 based kernel yields an essential improvement compared to RBF kernel when the input data are histograms (but not in the case of bitmap input). Thus the main conclusion is that *the χ^2-based kernel is very suitable for histogram classification*

As mentioned above, the number of bins in the histograms has been fixed to 64. After, some experiments have been carried out to study the influence of the number of bin on the accuracy of the recognition. As shown in figure 11, it turned out that 64 bins provides good results.

	256	128	64	32
Derivative	3.48	1.62	1.89	1.86
Transition	1.89	0.99	0.89	1.22

Figure 11: *Misclassification rate for different number of bins in the histograms*

For the sake of the comparison, several authors performed object recognition on the first 20 objects of coil database [Schmid and Mohr, 1996, Rao and Ballard, 1995, Schiele and Crowley, 1996, Murase and Nayar, 1995]. They use $\Delta\alpha = 10$ and most of them achieve 100% correct recognition. We also achieve perfect recognition on the first 20 objects using a χ^2-based kernel on luminance and transition histogram. For the whole database (100 objects), the recognition rate is 99.9%. This puts the SVM at the level of the state of the art.

5.1.3 PCA

The dimension of the input data is very large : 1024 for the bitmap images of the coil database and 4096 for each color histogram built with the Corel database. As mentioned in section 3.5, it seems interesting to see how a dimensionality reduction affects the classification accuracy.

For this purpose, a Principal Component Analysis has been the first step of the following experiments :

- Coil database, bitmap representation, polynomial kernel of degree 2
- Coil database, bitmap representation, linear kernel
- Corel database, transition histogram representation, polynomial kernel of degree 2

The 4 plots in figure 12 shows the accuracy in function of the number of components held.

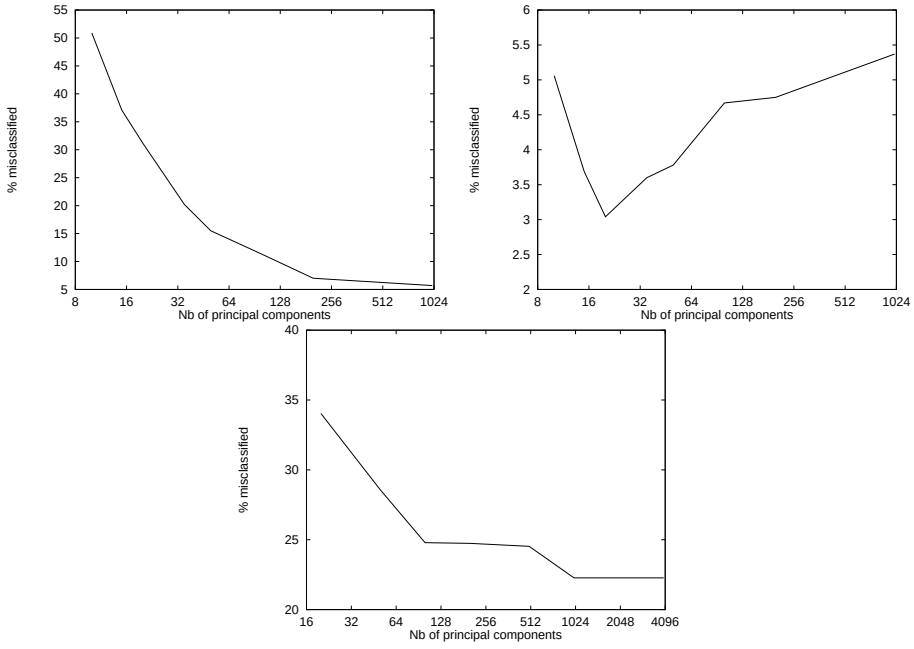

Figure 12: *Accuracy in function of the number of components. Top : Coil database and bitmap representation, left : linear kernel, right : polynomial kernel of degree 2. Bottom : Corel database with transition histogram and polynomial kernel of degree 2*

The only improvement is in the case of the Coil database with a polynomial kernel. These results are very difficult to interprete and a theoretical study on the influence of a PCA should be carried out.

5.2 Results for the Corel database

5.2.1 Influence of the data representation

Since the bitmap representation is not invariant regarding to translations and thus meaningless for image classification, we only compare the following histograms :

- Color histogram
- Transition histogram

- Color+Transition histograms
- Color+Transition histogram with subsampling (/2 and /4)

Each histogram has 16 bins per axis, hence a total of $16 \times 16 \times 16 = 4096$ bins. The number of components of the input data for each experiment is therefore 4096, 4096, 8192, 16384 respectively. The kernel is polynomial of degree 2 and the database is the Corel one with 14 classes.

Color	34.46
Transition	23.74
Color + Transition	21.43
Color + Transition + Subsampling	20.17

Figure 13: *Misclassification rate on the Corel database for various kind of histograms*

As for the Coil database, the transition histogram improves the performance compared to the color histogram.

5.2.2 Influence of the kernel

The same kernels as those used with the Coil database are compared for various kind of histograms. As already mentioned for the coil database, it turns out that χ^2-based kernel outperforms the RBF kernel (see figure 14).

	Linear	Poly 2	RBF	χ^2	KNN L_2	KNN χ^2
Color	36.4	34.5	29.2	14.7	47.7	26.5
Transition	26.3	23.7	21	13.2	38	26.5
Color + Transition	25.8	21.4	22.5	13.2	27	25.5
Color + Transition + Subsampling	23.5	20.2	20.4	12.8	25.1	25.9

Figure 14: *Misclassification rate for different kernels. The last two columns show the performance of the KNN algorithm with* $K = 1$

Figure 15 presents the class-confusion matrix corresponding to the result of the experiment involving the χ^2-based kernel on color and transitions histograms.

To assess the very good accuracy of Support Vector Machines, we conducted some experiments of image classification with a K Nearest Neighbors (KNN) algorithm. Distances chi^2 and L_2 have been tried. It turned out that the value $K = 1$ yields the best results. The last two columns of table 14 presents the results. As expected, the χ^2 distance is more suitable, but a χ^2*-based SVM is about twice better than the nearest neighbor algorithm*

Note that the same database has been used by [Carson et al., 1998] with a decision tree classifier and the accuracy was only about 50%

In order to be sure that the excellent performance of the χ^2 compared to a RBF kernel is not due to an inadequate choice of the constant σ^2 in the RBF kernel, we tried different values for σ^2 as shown in figure 16. This validates the superiority of χ^2 kernel compared to RBF one.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14
air-shows	1	31		1							1	1			
bears	2		26	2	2		2	1		1					
elephants	3		1	27				3						3	
tigers	4			1	32			1							
horses	5					34									
polar-bears	6						30					1		2	1
african-animals	7			1	1			30		1				1	
cheetahs	8						1		32			1			
eagles	9	1								33					
mountains	10	3								1	24	3	3		
fields	11			1				1			2	27	3		
deserts	12						2	1	1	2	1	3	24		
sunsets	13													34	
night scenes	14	1												2	31

Figure 15: *Class-confusion matrix. Entries are numbers. For example, row (1) indicates that on the 34 images of the air-shows category, 31 have been correctly classified and 3 have been classified in elephants, mountains and fields*

	RBF				χ^2
σ^2	5000	10000	20000	100000	
% misclassified	30.9	29.2	29.6	30.5	14.7

Figure 16: *Comparison of a RBF kernel with various values of σ^2 with a χ^2-based kernel*

The encouraging results obtained with the χ^2-based kernel told us into investigating with similar kernels. We compared the three following kernels of the form $K(\mathbf{x}, \mathbf{y}) = e^{-\frac{d(\mathbf{x},\mathbf{y})}{\sigma^2}}$ on colors histograms :

- χ^2 : $d(\mathbf{x}, \mathbf{y}) = \sum_i \frac{(x_i - y_i)^2}{x_i + y_i}$
- rotalink : $d(\mathbf{x}, \mathbf{y}) = \sum_i (\sqrt{x_i} - \sqrt{y_i})^2$
- L_1 : $d(\mathbf{x}, \mathbf{y}) = \sum_i |x_i - y_i|$

Results are summarized in table 17

χ^2	rotalink	L_1
14.7	14.7	14.9

Figure 17: *Misclassification rate on color histograms for kernels involving "linear" distances*

The results for these 3 distances are very similar and the common point between them is the linearity in respect of each component. Thus it seems that distances having this linearity characteristic are very suitable for histogram classifications. Further theoretical and experimental work on this kind of kernels need to be carried out.

To finish, figure 18 presents some experimental results obtained with 7 hand-labeled categories (see section 4.2 and 6).

	Linear	Poly 2	χ^2	KNN L_2	KNN χ^2
Color	42.7	38.9	21.6	51.4	35.4
Transition	36.8	30	20.1	42.4	31.4
Color + Transition	36.7	29.1	18.7	42.9	29.4

Figure 18: *Results obtained with the 7 hand-labeled categories of Corel*

The results are comparable to those with the first labeling (see figure 14), except that the accuracy is not so good. This is due to the fact that the labeling is more difficult

6 Conclusion

In this work, we have demonstrated the potential of Support Vector Machines in the problems of object recognition and image classification. It appears that unlike most learning techniques, SVM can be trained even if the number of examples is much lower that the dimensionality of the input space. We also pointed out the need to investigate into kernels which are well-suited for the data representation. For histograms classification, it turned out that a χ^2-based kernel provided excellent results. Thus this result can be extended to other problems and provides a general technique for histogram and density classification. Nevertheless, the image classification problem is still open since a color histogram may not provide enough information to obtain an efficient classifier.

Acknowledgments

I would to thank everybody working in team with me since they enabled me to have a very pleasant way of working and helped me at any time, especially Patrick Haffner and Vladimir Vapnik, my mentors, and also Pascal, Sayan and Dongwang.

References

[Bazaraa and Shetty, 1979] Bazaraa, M. and Shetty, C. (1979). *Nonlinear programming*. John Wiley, New York.

[Blanz et al., 1996] Blanz, V., Schölkopf, B., Bülthoff, H., Burges, C., Vapnik, V., and Vetter, T. (1996). Comparison of view-based object recognition algorithms using realistic 3d models. In *Artificial Neural Networks - ICANN'96*, pages 251–256, Berlin.

[Boser et al., 1992] Boser, B., Guyon, I., and Vapnik, V. (1992). A training algorithm for optimal margin classifier. In *Proc. 5th ACM Workshop on Computational Learning Theory*, pages 144–152, Pittsburgh, PA.

[Burges, 1996] Burges, C. (1996). Simplified support vector decision rules.

[Carson et al., 1998] Carson, C., Belongie, S., Greenspan, H., and Mlik, J. (1998). Color- and texture-based images segmentation using em and its application to image querying and classification. In *Submitted to IEEE*.

[Cortes and Vapnik, 1995] Cortes, C. and Vapnik, V. (1995). Support vector network. *Machine learning*, 20:1–25.

[Luntz and Brailovsky, 1969] Luntz, A. and Brailovsky, V. (1969). On estimation of characters obtained in statistical procedure of recognition. *Technicheskaya Kibernetica*, 3.

[Murase and Nayar, 1995] Murase, H. and Nayar, S. K. (1995). Visual learning of 3d objects from appearence. *International Journal of Computer Vision*, 14:5–24.

[Osuna and Girosi, 1998] Osuna, E. and Girosi, F. (1998). Reducing the run-time complexity of support vector machines. In *To appear in ICPR'98*, Brisbane, Australia.

[Pontil and Verri, 1996] Pontil, M. and Verri, A. (1996). Support vector machines for 3-d object recognition. unpublished.

[Rao and Ballard, 1995] Rao, R. P. N. and Ballard, D. (1995). Object indexing using an iconic sparse distribued memory. In *ICCV'95 Fifth international conference on Computer Vision*, pages 24–31.

[Schiele and Crowley, 1996] Schiele, B. and Crowley, J. (1996). Object recognition using multidimensional receptive field histograms. In *ECCV'96, Fourth European Conference on Computer Vision, Volume I*, pages 610–619.

[Schmid and Mohr, 1996] Schmid, C. and Mohr, R. (1996). Combining greyvalue invariants with local constraints for object recognition. In *International Conference on Computer Vision and Pattern Recognition*.

[Schölkopf et al., 1996] Schölkopf, B., Sung, K., Burges, C., Girosi, F., Niyogi, P., Poggio, T., and Vapnik, V. (1996). Comparing support vector machines with gaussian kernels to radial basis function classifiers. A.I. Memo No. 1599, Massachusetts Institute of Technology.

[Swain and Ballard, 1991] Swain, M. and Ballard, D. (1991). Indexing via color histograms. *Intern. Journal of Computer Vision*, 7:11–32.

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer, New York.

[Vapnik, 1998] Vapnik, V. (1998). Statistical learning theory. to appear.

[Weston and Watkins, 1998] Weston, J. and Watkins, C. (1998). Multi-class support vector machines. Technical Report CSD-TR-98-04, Royal Holloway, University of London.

[Y. LeCun and Haffner, 1997] Y. LeCun, L. Bottou, Y. B. and Haffner, P. (1997). Gradient-based learning applied to document recognition. In *Submitted to proceedings of the IEEE*.

A Proof of theorem 4

Theorem 6 *For optimal separating hyperplanes, the following inequality is valid*

$$\mathcal{L}_{N+1} \leq D^2\mathbf{w}^2 = 4R^2\mathbf{w}^2$$

where $\mathcal{L}_{N+1}$ is the number of errors made by the leave-one-out procedure, D and R are the diameter and the radius of the smallest sphere containing the support vectors and $\mathbf{w}$ the normal vector defining the optimal separating hyperplane

Proof : Suppose we are given the training set

$$(x_1, y_1), \ldots, (x_{N+1}, y_{N+1})$$

Let $\boldsymbol{\alpha}^0 = (\alpha_1, \ldots, \alpha_{N+1})$ be the solution of the optimization problem : maximize

$$W(\boldsymbol{\alpha}) = \sum_{i=1}^{N+1} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N+1} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$

subject to $\alpha_i \geq 0$ and $\sum_i y_i \alpha_i = 0$.

By enumerating the support vectors with $i = 1, \ldots, a$ the optimal separating hyperplane is defined by

$$\mathbf{w}_0 = \sum_{i=1}^{a} \alpha_i^0 y_i \mathbf{x}_i$$

and b_0 such that

$$y_i(\mathbf{w}_0 \cdot \mathbf{x}_i + b_0) = 1 \tag{23}$$

We are trying to bound the number of errors of the leave-one-out estimator. Consider we remove the support vector $\mathbf{x}_p$. Let us denote by $\boldsymbol{\alpha}^p$ the vector providing the maximum for the function $W(\boldsymbol{\alpha})$ under constraints

$$\begin{aligned} \alpha_p &= 0 \\ \alpha_i &\geq 0, \ i \neq p \\ \sum_i y_i \alpha_i &= 0 \end{aligned} \tag{24}$$

Let the vector

$$\mathbf{w}_p = \sum_i \alpha_i^p y_i \mathbf{x}_i$$

define the corresponding hyperplane. $\mathbf{w}_p$ is the optimal separating hyperplane once the support vector $\mathbf{x}_p$ has been removed. We suppose that the leave-one-out procedure makes an error on the vector $\mathbf{x}_p$, i.e.

$$y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p) < 0 \tag{25}$$

Now, let us make the intermediate following computation. Let $\boldsymbol{\beta}$ be a vector such that $\sum_{i=1}^{a} y_i \beta_i = 0$. Then,

$$\begin{aligned} W(\boldsymbol{\alpha}^0 - \boldsymbol{\beta}) &= \sum_i \alpha_i^0 - \beta_i - \frac{1}{2} \sum_{i,j} (\alpha_i^0 - \beta_i)(\alpha_j^0 - \beta_j) y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ &= W(\boldsymbol{\alpha}^0) - \sum_i \beta_i + \sum_{i,j} \alpha_i^0 \beta_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \frac{1}{2} \sum_{i,j} \beta_i \beta_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ &= W(\boldsymbol{\alpha}^0) - \sum_i \beta_i (1 - y_i \mathbf{w}_0 \cdot \mathbf{x}_i) - \frac{1}{2} \sum_{i,j} \beta_i \beta_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \end{aligned}$$

Taking into account (23) and $\sum_{i=1}^{a} y_i\beta_i = 0$, we have

$$W(\boldsymbol{\alpha}^0 - \boldsymbol{\beta}) = W(\boldsymbol{\alpha}^0) - \frac{1}{2}\sum_{i,j}\beta_i\beta_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{26}$$

Let us denote

$$\alpha^+ = \sum_{y_i = y_p} \alpha_i^0$$

$$\alpha^- = \sum_{y_i \neq y_p} \alpha_i^0$$

From $\sum_i y_i\alpha_i^0 = 0$, we have $\alpha^+ = \alpha^-$. Now consider the following vector $\boldsymbol{\beta}$:

$$\beta_p = \alpha_p^0$$

$$\beta_i = \frac{\alpha_i^0\alpha_p^0}{\alpha^+},\ y_i \neq y_p$$

$$\beta_i = 0,\ y_i = y_p,\ i \neq p$$

Since $\alpha^+ = \alpha^-$, $\sum_{i=1}^{a} y_i\beta_i = 0$. Moreover $\forall i, \alpha_i^0 - \beta_i \geq 0$ and $\alpha_p^0 - \beta_p = 0$. Thus, the vector $\boldsymbol{\alpha}^0 - \boldsymbol{\beta}$ satisfies the system of constraints (24). Since the maximum of the function $W(\boldsymbol{\alpha})$ under those constraints is obtained for $\boldsymbol{\alpha}^p$, the following inequality is true

$$W(\boldsymbol{\alpha}^p) \geq W(\boldsymbol{\alpha}^0 - \boldsymbol{\beta}) \tag{27}$$

Combining eq (26) and eq (27),

$$\frac{1}{2}(\sum_i \beta_i y_i \mathbf{x}_i)^2 \geq W(\boldsymbol{\alpha}^0) - W(\boldsymbol{\alpha}^p) \tag{28}$$

Now let us try to maximize the right hand side of inequality (28). Consider we have the vector $\boldsymbol{\alpha}^p$ and we make one step in the maximization of $W(\boldsymbol{\alpha})$ by computing $W(\boldsymbol{\alpha}+\boldsymbol{\gamma})$ with $\sum_i y_i\gamma_i = 0$, $\gamma_i \geq 0$ and $\gamma_p > 0$ We have

$$\begin{aligned}
W(\boldsymbol{\alpha}^p + \boldsymbol{\gamma}) &= \sum_i \alpha_i^p + \gamma_i - \frac{1}{2}\sum_{i,j}(\alpha_i^p + \gamma_i)(\alpha_j^p + \gamma_j) y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\
&= W(\boldsymbol{\alpha}^p) + \sum_i \gamma_i - \sum_{i,j}\alpha_i^p\gamma_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - \frac{1}{2}\sum_{i,j}\gamma_i\gamma_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\
&= W(\boldsymbol{\alpha}^p) + \sum_{i \neq p}\gamma_i(1 - y_i \mathbf{w}_p \cdot \mathbf{x}_i) + \gamma_p(1 - y_p \mathbf{w}_p \cdot \mathbf{x}_p) - \frac{1}{2}\sum_{i,j}\gamma_i\gamma_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j
\end{aligned}$$

Taking into account that

$$\forall i \neq p,\ y_i(\mathbf{w}_p \cdot \mathbf{x}_i + b_p) = 1$$

we obtain finally

$$W(\boldsymbol{\alpha}^p + \boldsymbol{\gamma}) = W(\boldsymbol{\alpha}^p) + \gamma_p(1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)) - \frac{1}{2}\sum_{i,j}\gamma_i\gamma_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{29}$$

Now consider the following vector $\boldsymbol{\gamma}$:

$$\gamma_p > 0$$

$$\gamma_i = \gamma_p\gamma_i',\ y_i \neq y_p \ \text{ for a given } \gamma_i' \geq 0 \text{ such that } \sum \gamma_i' = 1$$

$$\gamma_i = 0,\ y_i = y_p,\ i \neq p$$

For this $\boldsymbol{\gamma}$, eq (29) becomes

$$W(\boldsymbol{\alpha}^p + \boldsymbol{\gamma}) = W(\boldsymbol{\alpha}^p) + \gamma_p(1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)) - \frac{1}{2}\gamma_p^2(y_p\mathbf{x}_p + \sum_{y_i \neq y_p} \gamma_i' y_i \mathbf{x})^2 \tag{30}$$

The best value for γ_p is

$$\gamma_p = \frac{1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p)}{(y_p\mathbf{x}_p + \sum_{y_i \neq y_p} \gamma_i' y_i \mathbf{x})^2}$$

Increment of the function $W(\boldsymbol{\alpha})$ at this step equals

$$\Delta W_p = W(\boldsymbol{\alpha}^p + \boldsymbol{\gamma}) - W(\boldsymbol{\alpha}^p) = \frac{1}{2}\frac{(1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p))^2}{(y_p\mathbf{x}_p + \sum_{y_i \neq y_p} \gamma_i' y_i \mathbf{x})^2} \tag{31}$$

ΔW_p does not exceed the increment of the function $W(\boldsymbol{\alpha})$ for complete maximization :

$$W(\boldsymbol{\alpha}^0) - W(\boldsymbol{\alpha}^p) \geq \Delta W_p \tag{32}$$

Combining (32), (31) and (28), we obtain

$$\frac{1}{2}(\sum_i \beta_i y_i \mathbf{x}_i)^2 \geq \frac{1}{2}\frac{(1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p))^2}{(\mathbf{x}_p - \sum_{y_i \neq y_p} \gamma_i' \mathbf{x})^2}$$

Taking the value of β_i into account this inequality becomes

$$\frac{1}{2}(\alpha_p^0)^2(\mathbf{x}_p - \sum_{y_i \neq y_p} \frac{\alpha_i^0}{\alpha^+}\mathbf{x}_i)^2 \geq \frac{1}{2}\frac{(1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p))^2}{(\mathbf{x}_p - \sum_{y_i \neq y_p} \gamma_i' \mathbf{x})^2} \tag{33}$$

Since $\sum_{y_i \neq y_p} \gamma_i' = \sum_{y_i \neq y_p} \frac{\alpha_i^0}{\alpha^+} = 1$, we have

$$(\mathbf{x}_p - \sum_{y_i \neq y_p} \frac{\alpha_i^0}{\alpha^+}\mathbf{x}_i)^2 \leq D^2$$

$$(\mathbf{x}_p - \sum_{y_i \neq y_p} \gamma_i' \mathbf{x})^2 \leq D^2$$

Taking into account these inequalities and eq (25), inequality (33) gives

$$\alpha_p^0 D^2 \geq 1 - y_p(\mathbf{w}_p \cdot \mathbf{x}_p + b_p) \geq 1 \tag{34}$$

Thus if the optimal hyperplane makes an error classifying vector $\mathbf{x}_p$ in the leave-one-out procedure, then the inequality (34) holds. Therefore

$$\sum_{i=1}^{a} \alpha_i^0 \geq \frac{\mathcal{L}_{N+1}}{D^2} \tag{35}$$

Now from the Kuhn-Tucker condition (9) and eq (8), we have

$$\sum_{i=1}^{a} \alpha_i^0 = \mathbf{w}^2 \tag{36}$$

Combining eq (35) and eq (36), we prove the theorem

$$\mathcal{L}_{N+1} \leq \mathbf{w}^2 D^2$$

B COIL Database

Figure 19: *Images of 20 objects of the COIL database*

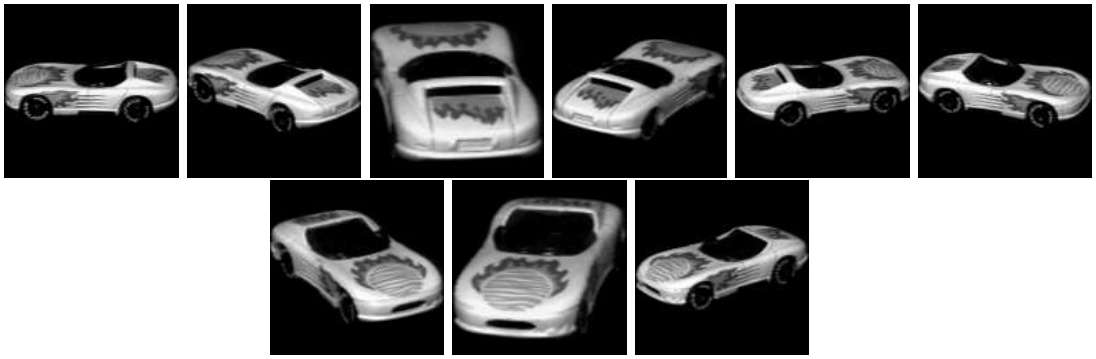

Figure 20: *9 of the 72 3D rotations of an object in the COIL database*

C COREL database

Figure 21: *Sample images of the Corel database : each row includes images from the following categories : Air Shows, Bears, Polar Bears, Elephants, Tigers, Arabian Horses, African Specialty Animals, Cheetahs-Leopards- Jaguars, Bald Eagles, Mountains, Fields, Night Scenes, Deserts, Sunrises-Sunsets.*

Figure 22: *Sample images of the Corel database with a hand-labeled categories: each row includes images from the following categories : Airplanes, Birds, Boats, Buildings, Fish, People, Cars*