Invariances in Kernel Methods: From Samples to Objects

Alexei Pozdnoukhov, Samy Bengio

IDIAP
CP 592, Rue de Simplon, 4
CH-1920 Martigny, Switzerland
e-mail: {pozd, bengio}@idiap.ch
Tel: +41 027 7217765
Fax: +41 027 7217712

Abstract

This paper presents a general method for incorporating prior knowledge into kernel methods such as Support Vector Machines. It applies when the prior knowledge can be formalized by the description of an object around each sample of the training set, assuming that all points in the given object share the same desired class. A number of implementation techniques of this method, based on hard geometrical objects and soft objects based on distributions are considered. Tangent vectors are extensively used for object construction. Empirical results on one artificial dataset and two real datasets of Electro-Encephalogram signals and face images demonstrate the usefulness of the proposed method. The method could establish a foundation for an information retrieval and person identification systems.

Key words: kernel methods, SVM, invariances, tangent vectors
PACS:

1 Introduction

Prior knowledge is often used in machine learning algorithms to constrain models towards reasonable solutions. One such class of prior knowledge relates to invariances. These are transformations of the inputs that leave the outputs unchanged. The general setting of including invariances into kernel methods was considered by Burges (1999). One of the widely used practical methods for incorporating invariances into Support Vector Machines (SVMs) is the Virtual Support Vector method, based on generating artificial samples from the current Support Vectors of the problem (Scholkopf et al. (1996)). The method performed particularly well on an optical digit recognition task. Another general way is to modify the cost function of the algorithm in order to penalize solutions not following the invariance properties. One such method (though not really suitable for large-scale datasets) was developed by Chapelle et al. (2002). Finally, a method of DeCoste and Burl (2000) called “kernel jittering” combines the generation of artificial examples with kernel modification. We do not consider here a number of application-specific methods.

In this paper, we present yet another approach to the problem, which assumes that the prior knowledge can be formalized as a mapping of a special kind. This mapping transforms each sample into an object in such a way that it includes the prior knowledge, similar to that done in the Tangent Distance approach of Simard et al. (1998) applied to neural networks. The method does not lead either to enlarged training sets or to modification of the cost function as opposed to other techniques. It simply exploits standard SVM optimization algorithms. It combines generative and discriminative approaches by use of local models (objects) based on training samples.

The rest of the paper is organized as follows. The general idea is presented in Section 2, as well as two implementations for hard geometrical (Section 3.1) and soft distribution-based objects (Section 3.2). We give links to several other apporaches in Section 4. Section 5 presents the experiments on artificial data where we illustrate the performance of the proposed method, and on two real datasets, where the first task is to classify EEG signals for a Brain-Computer Interface system and the second is devoted to a number of classification tasks in the area of face recognition or person identification. Section 6 completes the paper with a discussion and conclusions.

2 From Samples To Objects

Suppose we have some understanding of our data that can be formalized as a transformation of the inputs that leaves the outputs unchanged. For example, in a 2D image classification task we are often given the evident knowledge that small rotations and translations of the raw images do not affect the desired output class. Suppose the representation of the data (the set of features) allows us to describe the desired transformation as a mapping that leaves the outputs unchanged. The mapping applied to every sample produces a set of corresponding objects, which becomes a point of our consideration. In other words, we assume that given some understanding of the data we are able to generalize each sample into the equivalence class - the object in the input space. By doing this we aim to capture some prior similarities in the data. If this formalization is successfully performed, it is possible to deal with *objects* instead of samples when solving our particular learning problem. We will consider several kinds of such objects below.

2.1 Hard Objects

We define "hard" objects by the following geometrical transformation

$$x_i \mapsto S_{x_i} = \{\varphi_\alpha(x_i), \alpha \in \Lambda\} \tag{1}$$

where we denote the data samples by x_i. Function $\varphi_\alpha(.)$ defines a set in the input space through the admissible set Λ of parameters α. For example, one can consider segments instead of points x_i, as defined below in (10).

2.2 Soft Objects

Instead of using hard geometrical objects one can define an object as a local distribution of the kind

$$x_i \mapsto p(x|x_i, r_i), \tag{2}$$

where r_i is a vector of parameters of the local distribution $p(x|x_i, r_i)$. This distribution is constructed in a way to describe the desired local transformations of a sample. It represents the probability that a given point x is in fact a (transformed) sample x_i.

Though a uniform distribution on the bounded support can be considered as a hard object, we still discriminate the hard/soft cases due to the different underlying approaches used to define the kernel functions.

2.3 Objects Based on Tangent Vectors

One evident way to create objects from samples is to use the *tangent vector* approach. Tangent vectors were extensively used in the work of Simard et al. (1998) to introduce invariances. We will partly follow the notations of their paper. A good intuition for the following equations lies in considering 2D images on the plane (ξ, ψ). The intensity of the image is defined by some function $U(\xi, \psi)$. It provides a high-dimensional input vector x for a given discrete set of coordinates (ξ, ψ).

Suppose the 2D transformation of the image plane t_α we want to be invariant to is defined by the set of parameters α in some region of $D \subset R^2$:

$$t_\alpha : D \subset R^2 \mapsto t_\alpha(D) \subset R^2, \tag{3}$$

where α is a J-dimensional vector which parametrises the transformation. This transformation is assumed to be differentiable with respect to α and $(\xi, \psi) \in D$, and reduces to the identity transformation for some value of α^0. Then the object generated by this transformation and associated with an image U is defined by

$$S(U, \alpha) = U \circ t_\alpha^{-1}, \ \alpha \in \Lambda, \tag{4}$$

where Λ is some admissible set of parameters α. In the case of J local transformations one can linearly approximate $S(U, \alpha)$ as follows:

$$S_1(U, \alpha) = U + \sum_{j=1}^{J} (\alpha_j - \alpha_j^0) L_{\alpha_j}(U), \tag{5}$$

where $L_{\alpha_j}(U)$ are local transformations of U defined by:

$$L_{\alpha_j}(U) = \left. \frac{\partial S(U, \alpha)}{\partial \alpha_j} \right|_{\alpha = \alpha_0}. \tag{6}$$

Note that L_{α_j} are operators that generate the whole space of local transformations (a Lie algebra of local transformations). For example, three operators of X-translation, Y-translation and rotations about the origin produce a transformation (and a corresponding object) of all possible translations and rotations. *Tangent vectors* $\ell_j(x)$ can be obtained by discretising the result of applying the operators L_{α_j} to the continuous image U which correspond to a discrete sample x.

The examples of tangent vectors calculation for widely used transformations such as rotations and scaling are shown below:

- Rotation:

$$t_\alpha = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}, \quad L_\alpha^{rot} = \psi \frac{\partial}{\partial \xi} - \xi \frac{\partial}{\partial \psi}. \tag{7}$$

- Scaling:

$$t_\alpha = \begin{pmatrix} 1+\alpha & 0 \\ 0 & 1+\alpha \end{pmatrix}, \quad L_\alpha^{sc} = \xi \frac{\partial}{\partial \xi} + \psi \frac{\partial}{\partial \psi}. \tag{8}$$

If $\Lambda = R^J$, (4) is an J-dimensional differentiable manifold and (5) is a corresponding linear tangent manifold which is used in the tangent distance method of Simard et al.(1998). However, tangent vectors can hardly model the transformation of complex images, such as faces, providing acceptable results only for small values of α_i. Therefore, the bounded set of parameters $\alpha \in \Lambda$ can be used. Since every parameter α_i corresponds to a transformation we intend to be invariant to, then the choice of the set Λ defines the influence of this or that type of prior information on the invariances. Later, we will use tangent vectors for the construction of specific distributions which represent soft objects.

2.4 Objects Based on Sample Models

In general there is no need to build identical objects for every sample. Consider an example from Optical Character Recognition (OCR). There is no need to build an object which represents a set of symmetrically rotated characters for every sample of the database. A number of characters may appear to be rotated significantly already and the latter object representation would make them worse by rotating them unreasonably much.

So far, object construction can also be thought of as follows. Suppose we have an (output-independent) *model* $\Psi(.)$ for the samples we are dealing with. The model represents our knowledge of the sample's properties and can be of a very general kind. Given a sample from the dataset, the model transforms it into an *object* based on some estimated sample-dependent parameters: $object_i = \Psi(sample_i)$. Following the example from OCR, suppose we estimated the angle a digit character is rotated by, hence an object will be constructed to describe basically the rotation in the opposite direction.

However, practical implementation of this approach requires solving the tasks of application-dependent model construction and estimation of model parameters, which are rather complicated and will not be considered further here.

3 Kernels for Objects

There are two major ways to make use of objects in kernel methods. Generally, one would like to formulate a criterion for a learning algorithm directly for objects. For example, a criterion can be to maximise the margin between objects of different classes. One of the first attempts for constructing an algorithm of this type was recently proposed by Graepel et. al (2003). We explore here another approach based on defining a kernel function for objects, which can be used in a standard algorithm like SVM.

3.1 Kernels for Hard Objects

Since we apply our knowledge directly and deal with objects in the input space, it is reasonable to deal with distance-based kernels that have clear intuitive interpretation as a measure of similarity.

Suppose one uses a distance-based kernel, for example the commonly used Gaussian Radial Basis Function (RBF) kernel:

$$K(x_i, x_j) = e^{\frac{-\rho(x_i,x_j)^2}{2\sigma^2}}, \tag{9}$$

where sample-to-sample distance in the input space is defined by $\rho(x_i, x_j)^2 = \|x_i - x_j\|^2$ and σ^2 is the variance of the kernel.

Substituting the object-to-object distances into the kernel, one includes the prior knowledge into the algorithm. The problem here is to provide a way to compute distances between objects efficiently. In the following subsections we give some simple examples of distances that can be derived analytically and calculated efficiently.

3.1.1 Linear Scaling

Consider the linear transformation

$$x \mapsto G_x = \{\alpha x, \alpha \in [a, b]\} \tag{10}$$

where $a, b \in R^1$. This transformation corresponds to a brightness change of the image (given a raw image representation) or to an amplitude scaling of the signal. Note that in general for a finite range of α this transformation cannot be taken into account by simple normalization of the input data.

Consider the distance given by:

$$\rho(x, G(\hat{x}))^2 = (x - |\alpha^*|_{[a,b]}\hat{x})^2, \;\; \alpha^* = \frac{x \cdot \hat{x}}{\hat{x}^2}, \tag{11}$$

where $|g|_{[a,b]}$ is defined as a if $g < a$, b if $g > b$, and g otherwise. This is simply the distance between x and the segment $G_{\hat{x}} = [a\hat{x}, b\hat{x}]$ of the line corresponding to the directing vector $\hat{x}$. We will use a symmetrized sample-to-segment distance in the experiments below for an illustrative artificial example (Section 5.1). As we will see, even one-sided sample-to-object distance provide promising performance for introducing invariances into the classification model. This is also illustrated in the real task of EEG signals classification (Section 5.2). More generally, one has to take into account computational efficiency when deciding between one-sided and two-sided distances. If the two-sided distance can be computed reasonably fast, it should be the method of choice.

3.1.2 Translations

The second example is a particular case of translation invariance, i.e. the desired transformation is

$$x \mapsto P_x = \{e_i t_i + x; t_i \in [-t_i^{lim}, t_i^{lim}]\}, \tag{12}$$

where e_i are the basis vectors of the input space R^N and t_i^{lim} is the maximum allowed translation in dimension i, with $i = 1, 2, ..., N$. It corresponds to the mapping into an interior of the cuboid whose “center” is vector x and all the edges are parallel to the axes. This transformation corresponds to a speckle noise in the image and its use will be illustrated experimentally in Section 5.3.

The distance between vector x and cuboid $P_{\hat{x}}$ is given by minimization of

$$\rho(x, P_{\hat{x}})^2 = \min_{\vec{t}} (x - P_{\hat{x}}(\vec{t}))^2, \tag{13}$$

over the set of parameters $\vec{t} = \{t_1, t_2, ...t_N\}$ and can be calculated as follows:

$$\rho(x, P_{\hat{x}})^2 = \sum_{i=1}^{N} ((x^i - \hat{x}^i) - |x^i - \hat{x}^i|_{[-t_i^{lim}, t_i^{lim}]})^2, \tag{14}$$

where x^i, $\hat{x}^i$ are the components of the corresponding vectors.

The distance between two objects defined by (12) can be similarly computed with

$$\rho(P_x, P_{\hat{x}})^2 = \sum_{i=1}^{N} \Big| |x^i - \hat{x}^i| - 2t_i^{lim} \Big|_{[0,\infty]}. \tag{15}$$

We will use this distance later for noisy image classification (Section 5.3).

3.1.3 Object to Object Distances

Using the sample-to-object distances, we take into account some prior knowledge but still use a kernel matrix that might not be positive definite. One can use an average of two sample-to-object distances to make the kernel matrix symmetric. Ideally, an object-to-object distance may be preferable, but its calculation is often quite a difficult task and can not always be easily performed. For the considered examples it is possible to compute segment-to-segment and box-to-box distances (15).

In general, the computation of the Euclidean distance between the objects leads to a constrained optimization problem. Consider two objects $S(x, \alpha)$ and $\hat{S}(\hat{x}, \hat{\alpha})$. We approximate the Euclidean distance between them with the Euclidean distance between their linear approximations:

$$\rho(S, \hat{S})^2 \simeq \rho(S_1, \hat{S}_1)^2 = \min_{\alpha, \hat{\alpha}} \left(x - \hat{x} + \sum_{i=1}^{L} \alpha_i \ell_{\alpha_i}(x) - \hat{\alpha}_i \ell_{\hat{\alpha}_i}(\hat{x}) \right)^2, \qquad (16)$$

subject to

$$\alpha_i \in [\alpha_i^{min}, \alpha_i^{max}], \ \hat{\alpha}_i \in [\hat{\alpha}_i^{min}, \hat{\alpha}_i^{max}]. \qquad (17)$$

This problem can be considered as a Nearest Point Problem and a number of simple iterative methods such as Gilbert's or Mitchel-Demyanov-Malozyomov algorithms (which are also used for SVM training) can be applied for distance minimization (e.g. Keerthi et al. (1999) and references therein). By controlling the maximum number of iterations one defines a trade-off between accuracy and speed. The unconstrained problem (16) corresponds to the Tangent Distance method. Its direct application for SVM kernels was considered by Haasdonk and Keysers, 2002.

3.2 Kernels for Soft Objects

Given a soft object in a form of local distributions centered at each sample, one can apply a number of approaches developed in statistics for comparing two distributions. We first introduce here a special kind of distribution which makes use of tangent vectors. Next, we present some methods for defining the corresponding kernels.

3.2.1 Local Distributions based on Tangent Vectors

Suppose the transformation we want to be invariant to defines a differentiable manifold in the input space. Hence the tangent vectors can be defined as described above, and the whole set of tangent vectors can be used to model all the local linear transformations of the given image. Let us define the following function H which gives the measure of proximity of a given vector x to the linear span of some vector x' generated with a tangent vector ℓ_j:

$$H(x|x', \ell_j) = e^{-\frac{(x-x')^2 \ell_j^2 - ((x-x')\cdot \ell_j)^2}{2\gamma_w^2 \ell_j^2}}, \tag{18}$$

where γ_w is the parameter related to the width of the proximity region.

The following distribution K_s describes a similarity between given sample x and an object based on sample x' generated by a set of corresponding tangent vectors $\{\ell_1, ...\ell_J\}$:

$$K_s(x, x') = e^{-\frac{(x-x')^2}{2\sigma^2}} \cdot \prod_{j=1}^{J} (\eta + H(x|x', \ell_j)) \tag{19}$$

where σ is a bandwidth and the real number $\eta \in [0, 1]$ defines the shape of the distribution. The idea behind this formula is to combine several single proximity measures (18) into the general one. The extra parameters were introduced in order to control the influence of this or that type of invariance.

Assuming $\eta = 0$ in (19) and applying normalization, one can reduce (19) to a standard Gaussian:

$$\begin{aligned} K_s(x, x') &= \frac{e^{-(x-x')^T L_{x'}^{-1} (x-x')}}{(2\pi)^{N/2} |L_{x'}|^{1/2}}, \quad \text{where :} \\ L_{x'}^{-1} &= \left(\frac{1}{2\sigma^2} + \frac{J}{2\gamma_w^2}\right) I - \sum_{j=1}^{J} \frac{\ell_j \ell_j^T}{2\gamma_w^2 \ell_j^2}, \end{aligned} \tag{20}$$

where I is an identity matrix, and $|...|$ denotes the determinant. This representation will be extensively used below for kernel evaluation.

3.2.2 Tangent Vector Kernels

The simplest way to use the latter distribution for introducing invariances into kernel methods is to consider (19) as a one-sided (sample-to-object) similarity measure. Then, a two-sided kernel K_d can be obtained by taking the following average:

$$K_d(x, x') = \frac{1}{2}(K_s(x, x') + K_s(x', x)). \tag{21}$$

The proposed kernel combines the advantages of both VSV and Tangent Distance approaches. In this approach we not only analytically include the Virtual SV into the model (without putting them into the data), but also take into account all the linear combinations of invariant transformations of interest. Moreover, using all the tangent vectors which correspond to linear transformations, one can take into account all the possible local linear transformations of an image.

The proposed kernel (19)-(21) is not the only possible one to make use of the tangent vectors. Other kernels can be constructed in a similar way to the one presented by combining the terms (18) in a different manner.

3.2.3 Distribution-based Tangent Vector Kernels

To be consistent in the sample-to-object approach, let us consider the distributions (18) defined for every sample. We introduce the Distribution-based Tangent Vector Kernel (DB TVK) as follows. The kernel can be obtained by measuring the overlap of two distributions that correspond to the object based on samples x and x'. To do this we introduce the following kernel between two distributions:

$$K_B(x', x'') = \int K_s(x, x')^\rho K_s(x, x'')^\rho dx,\ 0 \le \rho \le 1, \tag{22}$$

which is a dot product in the space of functions $K_s(., x')$ and was called the probability product kernel in Kondor and Jebara (2004).

The closed form of $K_B(x', x'')$ can be obtained for a number of cases. For $\eta = 0$ and using equation (20), $K_B(x', x'')$ reduces to integration of Gaussians and can be expressed as follows:

$$\begin{aligned} &K_B(x', x'') = (2\pi)^{\frac{(1-2\rho)N}{2}} \left|\hat{L}\right|^{\frac{1}{2}} |L_{x'}|^{-\frac{\rho}{2}} |L_{x''}|^{-\frac{\rho}{2}} \\ &\exp(-\tfrac{\rho}{2} x'^T L_{x'}^{-1} x' - \tfrac{\rho}{2} x''^T L_{x''}^{-1} x'' + \tfrac{1}{2} \hat{x}^T \hat{L} \hat{x}) \end{aligned} \tag{23}$$

where $\hat{L} = (\rho L_{x'}^{-1} + \rho L_{x''}^{-1})^{-1}$ and $\hat{x} = \rho L_{x'}^{-1} x' + \rho L_{x''}^{-1} x''$.

A closed form equation for the distribution-based tangent vector kernel can also be derived for $\eta \neq 0$ and $\rho = 1$, which is more interesting but yields an even more cumbersome expression. We consider this case in the next section.

3.2.4 Making Distribution-based TVK practical

Direct implementation of the proposed DB TVK demands costly computations. Therefore, we propose here a practical way to compute (22). It consists of computing the approximation of the integral (22) for $\eta \neq 0$, $\rho = 1$ and $K_s(x, x')$ as presented in (19). Note that we fixed $\rho = 1$, hence the latter approximates the corresponding *expected likelihood* kernel. The following approximation can be used:

$$K_B(x', x'') = I_0 + (1 + \frac{\sigma^2}{2\gamma_w^2})^{\frac{1-D}{2}} \sum_{j=1}^{J} (I'_j + I''_j) + ..., \tag{24}$$

where

$$I_0 = e^{-\frac{(x'-x'')^2}{4\sigma^2}}. \tag{25}$$

The term I^0 correspond to the RBF kernel between samples x' and x''. The terms I'_j and I''_j correspond to the impact of j^{th} invariance of the samples x'' and x' to the samples x' and x'' correspondingly. The exact expressions of I'_j and I''_j are quite cumbersome. We present here the general idea only. The expansion of (22) with terms given by (19), $\rho = 1$ and $\eta = 1$ consist of the sum of products. One can neglect the terms which include the products of three and more exponents. Then the integration of the rest terms is analogous to (23). This approximation requires only $O(J \cdot N)$ operations to compute.

The impact of invariances reduces as the input dimension increases. For very high dimensional input spaces its influence vanishes, and the only term that matters is I_0. We've faced this problem in our experiments, which we describe later in Section 5.4.

4 Analysis and Links to Other Methods

4.1 Links with Kernel Jittering and Virtual SV

The distance between hard objects is a distance between some of the points the objects consist of. The points that give minimum to the distance effectively affect the model and can be considered as virtual samples. This allows interpretation of the described approach as a kind of virtual sample approach with automated choice of virtual samples, which may differ for every pair of objects. Furthermore, virtual samples can be used to replace tangent vectors with finite difference vectors. It appeared to be useful in our face classification experiments (see Section 5.3).

In analogy with the Virtual Support Vector approach of Scholkopf et al.(1996), one can define objects based on pre-determined support vectors only to enhance the speed of the algorithm.

The method of kernel jittering was proposed by DeCoste and Burl (2000). It combines artificial sample generation and kernel function modification as follows. Consider two samples, x_i and x_j and the corresponding non-jittered kernel function K_{ij}. Assume sample x_j could have been any of a set of values around x_j according to a "jittering" function. Consider the transformed ("jittered") forms of the sample x_j, including itself, and select one (x_{q^*}) closest to x_i in the feature space according to the Euclidean distance in the feature space:

$$q^* = \arg\min_q \sqrt{K_{ii} - 2K_{iq} + K_{qq}}. \tag{26}$$

The new "jittered" kernel for the examples x_i and x_j is simply K_{iq^*}. This idea can be interpreted as follows. Believing that transformed examples belong to the same class, kernel jittering corresponds to a kernel based on the distance between the sets generated from the examples by the allowed transformations.

The main drawback of the jittering approach is the need to do a lot of kernel calculations while selecting the minimal distance (26). The approach also requires that we do these calculations during the testing phase.

The distance-based methods proposed above can be considered as an analytical jittering. It does not suffer from the drawbacks described above, however it introduces valuable restrictions on the allowed transformations.

4.2 Vicinal Risk Minimization

Vapnik (2000) considered local distributions(soft objects) instead of samples to introduce the Vicinal Risk Minimization (VRM) learning principle. Defining the *vicinities* of the training samples and assuming some local density $p(x|x_i, r_i)$ therein, one obtains the following Vicinal Risk functional:

$$R_{vic}(\alpha) = \frac{1}{\ell}\sum_{i=1}^{\ell} L\left(y - \int f(x,\alpha)p(x|x_i,r_i)dx\right), \tag{27}$$

where x_i is a training sample and r_i is its vicinity parameter. Minimizing (27) instead of empirical risk is called the Vicinal Risk Minimization (VRM) principle.

Vapnik mentions how to use the VRM principle to incorporate an invariance into the learning algorithm. Using the density functions $p(x|x_i, r_i)$ defined on the non-symmetrical support that describes the invariance to the desired transformation, one enforces the learning algorithm to obey the invariance's properties.

The following Vicinal Support Vector algorithm is obtained by Vapnik (2000):

$$f(x) = \sum_{i=1}^{\ell} \beta_i^* D(x, x_i) + b, \tag{28}$$

where the β_i^* coefficients are such that

$$\beta^* = \arg\max_{\beta} \sum_{i=1}^{\ell} \beta_i - \frac{1}{2} \sum_{i,j=1}^{\ell} y_i y_j \beta_i \beta_j M(x_i, x_j), \tag{29}$$

subject to the constraints:

$$\begin{array}{l} \sum_{i=1}^{\ell} y_i \beta_i = 0, \\ 0 \leq \beta_i \leq C. \end{array} \tag{30}$$

Functions $D(x, x_i)$ and $M(x_i, x_j)$ are one- and two-vicinal kernels correspondingly:

$$D(x, x_i) = \int K(x, x') p(x'|x_i, r_i) dx', \tag{31}$$

$$M(x_i, x_j) = \iint K(x, x') p(x|x_i, r_i) p(x'|x_j, r_j) dx dx'. \tag{32}$$

4.2.1 Scaling Invariance

Let us now present a simple example. To obtain the invariance described by (1) with $\beta = 0$, consider the following vicinity density function:

$$p(x|x_i, \gamma) = \frac{1}{\sqrt{2\pi}\gamma} \int \delta(x - (1 - \alpha)x_i) e^{-\frac{\alpha^2}{2\gamma^2}} d\alpha, \tag{33}$$

where δ is the delta function, and the γ parameter defines the width of the vicinity and, hence, the influence of scaling invariance.

Substituting (33) in both (31) and (32) using the standard isotropic RBF kernel function given in (9) with the bandwidth parameter σ gives:

$$D(x, x_i) = \frac{\sigma}{\kappa} e^{-\frac{(x-x_i)^2}{2\kappa^2}} e^{-\frac{\gamma^2}{2\sigma^2\kappa^2}(x^2 x_i^2 - (x, x_i)^2)} \tag{34}$$

and

$$M(x_i, x_j) = \frac{\sigma^2}{\eta} e^{-\frac{\sigma^2(x_i - x_j)^2}{2\eta^2}} e^{-\frac{\gamma^2}{\eta^2}(x_i^2 x_j^2 - (x_i, x_j)^2)}, \tag{35}$$

where the following definitions were used:

$$\kappa^2 = \gamma^2 x_i^2 + \sigma^2, \tag{36}$$

$$\eta^2 = \gamma^2 \sigma^2 (x_i^2 + x_j^2) + \gamma^4 (x_i^2 x_j^2 - (x_i, x_j)^2) + \sigma^4. \tag{37}$$

The resulting kernels are still RBF-based. The “effective” kernel bandwidth depends both on the σ and γ parameters and on the samples x_i and x_j. One can note the similarity of (34)-(35) to the kernels presented before.

5 Applications and Experiments

It is often difficult to formulate real-life problems in a way suitable for object definition in the input space. For example, it is difficult to define objects that correspond to the invariances of interest in image processing such as 3D rotations with changing lighting conditions. This is one of the drawbacks of the described approaches.

We present a series of experiments illustrating the proposed approaches. These are an artificial two-class classification task, a problem of EEG signals classification and a number of face image classification tasks.

5.1 Artificial Data

To illustrate the action of the considered methods, we used an artificial dataset generated to be invariant to (10). The goal is thus to illustrate the influence of the modified kernels on the decision boundary.

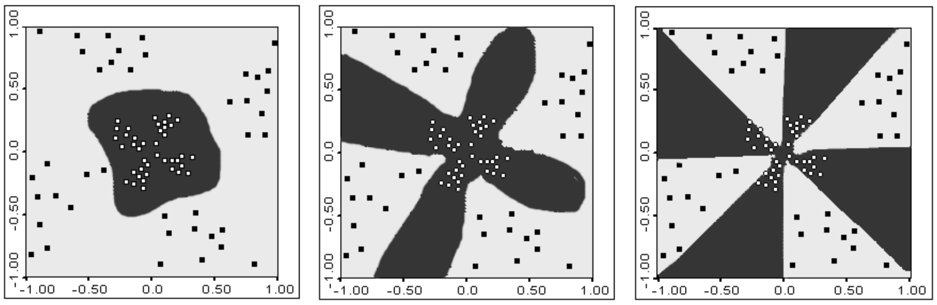

Fig. 1. Artificial two-class classification problem. Black training points have to be discriminated against white training points. Left: Original decision function of an SVM with RBF kernel ($\sigma = 0.2$), Center: decision function using slightly jittered kernel, Right: decision function facing full invariance.

Figure 1 illustrates the training data for both classes and the decision boundaries obtained with the following algorithms: the left image shows the original SVM with RBF kernel ($\sigma = 0.2$); the center one shows an SVM with RBF kernel ($\sigma = 0.2$) and distance defined by (11) with $a = 0.5$, $b = 2$; finally, on the right we see an SVM with RBF kernel ($\sigma = 0.2$) and distance defined by (11) with $a = 0.01$, $b = 10$. The substantial difference between the presented solutions lies in the number of support vectors, which is 20 for the standard solution (left figure) and 8 for the modified one (right figure). Note, that given the knowledge of global scaling invariance (Equation (10) with $\alpha \in R^1$) one could obtain the right solution by simply using input normalization. However, this is not the case if the scaling is bounded (Equation (10) with $\alpha \in [a, b]$). VRM-based approach described in Section 4.2 results in similar solutions.

5.2 EEG Signals Classification

The next series of experiments used EEG signals taken from the first competition devoted to Brain-Computer Interface system design. The competition was organized after the NIPS'01 Brain Computer Interface workshop. The task is to classify the signals that correspond to imaginary movements of the left or right hand. The original data consists of signals taken from different electrodes located on a human's head. The difference between two particular signals (from the C3 and C4 electrodes, according to the standard labeling) was taken as input for the algorithm. The data were resampled to 100Hz, the input dimension (the signal length) is 150. The dataset consists of 413 training and 100 testing samples. The details of data collecting and problem settings can be found at [http://newton.bme.columbia.edu/competition.htm].

Raw data usage may appear not to be the best way of carrying out classification. However, it was found to work well for SVMs. For example, the classification performance based on auto-regressive coefficients was significantly

Table 1
Experimental results on the EEG dataset

	Algorithm	Testing Error, %
EEG	SVM	9
	Segment-based SVM	6
	VRM-based SVM	6

worse. The evident properties of these data are the invariances to the signal amplitude and the selection of the reference point of the "zero" level of the signal. These findings are also justified by the physical conditions of the EEG signal measuring process.

The results for the baseline SVM classifier based on Gaussian RBF kernel and SVMs with modified kernels (as derived in Sections 3.1.1 and 4.2.1) are presented in Table 1. The hyper-parameters of all the algorithms were tuned according to cross-validation on the training set. The obtained values are $C = 25$, $\sigma = 1500$. The invariance-defining parameters are $\gamma = 0.55$ for VRM-based kernel, and for the kernel based on hard objects (segment-based SVM) the scaling range is $[0.5, 1.5]$.

Both methods provided an improvement of the classification performance according to the testing error. However, this improvement is hardly statistically significant (79% confidence only) since the size of the test set is only 100 samples. This is a basic disadvantage of the competition setting caused by difficulties in data collection.

5.3 Face Recognition Experiments

The next example presented here deals with a real dataset obtained from a face detector. These are faces detected on every fifth frame of a movie using a face detector from Schneiderman and Kanade (2000). Image dimension is 81 by 81 and greyscale level is 8 bit. There were 2899 images in the database. The data is available at [http://www.robots.ox.ac.uk/~vgg/data]. We present an approach to the problem of binary classification of the main actor against all the other images captured. Hence, this task can be seen either as a person identification or an information retrieval task.

The training set consists of every tenth image of the database, while the testing set consists of all the other ones. We used the first thousand images of the database, ending up with training and testing sets of 100 and 900 samples correspondingly. Example images are presented in Figure 2.

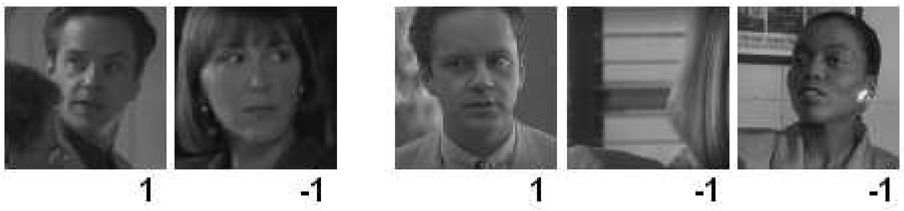

Fig. 2. Examples of clean samples. Left: two random training samples. Right: three random testing samples. The labels for class membership are shown below the images.

5.3.1 Noisy Image Classification

To illustrate the use of the method described in Section 3.1.2 we have corrupted the images with an additive speckle noise. The noise is generated from a uniform distribution with zero mean and variance 30. Example images with noise are shown in Figure 3. Another noisy testing set was obtained by corrupting the clean testing set with the same noise, and “outliers”: random 10% of the pixels were corrupted with uniform noise with zero mean and variance 70.

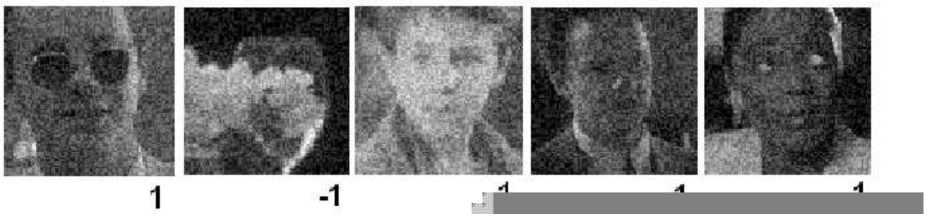

Fig. 3. Examples of noisy samples. The labels for class membership are shown below the images.

To show the performance of the method, we added noise to the testing set only. The objective is to obtain an algorithm robust to a known type of noise while given a clean training set only. Hence we are given a training set and a prior knowledge about the type of noise that occurs in the testing samples.

We used the raw image as input. Standard SVMs with Gaussian RBF kernel (9) were trained on the clean training set. The parameters were chosen according to the minima of the cross-validation error. The parameters are: $\sigma = 3000$, $C = 100$. Classification error on the clean testing data is 9%, 17% on the noisy testing data and 37% on the noisy data with outliers. One possible solution to handle noise is to use denoising techniques to preprocess the testing data before applying the SVM classifier. Different denoising techniques such as Wiener filtering, median filtering and Gaussian bluring were used. The best result achieved was 14% of testing error for noisy data and 34% for the noisy data with outliers.

The SVM with an object-based kernel (15) was applied to the problem. The testing error for various values of the prior parameter t^{lim} is presented in Figure 4 for both noisy testing sets. The minima of the testing error is achieved for the values of prior parameter t^{lim} which correspond to the standard deviation of the noise. The modified algorithm significantly outperforms standard SVM on the noisy testing data. However, testing error of the modified algorithm with $t^{lim} = 5$ gives 10.8% of the testing error on the clean testing data.

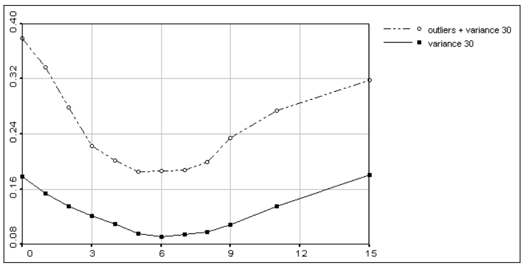

Fig. 4. Testing error curve of the SVM with object-based kernel for both noisy testing sets. X-axis: t^{lim} parameter, Y-axis: testing classification error rate. Testing errors at $t^{lim} = 0$ (37% and 0.18%) correspond to standard SVM.

5.4 Invariant Face Images Classification

In order to test the proposed approaches of Section 3.2, we conducted experiments using images of the faces from the database described above. All the 2899 images of the database were used. We used subsets of 300 training and 2599 testing samples. The resolution was decreased to 60x60 picsels.

We compared standard SVM with RBF kernel, Virtual Support Vector method, Kernel Jittering, and the proposed approaches of Sections 3.2.2 and 3.2.3. Two types of invariant transformations were studied: rotations (7) and scalings (8). Some considerations of practical implementation of the approaches are described below starting with tangent vectors evaluation.

5.4.1 Tangent Vectors and Finite Difference Vectors

There are some noticeable limitations in computing the tangent vectors. An input image has to be smooth enough to compute gradients that would approximate local transformations of the original image. The original method works well for binary images of digits, which were blurred with Gaussian filter for computing the gradients. We applied the method for our data using different Gaussian smoothing and found that the obtained approximation from these tangent vectors was not sufficient to describe real transformations. Instead we generated virtual samples by applying a finite desired transformation and used them for computing the finite differences that were used to approximate the tangent vectors. Example transformed images obtained by rotations with original gradient-based tangent vectors and finite differences are shown in Figure 5.

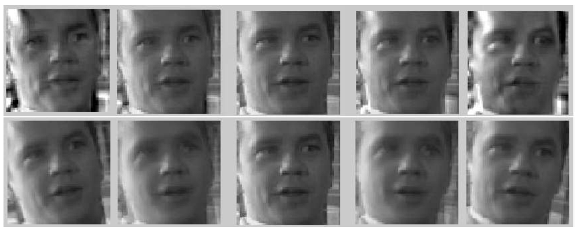
Fig. 5. Two Types of Virtual Images.

The first line in Figure 5 presents images obtained by applying direct calculation of tangent vectors according to (7). We can thus see that despite the accurate tuning of Gaussian filtering and other "tricks", only very local rotations are reasonable.

The second line in Figure 5 presents the original sample image x in the center; virtual samples obtained from x by applying rotations of 10 degrees are shown on the left and right of the figure. Let us denote them as $x+\ell^{exp}_{left}$ and $x+\ell^{exp}_{right}$. The intermediate images in between are $x + 0.5\ell^{exp}_{left}$ and $x + 0.5\ell^{exp}_{right}$.

The problem described here complicates the approach. Generally, one of the baseline aspect in the proposed approach is a definition of the objects. This has to be done in a way to describe the desired invariance in the best possible way. For our case, it was achieved by introducing the finite difference vectors, since their use allows for better modelling of the real-life invariances.

5.4.2 Scaling and Rotational Invariances with TVK

Since this approach implied that left and right rotations correspond to different tangent vectors, we used the following modified Tangent Vector Kernel:

$$K^{fd}_s(x,x') = e^{-\frac{(x-x')^2}{2\sigma^2}} + \sum_{j=1}^{J} H(x|x',\ell_j) \cdot e^{-\frac{(x-x'-\ell_j)^2}{2\gamma_r^2}}, \tag{38}$$

where we introduced one extra parameter γ_r corresponding to the length of proximity region. As one can see, this kernel is very similar to the particular case of the original kernel (19), for fixed $\eta = 1$ and $\gamma_r = \sigma$. Then, due to the reasons, discussed above in Section 5.4.1, the kernel is modified to take into account not symmetric finite difference vectors. Note, that with this modification, one uses more than one "tangent vector" per invariance, obtaining better modelling of the real-life invariant transformations.

With a proper choice of parameters in (38) ($\gamma_w \sim \infty$, $\gamma_r = \sigma$), the resulted model is closely linked to VSV. The noticeable difference is that in the VSV approach every virtual sample is included in the decision function with its own weight, while in our case all the virtual samples form an object hence share the same weight.

The parameters of the algorithms were chosen according to the minimum of cross-validation error over the training set, resulting in $\sigma = 600$, $C = 100$. Parameters γ_w and γ_r in (38) can be chosen by the following heuristics: $\gamma_w \sim \sigma$, and $\gamma_r^2 \sim Var(\ell_{ij})$, i.e. the variance of tangent vectors. We used $\gamma_w = 500$ and $\gamma_r = 1000$.

5.4.3 Scaling and Rotational Invariances with DB-TVK

Despite of the problems described above, we used non modifyed Distribution-Based TVK, as it was introduced in Section 3.2.3. The parameters were as follows: $\sigma = 600$, $C = 100$, $\gamma_w = 1000$.

As it was mentioned above, DB-TVK has worse performance for high-dimensional input spaces. It is clearly seen from equations in Section 3.2.4, that the impact of “invariant” terms of the kernel reduces with dimensionality. However, we obtained reasonable results in the presented case study.

5.4.4 Experimental Results

Table 2 presents testing errors obtained with SVM with Gaussian RBF kernel (SVM), SVM trained with virtual samples (VSV SVM), SVM with jittered kernel (KJ SVM) and SVM with Tangent Vector and Distribution-based Tangent Vector Kernel (TVK SVM and DB-TVK SVM). We used the same virtual samples both for VSV and KJ SVM and for computing the finite difference vectors in TVK and DB-TVK. This is the reason of similar results obtained with all the methods. The improvement of the testing error in comparison to the baseline SVM is statistically significant with a 95% confidence interval.

Table 2
Testing Error

	Algorithm	Testing Error, %
	SVM	11.2
	VSV SVM	9.8
	KJ SVM	10.0
	TVK SVM	9.7
	DB-TVK SVM	9.9

Another interesting experiment is to show the relative importance of prior knowledge with respect to the amount of available training data. We thus split the data using every N-th sample of the entire data for training, while the rest of the data were used for testing. Figure 6 shows the testing errors obtained for these different partitions. The X-axis in Figure 6 corresponds to the logarithm of the training set size and the Y-axis corresponds to the testing error. As expected, when the number of training examples is very small, prior knowledge is of prime importance, while its importance eventually decreases with increased amount of training examples.

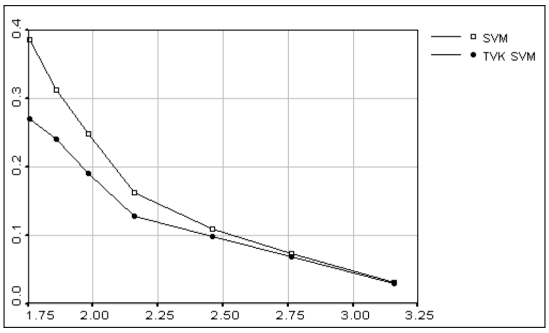

Fig. 6. SVM with RBF and TVK kernels.

6 Discussion and Conclusions

The method of prior knowledge incorporation considered in this paper results in a kernel modification and exploits the standard SVM algorithm. The main idea of the kernel construction is to consider an object in the input space: a set which can be derived for each training sample by applying all known invariant transformations. Then the kernel is defined for pairs of objects. Kernel calculation can appear to be a computationally expensive part of the algorithm, although in the considered examples it was not the case. The method does not lead to enlarging the training set, as it is the case for traditional virtual samples approaches of Girosi and Chan (1995), Niyogi et al. (1998).

The proposed approach has close links with the regularization framework. Loosely speaking, regularization is used to enforce smoothness of the function in the vicinity of the training points. For a learning algorithm based on the squared loss function it is shown by Leen (1999) that, under certain assumptions, the approaches of adding virtual samples to the training set and adding a regularization term to the cost function are equivalent. Our approach generalizes the virtual sample approach, and obviously it has regularizing properties.

Since we propose an object definition based on combining the sample and some prior knowledge, the presented method naturally establishes a link between kernel methods and generative models. Considering the whole structure of local distributions, we somehow model the class density. The general approach in this field is given in Jaakkola and Haussler (1999), where the Fisher kernel based on a metric defined on a parametric generative probability model is presented.

Some similar approaches were recently proposed by Kondor and Jebara (2003). The idea there is to make a transition from samples to the sample-characterizing distributions which are then used for kernel definition. This approach mainly uses the distributions (objects) for data representation. As the evolution of the previous research, Kondor and Jebara (2004) presented a similar sample-to-object framework. This transition step was used as an intermediate one to introduce probability product kernels. The aim of the presented research is to focus on the prior knowledge incorporation.

In conclusion, in this paper we presented a general method to incorporate prior knowledge into kernel methods. It is based on modifying the setting of the problem by a transition from samples to objects, which are generated from them using some prior knowledge. We mainly considered these objects in the form of local distributions. Tangent Vectors were extensively used for the construction of the latters. Several methods of kernel definition were presented and tested in experiments on artificial and real-life data.

Acknowledgments

This research has been partially carried out in the framework of the European project LAVA, funded by the Swiss OFES project number 01.0412. It supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778, funded in part by the Swiss OFES. It was also partially funded by the Swiss NCCR project (IM)2.

References

C.J.C. Burges, 1999. Geometry and invariance in kernel-based methods. In: B.Scholkopf, C.J.C. Burges, and A.J. Smola (eds.), *Advances in Kernel Methods - Support Vector Learning*, MIT Press.

B. Scholkopf, C. Burges, and V. Vapnik, 1996. Incorporating invariances in support vector learning machines. In: C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, and B. Sendhoff, (eds.), Artificial Neural Networks

ICANN‘96, pp. 47-52, Berlin. Springer Lecture Notes in Computer Science, Vol. 1112.

O. Chapelle and B. Scholkopf, 2002. Incorporating invariances in nonlinear SVMs. In: T.G. Dietterich, S. Becker and Z. Ghahramani, (eds.),*Advances in Neural Information Processing Systems*, vol. 14, pp. 609-616. MIT Press, Cambridge, MA, USA.

D. DeCoste, M.C. Burl, 2000. Distortion-invariant recognition via jittered queries. In *Computer Vision and Pattern Recognition, CVPR-2000*, June.

Haasdonk, D. Keysers. Tangent Distance Kernels for Support Vector Machines. In the proc. of ICPR'02, Vol.2, pp. 864-868.

P. Simard, Y. LeCun, J. Denker, B. Victorri, 1998. Transformation invariance in pattern recognition, tangent distance and tangent propagation. In: G. Orr and K. Muller, (eds.), *Neural Networks: Tricks of the trade.* Springer.

V. Vapnik, 1998. Statistical Learning Theory. J.Wiley, NY, 1998.

O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, 2001. Vicinal risk minimization. In: T.K. Leen, T.G. Dietterich, and V. Tresp, (eds.), *Advances in Neural Information Processing Systems*, vol. 13, pp. 416-422.

V. Vapnik, 2000. The Nature of Statistical Learning Theory. Second edition, Springer-Verlag, NY.

T.K. Leen, 1995. From data distributions to regularization in invariant learning. Neural Computation, vol. 7, no. 5, pp. 974-981.

T. Jaakkola, and D. Haussler, 1999. Exploiting generative models in discriminative classifiers. In: M.S.Kearns, S.A.Solla, D.A.Cohn (eds.) *Advances in Neural Information Processing Systems*, vol. 11, pp. 487-493, MIT Press.

S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, K.R.K. Murthy, 2000. A Fast Iterative Nearest Point Algorithm for Support Vector Machine Classifier Design. IEEE Transactions on Neural Networks, 11(1), pp.124−136.

R. Kondor, T. Jebara, 2003. A Kernel Between Sets of Vectors. In proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC.

R. Kondor, T. Jebara, A. Howard, 2004. Probability Product Kernels. Journal of Machine Learning Research 5(2004), pp.819−844.

F. Girosi, and N. Chan, 1995. Prior Knowledge and the Creation of Virtual Examples for RBF Networks. Neural Networks Signal Processing Proceedings of the 1995/IEEE-SP/Workshop, IEEE Signal Processing Society, Cambridge, MA, 201-210, September 1995.

P. Niyogi, T. Poggio, and F. Girosi. Incorporating Prior Information in Machine Learning by Creating Virtual Examples. IEEE Proceedings on Intelligent Signal Processing, Vol. 86, No 11, 2196-2209, 1998.