Incorporating Prior Knowledge in Support Vector Machines for Classification: a Review

Fabien Lauer, Gérard Bloch

Incorporating Prior Knowledge in Support Vector Machines for Classification: a Review

Fabien Lauer * and Gérard Bloch

Centre de Recherche en Automatique de Nancy (CRAN - UMR 7039), Nancy–University, CNRS, CRAN–ESSTIN, Rue Jean Lamour, 54519 Vandœuvre Cedex, France

Abstract

For classification, support vector machines (SVMs) have recently been introduced and quickly became the state of the art. Now, the incorporation of prior knowledge into SVMs is the key element that allows to increase the performance in many applications. This paper gives a review of the current state of research regarding the incorporation of two general types of prior knowledge into SVMs for classification. The particular forms of prior knowledge considered here are presented in two main groups: class-invariance and knowledge on the data. The first one includes invariances to transformations, to permutations and in domains of input space, whereas the second one contains knowledge on unlabeled data, the imbalance of the training set or the quality of the data. The methods are then described and classified in the three categories that have been used in literature: sample methods based on the modification of the training data, kernel methods based on the modification of the kernel and optimization methods based on the modification of the problem formulation. A recent method, developed for support vector regression, considers prior knowledge on arbitrary regions of the input space. It is exposed here when applied to the classification case. A discussion is then conducted to regroup sample and optimization methods under a regularization framework.

Key words: pattern recognition, classification, support vector machines (SVM), prior knowledge, invariance

* Corresponding author.
Email addresses: fabien.lauer@esstin.uhp-nancy.fr (Fabien Lauer), gerard.bloch@esstin.uhp-nancy.fr (Gérard Bloch).

1 Introduction

Pattern recognition is a very active field of research intimately bound to machine learning. As part of this area, classification aims at building classifiers that can determine the class of an input pattern. An extensive amount of work has been done in the past decades to develop classifiers that can learn from data to perform recognition tasks. Typical fields of application include image recognition such as character recognition, text categorization, speech recognition, biometric applications, bioinformatics, fault detection, diagnostic applications, decision support, network intrusion detection and so on. The classification problems can be divided into binary problems and multi-class problems. Many classifiers like support vector machines (SVMs) [63] consider the first case where the patterns can be of two classes. For multi-class applications, there also exists learning machines that can tackle directly multi-class applications such as neural networks [4,33] or even multi-class support vector machines (MSVM) [69,21,8]. Nonetheless, a very common approach consists in building a set of binary classifiers, each one either trained to separate one class from the others (the *one-against-all* method) or only to distinguish between two classes (the *one-against-one* method). Thus, this paper will focus on binary SVMs without restricting the area of applications.

Support vector machines aim at learning an unknown decision function based only on a set of N input-output pairs $(\boldsymbol{x}_i, y_i)$. Nonetheless, in real world applications, a certain amount of information on the problem is usually known beforehand. For instance, in character recognition, if an image is slightly translated or rotated it still represents the same character. This prior knowledge indicates that one should incorporate invariance to translations and rotations into the classifier.

This paper gives a review of the current state of research regarding the incorporation of two main types of prior knowledge into SVMs for classification. The different forms of prior knowledge considered here are presented hierarchically and divided into two main groups: class-invariance and knowledge on the data. The first one includes invariances to transformations, to permutations and in domains of input space, whereas the second one contains knowledge on unlabeled data, the imbalance of the training set or the quality of the data. This review chooses to present general methods that can be used for different applications rather than to attempt to provide an exhaustive list of application specific prior knowledge with its practical implementation into SVMs. However, some interesting methods derived from an application specific point of view can still be used in other fields and thus deserve to be presented. This paper focuses on the methods and reuses a categorization from the literature based on the component of the problem (the samples, the kernel or the optimization program) which is modified to include the prior knowledge rather

than on the prior knowledge itself. A regularization framework is then used to regroup the sample and optimization based methods.

The present paper aims at giving an up-to-date review that synthesizes the existing methods. In the last decade, authors considered the introduction of prior knowledge into SVMs and some reviews can be found in the literature. A chapter of the well-known book [53] is dedicated to the incorporation of invariances, but deals only with transformation-invariances. Thus, the authors do not present methods to include knowledge on the data or class-invariance in a domain (which were not available at the time). Nonetheless, they expose the three different ways of exploiting prior knowledge on which relies our categorization of the methods into three groups: sample methods, kernel methods and optimization methods. In [22], an overview of the works in the field is also given. However, this overview focuses on invariant kernel methods for pattern recognition in general. Here, we are interested in different types of prior knowledge (not only invariance) that can be included in the particular learning machine known as SVM, either in the kernel or not. In particular, this review focuses on and is limited to two main types of prior knowledge: class-invariance and knowledge on the data.

The paper is organized as follows. The SVM principles and the different problem formulations (QP and LP) are first introduced in Section 2, before giving a definition and categorization of the types of prior knowledge that we consider in Section 3. A review of the literature for the incorporation of prior knowledge into SVMs is then presented in Section 4 where the methods are classified in 3 categories: sample methods (Sect. 4.1) based on the modification of the training data, kernel methods (Sect. 4.2) based on the modification of the kernel and optimization methods (Sect. 4.3) based on the modification of the problem formulation. This last subsection includes the presentation of a recently developed approach considering prior knowledge on arbitrary regions of the input space for support vector regression [38] to propose its application to the classification case. The links and differences between the methods are discussed in Section 5 with the unifying framework before the exposition of some perspectives regarding the combination of methods. Finally, the conclusion is given in Section 6.

Notations: all vectors are column vectors written in boldface and lowercase letters whereas matrices are boldface and uppercase, except for the ith column of a matrix $\boldsymbol{A}$ that is denoted $\boldsymbol{A}_i$. The vectors $\mathbf{0}$ and $\mathbf{1}$ are vectors of appropriate dimensions with all their components respectively equal to 0 and 1. For $\boldsymbol{A} \in \mathbb{R}^{d\times m}$ and $\boldsymbol{B} \in \mathbb{R}^{d\times n}$ containing d-dimensional sample vectors, the "kernel" $\boldsymbol{K}(\boldsymbol{A}, \boldsymbol{B})$ maps $\mathbb{R}^{d\times m} \times \mathbb{R}^{d\times n}$ in $\mathbb{R}^{m\times n}$ with $\boldsymbol{K}(\boldsymbol{A}, \boldsymbol{B})_{i,j} = k(\boldsymbol{A}_i, \boldsymbol{B}_j)$, where $k : \mathbb{R}^{d\times d} \to \mathbb{R}$ is the kernel function. In particular, if $\boldsymbol{x} \in \mathbb{R}^d$ is a column vector then $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{B})$ is a row vector in $\mathbb{R}^{1\times n}$. The matrix $\boldsymbol{X} \in \mathbb{R}^{N\times d}$ contains all the training samples $\boldsymbol{x}_i$, $i = 1, \ldots, N$, as rows. The kernel matrix

$\boldsymbol{K}(\boldsymbol{X}^{\boldsymbol{T}}, \boldsymbol{X}^{\boldsymbol{T}})$ will be written $\boldsymbol{K}$ for short. The symbol $\langle ., . \rangle$ stands for the inner product (or dot product).

2 Support vector machines for classification

The classifier is built from a training set of N samples

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_i, y_i), \ldots, (\boldsymbol{x}_N, y_N) \tag{1}$$

where $\boldsymbol{x}_i \in \mathbb{R}^d$ is the input vector corresponding to the ith sample labelled by $y_i \in \{-1, +1\}$ depending on its class (only binary problems are considered here). For the linear case, the machine implements the decision function

$$f(\boldsymbol{x}) = \text{sign } (\langle \boldsymbol{x}, \boldsymbol{w} \rangle + b) \tag{2}$$

of parameters $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$. This function determines on which side of the separating hyperplane $(\langle \boldsymbol{x}, \boldsymbol{w} \rangle + b = 0)$ the sample $\boldsymbol{x}$ lies.

Support vector machines (SVMs) were first introduced as large margin classifiers [63]. For a separable training set, the margin is defined as the minimum distance between the points of the two classes, measured perpendicularly to the separating hyperplane. Maximizing this margin is a way for a learning algorithm to control the capacity and the complexity of the machine, and to select the *optimal separating hyperplane* amongst all the hyperplanes that separate the two classes of the training set. The control of the capacity allows to bound the generalization error [63] which is the probability of misclassification for new test samples [46].

In its original form, the SVM learning leads to a quadratic program which is a convex constrained optimization problem and thus has a unique solution. This is a large advantage in comparison to other learning algorithms such as the back-propagation for neural networks [48,4]. The SVM problem can be formulated as follows: find the parameters $\boldsymbol{w}$ (also called the weights) and b that maximize the margin while ensuring that the training samples are well classified. This can be written as the QP optimization problem [9]

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 \tag{3}$$

$$\text{s.t. } y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) \geq 1, \quad i = 1, \ldots, N \tag{4}$$

whose solution corresponds to the saddle point of the primal Lagrangian

$$L_P = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i [y_i(\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) - 1] \tag{5}$$

where the $\alpha_i \geq 0$ are the Lagrange multipliers. The problem is equivalently solved by maximizing the dual Lagrangian with respect to α_i as

$$\begin{aligned} \max\ & L_D = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j y_i y_j \langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle \\ \text{s.t. } & \alpha_i \geq 0,\ i = 1, \ldots, N \\ & \textstyle\sum_{i=1}^{N} \alpha_i y_i = 0 \end{aligned} \tag{6}$$

The resulting decision function is given [9] by

$$f(\boldsymbol{x}) = \text{sign} \left(\sum_{\alpha_i > 0} y_i \alpha_i \langle \boldsymbol{x}, \boldsymbol{x}_i \rangle + b \right) \tag{7}$$

where the $\boldsymbol{x}_i$ are the support vectors (SVs), i.e. with non-zero corresponding Lagrange multipliers α_i. The SVs are the training patterns that lie on the margin boundaries. An advantage of this algorithm is its sparsity since only a small subset of the training samples is finally retained for the classifier.

In order to deal with non-separable data, the soft margin hyperplane is used. A set of slack variables ξ_i is introduced to allow errors or points inside the margin and a hyperparameter C is used to tune the trade-off between the amount of accepted errors and the maximization of the margin

$$\begin{aligned} \min\ & \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{N} \xi_i \\ \text{s.t. } & y_i (\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle + b) \geq 1 - \xi_i, \quad i = 1, \ldots, N \\ & \xi_i \geq 0 \end{aligned} \tag{8}$$

This new formulation leads to the same dual problem (6) but with the addition of an upper bound on the Lagrange multipliers [9]

$$0 \leq \alpha_i \leq C, \quad i = 1, \ldots, N \tag{9}$$

For non-linear classification problems, the data are first mapped into a higher dimensional feature space F by

$$\mathbb{R}^d \ni \boldsymbol{x} \overset{\Phi}{\mapsto} \boldsymbol{\Phi}(\boldsymbol{x}) \in F \tag{10}$$

in which a separating hyperplane is built. This leads to the decision function $f(\boldsymbol{x}) = \text{sign}\ (\langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{w} \rangle + b)$, where $\boldsymbol{w}$ is now a vector of F. To avoid the curse of dimensionality [9], the "kernel trick" is used, which leads, for (7),

$$f(\boldsymbol{x}) = \text{sign} \left(\sum_{\alpha_i > 0} y_i \alpha_i k(\boldsymbol{x}, \boldsymbol{x}_i) + b \right) \tag{11}$$

where $k(\boldsymbol{x}, \boldsymbol{x}_i) = \langle \boldsymbol{\Phi}(\boldsymbol{x}), \boldsymbol{\Phi}(\boldsymbol{x}_i) \rangle$ stands for the kernel function, or, in matrix form

$$f(\boldsymbol{x}) = \text{sign } \Big(\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X^T}) \boldsymbol{D\alpha} + b \Big) \tag{12}$$

where $\boldsymbol{D} = \text{diag}(y_1, \ldots, y_i, \ldots, y_N)$ and $\boldsymbol{\alpha} = [\alpha_1 \ \ldots \ \alpha_i \ \ldots \ \alpha_N]^T$.

For the training of a non-linear SVM, one only has to replace the inner product $\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle$ in (6) by the kernel function $k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that corresponds to the inner product in the feature space F. To be an admissible kernel, this function must satisfy Mercer's conditions (positive semidefiniteness of the kernel matrix $\boldsymbol{K}$) [9]. Typical kernel functions used for classification are the linear, Gaussian RBF or polynomial kernels.

The training of SVMs can also result in a linear program. This is now presented, since some of the methods exposed in the next section for the incorporation of prior knowledge use this form of SVMs. Following the approach of [37], the ℓ_1-norm of the parameters $\boldsymbol{\alpha}$ in (12) is minimized instead of the ℓ_2-norm of the weights $\boldsymbol{w}$ as in (8). In practice, to yield a linear program, a new set of variables $\boldsymbol{a}$ bounding the ℓ_1-norm of the parameters is used. In matrix form, the linear program corresponding to the soft margin SVM is

$$\begin{array}{rll} \min\limits_{(\boldsymbol{\alpha}, b, \boldsymbol{\xi}, \boldsymbol{a})} & \mathbf{1}^T \boldsymbol{a} + C \mathbf{1}^T \boldsymbol{\xi} & \\ \text{s.t.} & \boldsymbol{D}(\boldsymbol{K}\boldsymbol{D}\boldsymbol{\alpha} + b\mathbf{1}) \geq \mathbf{1} - \boldsymbol{\xi} & \\ -\boldsymbol{a} \leq & \boldsymbol{\alpha} \leq \boldsymbol{a} & \\ & \boldsymbol{\xi} \geq 0 & \end{array} \tag{13}$$

In this formulation, no assumption on the symmetry or positive definiteness of the kernel matrix $\boldsymbol{K}$ is needed [37]. The form of the resulting output function (12) remains unchanged. Here, the sparsity is enforced by the minimization of the ℓ_1-norm of the parameters $\boldsymbol{\alpha}$ which makes some α_i vanish to zero. It has also been noticed [3] that, compared to the original QP formulation, this approach offers an increased sparsity of support vectors.

One advantage of the SVMs is the form of the learning problems, since many general optimization softwares such as CPLEX, LOQO, Matlab linprog and quadprog are capable of solving the linear and quadratic programs derived from SVMs. Nonetheless, the scale of the problems led to develop specific methods such as chunking [29], decomposition [41] or its extreme case known as the Sequential Minimal Optimization (SMO) algorithm [42], of which modifications have been proposed [30]. Many softwares, usually available on the Internet, have been developed in the last years to speed up the training time or to deal with large data sets such as SVM^{light} [25], libSVM [6,15] and libsvmTL [47], HeroSVM [12,13] or Core Vector Machine (CVM) [61].

3 Prior knowledge for classification

This Section starts by giving a definition of prior knowledge as considered in this paper. Different types of prior knowledge encountered in pattern recognition are then regrouped under two main categories: class-invariance and knowledge on the data. The tree-like diagram of Figure 1 categorizes the types of prior knowledge from the most general at the top to the particular at the bottom where arrows point to suited methods. In particular, this review focuses on two main types of prior knowledge: class-invariance and knowledge on the data. The next Section provides the description and categorization of the suited methods.

3.1 Definition

In this review, *prior knowledge* is defined as in [53] and refers to all information about the problem available in addition to the training data. Determining a model from a finite set of samples without prior knowledge is an ill-posed problem, in the sense that, for instance, a unique model may not exist. Many classifiers incorporate the general smoothness assumption that a test pattern similar to one of the training samples tends to be assigned to the same class. Also, choosing the soft margin version of SVMs can be seen as a use of prior knowledge on the non-separability of the data or the presence of outliers and noise in the training set. However, in both cases, these assumptions are intrinsically made by the support vector learning and are thus excluded from the definition of prior knowledge in the remainder of the paper. In machine learning, the importance of prior knowledge can be seen from the No Free Lunch theorem [71] which states that all the algorithms perform the same when averaged over the different problems and thus implies that to gain in performance one must use a specialized algorithm that includes some prior knowledge about the problem at hand.

3.2 Class-invariance

A very common type of prior knowledge in pattern recognition is the invariance of the class (or the output of the classifier) to a transformation of the input pattern. Throughout this paper this type of knowledge will be referred to as *transformation-invariance*. Incorporating the invariance to a transformation $T_\theta : \boldsymbol{x} \mapsto T_\theta \boldsymbol{x}$, parametrized in θ, into a classifier of output $f(\boldsymbol{x})$ for an input pattern $\boldsymbol{x}$ corresponds to enforce the equality

$$f(\boldsymbol{x}) = f(T_\theta \boldsymbol{x}), \quad \forall \boldsymbol{x}, \theta \tag{14}$$

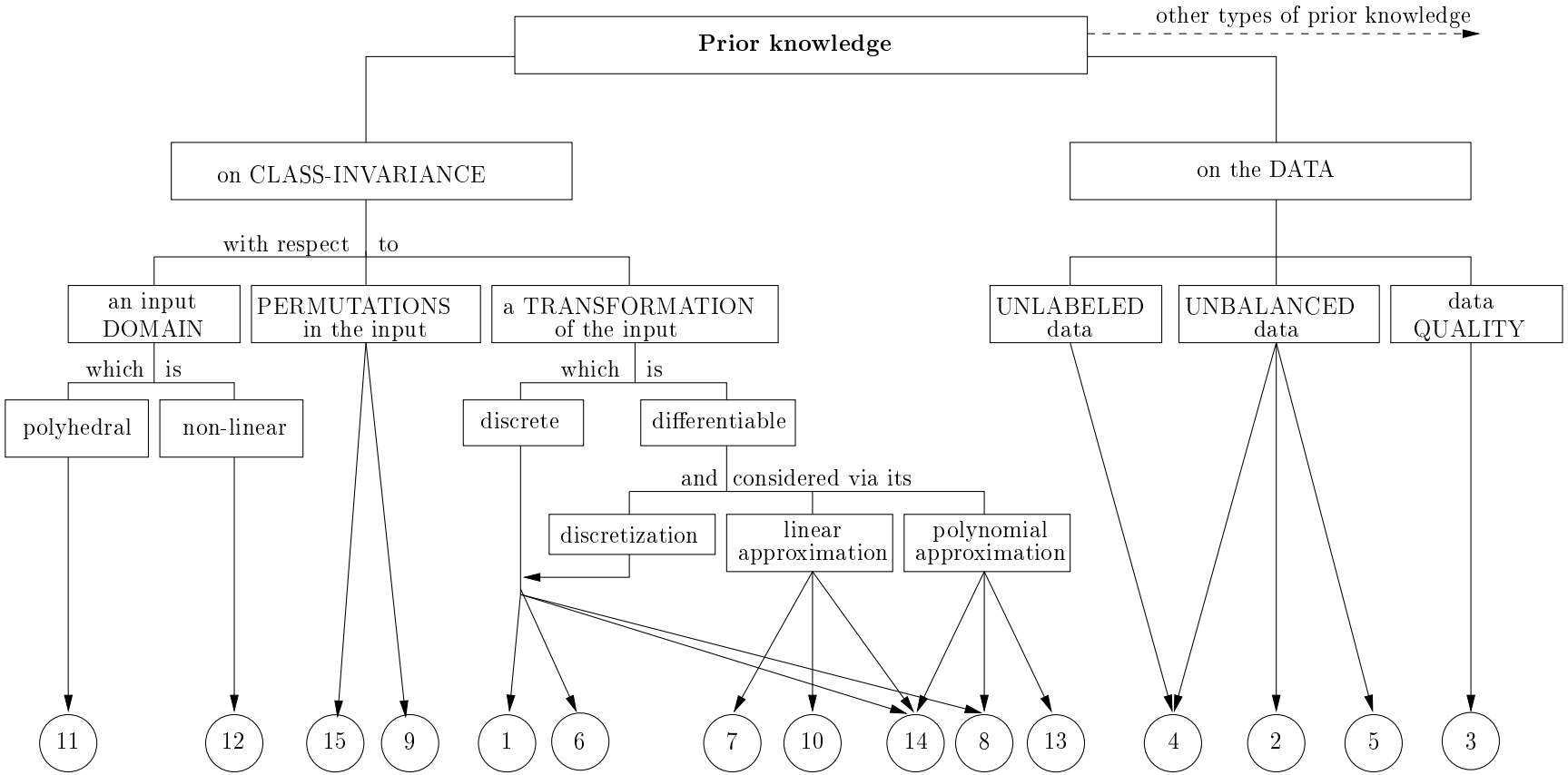

Fig. 1. Two main types of prior knowledge that can be incorporated into SVM with the corresponding methods expressed by their number given in the companion diagram of Figure 2.

However, local invariance is sometimes considered instead. In this case, the invariance is only imposed around a fixed value of θ. For a transformation centered at $\theta = 0$, so that $T_0\boldsymbol{x} = \boldsymbol{x}$, local invariance can be enforced by the constraint

$$\left.\frac{\partial}{\partial\theta}\right|_{\theta=0} f(T_\theta \boldsymbol{x}) = 0 \tag{15}$$

thus limiting the variation of f for a variation of θ. It must be noted that f in (15) is considered to be the real-valued output of the classifier, i.e. without the sign function in (7) or (11).

Some methods are based on another approach, which is to consider the class-invariance with respect to a *domain of the input space* instead of a transformation. In this case, the problem becomes finding f so that

$$f(\boldsymbol{x}) = y_{\mathcal{P}},\ \forall \boldsymbol{x} \in \mathcal{P} \tag{16}$$

where $y_{\mathcal{P}}$ is the class label of the region $\mathcal{P}$ of the input space. In practice, this approach is particularly useful to provide prior knowledge in regions of input space that lack training samples.

Another type of class-invariance found in pattern recognition is the *permutation-invariance*, i.e. invariance of the class to a permutation of elements in a structured input. A typical application is a classifier invariant to permutations of rows in matrix inputs. Since permutations are no more than particular transformations, permutation-invariance can also be considered as transformation-invariance.

3.3 Knowledge on the data

Other forms of prior knowledge than class-invariance concern the data more specifically and are thus of particular interest for real-world applications. In this review, the three particular cases that most often occur when gathering data are studied.

- *Unlabeled samples* are available with supposed class-memberships.
- *Imbalance* of the training set is encountered when a high proportion of samples is of the same class.
- *Quality of the data* may vary from a sample to another.

Prior knowledge in relationship with these cases can enhance the quality of the recognition if included in the learning. Moreover, not taking into account the poor quality of some data or a large imbalance between the classes can mislead the decision of a classifier.

4 Incorporating prior knowledge into SVM: a survey

In this Section, a review of the methods for incorporating prior knowledge into SVMs, however restricted to class-invariance and knowledge on the data, is given. These methods are classified in three categories, as defined by [53], depending on the means used to include the prior knowledge and the component of the problem that is modified. As presented in Figure 2, these three groups of methods are:

- *sample methods* that incorporate the prior knowledge either by generating new data or by modifying the way they are taken into account;
- *kernel methods* that incorporate the prior knowledge in the kernel function either by selecting the most appropriate kernel or by creating a new kernel;
- *optimization methods* that incorporate the prior knowledge in the problem formulation either by adding constraints to the original problem or by defining a new formulation which includes intrinsically the prior knowledge.

The methods are detailed in the following. Further discussions and links between the methods can be found in the next Section.

4.1 Sample methods

Two different approaches will be exposed in this section. The first one is based on the generation of virtual samples, while the second one aims at weighting the influence of different samples. Whereas the virtual samples methods focus on the incorporation of transformation-invariance, the weighting of samples allows to include knowledge on the data.

4.1.1 Virtual samples

In machine learning, the generalization ability of the obtained model depends on the number of data at hand. The more representative samples we have, the better we learn. Based on this simple fact, the idea of creating new samples to enlarge the training set was first introduced by [43] as virtual samples and in [1,2] as "hints". In [40], learning on an extended training set by virtual samples was linked to regularization and it thus showed a justification for the method.

The basic idea of the virtual samples approach [40] is to incorporate a known transformation-invariance as defined by (14). The new samples are generated

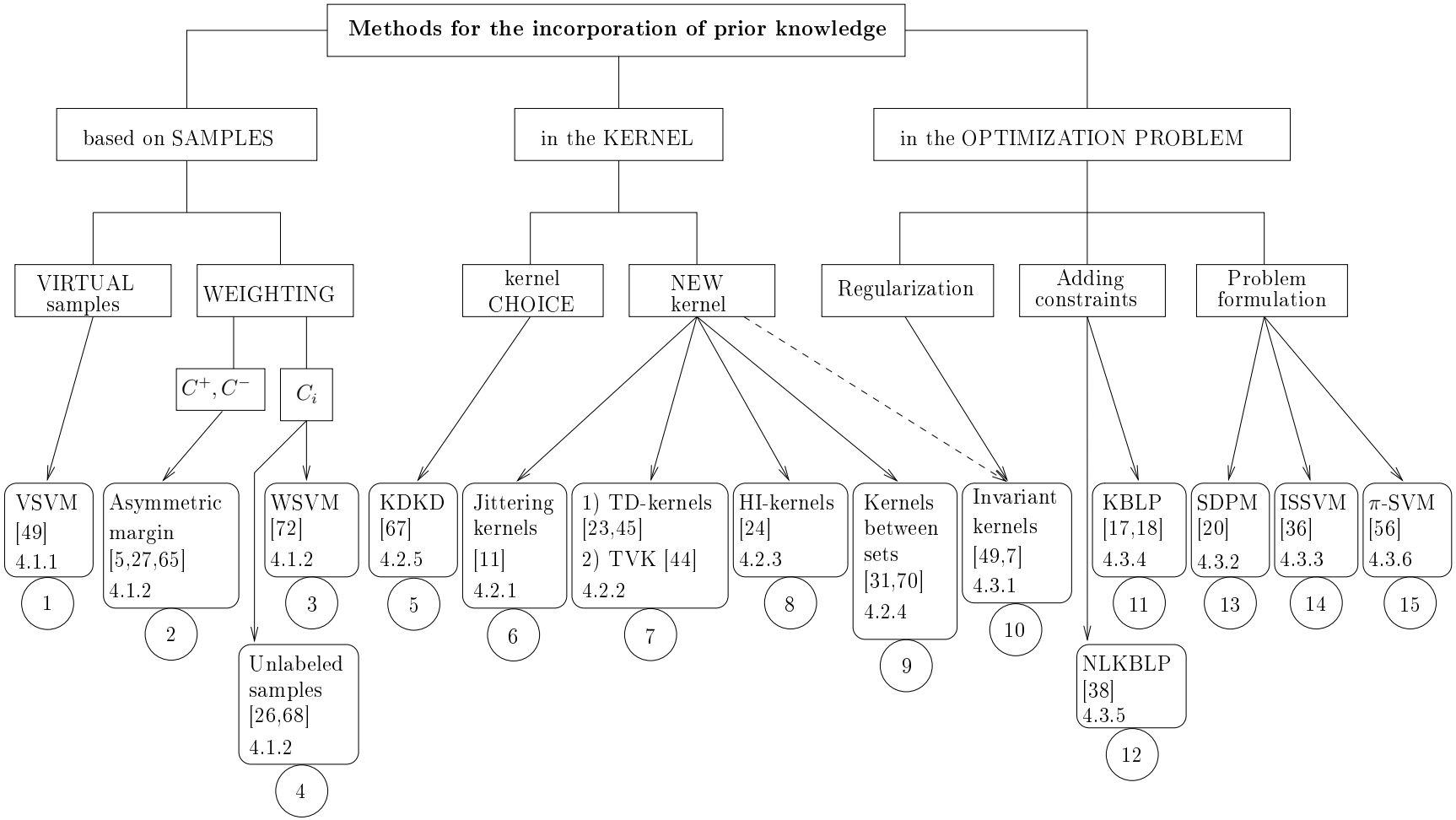

Fig. 2. Hierarchy of the methods for the incorporation of prior knowledge into SVM from the component of the problem which is modified at the top to the methods at the bottom. For each method are given the main references, the number of the corresponding section and a reference number in a circle.

from the training data as follows

$$(\boldsymbol{x}_i, y_i) \mapsto (T\boldsymbol{x}_i, y_i), \quad i = 1, \ldots, N \tag{17}$$

This method can be easily implemented in the context of pattern recognition. For instance, in image recognition, invariances to translations or rotations are often considered.

In the neural networks framework, *hints* introduced in [1] are specific properties the output function f of the network must satisfy such as oddness or invariance to a transformation. An error measure e_m is defined to introduce the mth hint H_m in the learning process. For instance, for the invariance hint that implies that the samples $\boldsymbol{x}_p$ and $\boldsymbol{x}_i$ are of the same class, we have

$$e_m(\boldsymbol{x}_p) = \left(f(\boldsymbol{x}_p) - f(\boldsymbol{x}_i)\right)^2 \tag{18}$$

which is zero when the two estimated classes are the same. A set $\mathcal{X}_m$ of N_m new samples is generated and the overall error E_m for the hint H_m is estimated by $E_m = 1/N_m \sum_{\boldsymbol{x}_p \in \mathcal{X}_m} e_m(\boldsymbol{x}_p)$ to test the accordance of f with the hint H_m. It must be noticed that the training set itself can be considered as a hint H_0 for which $e_0(\boldsymbol{x}_i) = (f(\boldsymbol{x}_i) - y_i)^2$ and the estimated overall error is $E_0 = 1/N \sum_{i=1}^{N} (f(\boldsymbol{x}_i) - y_i)^2$. Originally applied to a neural network that minimizes a least squares criterion, the learning process can be summarized as

$$\min \sum_m \lambda_m E_m = \sum_m \frac{\lambda_m}{N_m} \sum_{\boldsymbol{x}_p \in \mathcal{X}_m} e_m(\boldsymbol{x}_p) \tag{19}$$

where λ_m is a weighting factor for the hint H_m.

In addition, in [49], the Virtual SVM (VSVM) is introduced to incorporate transformation-invariance into SVMs. The idea is to generate the virtual samples only from the support vectors (SVs) since they contain all the information about the problem. The virtual samples are thus called "virtual SVs" (VSVs). The proposed procedure requires two SVM trainings. The first one extracts the SVs from the training set while the second one is performed on a dataset composed of the SVs and the VSVs.

In character recognition, the generation of virtual samples became very popular and almost necessary to achieve first-class performances. It is clear that an image representing a character will still represent the same character if, for instance, translated by one pixel. Thus, one often looks for classifiers that can incorporate some translation-invariance of the output as prior knowledge. Whereas simple distortions such as translations, rotations and scaling are generated by applying affine displacement fields to images, elastic distortion, introduced in [58] to imitate the variations of the handwriting, uses random displacement fields. Other transformations, such as morphing [28], were specifically developed and it appears that the best results (at least on the MNIST

database [34]) are obtained by elastic distortions [58,32] even if it is based on random displacement of pixels in the image. This highlights the fact that more samples help to learn better even if they are not absolutely accurate.

4.1.2 Weighting of samples

The weighting of samples allows to include other forms of prior knowledge than transformation-invariance. It is typically used to express knowledge on the data such as an imbalance between classes, the relative quality of the samples or prior knowledge on unlabeled samples. In practice this amounts to weight the errors or to choose a different trade-off parameter C for different samples.

Originally, different misclassification costs C_i were used to deal with unbalanced data, i.e. providing much more samples of a class than of the other [5,27]. C_i is set to a higher value for the less represented class, thus penalizing more the errors on this class. Besides, in [9], an equivalence is shown between the soft margin SVM using the ℓ_2-norm of the errors and a hard margin SVM trained with a modified kernel matrix

$$\boldsymbol{K}' = \boldsymbol{K} + \frac{1}{C}\boldsymbol{I} \tag{20}$$

This idea is extended in [65] to deal with unbalanced data when different misclassification costs C^+ and C^- are assigned for the positive and negative classes. The kernel matrix is then given by

$$\boldsymbol{K}' = \boldsymbol{K} + \boldsymbol{D} \tag{21}$$

where $\boldsymbol{D}$ is a diagonal matrix with components $D_{ii} = 1/C^+$ for $y_i = +1$ and $D_{ii} = 1/C^-$ for $y_i = -1$. This method amounts to define an asymmetric margin keeping further away from the decision boundary the class with a higher C. Heuristics are proposed in [5] to tune D_{ii} based on the knowledge of unbalanced data: set $D_{ii} = \lambda n^+/N$ for $y_i = +1$ and $D_{ii} = \lambda n^-/N$ for $y_i = -1$, where n^+ and n^- are respectively the numbers of positive and negative samples in the training set of size N and where λ is a scaling factor.

Another approach is developed in [72] in order to incorporate prior knowledge on the quality of the training data which may vary from a sample to another. The method, known as Weighted-SVM (WSVM), proposes to set a different cost C_i for each sample with respect to a confidence value based on some knowledge of the data acquisition or labeling procedure.

In [68], prior knowledge on unlabeled samples is considered. Based on supposed class-memberships, pseudo-labels are assigned to these samples which are then added to the training set with a different weight C_i in the cost function. The

incorporation of test samples as unlabeled data in the training is introduced in [64] as "transductive learning". A transductive learning for SVMs, in which the prior knowledge takes the form of the number num_+ of positive samples in the test set, has also been proposed for text classification [26]. In this scheme, the test samples are assigned a misclassification cost C^* which is different of the one used for the training samples and is further refined for each class as C^*_+ and C^*_- in accordance with the number num_+ to deal with unbalanced data.

4.2 Kernel methods

The following presents five methods based on the direct modification of the kernel function: the jittering kernels, the tangent distance, the Haar-integration kernels, the kernels between sets and the knowledge-driven kernel design. The first three methods aim at building invariant kernels k that can provide the same value for a sample $\boldsymbol{x}$ and its transformed $T\boldsymbol{x}$:

$$k(\boldsymbol{x}, \boldsymbol{z}) = k(T\boldsymbol{x}, \boldsymbol{z}) \tag{22}$$

thus leading to the transformation-invariance (14). Besides, the kernels between sets introduce permutation-invariance into the learning, which is another form of class-invariance. The last method considers the problem of selecting the kernel amongst admissible kernels with respect to prior knowledge on the imbalance of the training set.

4.2.1 Jittering kernels

Jittering kernels were first developed for kernel k-Nearest-Neighbors [10] and then presented for the incorporation of transformation-invariance into SVMs [11]. This approach is related to the virtual SV (VSV) method (see Sect. 4.1.1). Instead of considering an extended training set with all the jittered forms (translated, rotated...) of the training samples, these forms are considered in the kernel itself. Using the notation $k_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, the jittered form k^J_{ij} of the kernel k_{ij} is computed in two steps [11]:

(1) consider the sample $\boldsymbol{x}_i$ and all its jittered forms, and select the one, $\boldsymbol{x}_q$, closest to $\boldsymbol{x}_j$ by minimizing the distance between $\boldsymbol{x}_q$ and $\boldsymbol{x}_j$ in the space induced by the kernel:
$$\sqrt{k_{qq} - 2k_{qj} + k_{jj}} \tag{23}$$
(2) let $k^J_{ij} = k_{qj}$

Using such a jittering kernel may provide an output invariant to transformations. For instance, for a sample $\boldsymbol{x}$ and a sample issued from a translation of

this sample $T\boldsymbol{x}$, the jittering kernel function can yield $k(T\boldsymbol{x}, \boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{x})$.

The computation of such a kernel can be time consuming. However, for a RBF kernel, only k_{qj} needs to be considered for the minimization since k_{qq} and k_{jj} are constants and equal 1. Moreover it is argued that, compared to the VSVM method, the jittering kernel can still be faster for the training by making use of kernel caching. Nonetheless, in testing phase, this kernel might be slower since it requires to repeat the steps (1) and (2) for each new test sample.

4.2.2 Tangent distance

Another approach to incorporate knowledge of transformation-invariance is via the distance measurement, not in the space induced by the kernel but in the input space. In order to do so, one can implement a different distance $\rho(\boldsymbol{x}, \boldsymbol{z})$ instead of the Euclidean distance commonly used in radial basis kernels. For the Gaussian kernel, this yields

$$k(\boldsymbol{x}, \boldsymbol{z}) = \exp\left(\frac{-\rho(\boldsymbol{x}, \boldsymbol{z})^2}{2\sigma^2}\right) \tag{24}$$

The use of another distance for transformation-invariance has been extensively studied in [57] for neural networks under the name of tangent distance (TD) and originally incorporated in SVMs as TD kernels by [23]. The main idea is to measure the distance not between the samples $\boldsymbol{x}$ and $\boldsymbol{z}$ but between the sets $P_{\boldsymbol{x}}$ and $P_{\boldsymbol{z}}$. These sets contain all the patterns generated by the transformation T of the samples $\boldsymbol{x}$ and $\boldsymbol{z}$ that leaves the label unchanged. Thus, the distance between a sample and its translation can be made so that it equals zero.

As simple image transformations can correspond to highly non-linear transformations in the input space, the tangent distance uses linear approximations. Considering the transformation $T\boldsymbol{x}$ as a combination of n_L local and simple transformations L_{α_k} of parameter α_k in $[\alpha_k^{min}, \alpha_k^{max}]$, it can be linearly approximated by using the tangent vectors $\boldsymbol{\ell}_{\alpha_k}(\boldsymbol{x})$ as

$$T\boldsymbol{x} \approx \boldsymbol{x} + \sum_{k=1}^{n_L} \alpha_k \boldsymbol{\ell}_{\alpha_k}(\boldsymbol{x}) \tag{25}$$

The tangent distance is thus defined as the minimal Euclidean distance between the linearly approximated sets of all transformed samples

$$\begin{aligned} \rho(P_{\boldsymbol{x}}, P_{\boldsymbol{z}})^2 = & \min_{\alpha_1,\ldots,\alpha_{n_L},\beta_1,\ldots,\beta_{n_L}} \left(\boldsymbol{x} - \boldsymbol{z} + \sum_{k=1}^{n_L} (\alpha_k \boldsymbol{\ell}_{\alpha_k}(\boldsymbol{x}) - \beta_k \boldsymbol{\ell}_{\beta_k}(\boldsymbol{z}))\right)^2 \\ & \text{s.t. } \alpha_k \in [\alpha_k^{min}, \alpha_k^{max}],\ \beta_k \in [\beta_k^{min}, \beta_k^{max}],\ k = 1, \ldots, n_L \end{aligned} \tag{26}$$

where β_k and $\boldsymbol{\ell}_{\beta_k}(\boldsymbol{z})$ correspond to the parameter and the tangent vector for the kth local transformation of $\boldsymbol{z}$.

A similar approach was originally taken in [16] where a joint manifold distance was defined by minimizing a distance between sets of transformed samples. When the transformation is approximated by a Taylor expansion, this method is analogous to the tangent distance method of [57]. In [45], these concepts are considered under the name of object-to-object distance, where an object corresponds to the set of all transformed samples $P_{\boldsymbol{x}}$ or $P_{\boldsymbol{z}}$. Sample-to-object distance is also considered, in which case the transformation of only one of the two samples is allowed. This corresponds to a one-sided tangent distance which is computed for a sample $\boldsymbol{x}$ and an object $P_{\boldsymbol{z}}$ by

$$\rho(\boldsymbol{x}, P_{\boldsymbol{z}})^2 \approx \min_{\beta_1,\ldots,\beta_{n_L}} \left(\boldsymbol{x} - \boldsymbol{z} - \sum_{k=1}^{n_L} \beta_k \boldsymbol{\ell}_{\beta_k}(\boldsymbol{z}) \right)^2 \quad \text{s.t. } \beta_k \in [\beta_k^{min}, \beta_k^{max}],\ k = 1, \ldots, n_L \tag{27}$$

The sample-to-object (or one-sided tangent distance) method can be related to the jittering kernel method. They both amount to compute the kernel $k(\boldsymbol{x}, \boldsymbol{z})$ between the sample $\boldsymbol{x}$ and the closest pattern generated around the center $\boldsymbol{z}$ by a transformation that does not change the class label. The main difference lies in the implementation where the sample-to-object distance can be considered as an analytical form of the one used in jittering kernels, these latter requiring to test every jittered form of the center. Therefore, the sample-to-object method can be faster but introduces restrictions on the class of admissible transformations [45].

Objects can also be coded by local distributions centered at the samples. In this case they are called *soft-objects*. One has then to define a similarity measure between a sample and such an object. Here, tangent vectors can be used to locally approximate the transformations [45] and lead to the Tangent Vector Kernels (TVK) introduced in [44].

4.2.3 Haar-integration kernels

Haar-integration has been introduced in [54] for the construction of invariant features. In a similar approach, Haar-integration has been used to generate invariant kernels known as Haar-integration kernels (HI-kernels) [24]. Consider a standard kernel k_0 and a transformation group $\mathcal{T}$ which contains the admissible transformations (see [54] for a complete definition). The idea is to compute the average of the kernel output $k_0\left(T\boldsymbol{x}, T'\boldsymbol{z}\right)$ over all pairwise combinations of the transformed samples $\left(T\boldsymbol{x}, T'\boldsymbol{z}\right),\ \forall T, T' \in \mathcal{T}$. The HI-kernel k of k_0 with respect to $\mathcal{T}$ is thus

$$k(\boldsymbol{x}, \boldsymbol{z}) = \int_{\mathcal{T}} \int_{\mathcal{T}} k_0\left(T\boldsymbol{x}, T'\boldsymbol{z}\right) dT dT' \tag{28}$$

under the condition of existence and finiteness of the integral (which can be satisfied for instance by discretization of $\mathcal{T}$). An interpretation of this kernel in the feature space F can be given thanks to the following equality [24]

$$\langle \int_{\mathcal{T}} \mathbf{\Phi}\left(T\boldsymbol{x}\right) dT, \int_{\mathcal{T}} \mathbf{\Phi}\left(T'\boldsymbol{z}\right) dT' \rangle = \int_{\mathcal{T}} \int_{\mathcal{T}} \langle \mathbf{\Phi}\left(T\boldsymbol{x}\right), \mathbf{\Phi}\left(T'\boldsymbol{z}\right) \rangle \; dT dT' = k(\boldsymbol{x}, \boldsymbol{z}) \tag{29}$$

where the mapping Φ is defined by (10). In other words, averaging over $k_0\left(T\boldsymbol{x}, T'\boldsymbol{z}\right)$ is equivalent to computing the inner product between the averages $\overline{\mathbf{\Phi}(\boldsymbol{x})}$ and $\overline{\mathbf{\Phi}(\boldsymbol{z})}$ of the sets of transformed samples $\{\mathbf{\Phi}(T\boldsymbol{x})|\ T \in \mathcal{T}\}$ and $\{\mathbf{\Phi}(T'\boldsymbol{z})|\ T' \in \mathcal{T}\}$.

4.2.4 Kernels between sets

In [31], a kernel between sets of vectors is proposed. The idea is to classify the samples defined as sets of d-dimensional vectors $\boldsymbol{x}_i$ and now written

$$\chi = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_i, \ldots, \boldsymbol{x}_n\} \tag{30}$$

where n is the size of the set. This representation allows to intrinsically incorporate invariance to permutations of vectors $\boldsymbol{x}_i$ in the set. For instance, in image recognition, a vector $\boldsymbol{x}_i = [x, y, \gamma]^T$ represents a point of the image identified by its coordinates (x, y) and a gray-level γ. A sample χ is then composed of all the points corresponding to an image. It is clear that the ordering of the vectors inside this sample is irrelevant for the image classification. Thus, the recognition algorithm must include an invariance to permutations of vectors inside a sample, which is included here in the kernel.

The kernel between two sets χ and χ' is defined as the Bhattacharyya's affinity between the distributions p and p' fitted to the sets χ and χ'

$$k(\chi, \chi') = k(p, p') = \int \sqrt{p(\boldsymbol{x})}\sqrt{p'(\boldsymbol{x})}\; d\boldsymbol{x} \tag{31}$$

The approach here is to consider χ and χ' as i.i.d. samples from unknown distributions p and p' from a parametric family $\mathcal{P}$. The kernel requires to fit the distributions p and p' to the sets as an intermediate step, which ensures permutation-invariance [31]. When $\mathcal{P}$ is chosen as the family of multivariate normal distributions $\mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, p and p' are fitted by setting $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$ to their maximum likelihood estimates given by the sample mean and the empirical covariance matrix.

So far, the distributions are fitted in the input space $\mathbb{R}^d$, which might be limited (for instance, $d = 3$ for image recognition). However, the method is extended in [31] with an additional kernel $\kappa : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ defined between the vectors $\boldsymbol{x}$ in order to consider p and p' as distributions on the feature

space induced by κ. In this case, a regularized estimate of the covariance matrix, involving the computation of eigenvectors in the feature space by Kernel Principal Component Analysis (KPCA) [52], is used.

Another similar approach, independently developed at the same time, can be found in [70], where a positive definite kernel is defined over sets of vectors represented as matrices. Considering two matrices of identical sizes $\boldsymbol{A} = [\boldsymbol{\Phi}(\boldsymbol{a}_1), \ldots, \boldsymbol{\Phi}(\boldsymbol{a}_n)]$ and $\boldsymbol{B} = [\boldsymbol{\Phi}(\boldsymbol{b}_1), \ldots, \boldsymbol{\Phi}(\boldsymbol{b}_n)]$, where $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ are vectors of $\mathbb{R}^d$ and $\boldsymbol{\Phi}(.)$ is defined by (10), the proposed kernel between these matrices with column order invariance is given by

$$k(\boldsymbol{A}, \boldsymbol{B}) = \prod_{i=1}^{n} \cos(\theta_i) \tag{32}$$

where the θ_i stand for the principal angles in the feature space. In [70], a method is proposed to compute the principal angles in feature space using only inner products between columns of the input matrices. This extension allows to introduce an additional kernel as in the method of [31] in order to deal with non-linear cases without requiring to explicitly compute the feature map.

4.2.5 Knowledge-driven kernel selection

All previously described methods involving a modification of the kernel aim at building invariant kernels. The following method applies to unbalanced training sets and the cases where prior knowledge indicates that the negative class includes a wide variety of samples with only a few available for training. For instance, in face recognition, when the positive class includes the training images of a particular man, the training set cannot contain all the possible faces of the other men for the negative class.

In image retrieval, this problem has been tackled in [67] by a knowledge-driven kernel design (KDKD) procedure. The authors highlight the fact that, when the training set is small, the data cannot effectively represent the true distributions of the positive and negative classes, especially the negative one. Based on this prior knowledge, the kernel is designed so that, in the feature space, the positive samples are tightly clustered while the negative samples are pushed away from the positive ones, anywhere, but scattered.

In practice, the kernel k is designed by maximizing the ratio between the scatter $tr(\boldsymbol{S}_{np}^{\Phi})$ of the negative samples ($\boldsymbol{x}_i \in \mathcal{D}_n$) and the scatter $tr(\boldsymbol{S}_{p}^{\Phi})$ of the positive ones ($\boldsymbol{x}_i \in \mathcal{D}_p$), with respect to the mean $\boldsymbol{m}_p^{\Phi}$ of the positive ones

$$\mathcal{J}(k, \boldsymbol{\theta}) = \frac{tr(\boldsymbol{S}_{np}^{\Phi})}{tr(\boldsymbol{S}_{p}^{\Phi})} \tag{33}$$

where the matrices $\boldsymbol{S}_{np}^{\Phi}$ and $\boldsymbol{S}_{p}^{\Phi}$ are defined by

$$\boldsymbol{S}_{np}^{\Phi} = \sum_{\boldsymbol{x}_i \in \mathcal{D}_n} \left(\boldsymbol{\Phi}(\boldsymbol{x}_i) - \boldsymbol{m}_p^{\Phi}\right) \left(\boldsymbol{\Phi}(\boldsymbol{x}_i) - \boldsymbol{m}_p^{\Phi}\right)^T \tag{34}$$

$$\boldsymbol{S}_{p}^{\Phi} = \sum_{\boldsymbol{x}_i \in \mathcal{D}_p} \left(\boldsymbol{\Phi}(\boldsymbol{x}_i) - \boldsymbol{m}_p^{\Phi}\right) \left(\boldsymbol{\Phi}(\boldsymbol{x}_i) - \boldsymbol{m}_p^{\Phi}\right)^T \tag{35}$$

where $\boldsymbol{\Phi}(\boldsymbol{x}_i)$ is defined by (10). The traces of the matrices can be computed from the kernel function as described in [67] and the criterion has continuous first and second order derivatives with respect to the kernel parameters as long as the kernel function has. However, non-linear optimization techniques are required to solve the problem.

The maximization of the ratio (33) can be used to find the optimal kernel function amongst a set of admissible kernels, but also to tune the parameters of a previously chosen kernel.

4.3 Optimization methods

This Section presents methods that incorporate prior knowledge directly in the problem formulation: invariant kernels, Semidefinite Programming Machines (SDPM), Invariant SimpleSVM (ISSVM), Knowledge-Based Linear Programming (KBLP), Non-Linear Knowledge-Based Linear Programming (NLKBLP) and π-SVM. Though it may be argued that the first one belongs to the category of kernel methods, it is derived from a regularization approach minimizing a composite criterion. It is thus categorized as an optimization based method. The first three methods of the list aim at incorporating transformation-invariance, whereas the KBLP method considers class-invariance in polyhedral regions of the input space. From a method originally formulated for support vector regression, an extension of KBLP to arbitrary non-linear domains, NLKBLP, is proposed in Sect. 4.3.5 for classification. The last method exposed in this review, π-SVM, concerns permutation-invariance for SVMs that classify sets of elements instead of vectors.

4.3.1 Invariant kernels

Regularization of the cost function has been extensively used for neural networks [19,4], allowing to incorporate prior knowledge on a property of the function to estimate (usually the smoothness). But the implicit inclusion of regularization in SVMs, equivalent to regularization networks [59,14], might explain why few articles studied the application of regularization techniques for the incorporation of other forms of prior knowledge into SVMs. However, in the case of classification, the addition of a term to be minimized in the

cost function for this purpose has been considered. Nonetheless, it results in a modification of the kernel rather than a modification of the optimization problem as exposed in the following.

The authors of [51] incorporated local invariance in the sense of (15) and proposed invariant kernels. Defining the tangent vectors by

$$d\boldsymbol{x}_i = \left.\frac{\partial}{\partial\theta}\right|_{\theta=0} T_\theta \boldsymbol{x}_i \tag{36}$$

allows to include the local invariance (15) in the learning of a linear SVM by minimizing

$$\frac{1}{N}\sum_{i=1}^{N}\left(\boldsymbol{w}^T(d\boldsymbol{x}_i)\right)^2 \tag{37}$$

and thus making the weight vector $\boldsymbol{w}$ as orthogonal as possible to the tangent vectors. For the original QP formulation and a linear kernel, the regularized cost becomes

$$J(\boldsymbol{w}) = (1-\gamma)\|\boldsymbol{w}\|^2 + \gamma\sum_{i=1}^{N}\left(\boldsymbol{w}^T(d\boldsymbol{x}_i)\right) \tag{38}$$

where $\gamma \in [0,1]$ controls the trade-off between the standard SVM ($\gamma = 0$) and a full enforcement of the orthogonality between the hyperplane and the invariance directions ($\gamma \to 1$). Let define $\boldsymbol{C}_\gamma$ as the square root of the regularized covariance matrix of the tangent vectors

$$\boldsymbol{C}_\gamma = \left((1-\gamma)\boldsymbol{I} + \gamma\sum_{i=1}^{N} d\boldsymbol{x}_i d\boldsymbol{x}_i^T\right)^{\frac{1}{2}} \tag{39}$$

Then, minimizing the regularized cost (38) under the original constraints (4) leads to a standard SVM problem [51], yielding the output function

$$f(\boldsymbol{x}) = \sum_{i=1}^{N}\alpha_i y_i \langle \boldsymbol{C}_\gamma^{-1}\boldsymbol{x}_i, \boldsymbol{C}_\gamma^{-1}\boldsymbol{x}\rangle + b \tag{40}$$

Thus a linear invariant SVM is equivalent to a standard SVM where the input is first transformed via the linear mapping $\boldsymbol{x} \mapsto \boldsymbol{C}_\gamma^{-1}\boldsymbol{x}$.

In order to extend directly the approach to non-linear kernels [7], one would have to compute the matrix $\boldsymbol{C}_\gamma$ in the feature space F by

$$\boldsymbol{C}_\gamma = \left((1-\gamma)\boldsymbol{I} + \gamma\sum_{i=1}^{N} d\boldsymbol{\Phi}(\boldsymbol{x}_i) d\boldsymbol{\Phi}(\boldsymbol{x}_i)^T\right)^{\frac{1}{2}} \tag{41}$$

where $\boldsymbol{\Phi}(\boldsymbol{x}_i)$ is defined by (10), and use the new kernel

$$k(\boldsymbol{x}_i, \boldsymbol{x}) = \langle \boldsymbol{C}_\gamma^{-1}\boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{C}_\gamma^{-1}\boldsymbol{\Phi}(\boldsymbol{x})\rangle = \boldsymbol{\Phi}(\boldsymbol{x}_i)^T\boldsymbol{C}_\gamma^{-2}\boldsymbol{\Phi}(\boldsymbol{x}) \tag{42}$$

which is impossible to do directly because of the high dimensionality of F and the implicit nature of Φ. Nonetheless, two methods were proposed in [7] but still suffer from computational problems when applied to large datasets or when more than one invariances are considered.

This approach for the incorporation of invariance can be related to the virtual samples method that simply adds the transformed samples to the training set (see Section 4.1.1) and some equivalence between the two methods can be shown [35]. However, the invariant SVM does not only make the *class* invariant to the transformation but also the *real value* of the output (which can be considered as a class-conditional probability) [7].

4.3.2 Semidefinite Programming Machines (SDPM)

In [20], another formulation for the large margin classifier is developed for the incorporation of transformation-invariance. The aim is to find an optimal separating hyperplane between trajectories rather than between points. In practice, the trajectories, defined as sets of the type $\{T_\theta \boldsymbol{x}_i : \theta \in \mathbb{R}\}$, are based on training samples $\boldsymbol{x}_i$ and a differentiable transformation T of parameter θ to which the class is known to be invariant. The problem can be solved by approximating T by a transformation $\tilde{T}$ polynomial in θ that can be a Taylor expansion of the form

$$T_\theta \boldsymbol{x}_i \approx \tilde{T}_\theta \boldsymbol{x}_i = \sum_{j=0}^{r} \theta^j \left(\frac{1}{j!} \left. \frac{d^j T_\theta \boldsymbol{x}_i}{d\theta^j} \right|_{\theta=0} \right) = \tilde{\boldsymbol{X}}_i^T \boldsymbol{\theta} \tag{43}$$

where the $(r+1) \times d$-dimensional matrix $\tilde{\boldsymbol{X}}_i$ contains the derivative components and $\boldsymbol{\theta} = [1 \ \theta \ \theta^2 \ \ldots \ \theta^r]^T$. For this type of transformations, the problem of finding the optimal separating hyperplane between trajectories can be formulated as

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2, \quad \text{s.t.} \ \ y_i \boldsymbol{w}^T \tilde{\boldsymbol{X}}_i^T \boldsymbol{\theta} \leq 0, \ \forall \theta \in \mathbb{R}, \ i = 1, \ldots, N \tag{44}$$

For this problem, the authors of [20] propose an equivalent semidefinite program (SDP) [62], for which efficient algorithms exist. They also show that the resulting expansion of the optimal weight vector $\boldsymbol{w}^*$ in terms of $\tilde{\boldsymbol{X}}_i$ is sparse. Moreover, not only the examples $\tilde{\boldsymbol{X}}_i$ are determined but also their corresponding optimal transformation parameter θ_i^*. Thus, the so-called Semidefinite Programming Machines (SDPM) extends the idea of virtual samples (Sect. 4.1.1), since truly virtual samples that are not in the training set are used as SVs.

However, practical issues regarding the application of SDPM to non-linear classification with kernels remain open.

4.3.3 Invariant SimpleSVM

The authors of [36] propose a general framework for the incorporation of transformation-invariance into the learning based on a modification of the problem formulation. For the hard-margin SVM, the problem reads

$$\begin{aligned} &\min_{g,b} \ \frac{1}{2}\|g\|_H^2 \\ &\text{s.t. } y_i \left(g(T_\theta \boldsymbol{x}_i) + b\right) \geq 1, \ i = 1, \ldots, N, \ \theta \in \Theta \end{aligned} \tag{45}$$

where the function g lies in the Hilbert space H of admissible functions. This setting allows to ensure that not only the training sample $\boldsymbol{x}_i = T_0 \boldsymbol{x}_i$, but also the admissible transformations of this sample $T_\theta \boldsymbol{x}_i$, are classified as belonging to class the y_i. To solve the problem, the method requires a discretization of the parameter space Θ based on the assumption that only a finite number of values for θ will yield support vectors. This is reasonable since for the hard-margin SVM, only samples that lie exactly on the margin borders are SVs. In this case, a dual form of the problem with a finite number of Lagrange multipliers $\alpha_i(\theta)$ can lead to the solution. However, for the soft-margin case, any sample lying inside the margin corresponds to a SV. Thus the trajectory of a transformation that goes through the margin would yield an infinite number of SVs.

As highlighted in [36], this method can include other methods that incorporate invariance as follows:

- the virtual samples approach (see Sect. 4.1.1) is recovered after a discretization of the parameter space Θ;
- the tangent distance based methods (Sect. 4.3.1 and 4.2.2) are recovered by approximating the transformation $T_\theta \boldsymbol{x}_i$ by a first order polynomial;
- the semidefinite programming machine (SDPM, Sect. 4.3.2) is recovered if the transformation $T_\theta \boldsymbol{x}_i$ is approximated by a transformation polynomial in θ, such as a Taylor expansion.

In this framework, the authors proposed an efficient algorithm called Invariant SimpleSVM (ISSVM) based on the SimpleSVM algorithm developed in [66]. However, this algorithm requires the discretization of the parameter space Θ for the non-separable case, which is not necessary for SDPM which considers all θ in $\mathbb{R}$.

4.3.4 Knowledge-Based Linear Programming (KBLP)

The following methods consider the incorporation of prior knowledge into support vector learning by the addition of constraints to the optimization problem. In this framework, class-invariance inside polyhedral regions has been

introduced for linear classification in [17] and was then extended to the non-linear case via a reformulation of the kernel in [18]. These two methods are regrouped under the name Knowledge-Based Linear Programming (KBLP).

The learning machine considered here uses the linear programming formulation (13). Assuming as prior knowledge that all the points $\boldsymbol{x}$ on a polyhedral domain $\mathcal{P} = \{\boldsymbol{x}|\ \boldsymbol{B}_+\boldsymbol{x} \leq \boldsymbol{d}_+\}$ are positive samples ($y = +1$) can be written as the implication

$$\boldsymbol{B}_+\boldsymbol{x} \leq \boldsymbol{d}_+ \ \Rightarrow\ \boldsymbol{w}^T\boldsymbol{x} + b \geq 1 \tag{46}$$

with, for a domain of dimension n, $\boldsymbol{B}_+ \in \mathbb{R}^{n\times d}$, $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{d}_+ \in \mathbb{R}^n$. The implication (46) can be transformed for a linear SVM in an equivalent system of linear inequalities having the solution $\boldsymbol{u} \in \mathbb{R}^n$ [17]

$$\boldsymbol{B}_+^T\boldsymbol{u} + \boldsymbol{w} = \boldsymbol{0}, \quad \boldsymbol{d}_+^T\boldsymbol{u} - b + 1 \leq 0,\ \boldsymbol{u} \geq \boldsymbol{0} \tag{47}$$

For prior knowledge on negative samples ($y = -1$), we have

$$\boldsymbol{B}_-\boldsymbol{x} \leq \boldsymbol{d}_- \ \Rightarrow\ \boldsymbol{w}^T\boldsymbol{x} + b \leq -1 \tag{48}$$

which is equivalent to

$$\boldsymbol{B}_-^T\boldsymbol{u} - \boldsymbol{w} = \boldsymbol{0}, \quad \boldsymbol{d}_-^T\boldsymbol{u} + b + 1 \leq 0,\ \boldsymbol{u} \geq \boldsymbol{0} \tag{49}$$

This result is then extended for SVMs with non-linear kernels by assuming that $\boldsymbol{x} = \boldsymbol{X}^T\boldsymbol{t}$ is a linear combination of the training samples. For the positive class, the "kernelized" prior knowledge becomes

$$\boldsymbol{K}(\boldsymbol{B}_+^T, \boldsymbol{X}^T)\boldsymbol{t} \leq \boldsymbol{d}_+ \ \Rightarrow\ \boldsymbol{\alpha}^T\boldsymbol{D}\boldsymbol{K}\boldsymbol{t} + b \geq 1 \tag{50}$$

with the diagonal matrix $\boldsymbol{D} = \mathrm{diag}(y_1, \ldots, y_i, \ldots, y_N)$ and $\boldsymbol{\alpha}$ defined as in (12), the following system of linear inequalities is equivalent

$$\boldsymbol{K}(\boldsymbol{X}^T, \boldsymbol{B}_+^T)\boldsymbol{u} + \boldsymbol{K}\boldsymbol{D}\boldsymbol{\alpha} = \boldsymbol{0}, \quad \boldsymbol{d}_+^T\boldsymbol{u} - b + 1 \leq 0,\ \boldsymbol{u} \geq \boldsymbol{0} \tag{51}$$

These inequalities can then be easily incorporated to the linear program (13). With the introduction of $N + 1$ slack variables $\boldsymbol{z} = [z_1, \ldots, z_i, \ldots, z_N]^T$ and ζ, this leads to the addition of $N + 1$ linear constraints (not counting $\boldsymbol{u} \geq \boldsymbol{0}$

and $\zeta \geq 0$) as

$$\min_{(\boldsymbol{\alpha},b,\boldsymbol{\xi}\geq\mathbf{0},\boldsymbol{a},\boldsymbol{u}\geq\mathbf{0},\boldsymbol{z},\zeta\geq 0)} \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} + \mu_1\mathbf{1}^T\boldsymbol{z} + \mu_2\zeta$$

$$\begin{array}{lcll} \text{s.t.} & & \boldsymbol{D}(\boldsymbol{K}\boldsymbol{D}\boldsymbol{\alpha} + b\mathbf{1}) & \geq \mathbf{1} - \boldsymbol{\xi} \\ & -\boldsymbol{a} \leq & \boldsymbol{\alpha} & \leq \boldsymbol{a} \\ & -\boldsymbol{z} \leq & \boldsymbol{K}(\boldsymbol{X}^T, \boldsymbol{B}_+^T)\boldsymbol{u} + \boldsymbol{K}\boldsymbol{D}\boldsymbol{\alpha} & \leq \boldsymbol{z} \\ & & \boldsymbol{d}_+^T\boldsymbol{u} - b + 1 & \leq \zeta \end{array} \tag{52}$$

where μ_1 and μ_2 are two trade-off parameters between the training on the data and the learning of the prior knowledge.

Similar constraints can be derived for prior knowledge on negative samples and added to the problem [18]. As prior knowledge on a polyhedral set only requires the addition of a set of linear constraints, knowledge on many regions for the two classes can be easily combined and included to the problem.

It must be noticed that the three kernels appearing in (52) could be distinct kernels and do not need to be positive semidefinite.

4.3.5 Non-Linear Knowledge Based Linear Programming

In the framework of regression and kernel approximation of functions, Mangasarian and his coworkers proposed a new approach based on a non-linear formulation of the knowledge [38] to overcome the lacks of the previously described KBLP method. It is presented here to allow its extension to classification and thus to class-invariance inside an input domain. The prior knowledge can now be considered on any non-linear region of the input space and takes the general form

$$\boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0} \;\Rightarrow\; f(\boldsymbol{x}) \geq h(\boldsymbol{x}) \tag{53}$$

which is equivalent to

$$f(\boldsymbol{x}) - h(\boldsymbol{x}) + \boldsymbol{v}^T\boldsymbol{g}(\boldsymbol{x}) \geq 0,\ \forall \boldsymbol{x} \tag{54}$$

for $\boldsymbol{v} \geq \mathbf{0}$ [38]. In this formulation, $f, \boldsymbol{g}, h$ are arbitrary non-linear functions. Indeed the demonstration of the equivalence requires $f, \boldsymbol{g}, -h$ to be convex but only for the implication (53) $\Rightarrow$ (54). The implication (54) $\Rightarrow$ (53), in which we are interested, holds for any $f, \boldsymbol{g}, h$. Nonetheless, the inequality (54) must be verified $\forall \boldsymbol{x}$ and thus cannot directly be included in the linear program as a finite set of constraints. To overcome this, a discretization of the knowledge is performed over a set of points $\{\boldsymbol{x}_p\} \subset \{\boldsymbol{x} | \boldsymbol{g}(\boldsymbol{x}) \leq \mathbf{0}\}$, yielding a finite set of constraints.

For the classification case, we can set $f(\boldsymbol{x})$ to the real value of the output of the classifier (the bracketed expression in (12) without the sign function) $f(\boldsymbol{x}) = \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)\boldsymbol{D}\boldsymbol{\alpha} + b$ and $h(\boldsymbol{x}) = +1$. Of course, if the region considered is known to belong to the negative class instead of the positive class, the setting becomes $f(\boldsymbol{x}) = -\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X}^T)\boldsymbol{D}\boldsymbol{\alpha} - b$ and $h(\boldsymbol{x}) = -1$. Associating slack variables z_p to the N_p points of discretization $\{\boldsymbol{x}_p\}$ gives the Nonlinear Knowledge-Based Linear Program (NLKBLP)

$$\begin{aligned} \min_{(\alpha, b, \boldsymbol{\xi}\geq\boldsymbol{0}, \boldsymbol{a}, \boldsymbol{v}\geq\boldsymbol{0}, z_p\geq 0)} \quad & \mathbf{1}^T\boldsymbol{a} + C\mathbf{1}^T\boldsymbol{\xi} + \mu\textstyle\sum_p^{N_p} z_p & \\ \text{s.t.} \quad & \boldsymbol{D}(\boldsymbol{K}\boldsymbol{D}\boldsymbol{\alpha} + \mathbf{1}b) & \geq \mathbf{1} - \boldsymbol{\xi} \\ -\boldsymbol{a} \leq & \boldsymbol{\alpha} & \leq \boldsymbol{a} \\ & f(\boldsymbol{x}_p) - h(\boldsymbol{x}_p) + \boldsymbol{v}^T\boldsymbol{g}(\boldsymbol{x}_p) + z_p \geq 0, \ p = 1, \ldots, N_p & \end{aligned} \tag{55}$$

A difference between this method and the polyhedral method is that the constraints are applied on an arbitrary discretized domain. With a polyhedral domain, the constraints hold on the whole domain (without discretization).

It is thus possible to include prior knowledge such as the class-membership of an arbitrary region into support vector learning. For each region a set of constraints is added to the linear program. The method is thus only limited by computational power.

4.3.6 Permutation-invariant SVM (π-SVM)

The paper [56] focuses on the issue of Section 4.2.4, i.e. to build a classifier separating sets of vectors (here in matrix form) that incorporates permutation-invariance (here between rows of matrices). But the approach is rather different from the ones of [31] or [70]. Here, the permutation invariance is not incorporated in the kernel but in the learning machine itself. To do so, a SVM with matrix inputs $\boldsymbol{Z}^i \in \mathbb{R}^{m\times d}$ instead of vectors $\boldsymbol{x}_i$ is considered by defining a function $\pi : \mathbb{R}^{m\times d} \times \mathbb{R}^{m\times d} \to \mathbb{R}$ as

$$\pi(\boldsymbol{A}, \boldsymbol{B}) = \sum_{j=1}^{d} \langle \boldsymbol{A}_j, \boldsymbol{B}_j \rangle = \sum_{j=1}^{d}\sum_{k=1}^{m} A_{jk}B_{jk} \tag{56}$$

and the norm of a matrix as $\|\boldsymbol{A}\| = \sqrt{\sum_{j=1}^{d} \|\boldsymbol{A}_j\|^2}$. The training of such a SVM can still be written as (8) by replacing the vector $\boldsymbol{w}$ by a matrix $\boldsymbol{W}$ in $\mathbb{R}^{m\times d}$ and replacing the inner product $\langle \boldsymbol{x}_i, \boldsymbol{w} \rangle$ in the constraints by $\pi(\boldsymbol{Z}^i, \boldsymbol{W})$. Similarly, the class of a sample $\boldsymbol{Z}$ is given by $f(\boldsymbol{Z}) = \text{sign}(\pi(\boldsymbol{Z}, \boldsymbol{W}) + b)$. The margin M of this classifier can be defined as the minimal value for $y_i(\pi(\boldsymbol{Z}^i, \boldsymbol{W}) + b)$ over the training set, since maximizing M would lead to a better separation

of the training data.

The main steps of the proposed procedure to train a permutation-invariant SVM (π-SVM) [56] are as follows:

(1) compute the radius R and centroid of the smallest hypersphere enclosing all training data;
(2) solve the SVM on the training data composed of matrices;
(3) find the permutation of rows for each training sample that both minimizes the radius R and maximizes the margin (with a chosen trade-off parameter);
(4) permute the rows of the training matrices accordingly;
(5) repeat from step 1 until some criterion is met.

For all these steps, efficient algorithms, of which the description can be found in [56], exist. The idea here is to permute the rows of training matrices so as to minimize the bound on the generalization error based on the ratio between the radius of the data and the margin. The class of a test sample $\boldsymbol{Z}$ is then determined by $f(\boldsymbol{Z})$ computed for the permutation of rows of $\boldsymbol{Z}$ that yields the larger margin.

5 Discussions

This Section starts with some general remarks on the main directions for the incorporation of prior knowledge into SVM. Then, a technical discussion on particular aspects of the methods is proposed to highlight the links between the methods. Notably, a unifying framework from an implementation point of view is given in Section 5.2.1.

5.1 General issues

In this overview of the methods, the approaches to incorporating prior knowledge into SVM are classified in the three categories defined in [53]: sample methods, kernel methods and optimization methods.

Sample methods are often chosen in practice for their simplicity of use. On the contrary, kernel methods may suffer from computational issues. However, most of the current research aim at building invariant kernels. One reason is that these kernels may also be directly applied to other kernel-based classifiers such as Kernel Principal Component Analysis (KPCA) [52] or Kernel Fisher Discriminant Analysis (KFDA) [39].

Also, most of the work focus on one particular type of prior knowledge: class-invariance to a transformation of the input. This can be explained by the availability of this type of prior knowledge in many of the pattern recognition problems such as image recognition and all its various applications. However, on a smaller scale, other forms of invariances are also studied when, for instance, considering structured inputs such as matrices. In this setting invariance to permutations of rows as proposed by [31], [70] or [56] can be crucial for the problem. The methods proposed by [17], [18] and their extension based on [38] allow to include some class-invariance knowledge on regions of the input space, which might be interesting if, for instance, these regions lack training samples.

Methods that incorporate knowledge on the data, such as the Weighted-SVM [72], are also interesting since they allow to include knowledge on the experimental setup used to provide the training data. From a practical point of view, it is clear that, in real-world applications, the process of gathering data may suffer from a weak or variable accuracy. Being able to track this and learn in accordance is an important issue. As an example, the labeling of images in handwriting recognition is not always exact and can sometimes differ with respect to the person who is asked to label a pattern. As highlighted in [60], a certain amount of errors made by the classifiers are due to images that humans have difficulty in identifying them because of cursive writing, degradation and distortion due to the quality of the scanner or the width of the tip of the writing instrument. These problems are present and identifiable in most of the common databases (see [60] for an analysis on the MNIST, CENPARMI, USPS and NIST SD 19 databases) and should be taken into account to be able to improve the performances of the recognition systems.

5.2 Technical discussions

In the following, a regularization framework is used to regroup both the sample and optimization methods. A comparison of the invariant kernel methods is then proposed to show the different spaces in which the distance measures are considered by the different methods. The end of this Section is dedicated to the perspective of combining the methods.

5.2.1 Sample and constrained methods in a regularization framework

Though stemming from different approaches, many of the presented methods can be seen in a regularization framework from the way they are implemented. Actually both sample and constrained methods amount to minimize a com-

posite criterion J_C with additional constraints

$$\begin{aligned} \min \ & J_C = J_{SVM} + J_{PK} \\ \text{s.t. } & \boldsymbol{c}_{SVM} \leq \mathbf{0} \\ & \boldsymbol{c}_{PK} \leq \mathbf{0} \end{aligned} \tag{57}$$

where J_{SVM} and $\boldsymbol{c}_{SVM}$ correspond to the usual SVM criterion and constraints (8), whereas J_{PK} and $\boldsymbol{c}_{PK}$ are added to the problem to include the prior knowledge. Their setting for the different methods are derived in the following and summarized in Table 1.

All virtual sample methods (Sect. 4.1.1) basically add new samples to the training set, which simply corresponds to augmenting the size of the training set N by the number of virtual samples N_V in the optimization problem (8). Separating the training samples $\boldsymbol{x}_i$ from the virtual samples $\tilde{\boldsymbol{x}}_i$ in the writing yields the setting of Table 1 for (57).

Though originally developed for different purposes and from different points of view, all the methods based on weighting (Sect. 4.1.2) can be written as a standard SVM problem with the parameter C set to a different value C_i for each sample. For asymmetric margins, C_i can only take two values C^+ and C^- depending on the class of the sample $\boldsymbol{x}_i$: positive (for i in $\mathcal{P} = \{i : \ y_i = +1\}$) or negative (for i in $\mathcal{N} = \{i : \ y_i = -1\}$). Considering C as an average weight allows to write the problem as in (57) with the settings of Table 1 for asymmetric margin methods and the Weighted-SVM. Besides, the method of [68] for the incorporation of unlabeled samples can be seen as a mixture of virtual samples and weighting by considering the N_U unlabeled samples as extra samples $\tilde{\boldsymbol{x}}_i$ weighted by C_i.

The KBLP and NLKBLP problems (Sect. 4.3.4 and 4.3.5) are directly formulated as in (57), except for J_{SVM} and $\boldsymbol{c}_{SVM}$, which correspond now to the criterion and constraints used by the linear programming form of SVM (13). Because of the required discretization of the domain of knowledge, the NLKBLP method can be seen as adding virtual samples to the training set. In this case, the samples are not generated by transformations of other samples as usual, but chosen in the input space by discretization of a region that might actually contain no sample. An interesting point is that though basically corresponding to the same problem from an optimization viewpoint, the VSVM (Sect. 4.1.1) and NLKBLP methods are radically different. The first one aims at improving the boundary between two regions with training samples that are close but that belong to different classes. The second one becomes of particular interest when enhancing the decision function in a region of input space that lacks training samples but on which prior knowledge is available.

The methods of Section 4.3.1 for the incorporation of invariances conform obviously to this framework since they are initially formulated as the mini-

Table 1
Setting for the regularization framework with respect to the different methods.

Method	J_{PK}	$\boldsymbol{c}_{PK} \leq \mathbf{0}$
Virtual samples	$C \sum_{i=1}^{N_V} \tilde{\xi}_i$	$\tilde{y}_i(\langle \tilde{\boldsymbol{x}}_i, \boldsymbol{w} \rangle + b) \geq 1 - \tilde{\xi}_i, i = 1, \ldots, N_V$
Asymmetric margin	$(C^+ - C) \sum_{i \in \mathcal{P}} \xi_i + (C^- - C) \sum_{i \in \mathcal{N}} \xi_i$	
WSVM	$\sum_{i=1}^{N} (C_i - C) \xi_i$	
Unlabeled samples	$\sum_{i=1}^{N_U} C_i \tilde{\xi}_i$	$\tilde{y}_i(\langle \tilde{\boldsymbol{x}}_i, \boldsymbol{w} \rangle + b) \geq 1 - \tilde{\xi}_i$
KBLP	$\mu_1 \sum_{i=1}^{N} z_i + \mu_2 \zeta$	$-\boldsymbol{z} \leq \boldsymbol{K}(\boldsymbol{X}^T, \boldsymbol{B}^T)\boldsymbol{u} + \boldsymbol{K}\boldsymbol{D}\boldsymbol{\alpha} \leq \boldsymbol{z}$, $\boldsymbol{d}^T \boldsymbol{u} - b + 1 \leq \zeta$
NLKBLP	$\mu \sum_{p}^{N_p} z_p$	$f(\boldsymbol{x}_p) - h(\boldsymbol{x}_p) + \boldsymbol{v}^T \boldsymbol{g}(\boldsymbol{x}_p) + z_p \geq 0$, $p = 1, \ldots, N_p$

mization of a composite criterion. However, as they correspond in practice to a modification of the kernel, they do not appear in Table 1, which focuses on the practical implementation of the methods.

5.2.2 Kernel methods: invariance via distance measure, but in which space?

Most of the kernel methods presented in Section 4.2 implement transformation invariance by modifying the computation of the distance between the patterns. These methods are summarized in Table 2. It can be noticed that while the jittering kernel minimizes a distance in the feature space F, the tangent distance (TD) and the sample-to-object or object-to-object distances are considered in the input space. On the other hand, the Haar-integration kernel does not look for a minimal distance, but rather computes the distance in F between averages over transformed samples.

5.3 Combinations of methods

Combining methods from different categories with respect to our classification seems possible since they act on different parts of the problem. For example, nothing prevents us from using a jittering kernel on an extended training set with prior knowledge on polyhedral regions of input space. But this does not

Table 2
Invariant kernel functions. $\mathcal{Z}$ contains all the jittered forms of $\boldsymbol{z}$.

Method	kernel function	specificity
Jittering kernels	$k(\boldsymbol{x},\boldsymbol{z}) = k(\boldsymbol{x},\hat{\boldsymbol{z}})$	$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}\in\mathcal{Z}}{\arg\min} \|\boldsymbol{\Phi}(\boldsymbol{x}) - \boldsymbol{\Phi}(\boldsymbol{z})\|_F$
One-sided TD (sample-to-object)	$k(\boldsymbol{x},\boldsymbol{z}) = k(\boldsymbol{x},\hat{\boldsymbol{z}})$	$\hat{\boldsymbol{z}} = \underset{\boldsymbol{z}\in P_{\boldsymbol{z}}}{\arg\min} \|\boldsymbol{x} - \boldsymbol{z}\|$
Two-sided TD (object-to-object)	$k(\boldsymbol{x},\boldsymbol{z}) = k(\hat{\boldsymbol{x}},\hat{\boldsymbol{z}})$	$(\hat{\boldsymbol{x}},\hat{\boldsymbol{z}}) = \underset{(\boldsymbol{x},\boldsymbol{z})\in P_{\boldsymbol{x}}\times P_{\boldsymbol{z}}}{\arg\min} \|\boldsymbol{x} - \boldsymbol{z}\|$
Haar-integration	$k(\boldsymbol{x},\boldsymbol{z}) = \overline{k_0(\boldsymbol{x},\boldsymbol{z})}$	$\overline{k_0(\boldsymbol{x},\boldsymbol{z})} = \langle\overline{\boldsymbol{\Phi}(\boldsymbol{z})}, \overline{\boldsymbol{\Phi}(\boldsymbol{x})}\rangle$

exclude combinations of methods of the same category. Actually all the methods of Table 1 can be combined since they all amount to the addition of a term in the criterion and optionally some constraints. Such combinations become interesting when two methods are used for different purposes. Consider the example of character recognition for which we have seen that generating virtual samples by a known transformation such as translation helps the classifier to be invariant to translations. Also, random transformations such as elastic distortions lead to even better results [58] by incorporating invariance to small random variations of the image. But elastic distortion may sometimes yield very distorted images that become unrecognizable [32] and can mislead the classifier. This drawback could be diminished by combining elastic distortions with a Weighted-SVM and associating a smaller confidence to the virtual samples than to the original training ones. Thus, the combination of virtual samples and weighting of samples can be used to incorporate invariance to an approximately known transformation. The weights C_i associated to the virtual samples are then a confidence measure of the transformation that generated these samples.

However, the combination of methods leads to an increase of the algorithm complexity. The methods must be chosen with care by looking at their complementarity in order to yield a respectable improvment.

6 Conclusion

The fundamentals of the Support Vector Machines (SVMs) for classification have been presented together with the different formulations of the optimization problem resulting from the training of such machines. A review of the literature concerning the incorporation of prior knowledge into SVMs has been exposed. The methods are classified, with respect to the categorization of [53],

in three categories depending on the implementation approach (via samples, in the kernel or in the problem formulation). Two main types of prior knowledge that can be included by these methods have been considered: class-invariance and knowledge on the data. Most of the work in this field has been focused so far on transformation-invariance, either via the kernel function or via extended training sets. Nonetheless, a recent approach considers prior knowledge on a polyhedral domain of the input space for which the class is known. It has been extended here for arbitrary regions and results in the addition of linear constraints to the optimization problem, thus providing an easy mean to code knowledge on multiple regions for the two classes. Finally, a regularization framework has been used to regroup both the sample and constrained methods from an implementation point of of view.

Being able to include expert knowledge in the learning will be a key element in the future for the increase of classifiers performance on benchmark datasets and practical applications. Further research might explore other forms of prior knowledge together with optimized algorithms for their implementations. Also, the combination of different types of knowledge might be explored for practical applications. In addition, an interesting problem not covered in this review, though it may be regarded as the inclusion of prior knowledge, is the classification of structured inputs such as graphs or strings. Regarding this issue, the reader can refer to the book [55] and the references therein that provide a framework for building kernels for structured data.

References

[1] Y. S. Abu-Mostafa, Learning from hints in neural networks, Journal of Complexity 6 (2) (1990) 192–198.

[2] Y. S. Abu-Mostafa, Learning from hints, Journal of Complexity 10 (1) (1994) 165–178.

[3] K. P. Bennett, Combining support vector and mathematical programming methods for classification, in: Schölkopf et al. [50], pp. 307–326.

[4] C. M. Bishop, Neural Networks for Pattern Recognition, Oxford University Press, 1995.

[5] M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, T. Furey, M. Ares Jr, D. Haussler, Knowledge-based analysis of microarray gene expression data by using support vector machines, Proc. of the National Academy of Sciences 97 (2000) 262–267.

[6] C. Chang, C. Lin, LibSVM: a library for support vector machines, http://www.csie.ntu.edu.tw/∼cjlin/libsvm/ (2001).

[7] O. Chapelle, B. Schölkopf, Incorporating invariances in non-linear support vector machines, in: T. G. Dietterich, S. Becker, Z. Ghahramani (eds.), Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, USA, 2001, pp. 609–616.

[8] K. Crammer, Y. Singer, On the algorithmic implementation of multiclass kernel-based vector machines, Journal of Machine Learning Research 2 (2001) 265–292.

[9] N. Cristianini, J. Shawe-Taylor, An introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000.

[10] D. DeCoste, M. Burl, Distortion-invariant recognition via jittered queries, in: Proc. of the Conf. on Computer Vision and Pattern Recognition, Hilton Head Island, SC, USA, 2000, pp. 732–737.

[11] D. Decoste, B. Schölkopf, Training invariant support vector machines, Machine Learning 46 (2002) 161–190.

[12] J. X. Dong, A. Krzyzak, C. Y. Suen, A fast parallel optimization for training support vector machines, in: P. Perner, A. Rosenfeld (eds.), Proc. of the Int. Conf. on Machine Learning and Data Mining, Vol. 2734 of Lecture Notes in Computer Science, Springer, 2003, pp. 96–105.

[13] J. X. Dong, A. Krzyzak, C. Y. Suen, Fast SVM training algorithm with decomposition on very large data sets, IEEE Trans. on Pattern Analysis and Machine Intelligence 27 (4) (2005) 603–618.

[14] T. Evgeniou, M. Pontil, T. Poggio, Regularization networks and support vector machines, Advances in Computational Mathematics 13 (2000) 1–50.

[15] R.-E. Fan, P.-H. Chen, C.-J. Lin, Working set selection using the second order information for training SVM, Journal of Machine Learning Research 6 (2005) 1889–1918.

[16] A. Fitzgibbon, A. Zisserman, Joint manifold distance: a new approach to appearance based clustering, in: Proc. of the Conf. on Computer Vision and Pattern Recognition, Vol. 1, IEEE Computer Society, 2003, pp. 26–33.

[17] G. Fung, O. L. Mangasarian, J. W. Shavlik, Knowledge-based support vector machine classifiers, in: S. Becker, S. Thrun, K. Obermayer (eds.), Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, USA, 2003, pp. 521–528.

[18] G. Fung, O. L. Mangasarian, J. W. Shavlik, Knowledge-based nonlinear kernel classifiers., in: B. Schölkopf, M. K. Warmuth (eds.), Proc. of the Conf. on Learning Theory, Vol. 2777 of Lecture Notes in Computer Science, Springer, 2003, pp. 102–113.

[19] F. Girosi, M. Jones, T. Poggio, Regularization theory and neural networks architectures, Neural Computation 7 (2) (1995) 219–269.

[20] T. Graepel, R. Herbrich, Invariant pattern recognition by semi-definite programming machines, in: S. Thrun, L. K. Saul, B. Schölkopf (eds.), Advances in Neural Information Processing Systems, MIT Press, Cambridge, MA, USA, 2004, pp. 33–40.

[21] Y. Guermeur, A. Elisseeff, H. Paugam-Moisy, A new multi-class SVM based on a uniform convergence result, in: Proc. of the Int. Joint Conf. on Neural Networks, Como, Italy, Vol. 4, 2000, pp. 183–188.

[22] B. Haasdonk, H. Burkhardt, Invariant kernels for pattern analysis and machine learning, Tech. Rep. 3/05, IIF-LMB, Computer Science Department, University of Freiburg (2005).

[23] B. Haasdonk, D. Keysers, Tangent distance kernels for support vector machines, in: Proc. of the 16th Int. Conf. on Pattern Recognition, Québec, QC, Canada, Vol. 2, 2002, 864–868.

[24] B. Haasdonk, A. Vossen, H. Burkhardt, Invariance in kernel methods by Haar-integration kernels, in: H. Kälviäinen, J. Parkkinen, A. Kaarna (eds.), Scandinavian Conf. on Image Analysis, Vol. 3540 of Lecture Notes in Computer Science, Springer, 2005, pp. 841–851.

[25] T. Joachims, Making large-scale support vector machine learning practical, in: Schölkopf et al. [50], pp. 169–184.

[26] T. Joachims, Transductive inference for text classification using support vector machines, in: I. Bratko, S. Dzeroski (eds.), Proc. of the Int. Conf. on Machine Learning, Morgan Kaufmann, 1999, pp.200–209.

[27] T. Joachims, Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms, Kluwer Academic Publishers, 2002.

[28] S. Kambar, Generating synthetic data by morphing transformation for handwritten numeral recognition (with ν-svm), Master's thesis, Concordia University, Montréal, QC, Canada (2005).

[29] L. Kaufman, Solving the quadratic programming problem arising in support vector classification, in: Schölkopf et al. [50], pp. 147–167.

[30] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, K. R. K. Murthy, Improvements to Platt's SMO algorithm for SVM classifier design, Neural Computation 13 (3) (2001) 637–649.

[31] R. I. Kondor, T. Jebara, A kernel between sets of vectors, in: T. Fawcett, N. Mishra (eds.), Proc. of the 20th Int. Conf. on Machine Learning, Washington, DC, USA, AAAI Press, 2003, pp. 361–368.

[32] F. Lauer, C. Y. Suen, G. Bloch, A trainable feature extractor for handwritten digit recognition, Pattern Recognition 40 (6) (2007) 1816–1824.

[33] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86 (11) (1998) 2278–2324.

[34] Y. LeCun, C. Cortes, The MNIST database of handwritten digits, http://yann.lecun.com/exdb/mnist/index.html.

[35] T. K. Leen, From data distributions to regularization in invariant learning, Neural Computation 7 (5) (1995) 974–981.

[36] G. Loosli, S. Canu, S. V. N. Vishwanathan, A. J. Smola, Invariances in classification: an efficient SVM implementation, in: Int. Symp. on Applied Stochastic Models and Data Analysis, 2005.

[37] O. Mangasarian, Generalized support vector machines, in: A. Smola, P. Bartlett, B. Schölkopf, D. Schuurmans (eds.), Advances in Large Margin Classifiers, MIT Press, Cambridge, MA, USA, 2000, pp. 135–146.

[38] O. L. Mangasarian, E. W. Wild, Nonlinear knowledge in kernel approximation, Tech. Rep. 05-05, Data Mining Institute, University of Wisconsin (2005).

[39] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, K.-R. Müller, Fisher discriminant analysis with kernels, in: Proc. of the IEEE Conf. on Neural Networks for Signal Processing IX, 1999, pp. 41–48.

[40] P. Niyogi, F. Girosi, T. Poggio, Incorporating prior information in machine learning by creating virtual examples, Proceedings of the IEEE 86 (1998) 2196–2209.

[41] E. Osuna, R. Freund, F. Girosi, An improved training algorithm for support vector machines, in: Proc. of the IEEE Workshop on Neural Networks for Signal Processing, Amelia Island, FL, USA, IEEE Press, Cambridge, MA, USA, 1997.

[42] J. C. Platt, Fast training of support vector machines using sequential minimal optimization, in: Schölkopf et al. [50], pp. 185–208.

[43] T. Poggio, T. Vetter, Recognition and structure from one 2D model view: Observations on prototypes, object classes and symmetries, Tech. Rep. AIM-1347, Massachusetts Institute of Technology, Cambridge, MA, USA (1992).

[44] A. Pozdnoukhov, S. Bengio, Tangent vector kernels for invariant image classification with SVMs, in: Proc. of the 17th Int. Conf. on Pattern Recognition, Cambridge, UK, 2004, pp. 486–489.

[45] A. Pozdnoukhov, S. Bengio, Invariances in kernel methods: From samples to objects, Pattern Recognition Letters 27 (2006) 1087–1097.

[46] Š. Raudys, Statistical and neural classifiers: an integrated approach to design, Springer-Verlag, London, UK, 2001.

[47] O. Ronneberger, LibsvmTL: a support vector machine template library (2004), http://lmb.informatik.uni-freiburg.de/lmbsoft/libsvmtl/index.en.html.

[48] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning internal representations by error propagation, in: Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations, MIT Press, Cambridge, MA, USA, 1986, pp. 318–362.

[49] B. Schölkopf, C. Burges, V. Vapnik, Incorporating invariances in support vector learning machines, in: C. von der Malsburg, W. von Seelen, J. C. Vorbrüggen, B. Sendhoff (eds.), Proc. of the Int. Conf. on Artificial Neural Networks, Vol. 1112 of Lecture Notes in Computer Science, Springer, 1996, pp. 47–52.

[50] B. Schölkopf, C. J. Burges, A. J. Smola (eds.), Advances in kernel methods: Support vector learning, MIT Press, Cambridge, MA, USA, 1999 (1999).

[51] B. Schölkopf, P. Simard, A. J. Smola, V. Vapnik, Prior knowledge in support vector kernels, in: M. I. Jordan, M. J. Kearns, S. A. Solla (eds.), Advances in Neural Information Processing Systems, The MIT Press, Cambridge, MA, USA, 1998, pp. 640–646.

[52] B. Schölkopf, A. Smola, K. Müller, Nonlinear component analysis as a kernel eigenvalue problem, Neural Computation 10 (1998) 1299–1319.

[53] B. Schölkopf, A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.

[54] H. Schulz-Mirbach, Constructing invariant features by averaging techniques, in: Proc. of the 12th Int. Conf on Pattern Recognition, Jerusalem, Israel, Vol. 2, 1994, pp. 387–390.

[55] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.

[56] P. Shivaswamy, T. Jebara, Permutation invariant SVMs, in: Proc. of the 23th Int. Conf. on Machine Learning, Pittsburgh, PA, USA, 2006.

[57] P. Simard, Y. LeCun, J. Denker, B. Victorri, Transformation invariance in pattern recognition, tangent distance and tangent propagation, in: G. Orr, K. Müller (eds.), Neural Networks: Tricks of the trade, vol. 1524 of Lecture Notes in Computer Science, Springer, 1998, pp. 239–274.

[58] P. Y. Simard, D. Steinkraus, J. C. Platt, Best practices for convolutional neural networks applied to visual document analysis, in: Proc. of the 7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland, UK, vol. 2, 2003, pp. 958–962.

[59] A. J. Smola, B. Schölkopf, K. R. Müller, The connection between regularization operators and support vector kernels, Neural Networks 11 (4) (1998) 637–649.

[60] C. Y. Suen, J. Tan, Analysis of errors of handwritten digits made by a multitude of classifiers, Pattern Recognition Letters 26 (2005) 369–379.

[61] I. W. Tsang, J. T. Kwok, P.-M. Cheung, Core vector machines: Fast SVM training on very large data sets, Journal of Machine Learning Research 6 (2005) 363–392.

[62] L. Vandenberghe, S. Boyd, Semidefinite programming, SIAM Review 38 (1) (1996) 49–95.

[63] V. N. Vapnik, The nature of statistical learning theory, Springer-Verlag, New York, NY, USA, 1995.

[64] V. N. Vapnik, Statistical learning theory, John Wiley, New York, NY, USA, 1998.

[65] K. Veropoulos, C. Campbell, N. Cristianini, Controlling the sensitivity of support vector machines, in: Proc. of the Int. Joint Conf. on Artificial Intelligence, 1999, pp. 55–60.

[66] S. V. N. Vishwanathan, A. J. Smola, M. N. Murty, SimpleSVM, in: T. Fawcett, N. Mishra (eds.), Proc. of the 20th Int. Conf. on Machine Learning, Washington, DC, USA, AAAI Press, 2003, pp. 760–767.

[67] L. Wang, Y. Gao, K. L. Chan, P. Xue, W.-Y. Yau, Retrieval with knowledge-driven kernel design: An approach to improving SVM-based CBIR with relevance feedback, in: Proc. of the Int. Conf. on Computer Vision, IEEE Computer Society, 2005, pp. 1355–1362.

[68] L. Wang, P. Xue, K. L. Chan, Incorporating prior knowledge into SVM for image retrieval, in: Proc. of the 17th Int. Conf. on Pattern Recognition, Cambridge, UK, Vol. 2, 2004, pp. 981–984.

[69] J. Weston, C. Watkins, Multiclass support vector machines, Tech. Rep. CSD-TR-98-04, Royal Holloway, University of London (1998).

[70] L. Wolf, A. Shashua, Learning over sets using kernel principal angles, Journal of Machine Learning Research 4 (10) (2003) 913–931.

[71] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Trans. on Evolutionary Computation 1 (1) (1997) 67–82.

[72] X. Wu, R. Srihari, Incorporating prior knowledge with weighted margin support vector machines, in: Proc. of the tenth ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Seattle, WA, USA, ACM Press, New York, NY, USA, 2004, pp. 326–333.