Invariant Kernel Functions for Pattern Analysis and Machine Learning

Bernard Haasdonk [1] and Hans Burkhardt [2]

[1]Applied Mathematics Department
Albert-Ludwigs-University Freiburg
Hermann-Herder-Str. 10
79104 Freiburg, Germany
haasdonk@mathematik.uni-freiburg.de

[2]Computer Science Department
Albert-Ludwigs-University Freiburg
Georges-Köhler-Allee 52
79110 Freiburg, Germany
burkhardt@informatik.uni-freiburg.de

Abstract

In many learning problems prior knowledge about pattern variations can be formalized and beneficially incorporated into the analysis system. The corresponding notion of *invariance* is commonly used in conceptionally different ways. We propose a more distinguishing treatment in particular in the active field of kernel methods for machine learning and pattern analysis. Additionally, the fundamental relation of invariant kernels and traditional invariant pattern analysis by means of invariant representations will be clarified. After addressing these conceptional questions, we focus on practical aspects and present two generic approaches for constructing invariant kernels. The first approach is based on a technique called invariant integration. The second approach builds on invariant distances. In principle, our approaches support general transformations in particular covering discrete and non-group or even an infinite number of pattern-transformations. Additionally, both enable a smooth interpolation between invariant and non-invariant pattern analysis, i.e. they are a covering general framework. The wide applicability and various possible benefits of invariant kernels are demonstrated in different kernel methods.

DOI 10.1007/s10994-007-5009-7.
Electronic access: http://www.springerlink.com/content/f07m327r45151535/fulltext.pdf

1 Introduction

Machine learning, pattern analysis and pattern recognition all benefit largely from the active field of kernel methods, which has developed to state-of-the-art during the last decade, cf. (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004). Numerous kernel methods and kernel functions have emerged which hereby enhance the flexibility, applicability and manifoldness of these methods. The famous *no free lunch* and *ugly duckling* theorems state formally, that prior knowledge or assumptions of a problem at hand must be incorporated into the solution, e.g. (Duda et al., 2001). Without prior knowledge, no best classification system or best pattern representation exist. Also empirical studies uniformly demonstrate the importance of prior knowledge for the generalization ability of pattern analysis systems. We focus on a traditional type of problem specific prior knowledge, namely knowledge about pattern variations or invariances. This means that certain transformations of the objects under investigation are known, which leave the inherent object meaning unchanged. In classical pattern recognition this is solved by a pattern pre-processing step of *invariant feature extraction*. In view of the new armada of kernel methods, it is valuable to ask for general ways to incorporate the invariance knowledge into kernel functions and to understand, how this conceptionally extends the traditional invariant feature approach. These will be the main topics of the present study.

Goals and Paper Organization

We will briefly list some desired properties of a general approach for incorporating invariance knowledge.

1. **Various Kernel Methods:** The approaches should not be learning-task-specific, especially not restricted to support vector machines (SVMs) or other optimization-based techniques. In principle, the approaches should support arbitrary learning methods implicitly working in a kernel-induced feature space.

2. **Various Kernel types:** The approaches should be applicable to various kernel functions. In particular, arbitrary distance-based or inner-product-based kernels should be supported.

3. **Various Transformations:** The methods should allow to model both infinite and finite sets of transformations. The transformations should comprise group- and non-group transformations, discrete and continuous ones.

4. **Adjustable Invariance:** The extent of the invariance should be explicitly adjustable from the non-invariant to the totally invariant case hereby smoothly interpolating between invariant and non-invariant pattern recognition.

5. **Applicability:** The applicability on real world problems should be possible with respect to both computational demands and good generalization performance. At

least standard benchmark datasets must be possible to treat. The methods should compete with or outperform existing approaches.

These points will be addressed and satisfied by the proposed approaches. In the following we concentrate explicitly on methods modifying the learning method by designing suitable kernel functions. Hereby we automatically obtain full generality according to point 1 above.

The article will first provide the required notions related to kernel methods and invariance in the next section. The fundamental relation of invariant kernels and invariant features is explained. The consecutive Section 3 proposes two generic methods for obtaining invariant kernel functions and comments on the basic properties. The first one is denoted *transformation integration kernels*, the second one *invariant distance substitution kernels*. Experimental applicability of the concepts are presented in Sec. 4 with various kernel methods. We comment on the real world applicability in different applications, which are extended by the present general framework. We conclude with Sec. 5.

Related Work

Some existing methods rely on finitely many explicit pattern transformations, so they lack some generality as demanded in goal 3. The *virtual support vector (VSV) method* (Schölkopf et al., 1996) is a two step method designed for a special kernel method, namely SVM (additionally restricting goal 1): First an ordinary training is performed, then the set of resulting support vectors (SVs) is extracted, which usually has a largely reduced size, this set of SVs is multiplied by application of the finitely many transformations, finally a second SVM training is performed on this extended set of samples. The advantage of training set modification is that no kernel modification is required. All standard kernels can be applied. Particularly, positive definiteness is guaranteed. This is the reason for the VSV method to be the most widely accepted method for invariances in SVMs.

Instead of performing these transformations before training, the *jittering kernels* approach performs them during kernel evaluation (DeCoste and Schölkopf, 2002). Starting with an arbitrary kernel function k, the computation of the jittered kernel $k_J(x, x')$ is done in two steps, where we denote the set of transformed patterns of x with T_x: Firstly, determine the points minimizing the distance of the sets $\Phi(T_x)$ and $\Phi(T_{x'})$ in the kernel induced feature space by explicitly performing all transformations and computing the squared distances. Secondly, take the original kernel at these minimizing points. This scheme nicely reflects the idea of operating in the kernel-induced feature space. It is, however, only well-defined for kernels with $k(x, x) = const$ like distance-based kernels. For other kernels, e.g. the linear or polynomial, the definition is not proper as occasionally multiple minima (i, i') of the distance minimization problem can occur. This additionally restricts the generality with respect to goal 2.

Another relevant study focussing on the theoretical question of invariance in kernel functions is (Burges, 1999). Under the assumption of differentiable transformations, the requirement of invariance results in partial differential equations (PDEs), which must

be solved to obtain invariant kernel functions. This is a highly nontrivial task as many integrals have to be determined. For academic examples of vectors with few entries this seems to be applicable. However, even for small sized image data, the method is far from being practically applicable, missing goal 5.

Many approaches rely on differentiable transformations (limiting target 3) and do not use the precise transformations but make use of linear approximations resulting in tangent vectors. Directly involving these tangents into arbitrary distance-based kernels results in *tangent distance kernels* (Haasdonk and Keysers, 2002), inserting into a Gaussian rbf can alternatively produce *tangent vector kernels* (Pozdnoukhov and Bengio, 2004). Further methods such as the *invariant hyperplane* (Schölkopf et al., 1998), the nonlinear extension called *invariant SVM* (Chapelle and Schölkopf, 2002), the *invariant simple SVM* (Loosli et al., 2005) or the *kernel Fisher discriminant* (Mika et al., 2000) concentrate on a specific kernel method (limiting goal 1). The majority of these modify the method's specific optimization target using the tangents of the training samples in the so called *tangent covariance matrix*. Hereby, they alter the hyperplane in such a way, that it globally fits optimally to all local invariant directions. This turns out to be equivalent to a pre-whitening in feature space along those directions which align best to all local invariance directions simultaneously. However, this pre-whitening involves (kernel) PCA and appears to be computationally very hard in the nonlinear case.

Other conceptionally appealing methods consist of sophisticated new optimization problems, which encode the sets of transformed samples. By enforcing separability constraints as in the SVM, new classifiers are produced. For instance, assuming polyhedral sets results in the approach of the *knowledge-based SVM* (Fung et al., 2004). If alternatively polynomial trajectories of the patterns are assumed, the resulting infinitely many constraints are condensed in a semi-definite programming (SDP) problem (Graepel and Herbrich, 2004). These problems are more complex, but can still be solved for small sizes, however they are problematic for real world applications as desired in goal 5.

Basically, (Schölkopf and Smola, 2002) distinguishes between methods for introducing transformation knowledge into the object representation, the training set or the learning method itself. Although this is an intuitive categorization, certain methods can be interpreted in multiple of these categories. For instance, as explained above, the *invariant hyperplane* method can be seen as modifying the object representations by a pre-whitening operation, but it was initially motivated as modifying the learning target of a SVM. In (Leen, 1995) it is argued that also the extension of the training set is in general equivalent to adding a suitable regularization term into the learning target.

2 Notions and Conceptional Relations

In this section we will introduce the notions and notation related to kernel methods and invariance. We will explain, why different types of invariance are relevant in case of kernel functions and should be distinguished. We discuss the relation of invariant kernels and traditional methods such as invariant feature extraction and template matching.

Kernel Methods

Kernel methods are meanwhile well established approaches for general machine learning tasks. We refer to (Shawe-Taylor and Cristianini, 2004; Schölkopf and Smola, 2002) and their bibliographic references for details on the notions and concepts.

The general assumption is the availability of observations or objects x stemming from some pattern space $\mathcal{X}$. If the space occasionally is a vector space, the vectorial objects will be denoted boldface $\mathbf{x} \in \mathcal{X}$. In general, kernel methods are not restricted to vectorial representations of the objects in contrast to many traditional methods. Instead a wide range of structured or unstructured data types, e.g. general discrete structures (Haussler, 1999), data-sequences (Watkins, 2000), strings (Lodhi et al., 2002), weighted automata (Cortes et al., 2003), dynamical systems (Smola et al., 2004) etc. can be processed. The main notion is that of a *kernel* function on the pattern space. We restrict ourselves to real valued kernels, as the resulting applications only require those, though complex valued definitions are possible, cf. (Berg et al., 1984).

Definition 1 (Kernel, Kernel Matrix)**.** *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which is symmetric is called a* kernel*. Given a set of observations $x_i \in \mathcal{X}$, $i = 1, \ldots, n$ the matrix $\mathbf{K} := (k(x_i, x_j))_{i,j=1}^n$ is called the* kernel matrix*.*

We emphasize that this use of the notion kernel is wider than frequently used in literature, which often requires *positive definiteness*, defined below. As we also will discuss non positive definite functions to be used, we extend the notion kernel also covering these cases.

Definition 2 (Definiteness)**.** *A kernel k is called* positive definite (pd)*, if for all n and all sets of data points $(x_i)_{i=1}^n \in \mathcal{X}^n$ the kernel matrix $\mathbf{K}$ is positive semi-definite, i.e. for all vectors $\mathbf{v} \in \mathbb{R}^n$ holds $\mathbf{v}^T\mathbf{K}\mathbf{v} \geq 0$. If this is only satisfied for those $\mathbf{v}$ with $\mathbf{1}_n^T\mathbf{v} = 0$, then k is called* conditionally positive definite (cpd)*. A kernel is* indefinite*, if a kernel matrix $\mathbf{K}$ exists, which is indefinite, i.e. vectors $\mathbf{v}$ and $\mathbf{v}'$ exist with $\mathbf{v}^T\mathbf{K}\mathbf{v} > 0$ and $\mathbf{v}'^T\mathbf{K}\mathbf{v}' < 0$.*

We denote some particular inner-product- and distance-based kernels by

$$k^{\text{lin}}(\mathbf{x}, \mathbf{x}') := \langle \mathbf{x}, \mathbf{x}' \rangle \qquad k^{\text{nd}}(\mathbf{x}, \mathbf{x}') := -\left\|\mathbf{x} - \mathbf{x}'\right\|^\beta, \beta \in [0, 2]$$
$$k^{\text{pol}}(\mathbf{x}, \mathbf{x}') := (1 + \gamma \langle \mathbf{x}, \mathbf{x}' \rangle)^p \qquad k^{\text{rbf}}(\mathbf{x}, \mathbf{x}') := e^{-\gamma\left\|\mathbf{x}-\mathbf{x}'\right\|^2}, p \in \mathbb{N}, \gamma \in \mathbb{R}_+.$$

Here, the linear k^{lin}, polynomial k^{pol} and Gaussian radial basis function (rbf) k^{rbf} are pd for the given parameter ranges. The negative distance kernel k^{nd} is cpd, which is completely sufficient for application in certain kernel methods such as support vector machines (SVMs), cf. (Berg et al., 1984; Schölkopf, 2001).

In general, a kernel method is a nonlinear data analysis method for patterns from the set $\mathcal{X}$, which is obtained by application of the *kernel trick* on a given linear method: Assume some analysis method operating on vectors $\mathbf{x}$ from some Hilbert space $\mathcal{H}$, which only accesses patterns $\mathbf{x}$ in terms of the bilinear inner product $\langle \mathbf{x}, \mathbf{x}' \rangle$. Examples of such methods are *principal component analysis (PCA)*, linear classifiers like the *large*

margin hyperplane, the *perceptron* or *Fisher linear discriminant*, but also more expressive methods like the *k-nearest-neighbour* classifier, etc. If we assume some nonlinear mapping $\Phi : \mathcal{X} \to \mathcal{H}$, the initial analysis method can be applied on the images $\Phi(x)$ as long as the inner products $\langle \Phi(x), \Phi(x') \rangle$ are available. This results in a nonlinear analysis method on the original space $\mathcal{X}$. The *kernel trick* now consists in replacing these inner products by a kernel function $k(x, x') := \langle \Phi(x), \Phi(x') \rangle$: As soon as the kernel function k is known, the Hilbert space $\mathcal{H}$ and the particular embedding Φ are no longer required. For suitable choice of kernel function k, one obtains methods, which are very expressive due to the nonlinearity and cheap to compute, as explicit embeddings are omitted. The resulting methods are the meanwhile well known *kernel principal component analysis*, *support vector machine*, *kernel perceptron*, *kernel Fisher discriminant*, etc.

The question, whether a given kernel allows a representation in a Hilbert space as $k(x, x') = \langle \Phi(x), \Phi(x') \rangle$ is interestingly completely characterized by the positive definiteness of the kernel. Various methods of explicit feature space construction can be given. Theoretically most relevant is the embedding in a so called *reproducing kernel Hilbert space (RKHS)*, which enables the embedding of the whole space $\mathcal{X}$ into a Hilbert space of functions, cf. (Schölkopf and Smola, 2002, Sec. 2.2.2) for details.

Definition 3 (Reproducing Kernel Hilbert Space Embedding)**.** *For any given positive definite kernel k on the nonempty set $\mathcal{X}$ the mapping $\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}$ with $\Phi(x) := k(x, \cdot)$ defines a pre-Hilbert space $\mathcal{H}_0 := \operatorname{span}(\Phi(\mathcal{X}))$ with inner product*

$$\left\langle \sum_i a_i k(x_i, \cdot), \sum_j b_j k(x'_j, \cdot) \right\rangle := \sum_{i,j} a_i b_j k(x_i, x'_j).$$

The completion of this space with respect to the induced norm yields a so called reproducing kernel Hilbert space (RKHS) $\mathcal{H}$ *with* reproducing kernel k*.*

The notion *reproducing kernel* is motivated as k satisfies the so called *reproducing property* $\langle k(x, \cdot), f \rangle = f(x)$ for any $f \in \mathcal{H}$. For the embedded points $\Phi(x)$ this results in $\langle \Phi(x), \Phi(x') \rangle = \langle k(x, \cdot), k(x', \cdot) \rangle = k(x, x')$ for all x, x'.

By the kernel trick, various kernel methods have been developed and successfully applied during the last decade. This enables a wide variety of analysis and learning algorithms, once a suitable kernel is chosen for the data. This modularity of choice of kernel function and choice of learning method is a major feature of kernel methods. Additionally, this emphasizes the importance of the kernel choice: The only view on the data, which the analysis algorithm obtains, is the kernel matrix $\mathbf{K}$, i.e. the kernel evaluated on all pairs of input objects. Therefore, the kernel matrix is very reasonably denoted an *information bottleneck* in any such analysis system (Shawe-Taylor and Cristianini, 2004). Hence, a good solution for any learning task will require a well designed kernel function based on the available problem specific prior knowledge.

Variants of Invariance

The prior knowledge that we assume is a structural assumption on the generation of the patterns. We restrict to the knowledge of a set of *transformed patterns* for each sample with the assumption that these patterns have equal or similar meaning as the original pattern itself. So, replacing an individual point by one of its transformed patterns should keep the output of the analysis or learning task roughly unchanged. As argued in the preceding section under goal 3, these transformations can be of different kind, which are all covered by the following formalization:

Definition 4 (Transformation Knowledge)**.** *We assume to have a set T of* transformations $t : \mathcal{X} \to \mathcal{X}$ *including the identity mapping which define a set of* transformed patterns $T_x := \{t(x) | t \in T\} \subset \mathcal{X}$ *for any x. These patterns are assumed to have identical or similar inherent meaning as the pattern x itself.*

At this point we do not put any further assumptions on T. In particular, we do not assume an explicit parameterization of these sets, nor assume that they are finite. We neither require specific relations between the $T_x, T_{x'}$ of different patterns. They may be disjoint, may be equal or may intersect. Of course for computationally dealing with these sets, one must assume enumerability of the sets, characterization of the sets by constraints or explicit parameterization of the transformations.

A traditional way in pattern analysis for involving such transformation knowledge is so called *template matching*, which means that all patterns are explicitly transformed and the best fitting match between two transformed objects is used in the analysis task. Another method omitting these explicit transformations is to perform a pre-processing step by mapping the objects into an invariant representation in some real vector space $\mathcal{H}$. Instead of working on the original patterns, the transformed samples are taken as a basis for investigation. This step is called *feature extraction*.

Definition 5 (Invariant Function, Single Argument)**.** *We call a function $f : \mathcal{X} \to \mathcal{H}$* invariant *with respect to T, if for all patterns x and all transformations $t \in T$ holds $f(x) = f(t(x))$. The vector $f(x)$ is then called an* invariant representation *or* invariant feature vector *of x.*

To emphasize the invariance of an arbitrary function $f(x)$, we will occasionally denote it $I(x)$. In traditional invariant pattern recognition, this notion is used for transformation groups $T = G$, cf. (Schur, 1968; Schulz-Mirbach, 1995; Wood, 1996). In this case the pattern space $\mathcal{X}$ is nicely partitioned into equivalence classes T_x, which correspond to the orbits of the patterns under the group action. Then, invariants are exactly those functions, which are constant on each equivalence class. Various methods for constructing such invariant features are known, e.g. normalization approaches like moments (Canterakis, 1999), averaging methods (Schulz-Mirbach, 1994) or differential approaches. For a general overview of invariance in pattern recognition and computer vision we refer to (Burkhardt and Siggelkow, 2001; Mundy et al., 1994).

Learning targets can often be modelled as functions of several input objects, for instance depending on the training data and the data for which predictions are required.

For such functions different notions of invariance are used in literature, each with its own practical relevance. Thus, a more distinguishing treatment of the notion *invariance* is required in particular for kernel methods. Therefore, we introduce discriminating extensions of Def. 5, which will be used throughout the presentation.

Definition 6 (Invariant Function, Several Arguments)**.** *We call a function* $f : \mathcal{X}^n \to \mathcal{H}$

i) simultaneously invariant *with respect to* T*, if for all patterns* $x_1, \ldots, x_n \in \mathcal{X}$ *and transformations* $t \in T$ *holds*

$$f(x_1, \ldots, x_n) = f(t(x_1), \ldots, t(x_n)).$$

ii) totally invariant *with respect to* T*, if for all patterns* $x_1, \ldots, x_n \in \mathcal{X}$ *and transformations* $t_1, \ldots, t_n \in T$ *holds*

$$f(x_1, \ldots, x_n) = f(t_1(x_1), \ldots, t_n(x_n)).$$

Obviously, for the case of a function with a single argument, both definitions correspond to the *invariance* according to Def. 5. The first notion *i)* is used in (Schur, 1968) for polynomial functions under group transformations. In general, this is a common understanding of invariance. The function does not change if the whole space $\mathcal{X}$ is globally transformed, i.e. all inputs are transformed *simultaneously* with an identical transformation. For example, the Euclidean distance is called *translation invariant*, the standard inner product *rotation invariant* (Veltkamp, 2001; Schölkopf and Smola, 2002). From a practical viewpoint, this type of invariance is useful, as it guarantees that the function is independent of the global constellation of the data. By this it is unaffected, e.g. by changes of the experimental setup: A simultaneously translation-invariant system can operate on data without pre-processing like centering. A simultaneously scale-invariant system will produce the same output on differently scaled datasets, making a uniform scale-normalization superfluous, etc. So, these transformations can be ignored in the consecutive analysis chain. Some more studies investigate the behaviour of an SVM concerning uniform global transformations. For instance, (Joachims, 1999, Lemma 2) states that the SVM solution is invariant with respect to global addition of real values c to the kernel function. The simultaneous rotation invariance of the Euclidean inner product and the additional translation invariance of the induced distance transfers to similar transformation behaviour for resulting SVM solutions with various distance or inner product kernels, cf. (Abe, 2003; Sahbi and Fleuret, 2002).

This notion of simultaneous invariance, however, does not capture the transformation knowledge as given in Def. 4: It only guarantees to remain constant under *global* transformation of the whole input space. However, if we only translate/rotate one of the several patterns, the Euclidean distance and the inner product will in general change. Therefore, we introduce the notion *ii)* of *total invariance* to denote functions, which are guaranteed to maintain their value, if any single argument is (or equivalently all simultaneously are) transformed independently. Note that this is equivalent to the statement

that they are *invariant* as functions of one argument fixing the remaining ones arbitrarily. The *total invariance ii)* implies the *simultaneous invariance i)*. Further variations of *invariance* exist in invariant theory, such as *relative* versus *absolute* invariance, *covariance*, *semi*-invariance, etc. (Schur, 1968). These notions, however, are not relevant in the sequel.

Note that the requirement of precise invariance is frequently too strict for practical problems. The points within T_x are sometimes not to be regarded as identical to x, but only as similar, where the similarity can even vary over T_x. Such approximate invariance is called *transformation tolerance* (Wood, 1996), *probabilistic invariance* (Lenz, 1991), *quasi-invariance* (Binford and Levitt, 1993) or denoted *additive invariance* (Burges, 1999). A well-known and intuitive example is optical character recognition (OCR): The sets T_x of 'similar' patterns might be defined as rotations of the pattern x under small rotation angles. Exact invariance is not wanted with respect to these transformations. An invariant function f will not only be constant on the set of small rotations of a pattern, but by transitivity it must be constant for all rotations. This results in the characters M/W, Z/N, 6/9 etc. not being discriminable.

We want to cover both the exact invariance and these *transformation tolerant* cases. Therefore, in all cases where exact invariance is wanted and the sets T_x form a partition of $\mathcal{X}$, the proposed tools will result in totally invariant functions. Additionally, they will allow to relax the strict invariance if required, such that they smoothly can interpolate to the non-invariant case, where a given base-kernel is reproduced. Hereby the invariance degree can be adjusted.

Relation of Invariant Kernels and Features

Conceptionally more interesting than relating different invariant kernel types is the relation of invariant kernel functions to traditional elements of invariant pattern analysis. Specifically, we address the relation to invariant features and template matching. The main fundamental insight is that totally invariant kernels and invariant representation have a direct correspondence.

Proposition 7 (Correspondence of Invariant Features and Kernels)**.**
Let T be a set of transformations on the space $\mathcal{X}$ including the identity.

i) Any invariant function $I(x)$ induces a positive definite and totally invariant kernel by the standard inner product $k(x, x') := \langle I(x), I(x') \rangle$.

ii) For every positive definite and totally invariant kernel there exists an invariant function I mapping to some Hilbert space $\mathcal{H}$ such that k is the inner product between transformed patterns, i.e. $k(x, x') = \langle I(x), I(x') \rangle$ for all $x, x' \in \mathcal{X}$.

Proof. Part i) is trivially satisfied, as the inner product in a Hilbert space is a symmetric positive definite function and the resulting k is totally invariant by invariance of I:

$$k(t(x), t'(x')) = \langle I(t(x)), I(t'(x')) \rangle = \langle I(x), I(x') \rangle = k(x, x').$$

Part ii) is a consequence of the existence of a RKHS $\mathcal{H}$ for k with corresponding mapping Φ according to Def. 3. As k is assumed to be totally invariant and T includes the identity, we have $k(t(x), \mathrm{id}(x')) - k(x, x') = 0$ which implies $\langle \Phi(t(x)) - \Phi(x), \Phi(x') \rangle = 0$ for all x'. As every $\mathbf{x} \in \mathcal{H}$ is the limit of a sequence in $\mathrm{span}(\Phi(\mathcal{X}))$ we obtain $\langle \Phi(t(x)) - \Phi(x), \mathbf{x} \rangle = 0$ for all $\mathbf{x}$. This implies $\Phi(t(x)) = \Phi(x)$, so $I(x) := \Phi(x)$ is the wanted invariant transformation. □

So, invariant kernels and invariant representations are basically different views on identical concepts. However, the *kernel trick* is the reason, why invariant kernels can be practically more general and advantageous: Invariant kernels do not require to explicitly compute the invariant feature representations. If the invariant feature representation is very high or even infinite dimensional, which prevents practically accessing these spaces, the corresponding invariant kernel can enable to work with these spaces, despite of the dimensionality. In particular some instances of the *transformation integration kernels* described in the next section are an effective mean to operate on theoretically ideal *complete* sets of invariants, which have up to now not been accessible for real applications due to their high dimensional feature space.

A second conceptional relation of invariant kernels to invariant features and template matching can be found and is illustrated in Fig. 1. The basic operation of pairwise comparisons of objects with either of these methods can be divided into a stage of sample-wise precomputations and a stage where the precomputed quantities are combined to obtain the final pairwise similarity measure, i.e. the kernel evaluation. The main difference between the approaches can be seen in the distribution of the computational load indicated by the area of the blue rectangles. In the left it can be seen, that the main contribution is the expensive computation of invariant features, whereas the kernel evaluation is quite cheap by usually few arithmetic operations in the simple standard kernels. In case of template matching on the right, merely few precomputations are possible, the computational load is concentrated on the evaluation of the kernel, which involves matching all transformed patterns against each other. Invariant kernels between those extremes allow some sample-wise precomputations of (possibly non-invariant) quantities and require some more than trivial computations for kernel evaluation than the simple kernels. The main statement of this comparison is, that totally invariant kernels nicely interpolate between the well-known methods of invariant features applied in ordinary kernels and the other extreme of template matching. Totally invariant kernels are therefore conceptionally covering but also extending these traditional methods.

3 Constructing Invariant Kernels

After these conceptional considerations, we deal with applicational aspects of invariant kernels for the remainder of the presentation. This section proposes two generic methods for producing invariant kernels from a multitude of given base-kernels. This satisfies goal 2 stated in the introductory section. We discuss their properties, in particular the

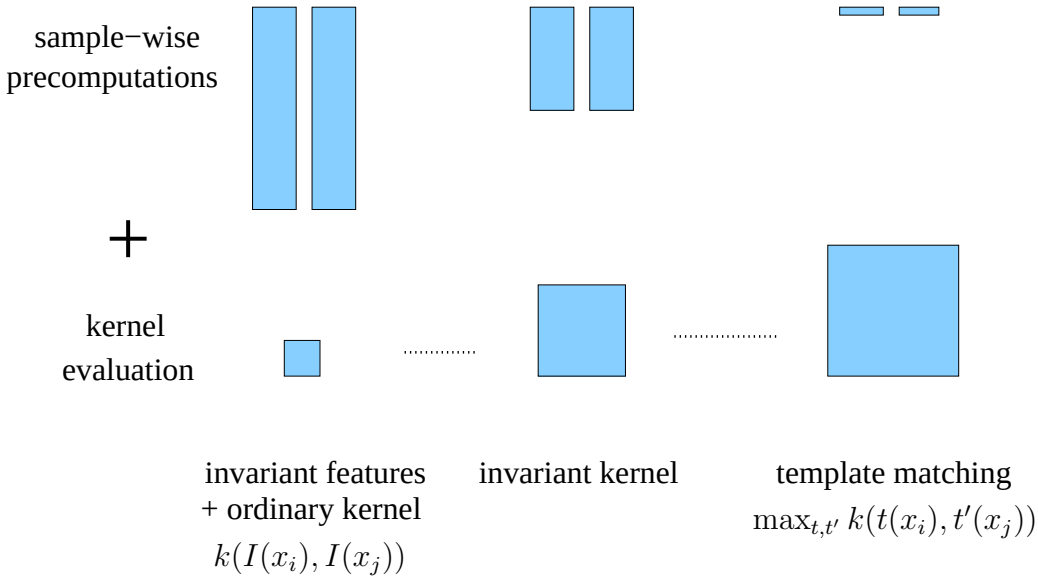

Figure 1: Qualitative complexity comparison of totally invariant kernels covering the whole spectrum from invariant feature extraction to template matching. The computational weight is shifted from sample-wise precomputations to the kernel evaluation.

adjustable invariance with respect to a multitude of transformations aiming at goals 3 and 4.

Transformation Integration Kernels

The motivation for the first invariant kernel construction method stems from a technique which produces invariant features, the so called Haar-integral invariants or features. These are invariant functions, which can be derived by an averaging procedure (Schulz-Mirbach, 1994, 1995). In this approach, the transformations T are assumed to be structured as a group G with invariant measure dg, the so called Haar-measure, which exists for locally compact topological groups and finite groups, cf. (Nachbin, 1965). Based on this, invariant feature representations of patterns are generated by integrating simple non-invariant functions h over the known transformation group (Schulz-Mirbach, 1994). This results in *Haar-integral invariants* defined as

$$f(x) = \int_G h(g(x))dg, \tag{1}$$

where h is chosen such that the integral exists and is finite for all x. It is intuitively clear and can be rigorously proven (Nachbin, 1965, p. 64) that $f(x)$ is an invariant function in the sense of Def. 5. Extending this concept of integration over transformations to kernel functions is the main motivating idea for the *transformation integration kernels*. This results in kernel functions, which are positive definite, have arbitrarily wide adjustable

invariance and can capture simultaneously various continuous or discrete transformations as desired. The following is a slight generalization of the presentation in (Haasdonk et al., 2005), here we omit the requirement of having a transformation group.

Definition 8 (TI-Kernels)**.** *Let T be a set of transformations operating on the set $\mathcal{X}$ with measure dt. Let k be a kernel on $\mathcal{X}$ such that for all x, x'*

$$k_{TI}(x, x') = \int_T \int_T k(t(x), t'(x'))dtdt' \tag{2}$$

exists and is finite. We denote this function the transformation integration kernel (TI-kernel) *of k with respect to T.*

The requirement of the integrability of k is practically mostly satisfied, e.g. after finite discretization of T. But the definition also covers infinite cases. For instance, a general assumption is to have transformations $t(\mathbf{x}, \mathbf{p})$, where the patterns stem from some real vector space and can be parameterized by a real parameter vector $\mathbf{p}$ from some compact set. If t and k are continuous and dt is the Lebesgue measure, the integrand is bounded and continuous for all $\mathbf{x}, \mathbf{x}'$, so the integral exists and is finite. If t and k are polynomial, the integrals can even be analytically solved with integration by parts.

The motivation of the integral (2) is demonstrated in Fig. 2 in two ways: a) in the original pattern space $\mathcal{X}$ and b) in a k-induced feature space $\mathcal{H}$. For simplicity, we assume the measure to be normalized to $dt(T) = 1$. In the left plot, two patterns x, x' are illustrated in the pattern space $\mathcal{X}$ with their sets of transformed patterns T_x and $T_{x'}$ indicated by rectangles. The TI-kernel generated by k is the kernel average over all pairwise combinations of T_x and $T_{x'}$.

In the right sketch b), the interpretation of the kernel in the feature space $\mathcal{H}$ is given: Instead of averaging over $k(x, x')$, the integration kernel is the inner product of the average (indicated by a bar) of the sets $\Phi(T_{x'})$, $\Phi(T_x)$, respectively, due to

$$\begin{aligned} \left\langle \int_T \Phi(t(x))dt, \int_T \Phi(t'(x'))dt' \right\rangle &= \int_T \int_T \langle \Phi(t(x)), \Phi(t'(x')) \rangle \, dtdt' \\ &= \int_T \int_T k(t(x), t'(x'))dtdt' = k_{TI}(x, x'). \end{aligned} \tag{3}$$

Interestingly, these kernels turn out to be very convenient, as definiteness properties of the base kernel k are transferred. In particular, the TI-kernels are (c)pd if the base kernel is (c)pd. The argumentation is identical to the case of group transformations (Haasdonk et al., 2005).

For these kernels, the practical gain by the kernel trick as mentioned in the previous section can be demonstrated. It is known, that for finite transformation groups G with n_T elements *complete* sets of Haar-integral invariants can be generated. This is obtained by using all monomials up to degree n_T as functions h in (1) (Schulz-Mirbach, 1995, Satz 3.5). Until now, this result was mainly a theoretical upper bound, as computation of these exponentially many features is infeasible. It can be shown, that the TI-kernel of

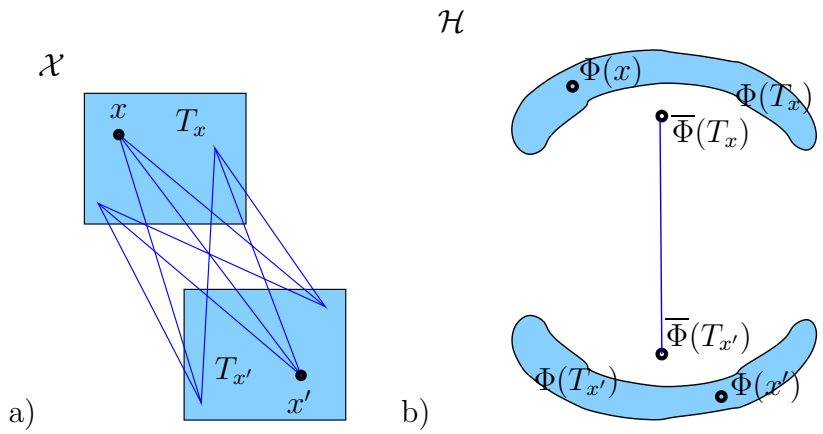

Figure 2: Geometric interpretation of TI-kernels. a) Original pattern space $\mathcal{X}$, b) kernel-induced feature space $\Phi(\mathcal{X}) \subset \mathcal{H}$.

$k(\mathbf{x}, \mathbf{x}') = (\langle \mathbf{x}, \mathbf{x}' \rangle + 1)^{n_T}$ exactly is operating on this complete set of invariant features. Therefore, the TI-kernels are indeed an efficient way to implicitly operate with high dimensional vectors of invariant features. For details on this argumentation, we refer to (Haasdonk, 2005a, Sec. 5.6).

The integration kernels allow various ways of time complexity reduction. As argued before, suitable caching strategies can accelerate the computation procedure, e.g. caching the transformed patterns throughout the computation of the kernel matrix. Additionally, various transformations allow the reduction of the double integral to a single integral, which is a reduction of the complexity by one square-root (Haasdonk et al., 2005).

Invariant Distance Substitution Kernels

A second generic method for constructing totally invariant kernels can be obtained by involving invariant distances. Assuming some distance function d on the space of patterns $\mathcal{X}$ enables to incorporate the invariance knowledge given by the transformations T into a new dissimilarity measure. Hereby, for a *distance* function we only require symmetry, non-negativity and zero-diagonal. So in particular the triangle inequality does not need to be valid. The invariant distance computation is then based on minimizing the distance between the sets of transformed samples. Similar formalizations of such distances are widely available (Vasconcelos and Lippman, 1998; Simard et al., 1998; Keysers et al., 2000). In particular this notion of invariant distance covers many specific examples in literature. In particular *tangent distance (TD)* (Simard et al., 1993), the *image distortion model (IDM)* (Keysers et al., 2000), general deformation models (Keysers et al., 2004), *dynamic time warping (DTW)* (Rabiner and Juang, 1993), *Fréchet* distance (Alt and Guibas, 1999), invariant distances between point sets (Werman and Weinshall, 1995) or *two-sided manifold distance* (Fitzgibbon and Zisserman, 2003).

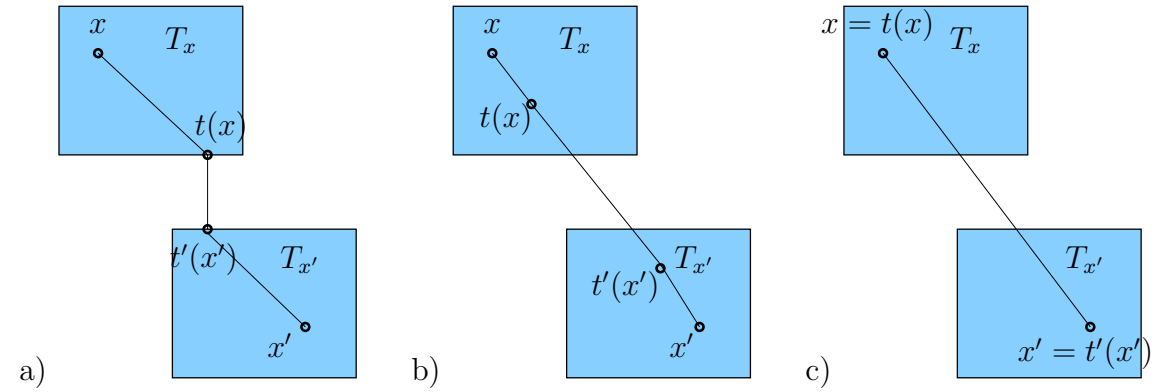

Figure 3: Illustration of the invariant distance with varying λ. a) Total invariance by $\lambda = 0$, b) approximate invariance by $\lambda > 0$, c) no invariance by $\lambda = \infty$.

Definition 9 (Invariant Distances)**.** *For a given distance d on the set $\mathcal{X}$ and some cost function $\Omega : T \times T \to \mathbb{R}_+$ with $\Omega(t, t') = 0 \Leftrightarrow t = t' = \mathrm{id}$, we define the* two-sided invariant distance *as*

$$d_{2S}(x, x') := \inf_{t,t' \in T} d(t(x), t'(x')) + \lambda \Omega(t, t'). \tag{4}$$

Note, that the resulting distance measure again is guaranteed to be symmetric, non-negative and will have a zero-diagonal.

The motivation for this formalization is illustrated in Fig. 3. We again see two patterns x, x' and their corresponding sets of transformed patterns as blue rectangles. We additionally plot the transformed samples $t(x)$ and $t'(x')$ which are obtained from the minimization process in (4) for different values of λ. If the points within each set are assumed to have identical meaning, a dissimilarity measure should incorporate this by rather measuring the distance of the sets $T_x, T_{x'}$ than the original point-distance. The distance of point-sets is usually defined by taking the infimum over distances of point-pairs, which results in the first term of (4), or equivalently, setting $\lambda = 0$ as indicated in plot a).

However, in general, not all points in the set of transformed patterns T_x have exact identical meaning, the likeliness of $t(x)$ to x will decrease with the extent of the transformation. This 'unlikeliness' of large transformations can be modelled by a nonnegative cost term $\Omega(t, t')$ in (4). By increasing λ, unlikely large transformations are penalized and a larger distance value is allowed as indicated in plot b). For $\lambda \to \infty$ we obtain the case c), where the original distance measure is reproduced.

With the notion of invariant distance we can define the *invariant distance substitution kernels* as follows:

Definition 10 (IDS-Kernels)**.** *For any* distance-based kernel *k, i.e. $k(\|\mathbf{x} - \mathbf{x}'\|)$, and invariant distance measure d_{2S} we call $k_{IDS}(x, x') := k(d_{2S}(x, x'))$ its* invariant distance substitution kernel *(IDS-kernel). Similarly, for an* inner-product-based *kernel k, i.e.*

$k(\langle \mathbf{x}, \mathbf{x}' \rangle)$, *we call* $k_d(x, x') := k(\langle x, x' \rangle^O)$ *its IDS-kernel, where* $O \in \mathcal{X}$ *is an arbitrary* origin *and a generalization of the inner product is given by the function*

$$\langle x, x' \rangle^O := -\frac{1}{2}(d_{2S}(x, x')^2 - d_{2S}(x, O)^2 - d_{2S}(x', O)^2). \tag{5}$$

This notation for the inner product reflects the fact that in case of d_{2S} being the L^2-norm in a Hilbert space $\mathcal{X}$, $\langle x, x' \rangle^O$ corresponds to the inner product in this space with respect to the origin O. Note that this is only a formal definition, usual properties like bilinearity are not defined. Still a suitable interpretation as an inner product exists after suitable embeddings into pseudo-Euclidean spaces, cf. (Haasdonk, 2005a).

So the notion *distance substitution* in case of inner-product-based kernels is also reasonable as indeed distances are substituted. Note that these kernels are well-defined for these inner-product-based cases in contrast to the jittering kernels, cf. the comments in the introductory section. In particular, for the simple linear, negative-distance, polynomial and Gaussian kernels the IDS kernels are well-defined. Of course, more general distance- or dot-product-based kernels exist and corresponding IDS-kernels can be constructed, e.g. sigmoid, multiquadric, B_n-spline, exponential, rational quadric, Mátern kernels (Schölkopf and Smola, 2002; Genton, 2001), etc.

Frequently, it can be difficult to construct all possible variations T_x of an object x by explicit transformations, e.g. if the sets T_x are infinite or the required number of transformation parameters is high. Practical approaches for computation therefore require further structural assumptions on the sets T_x in order not to perform the transformations explicitly. This is the most important point for practically employing these kernels, as immense complexity reductions compared to matching are possible. For instance, if some recursive formulation of the transformations in T can be defined, methods like dynamic programming can be applied to implicitly operate on the exponentially many transformations. An example of this is the DTW distance (Rabiner and Juang, 1993; Bahlmann et al., 2002), which is a totally invariant distance measure in our terms and applied in online handwriting recognition (Bahlmann et al., 2002). This distance measure can be determined in a complexity, which scales quadratically with the length of the point sequences in contrast to an exponentially growing complexity for matching. If in other cases the set T is composed of exponentially many combinations of transformations, which operate locally on independent parts of the pattern x, efficient computation can be performed by sequentially addressing the different object parts and their local transformations. An example for this is the IDM (Keysers et al., 2000) for images, which can be evaluated in complexity growing linear in the number of pixels. If in other cases the assumption of linear representation of the sets T_x holds, projection methods can be used to perform the exact minimization over infinitely many transformations very efficiently. For instance, the TD-kernels (Haasdonk and Keysers, 2002) can be evaluated in a complexity growing only polynomial with the dimension of the transformation set. An additional computational benefit of the IDS-kernels is the possibility to precompute the distance matrices. By this, the final kernel evaluation is very cheap and ordinary fast model selection can be performed.

An important property worth noting is, that the IDS-kernels in general are indefinite, even if the base-kernel was pd. This results from the fact, that the invariant distances frequently are not Hilbertian metrics, e.g. they violate the triangle inequality. This should be kept in mind, if they are involved in learning algorithms. For certain kernel methods this is no problem, e.g. kernel principal component analysis can work with both pd or indefinite kernels, the SVM is known to tolerate indefinite kernels (Haasdonk, 2005a) and further kernel methods are developed, which can work with such kernels, (Ong et al., 2004).

We want to conclude this section with mentioning a relation to an existing method. It can be shown that the IDS-kernels for $\lambda = 0$ generalize the jittering kernels in case of finite T and distance-based invertible functions $k(\|\cdot\|)$, which monotonically decrease with $\|\cdot\|$. This particularly comprises k^{rbf} and k^{nd}. Still, the IDS-kernels are more general: They allow for regularization, they can capture infinitely many transformed patterns, by suitable distance measures, which is not possible by explicit discrete jittering, and they naturally extend to linear/polynomial kernels, where the jittering kernels are not well defined.

Adjustable Invariance

The IDS and TI-kernels are conceptionally an elegant seamless connection between non-invariant and invariant data-analysis: The size of T can be adjusted from the non-invariant case $T = \{\mathrm{id}\}$, which recovers the base kernel, to the fully invariant case, where T equals a transformation group $T = G$. This can both be stated theoretically and demonstrated experimentally.

Proposition 11 (Adjustable Invariance of TI-Kernels)**.**

i) If $T = \{\mathrm{id}\}$ and dt satisfies $dt(T) = 1$, then $k_{TI} = k$.

ii) If T is a transformation group and the measure dt is the invariant Haar-measure then the resulting integration kernel k_{TI} is totally invariant.

Proof. Statement *i)* is obvious. For *ii)* it is sufficient to argue, that variation of a single argument maintains the value of k_{TI}. This is true, as the TI-kernels are Haar-integral features if $T = G$ for some group G, dt is the corresponding Haar-measure dg and one argument is fixed:

$$k_{TI}(x, x') = \int_G h(g(x))dg \quad \text{with} \quad h(x) := \int_G k(x, g'(x'))dg'.$$

For such Haar-integral features, the invariance is known, cf. (Nachbin, 1965, p. 64), which particularly implies the desired one-sided invariance $k_{TI}(g(x), x') = k_{TI}(x, x')$. □

Similar statements can be made for the IDS-kernels, these however offer two possibilities for controlling the invariance extend and hereby interpolating between the invariant

and non-invariant case. Firstly, the size of T can be adjusted, secondly, the regularization parameter λ can be increased to reduce the invariance. The proof of the following is omitted, as it can easily be obtained from the definitions.

Proposition 12 (Adjustable Invariance of IDS-Kernels)**.**

i) If $T = \{\mathrm{id}\}$ *and* d *is the ordinary Euclidean distance, then* $k_{IDS} = k$*.*

ii) If d *is the ordinary Euclidean distance, then* $\lim_{\lambda\to\infty} k_{IDS} = k$*.*

iii) If $\lambda = 0$ *and* T *is a transformation group, then* k_{IDS} *is totally invariant.*

We give simple illustrations of the proposed kernels. From the multitude of possible distance and inner-product-based kernels we restrict to well known representatives, the linear k^{lin} and Gaussian k^{rbf} kernel as other kernels like the polynomial, exponentials, etc. behave very similarly. We treat the TI- and IDS-kernels in parallel to emphasize and comment on similarities and differences. For the illustrations, our objects are simply points in two dimensions and several transformations of the points define sets of points to be regarded as similar. These transformations will comprise discrete, continuous, linear, nonlinear, highly nonlinear transformations and combinations thereof. This demonstrates the achievement of goal 3. All plots generated in the sequel can be reproduced by the MATLAB library *KerMet-Tools* (Haasdonk, 2005b), which is designed for working with invariant kernels in different kernel methods. This 2D scenario is not only an academic problem setting, but might be relevant in image-based applications: For instance, interest point extractors are fundamental ingredient in many image analysis systems. These produce two-dimensional points which are subject to the same geometric coordinate-transformations as the object displayed in the image.

We start with the total invariance of all proposed kernels in both discrete and continuous cases. We fix one argument $\mathbf{x}'$ (denoted with a black dot) of the kernel, and the other argument $\mathbf{x}$ is varying over the square $[-1, 2]^2$ in the Euclidean plane. We plot the different resulting kernel values $k(\mathbf{x}, \mathbf{x}')$ color coded. The upper row of Fig. 4 illustrates the non-invariant linear kernel in a) and the corresponding y-axis-reflection invariant versions, the IDS-kernel in b) and the TI-kernel in c). The invariance is clearly obtained, though being qualitatively different. The lower row demonstrates that the invariance is also obtained for continuous transformations, here the rotation. The non-invariant Gaussian in d) is made invariant by its IDS- and TI-kernels in subplot e) and f), respectively. The qualitative difference is, that the circle with maximum values of the IDS-kernel precisely captures the set of rotated patterns $T_{\mathbf{x}}$, whereas the circle with maximum values of the TI-kernel is slightly smaller. Overall, total invariance is perfectly obtained in all cases.

In addition to the total invariance, we claimed that the invariance can be adjusted to approximate invariance. We will demonstrate this in the remainder of the section. We focus on a linear shift along a certain slant direction while increasing the transformation extent. Figure 5 a) demonstrates the behaviour of the linear IDS-kernel, which perfectly aligns to the transformation direction. Interestingly, the TI-kernel of the linear kernel

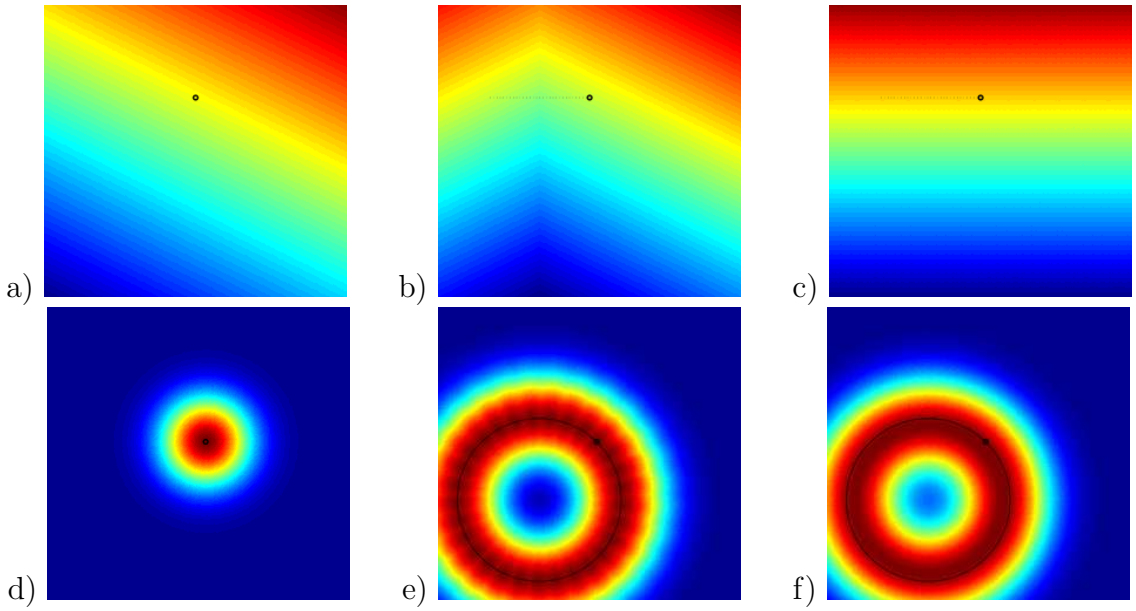

Figure 4: Total invariance with respect to discrete and continuous transformations of inner-product- and distance-based kernels. a) Non-invariant linear kernel k^{lin}, b) reflection-invariant kernel k^{lin}_{IDS}, c) reflection-invariant kernel k^{lin}_{TI}, d) Non-invariant Gaussian k^{rbf}, e) rotation-invariant kernel k^{rbf}_{IDS}, f) rotation-invariant kernel k^{rbf}_{TI}.

cannot capture invariance with respect to this highly symmetric linear transformation, illustrated in b). This can easily be explained by the feature-space interpretation in Fig. 2 b) as the mean of T_x always is identical to x. This problem does not appear for non-symmetric transformations. To demonstrate this, we choose a highly nonlinear transformation of the points, namely the shift along a sine curve in Fig. 6. It is obvious how both the IDS-kernel a) and the TI-kernel b) of the linear kernel are completely nonlinear due to the construction rules. In both cases, the invariance along the sine curve can be smoothly adjusted by increasing T from left to right. Similar behaviour is observed for other inner-product-based kernels.

Corresponding illustrations of the distance based kernel k^{rbf} are given in Fig. 7 where the adjustability of the IDS-kernel a) and TI-kernel b) are illustrated. It is striking, that the captured transformation range is much larger and more accurate for the IDS-kernels. In the nonlinear transformation case of Fig. 8 the situation is very similar. Again the maximum values of those kernels perfectly fit along $T_{\mathbf{x}}$ and even beyond until precisely twice the range of T_x.

The IDS-kernels have a second means for controlling the invariance extend, namely increasing the regularization parameter λ. This gives similar results as decreasing the transformation extent, but the λ-variation is also applicable for discrete transformations such as reflections, cf. Fig. 9. The figure additionally demonstrates the invariance with respect to combined transformations.

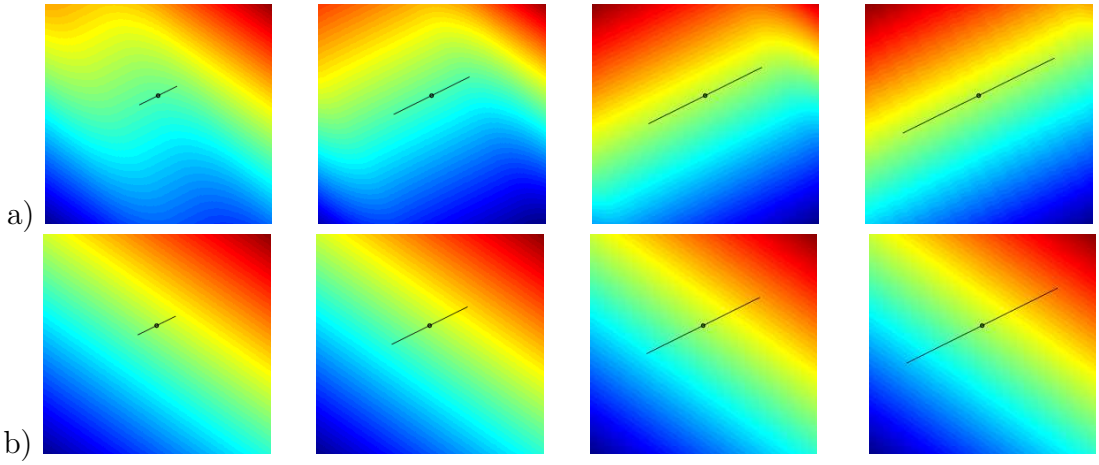

Figure 5: Adjustable invariance of linear kernel k^{lin} with respect to linear transformation. a) IDS-kernels with increasing transformation extent, b) TI-kernels.

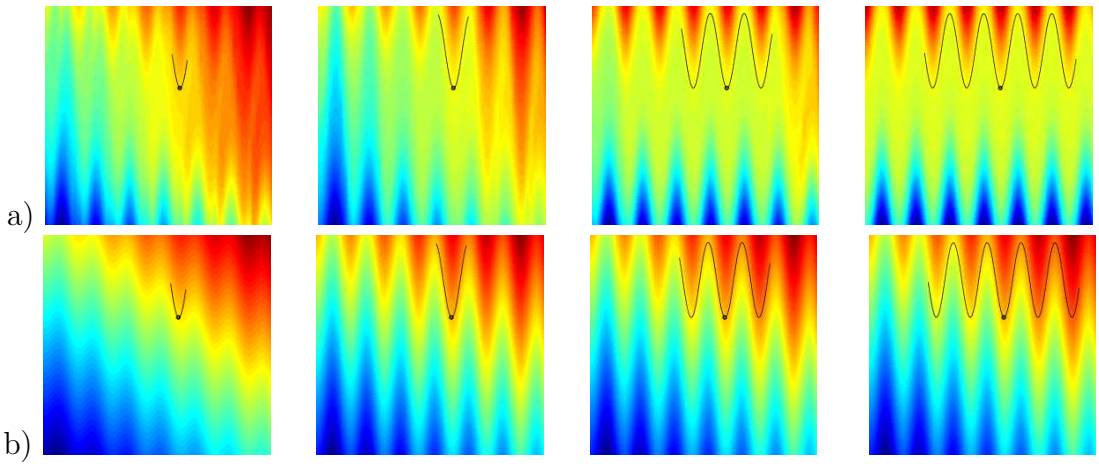

Figure 6: Adjustable invariance of linear kernel k^{lin} with respect to highly nonlinear transformation. a) IDS-kernels with increasing transformation extent, b) TI-kernels.

Comment on Simultaneous Invariance

Due to our definition of transformation knowledge, we are mostly interested in *totally invariant* kernels. As noted earlier, other notions like *simultaneous invariance* can also make sense, if a data normalization step is to be prevented. As total invariance implies simultaneous invariance, the presented kernels are similarly simultaneous invariant. But also some simple modifications of the TI- and IDS-kernels can be defined, which result in more general simultaneous invariant kernels.

Proposition 13 (Simultaneous Invariant Modifications)**.**

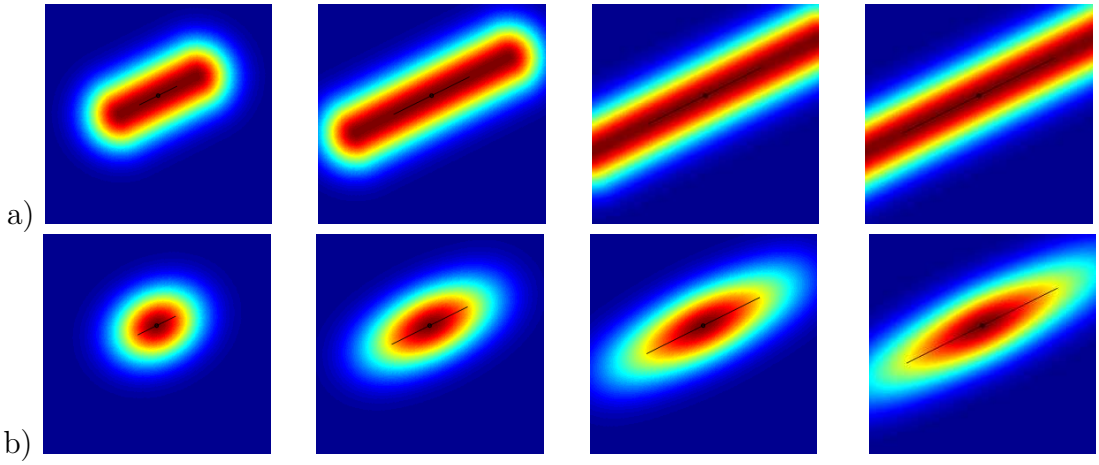

Figure 7: Adjustable invariance of Gaussian kernel k^{rbf} with respect to linear transformation. a) IDS-kernels with increasing transformation extent, b) TI-kernels.

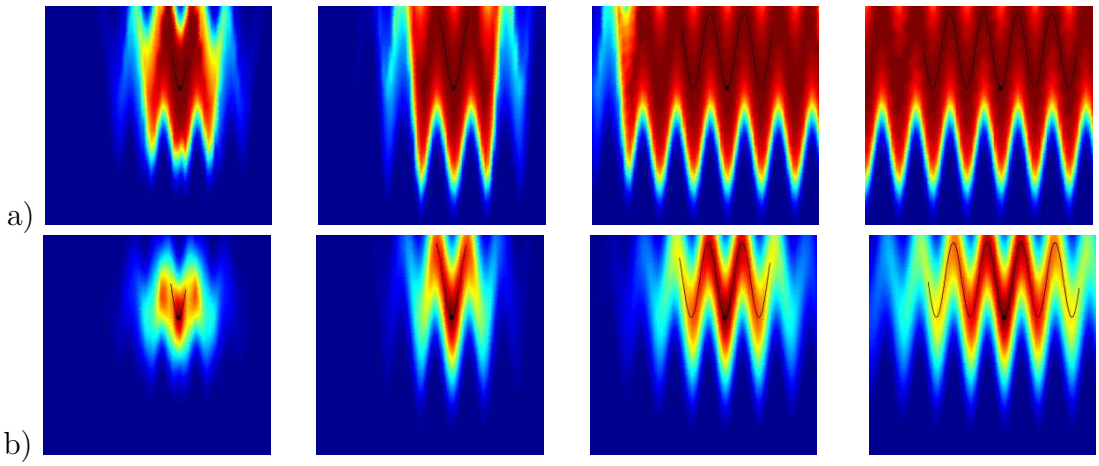

Figure 8: Adjustable invariance of Gaussian kernel k^{rbf} with respect to highly nonlinear transformation. a) IDS-kernels with increasing transformation extent, b) TI-kernels.

Let us define alternatively to (2) and (4)

$$k_{TI}(x,x') = \int_T k(t(x),t(x'))dt \quad \textit{and} \quad d_{2S}(x,x') := \inf_{t\in T} d(t(x),t(x')) + \lambda\Omega(t,t).$$

If we again assume that T is a transformation group

i) the kernel k_{TI} is simultaneously invariant.

ii) for all distance-based kernels k, the IDS-kernel k_{IDS} is simultaneously invariant.

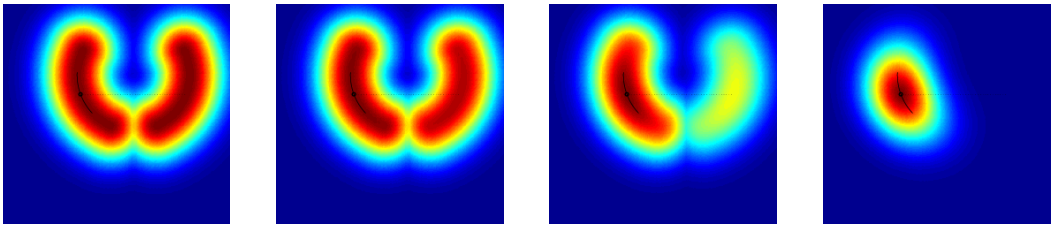

Figure 9: Adjustable invariance of Gaussian kernel k^{rbf}_{IDS} with respect to simultaneous discrete and continuous transformation by increasing regularization parameter λ.

Note that the *simultaneous* invariance in general does not transfer to inner-product-based IDS-kernels. From now on, we only regard totally invariant kernels and their adjustable invariant modifications.

4 Application in Kernel Methods

Until now, we only commented on and illustrated the kernel functions themselves. In this section we want to demonstrate, that they are applicable in various kernel methods ranging from classification, feature-extraction to novelty detection. This is an empirical demonstration of goal 1 in the introduction. In addition to the applicability, we will point out several benefits of invariant kernels compared to the non-invariant ordinary ones. The following algorithms are also implemented in the MATLAB library *KerMet-Tools* (Haasdonk, 2005b) used for the plots in the preceding section.

Kernel Nearest-Neighbour Classification

A simple kernel method is the kernel nearest-neighbour algorithm for classification. Despite the theoretical benefits of the standard nearest-neighbour algorithm (Duda et al., 2001, Sec. 4.5) (no training stage, asymptotic upper error bound in terms of the Bayes error, no parameters) the kernelized algorithm is not widely used. The kernelization hereby simply results from computing a distance measure from a given kernel by $d(x,x')^2 = k(x,x) - 2k(x,x') + k(x',x')$. One reason for this rare use is that for simple kernels the result is identical to the standard nearest-neighbour. In particular for the linear kernel and certain distance based kernels such as the Gaussian and negative distance kernel, this is the case as exemplified for a checkerboard pattern in Fig. 10, where plot a) is the result of the kernel 1-nearest-neighbour algorithm with the k^{rbf} kernel. So kernelization indeed has no benefit for these simple kernels. For invariant kernels the situation however changes: The simple structure of a base kernel can be remarkably made more complex, such that kernel nearest-neighbour indeed makes sense. In Fig b) and c) we assume that the transformation knowledge consists of a certain scaling indicated by the black lines. These transformations are obviously complementary to the information in the training data. The classification result for the IDS-kernel in b) and the TI-kernel in c) clearly

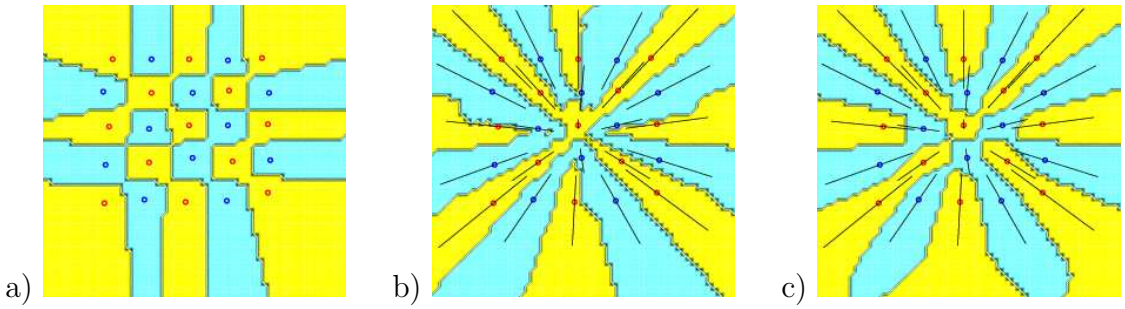

Figure 10: Illustration of kernel nearest-neigbour classification with rbf-kernel. a) No invariance, b) scale-invariance in IDS-kernel, c) scale-invariance in TI-kernel

captures this scale invariance, while the checkerboard data still is classified correctly. The IDS-kernel seems to be slightly advantageous over the TI-kernel as again the extent of the invariance is slightly larger than for the latter. In particular we see by this initial example, that the invariance properties of the kernel function transfer to the analysis method.

Kernel PCA Feature Extraction

The next method for demonstration is the kernel principal component analysis (KPCA) algorithm, cf. (Schölkopf et al., 2000). Traditional PCA searches for the directions of maximal variance of a dataset and projects points into this new coordinate system, these coordinates then are the features of a newly observed point. Hence, in two dimensions maximal 2 features can be extracted. This changes with the kernelized version, here the number of relevant features is bounded by the number of training samples, which can be remarkably higher than the dimension-restriction. We focus on two clusters of 4 points in Fig. 11 while using the indefinite k^{nd} kernel. The non-invariant KPCA correctly finds the illustrated principal component projection a), which separates the two clusters. If we however assume to have knowledge about rotation invariance, the *clusters* should more reasonably be separated by proximity of their transformation circles. This is nicely obtained for the IDS-kernel in b) and the TI-kernel in plot c). This is another interesting relation to the traditional method of invariant feature construction: Kernel methods with invariant kernels can be used to construct invariant features, which are then available for arbitrary (non-kernel-based) analysis methods.

Enclosing Hypersphere

As a further kernel method we choose an unsupervised method for novelty detection, which is the optimal enclosing hypersphere algorithm, cf. (Shawe-Taylor and Cristianini, 2004). With the linear kernel, this algorithm searches for the best fitting sphere around a data distribution while the fraction of outliers can be specified in advance. As illustrated

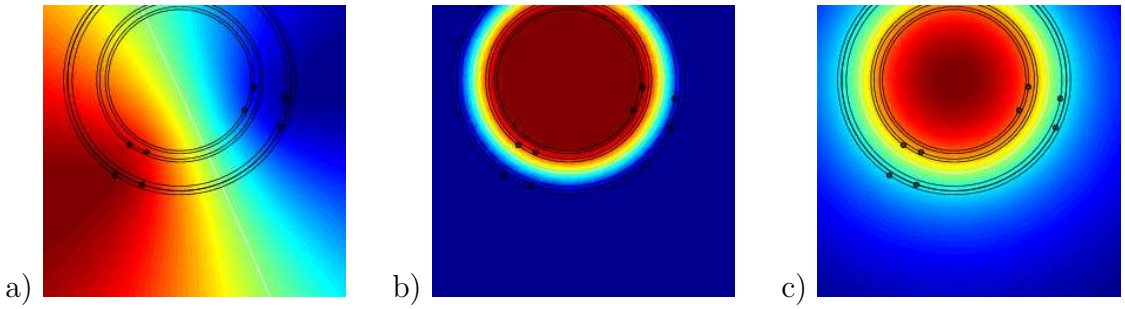

Figure 11: Illustration of the first principal component of KPCA analysis. a) Non invariant negative distance kernel k^{nd}, b) rotation invariant k^{nd}_{IDS}, c) rotation invariant k^{nd}_{TI}.

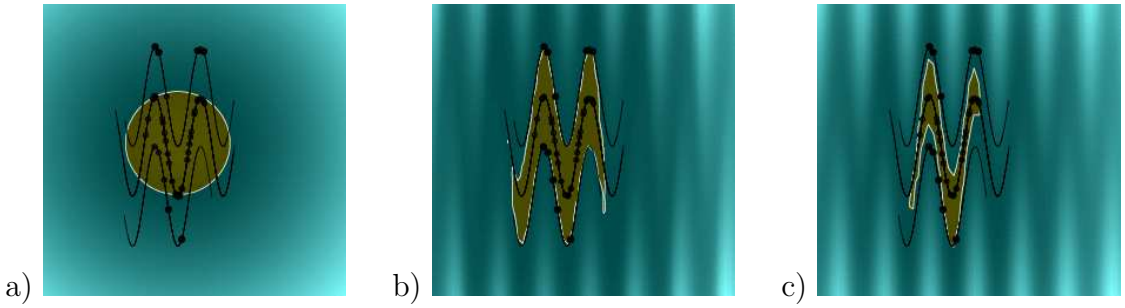

Figure 12: Illustration of novelty detection experiments. a) Non-invariant linear kernel, b) sine-invariant k^{lin}_{IDS}, c) sine-invariant k^{lin}_{TI}

in Fig. 12 we choose 30 points randomly lying on a sine-curve, which are interpreted as normal observations. We randomly add 10 points on slightly downward/upward shifted curves and want these points to be detected as novelties. So the outlier rate is 25% and this rate can be set in advance in the ν-formulation of the algorithm. Plot a) indicates that the linear kernel nicely results in an ordinary sphere, which however gives 5 false alarms, i.e. normal patterns detected as novelties, and 4 missed outliers, i.e. outliers detected as normal patterns. As soon as we involve the sine-invariance into the IDS-kernel in b) or the TI-kernel in c) we consistently obtain 0 false alarms and 0 misses. To demonstrate that this is not a singular random tendency, we performed this experiment 20 times resulting in false alarm rates with standard-deviation intervals of 4.75 ± 1.1180, 0 ± 0 and 0.1000 ± 0.3078 for the kernels corresponding to plots a), b) and c). The miss rates correspondingly are 4.3500 ± 0.9333, 0.4000 ± 0.5026 and 0.2500 ± 0.4443. So single misses/false alarms are possible with the invariant kernels, but overall they almost perform perfect in contrast to the non-invariant kernel. This is an example, where the invariance knowledge already was present in the dataset, but explicit formalization still gives a remarkable performance gain in terms of recognition or detection accuracy. Similar observations can be made with other predictive kernel-methods.

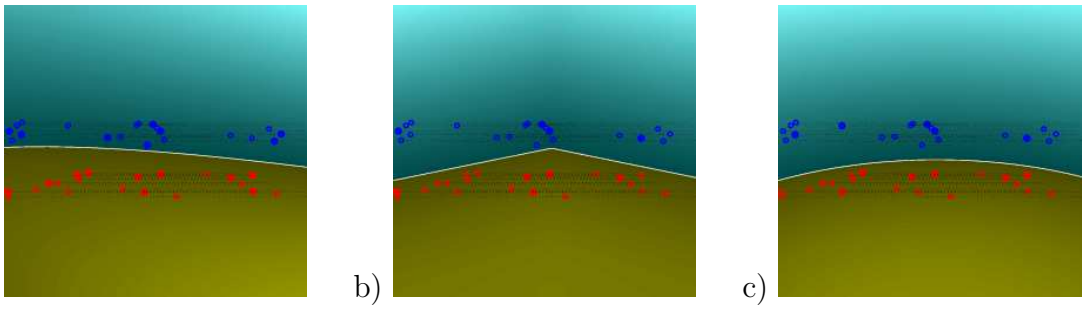

Figure 13: Kernel Perceptron results with polynomial kernel k^{pol} of degree 2. a) Non-invariant kernel, b) reflection-invariant k^{pol}_{IDS}, c) reflection-invariant k^{pol}_{TI}.

Kernel Perceptron

In this example we want to focus on another aspect, namely the convergence speed in online-learning algorithms. The folklore representative is the perceptron algorithm, which can also be kernelized, cf. (Herbrich, 2002). As an example we take the polynomial kernel k^{pol} of degree 2 and want to separate two random point sets of 20 points each lying uniformly distributed within two horizontal rectangular stripes indicated in Fig. 13. We want to investigate the effect of explicitly incorporating the reflection invariance of the problem into the kernel. The dataset is clearly separable with polynomials of degree 2, as is obtained by the non-invariant kernel in a). Similar well separation is obtained by the reflection invariant kernels k^{pol}_{IDS} in b) and k^{pol}_{TI} in c). Thus, accuracies are not interesting in this example, but another algorithmic aspect, namely the number of updates required before the solution is obtained. For the non-invariant kernel, the algorithm requires 18 update steps, while the invariant kernels converge much faster after 9 (IDS) or 11 (TI) updates. Again, this result is not a singular random outcome, but can be found systematically. We performed the random data distribution and the convergence analysis 20 times, which results in the one-standard-deviation intervals of 21.0000 ± 6.5855, 11.5500 ± 4.5361 and 12.5500 ± 2.3050 updates for the non-invariant, the IDS- and the TI-kernel, respectively. So indeed, the explicit invariance knowledge leads to improved convergence properties.

Support Vector Machine

We want to conclude the 2D experiments with aspects which are demonstrated on the well known SVM for classification. We want to investigate the effect of invariance knowledge on the resulting model complexities. For this we take two random sets of 20 points distributed uniformly on two concentric rings, cf. Fig. 14. We will involve rotation invariance explicitly by taking T as rotations by angles $\phi \in [-\pi/2, \pi/2]$. We clearly see, that the problem is separable by a standard SVM with Gaussian kernel in a) and its corresponding invariant versions b) IDS-kernel and c) TI-kernel. The main difference in addition to some clearly captured invariance is the model size in terms of number of

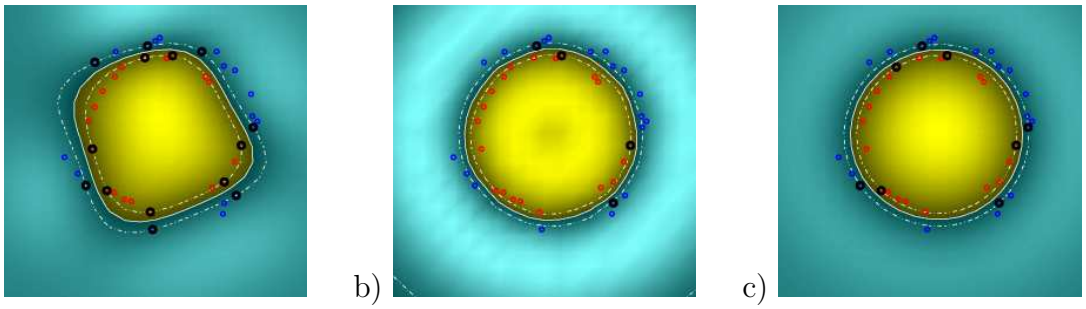

Figure 14: SVM model simplification by invariance. a) Non-invariant k^{rbf}, b) rotation invariant k^{rbf}_{IDS}, c) rotation invariant k^{rbf}_{TI}.

support vectors (SVs). This is a determining factor for the required storage, number of test-kernel evaluations and other theoretical statements like error estimates. In the example we obtain 14 SVs for the non-invariant case, where the IDS-kernel only returns 4 SVs and the TI-approach results in 8 SVs. Again, this is not only a singular observation, repeated evaluation over random drawings of the data yield similar results. So again there is a clear improvement by involving invariance, in this case expressed in the model size.

As the SVM is the most familiar algorithm of the presented examples, we additionally want to discuss the generalization aspect here. From theoretical considerations it is well known, that the number of support vectors n_{SV} of a SVM trained on n samples, denoted as f_n, can be related to the generalization error $R[f_{n-1}]$ of a SVM trained on $n-1$ samples. For instance, the fraction of support vectors n_{SV}/n is an upper bound on the leave-one-out error R_{LOO} and the latter is an unbiased estimate of the true risk of a learning machine, cf. (Schölkopf and Smola, 2002, Thm. 12.9) and references therein. This bound can be formulated as $E_{n-1}(R[f_{n-1}]) = E_n(R_{LOO}[f_n]) \leq E_n(n_{SV})/n$, where the expectations are to be taken over $n-1$ resp. n independently drawn samples. So a low expected value of n_{SV} for a SVM trained on n samples is a guarantee for a low expected value of the true risk for the SVM trained on all possible $n-1$ samples. This is only an averaged statement, the estimate does not regard the variance of the true risk and the number of training samples on both sides is differing. So the practical use of this estimate is limited, but still it indicates, that the LOO-error or the fraction of SVs is an informative quantity for assessing the generalization ability of a SVM.

Another quantity, the geometrical margin $\gamma := 1/\|\mathbf{w}\|$ of a SVM can also be related to generalization ability. Intuitively it is clear, that the margin alone cannot be used directly for comparing different kernels, but the scaling of the data has to be respected. The margin can always be increased by upscaling the data, i.e. the kernel function. So a certain data-dependent scaling must be performed for comparing different kernels. Examples are radius-margin quantities, which emerge from VC capacity bounds e.g. (Schölkopf and Smola, 2002, Thm. 5.5). Alternatively, the trace of the kernel matrix can be used to effectively estimate a data-dependent scaling factor, which allows to derive generalization

Table 1: Detailed experimental SVM results averaged over a random drawing of 20 training datasets.

	non-inv. k^{rbf}	rotation inv. k^{rbf}_{IDS}	rotation inv. k^{rbf}_{TI}
test error	0.048 ± 0.037	0 ± 0	0.001 ± 0.003
fraction of SVs n_{SV}/n	0.410 ± 0.042	0.085 ± 0.019	0.188 ± 0.025
trace-margin ratio $\frac{4}{n\gamma}\sqrt{\text{tr}(\mathbf{K})}$	12.68 ± 2.25	10.67 ± 1.26	10.17 ± 1.40

bounds from the margin. For instance (Shawe-Taylor and Cristianini, 2004, Thm. 7.22) states, that the generalization error of a SVM with bias can be bounded by an estimate which is dominated by the term $4\sqrt{\text{tr}(\mathbf{K})}/(n\gamma)$.

In addition to these formal indicators, the most practical indicator for the generalization ability is certainly the test-error on an independent test-set.

We have assessed these quantities empirically for the given example of rotational invariance in SVM and have performed 20 training runs with the non-invariant and both invariant Gaussian rbf-kernels. We list the test-error, the fraction of support vectors and the trace-margin ratio in Tab. 1. The test-error is obtained by drawing 100 i.i.d. points and classifying these by all trained classifiers. In addition to the average values, the table lists the one-standard-deviation intervals.

It is clearly visible, how the use of invariance improves all of the considered generalization indicators. The test error is consistently decreased by the invariant kernels. The fraction of SVs is lowered and the trace-margin ratio is also effectively reduced. These findings demonstrate and support the main benefit of invariant kernels in practice: The generalization ability can very well be expected to improve.

Real World Applications

In this paper we restrict our considerations to the conceptional aspects and the 2D-examples of the preceding section. However, the real world applicability of the kernels can also be demonstrated. Existing work and work in progress is covered by the framework presented here. The Haar-integration kernels of (Haasdonk et al., 2005) are a special case, namely restriction to transformation groups, of the general TI-kernels presented here. In that study, the real world applicability was demonstrated on SVM classification in an optical character recognition problem, where geometric transformations of the letters were involved. The USPS dataset was used, which consists of 7291 training and 2007 test-patterns of 16x16 grey-value images of 10 digit classes. We give some reference results in Tab. 2. For further details on this widely used dataset and further literature results we refer to (Schölkopf and Smola, 2002). The non-invariant k-nearest-neighbour method (with an extended training set called USPS+) and the SVM approach are clearly outperformed by incorporating invariance by the virtual support vector method or the TI-kernels. Overall, error-rates of 3.2% were obtained with the TI-kernels, which are

Table 2: Test-error rates on USPS digits dataset.

base method	type of invariance	test-error	reference
k-NN	no invariance, USPS+	5.9 %	(Simard et al., 1993)
SVM	no invariance, k^{rbf}	4.5 %	(Haasdonk et al., 2005)
SVM	VSV, x/y-shift, k^{pol}	3.2 %	(Schölkopf et al., 1996)
SVM	x/y-shift k_{TI}^{rbf}	3.2 %	(Haasdonk et al., 2005)

Table 3: LOO-error rates on Raman-spectra dataset.

base method	type of invariance	LOO-error	reference
k-NN	no invariance	21.06 %	(Peschke et al., 2006)
k-NN	baseline-shift + scaling, k_{IDS}^{rbf}	9.00 %	(Peschke et al., 2006)
SVM	no invariance k^{rbf}	4.20 %	(Haasdonk, 2005c)
SVM	baseline-shift + scaling, k_{IDS}^{rbf}	2.91 %	(Peschke et al., 2006)

comparable to those of the VSV-method. Certain acceleration methods resulted in training/testing speed outperforming/being comparable to the VSV-method.

Examples of the IDS-kernels are given in (Haasdonk and Keysers, 2002; Bahlmann et al., 2002; Peschke et al., 2006). In the first reference, an invariant distance called *tangent distance* was involved in the USPS handwritten digit recognition task. The results presented there have meanwhile been improved to slightly outperform the VSV-method, cf. (Haasdonk, 2005c). The second reference (Bahlmann et al., 2002) involves another invariant distance measure called *dynamic time warping* distance in the problem on online-handwriting recognition. Here, the infinite sets T_x of time-reparameterizations of point-sequences can be efficiently dealt with by dynamic programming. A recent application of invariant distances can be found in the field of bacteria recognition based on Raman-spectroscopic measurements (Peschke et al., 2006). The dataset used in this study consists of 2545 spectra of 20 classes, each consisting of 1833 intensity measurements. Chemometrically relevant pattern variations can be formulated and the invariant distances can efficiently be computed by projections on linear subspaces. The transformations, which are considered, involve an intensity scaling by multiplication and a base-line-shift by adding Lagrange-polynomials. We give the main results concerning the LOO-errors in Tab. 3, which again demonstrate, that the non-invariant k-nearest-neighbour and SVM approaches are clearly improved by involving the invariance by the IDS-kernels.

We refrain from further specific details on applications, but summarize, that the kernels give state-of-the-art results and are real-world applicable as soon as the transformation sets T_x are reasonably defined, such that fast computations are possible by omitting most of the pairwise explicit pattern transformations.

5 Conclusion

We exemplified that the notion of *invariance* is used in qualitatively different meanings in machine learning and pattern analysis and especially in kernel method research. We therefore have distinguished different notions and focussed on *totally invariant* kernels. We clarified the conceptional relation of such kernels to traditional pattern analysis by demonstrating that invariant kernels and invariant features have direct correspondences. In particular the main insight is that invariant kernels are practically more general than invariant features as they enable practical operation with high or infinite dimensional feature representations. Additionally, these kernels conceptionally cover and interpolate between the opposite traditional methods of invariant feature extraction and template matching. In addition to these conceptional relations, we proposed two generic practical methods for constructing invariant kernel functions under the assumption of very general transformation knowledge, which is mostly restricted in literature. The approaches support discrete, continuous, infinite and even non-group transformations. The approaches offer intuitive ways of adjusting the total invariance to approximate invariance until recovering the non-invariant case. By this they build a framework interpolating between invariant and non-invariant machine learning. In particular, both realize the goals formulated in Sec. 1. The main difference between the TI and the IDS-kernel approach seems to be that the former results in positive definite kernels, while the latter in general yields indefinite functions. The IDS-kernels, however, capture the invariance more accurately and the other benefits also turn out to be consistently better expressed. So overall, the IDS-kernels seem to be the first choice for invariance in kernel methods. If the desired kernel method turns out to have problems with indefinite kernels, the TI-kernels can be taken as a valuable alternative. Various practical possible benefits of invariant kernels have been demonstrated. In addition to the model-inherent invariance, when applying such kernels, further advantages can be the convergence speed in online-learning methods, model size reduction in SV approaches, or improvement of prediction accuracy. The latter the most relevant point in practice, and we exemplified that this improvement in generalization ability is supported by various error indicators.

Current perspectives in invariant kernel methods are investigation of further acceleration methods for the proposed kernels. Additionally, the use of indefinite kernels in machine learning is not completely understood. New algorithms are to be developed and existing algorithms must be investigated on suitability. Another interesting option is the construction of approximate invariant feature representations by kernel-methods for feature extraction, e.g. KPCA.

References

Abe, S.: 2003, 'On Invariance of Support Vector Machines'. In: *Proceedings of the 4th International Conference on Intelligent Data Engineering and Automated Learning.*

Alt, H. and L. J. Guibas: 1999, 'Discrete geometric shapes: Matching, interpolation, and

approximation - A survey'. In: *Handbook of Computational Geometry*. Elsevier Science Publishers B.V. North-Holland, pp. 121–153.

Bahlmann, C., B. Haasdonk, and H. Burkhardt: 2002, 'On-line Handwriting Recognition with Support Vector Machines - A Kernel Approach'. In: *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*. pp. 49–54.

Berg, C., J. P. R. Christensen, and P. Ressel: 1984, *Harmonic Analysis on Semigroups. Theory of Positive Definite and Related Functions*, Graduate Texts in Mathematics. Springer.

Binford, T. O. and T. S. Levitt: 1993, 'Quasi-Invariants: Theory and Exploitation'. In: *Proceedings of DARPA Image Understanding Workshop*. pp. 819–829.

Burges, C. J. C.: 1999, 'Geometry and Invariance in Kernel Based Methods'. In: B. Schölkopf, C. J. C. Burges, and A. J. Smola (eds.): *Advances in Kernel Methods — Support Vector Learning*. pp. 89–116.

Burkhardt, H. and S. Siggelkow: 2001, 'Invariant features in pattern recognition - fundamentals and applications'. In: *Nonlinear Model-Based Image/Video Processing and Analysis*. John Wiley & Sons, pp. 269–307.

Canterakis, N.: 1999, '3D Zernike moments and Zernike Affine Invariants for 3D Image Analysis and Recognition'. In: *Proceedings of the 11th Scandinavian Conference on Image Analysis*.

Chapelle, O. and B. Schölkopf: 2002, 'Incorporating invariances in nonlinear Support Vector Machines'. In: *Advances in Neural Information Processing Systems 14*. pp. 609–616.

Cortes, C., P. Haffner, and M. Mohri: 2003, 'Rational Kernels'. In: *Advances in Neural Information Processing Systems 15*.

DeCoste, D. and B. Schölkopf: 2002, 'Training Invariant Support Vector Machines'. *Machine Learning* **46**(1), 161–190.

Duda, R. O., P. E. Hart, and D. G. Stork: 2001, *Pattern Classification*. Wiley Interscience, 2nd edition.

Fitzgibbon, A. W. and A. Zisserman: 2003, 'Joint Manifold Distance: A new approach to appearance based clustering'. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*.

Fung, G. M., O. L. Mangasarian, and J. W. Shavlik: 2004, 'Knowledge-Based Support Vector Machine Classifiers'. In: *Advances in Neural Information Processing Systems 16*.

Genton, M. G.: 2001, 'Classes of Kernels for Machine Learning: A Statistics Perspective'. *Journal of Machine Learning Research* **2**, 299–312.

Graepel, T. and R. Herbrich: 2004, 'Invariant Pattern Recognition by Semidefinite Programming Machines'. In: *Advances in Neural Information Processing Systems 16*.

Haasdonk, B.: 2005a, 'Feature Space Interpretation of SVMs with Indefinite Kernels'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(4), 482–492.

Haasdonk, B.: 2005b, 'KerMet-Tools: A MATLAB Invariant Kernel Method Toolbox'. Software available at http://lmb.informatik.uni-freiburg.de/people/haasdonk/KerMet-Tools.

Haasdonk, B.: 2005c, 'Transformation Knowledge in Pattern Analysis with Kernel Methods - Distance and Integration Kernels'. Ph.D. thesis, Computer Science Department, University of Freiburg, Germany.

Haasdonk, B. and D. Keysers: 2002, 'Tangent Distance Kernels for Support Vector Machines'. In: *Proceedings of the 16th International Conference on Pattern Recognition*, Vol. 2. pp. 864–868.

Haasdonk, B., A. Vossen, and H. Burkhardt: 2005, 'Invariance in Kernel Methods by Haar-Integration Kernels'. In: *Proceedings of the 14th Scandinavian Conference on Image Analysis*. pp. 841–851.

Haussler, D.: 1999, 'Convolution Kernels on Discrete Structures'. Technical Report UCS-CRL-99-10, UC Santa Cruz.

Herbrich, R.: 2002, *Learning Kernel Classifiers*. MIT Press.

Joachims, T.: 1999, 'Estimating the Generalization Performance of a SVM Efficiently'. Technical Report LS8 Report 25, University of Dortmund, Germany.

Keysers, D., J. Dahmen, T. Theiner, and H. Ney: 2000, 'Experiments with an Extended Tangent Distance'. In: *Proceedings of the 15th International Conference on Pattern Recognition*, Vol. 2. pp. 38–42.

Keysers, D., C. Gollan, and H. Ney: 2004, 'Local Context in Non-linear Deformation Models for Handwritten Character Recognition'. In: *Proceedings of the 17th International Conference on Pattern Recognition*.

Leen, T. K.: 1995, 'From Data Distributions to Regularization in Invariant Learning'. In: *Advances in Neural Information Processing Systems 7*.

Lenz, R.: 1991, 'Group Theoretical Feature Extraction: Weighted Invariance and Texture Analysis'. In: *Proceedings of the 7th Scandinavian Conference on Image Analysis*. pp. 63–70.

Lodhi, H., C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins: 2002, 'Text Classification using String Kernels'. *Journal of Machine Learning Research* **2**, 419–444.

Loosli, G., S. Canu, S. Vishwanathan, and A. Smola: 2005, 'Invariances in Classification: an efficient SVM implementation'. In: *Proceedings of the 11th International Symposium on Applied Stochastic Models and Data Analysis*.

Mika, S., G. Rätsch, B. Schölkopf, A. Smola, J. Weston, and K.-R. Müller: 2000, 'Invariant Feature Extraction and Classification in Kernel Spaces'. In: *Advances in Neural Information Processing Systems 12*.

Mundy, J. L., A. Zisserman, and D. Forsyth (eds.): 1994, 'Applications of Invariance in Computer Vision, Proceedings of the of 2nd Joint European - US Workshop 1993'. Springer.

Nachbin, L.: 1965, *The Haar integral*. Princeton, N.J.: Van Nostrand.

Ong, C. S., X. Mary, S. Canu, and A. J. Smola: 2004, 'Learning with Non-Positive Kernels'. In: *Proceedings of the 21st International Conference on Machine Learning*. pp. 639–646.

Peschke, K., B. Haasdonk, O. Ronneberger, H. Burkhardt, P. Rösch, M. Harz, and J. Popp: 2006, 'Using Transformation Knowledge for the Classification of Raman Spectra of Biological Samples'. In: *Proceedings of the 4th IASTED International Conference on biomedical engineering*. pp. 288–293.

Pozdnoukhov, A. and S. Bengio: 2004, 'Tangent Vector Kernels for Invariant Image Classification with SVMs'. In: *Proceedings of the 17th International Conference on Pattern Recognition*.

Rabiner, L. and B. Juang: 1993, *Fundamentals of Speech Recognition*. Prentice Hall.

Sahbi, H. and F. Fleuret: 2002, 'Scale-Invariance of Support Vector Machines based on the Triangular Kernel'. Technical Report RR-4601, INRIA.

Schölkopf, B.: 2001, 'The Kernel Trick for Distances'. In: *Advances in Neural Information Processing Systems 13*. pp. 301–307.

Schölkopf, B., C. Burges, and V. Vapnik: 1996, 'Incorporating Invariances in Support Vector Learning Machines'. In: *Proceedings of the 6th International Conference on Artificial Neural Networks*. pp. 47–52.

Schölkopf, B., P. Simard, A. Smola, and V. Vapnik: 1998, 'Prior knowledge in support vector kernels'. In: *Advances in Neural Information Processing Systems 10*. pp. 640–646.

Schölkopf, B., A. Smola, R. Williamson, and P. Bartlett: 2000, 'New support vector algorithms'. *Neural Computation* **12**, 1083 – 1121.

Schölkopf, B. and A. J. Smola: 2002, *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press.

Schulz-Mirbach, H.: 1994, 'Constructing invariant features by averaging techniques'. In: *Proceedings of the 12th International Conference on Pattern Recognition*, Vol. 2. pp. 387–390.

Schulz-Mirbach, H.: 1995, 'Anwendung von Invarianzprinzipien zur Merkmalgewinnung in der Mustererkennung.'. Ph.D. thesis, Technical University Hamburg-Harburg, Germany.

Schur, I.: 1968, *Vorlesungen über Invariantentheorie*. Berlin: Springer.

Shawe-Taylor, J. and N. Cristianini: 2004, *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Simard, P. Y., Y. A. LeCun, and J. S. Denker: 1993, 'Efficient Pattern Recognition Using a New Transformation Distance'. In: *Advances in Neural Information Processing Systems 5*. pp. 50–58.

Simard, P. Y., Y. A. LeCun, J. S. Denker, and B. Victorri: 1998, 'Transformation Invariance in Pattern Recognition — Tangent Distance and Tangent Propagation'. In: *Neural Networks: Tricks of the Trade*. pp. 239–274.

Smola, A. J., R. Vidal, and S. V. N. V. Vishwanathan: 2004, 'Kernels and Dynamical Systems'. *Automatica*. Submitted.

Vasconcelos, N. and A. Lippman: 1998, 'Multiresolution Tangent Distance for Affine-invariant Classification'. In: *Advances in Neural Information Processing Systems 10*.

Veltkamp, R. C.: 2001, 'Shape Matching: Similarity Measures and Algorithms'. Technical Report UU-CS-2001-03, Department of Computing Science, Utrecht University, The Netherlands.

Watkins, C.: 2000, 'Dynamic Alignment Kernels'. In: *Advances in Large Margin Classifiers*. MIT Press, pp. 39–50.

Werman, M. and D. Weinshall: 1995, 'Similarity and Affine Invariant Distances Between 2D Point Sets'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **17**(8), 810–814.

Wood, J.: 1996, 'Invariant pattern recognition: a review'. *Pattern Recognition* **29**(1), 1–17.