1 Measuring Similarity with Kernels

1.1 Introduction

Over the last ten years, estimation and learning methods utilizing positive definite kernels have become rather popular, particularly in machine learning. Since these methods have a stronger mathematical slant than earlier machine learning methods (e.g., neural networks), there is also significant interest in the statistical and mathematical community for these methods. The present chapter aims to summarize the state of the art on a conceptual level. In doing so, we build on various sources (including Vapnik (1998); Burges (1998); Cristianini and Shawe-Taylor (2000); Herbrich (2002) and in particular Schölkopf and Smola (2002)), but we also add a fair amount of recent material which helps in unifying the exposition.

The main idea of all the described methods can be summarized in one paragraph. Traditionally, theory and algorithms of machine learning and statistics have been very well developed for the linear case. Real-world data analysis problems, on the other hand, often require nonlinear methods to detect the kind of dependences that allow successful prediction of properties of interest. By using a positive definite kernel, one can sometimes have the best of both worlds. The kernel corresponds to a dot product in a (usually high-dimensional) feature space. In this space, our estimation methods are linear, but as long as we can formulate everything in terms of kernel evaluations, we never explicitly have to work in the high-dimensional feature space.

1.2 Kernels

1.2.1 An Introductory Example

Suppose we are given empirical data

$$(x_1, y_1), \ldots, (x_n, y_n) \in \mathcal{X} \times \mathcal{Y}. \tag{1.1}$$

Here, the domain $\mathcal{X}$ is some nonempty set that the *inputs* x_i are taken from; the $y_i \in \mathcal{Y}$ are called *targets*. Here and below, $i, j = 1, \ldots, n$.

Note that we have not made any assumptions on the domain $\mathcal{X}$ other than it being a set. In order to study the problem of learning, we need additional structure. In

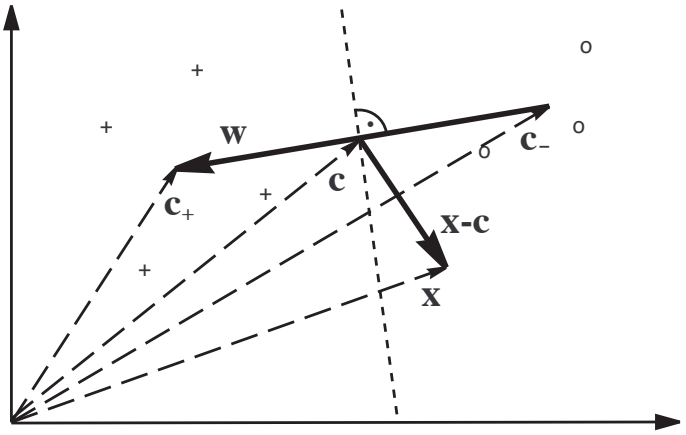

Figure 1.1 A simple geometric classification algorithm: given two classes of points (depicted by 'o' and '+'), compute their means c_+, c_- and assign a test input x to the one whose mean is closer. This can be done by looking at the dot product between $x - c$ (where $c = (c_+ + c_-)/2$) and $\mathbf{w} := c_+ - c_-$, which changes sign as the enclosed angle passes through $\pi/2$. Note that the corresponding decision boundary is a hyperplane (the dotted line) orthogonal to $\mathbf{w}$ (from Schölkopf and Smola (2002)).

learning, we want to be able to *generalize* to unseen data points. In the case of binary pattern recognition, given some new input $x \in \mathcal{X}$, we want to predict the corresponding $y \in \{\pm 1\}$. Loosely speaking, we want to choose y such that (x, y) is in some sense *similar* to the training examples. To this end, we need similarity measures in $\mathcal{X}$ and in $\{\pm 1\}$. The latter is easier, as two target values can only be identical or different.[1] For the former, we require a function

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \qquad (x, x') \mapsto k(x, x') \tag{1.2}$$

satisfying, for all $x, x' \in \mathcal{X}$,

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle , \tag{1.3}$$

where Φ maps into some dot product space $\mathcal{H}$, sometimes called the *feature space*. The similarity measure k is usually called a *kernel*, and Φ is called its *feature map*.

kernels and feature map

The advantage of using such a kernel as a similarity measure is that it allows us to construct algorithms in dot product spaces. For instance, consider the following simple classification algorithm, where $\mathcal{Y} = \{\pm 1\}$. The idea is to compute the means of the two classes in the feature space, $c_+ = \frac{1}{n_+} \sum_{\{i:y_i=+1\}} \Phi(x_i)$, and $c_- = \frac{1}{n_-} \sum_{\{i:y_i=-1\}} \Phi(x_i)$, where n_+ and n_- are the number of examples with

1. When $\mathcal{Y}$ has a more complex structure, things can get complicated — this is the main topic of the present book, but we completely disregard it in this introductory example.

positive and negative target values, respectively. We then assign a new point $\Phi(x)$ to the class whose mean is closer to it. This leads to

$$y = \operatorname{sgn}(\langle \Phi(x), c_+ \rangle - \langle \Phi(x), c_- \rangle + b) \tag{1.4}$$

with $b = \frac{1}{2}\left(\|c_-\|^2 - \|c_+\|^2\right)$. Substituting the expressions for $c_\pm$ yields

$$y = \operatorname{sgn}\left(\frac{1}{n_+}\sum_{\{i:y_i=+1\}} \langle \Phi(x), \Phi(x_i)\rangle - \frac{1}{n_-}\sum_{\{i:y_i=-1\}} \langle \Phi(x), \Phi(x_i)\rangle + b\right). \tag{1.5}$$

Rewritten in terms of k, this reads

$$y = \operatorname{sgn}\left(\frac{1}{n_+}\sum_{\{i:y_i=+1\}} k(x, x_i) - \frac{1}{n_-}\sum_{\{i:y_i=-1\}} k(x, x_i) + b\right), \tag{1.6}$$

where $b = \frac{1}{2}\left(\frac{1}{n_-^2}\sum_{\{(i,j):y_i=y_j=-1\}} k(x_i, x_j) - \frac{1}{n_+^2}\sum_{\{(i,j):y_i=y_j=+1\}} k(x_i, x_j)\right)$. This algorithm is illustrated in figure 1.1 for the case that $\mathcal{X}$ equals $\mathbb{R}^2$ and $\Phi(x) = x$.

Let us consider one well-known special case of this type of classifier. Assume that the class means have the same distance to the origin (hence $b = 0$), and that $k(., x)$ is a density for all $x' \in \mathcal{X}$. If the two classes are equally likely and were generated from two probability distributions that are correctly estimated by the Parzen windows estimators

$$p_+(x) := \frac{1}{n_+}\sum_{\{i:y_i=+1\}} k(x, x_i), \qquad p_-(x) := \frac{1}{n_-}\sum_{\{i:y_i=-1\}} k(x, x_i), \tag{1.7}$$

then (1.6) is the Bayes decision rule.

The classifier (1.6) is quite close to the *support vector machine (SVM)* that we will discuss below. It is linear in the feature space (see (1.4)), while in the input domain, it is represented by a kernel expansion (1.6). In both cases, the decision boundary is a hyperplane in the feature space; however, the normal vectors are usually different.[2]

1.2.2 Positive Definite Kernels

We have above required that a kernel satisfy (1.3), i.e., correspond to a dot product in some dot product space. In the present section, we show that the class of kernels that can be written in the form (1.3) coincides with the class of positive definite kernels. This has far-reaching consequences. There are examples of positive definite

2. For (1.4), the normal vector is $w = c_+ - c_-$. As an aside, note that if we normalize the targets such that $\hat{y}_i = y_i/|\{j : y_j = y_i\}|$, in which case the $\hat{y}_i$ sum to zero, then $\|w\|^2 = \langle K, \hat{y}\hat{y}^\top\rangle_F$, where $\langle ., .\rangle_F$ is the Frobenius dot product. If the two classes have equal size, then up to a scaling factor involving $\|K\|_2$ and n, this equals the *kernel-target alignment* defined by Cristianini et al. (2002).

kernels which can be evaluated efficiently even though via (1.3) they correspond to dot products in infinite-dimensional dot product spaces. In such cases, substituting $k(x, x')$ for $\langle \Phi(x), \Phi(x') \rangle$, as we have done when going from (1.5) to (1.6), is crucial.

1.2.2.1 Prerequisites

Definition 1 (Gram Matrix) *Given a kernel k and inputs $x_1, \ldots, x_n \in \mathcal{X}$, the $n \times n$ matrix*

$$K := (k(x_i, x_j))_{ij} \tag{1.8}$$

is called the Gram matrix *(or* kernel matrix*) of k with respect to $x_1, \ldots, x_n$.*

Definition 2 (Positive Definite Matrix) *A real $n \times n$ symmetric matrix K_{ij} satisfying*

$$\sum_{i,j} c_i c_j K_{ij} \geq 0 \tag{1.9}$$

for all $c_i \in \mathbb{R}$ is called positive definite*. If for equality in (1.9) only occurs for $c_1 = \cdots = c_n = 0$, then we shall call the matrix* strictly positive definite*.*

Definition 3 (Positive Definite Kernel) *Let $\mathcal{X}$ be a nonempty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which for all $n \in \mathbb{N}, x_i \in \mathcal{X}$ gives rise to a positive definite Gram matrix is called a* positive definite kernel*. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ which for all $n \in \mathbb{N}$ and distinct $x_i \in \mathcal{X}$ gives rise to a strictly positive definite Gram matrix is called a* strictly positive definite kernel*.*

Occasionally, we shall refer to positive definite kernels simply as a *kernels*. Note that for simplicity we have restricted ourselves to the case of real-valued kernels. However, with small changes, the below will also hold for the complex-valued case.

Since $\sum_{i,j} c_i c_j \langle \Phi(x_i), \Phi(x_j) \rangle = \left\langle \sum_i c_i \Phi(x_i), \sum_j c_j \Phi(x_j) \right\rangle \geq 0$, kernels of the form (1.3) are positive definite for any choice of Φ. In particular, if $\mathcal{X}$ is already a dot product space, we may choose Φ to be the identity. Kernels can thus be regarded as generalized dot products. While they are not generally bilinear, they share important properties with dot products, such as the Cauchy-Schwartz inequality:

Proposition 4 *If k is a positive definite kernel, and $x_1, x_2 \in \mathcal{X}$, then*

$$k(x_1, x_2)^2 \leq k(x_1, x_1) \cdot k(x_2, x_2). \tag{1.10}$$

Proof The 2×2 Gram matrix with entries $K_{ij} = k(x_i, x_j)$ is positive definite. Hence both its eigenvalues are nonnegative, and so is their product, K's determinant, i.e.,

$$0 \leq K_{11} K_{22} - K_{12} K_{21} = K_{11} K_{22} - K_{12}^2. \tag{1.11}$$

Substituting $k(x_i, x_j)$ for K_{ij}, we get the desired inequality. ∎

1.2.2.2 Construction of the Reproducing Kernel Hilbert Space

We now define a map from $\mathcal{X}$ into the space of functions mapping $\mathcal{X}$ into $\mathbb{R}$, denoted as $\mathbb{R}^{\mathcal{X}}$, via

$$\begin{aligned} \Phi : \mathcal{X} &\to \mathbb{R}^{\mathcal{X}} \\ x &\mapsto k(., x). \end{aligned} \tag{1.12}$$

Here, $\Phi(x) = k(., x)$ denotes the function that assigns the value $k(x', x)$ to $x' \in \mathcal{X}$.

We next construct a dot product space containing the images of the inputs under Φ. To this end, we first turn it into a vector space by forming linear combinations

$$f(.) = \sum_{i=1}^{n} \alpha_i k(., x_i). \tag{1.13}$$

Here, $n \in \mathbb{N}$, $\alpha_i \in \mathbb{R}$ and $x_i \in \mathcal{X}$ are arbitrary.

Next, we define a dot product between f and another function $g(.) = \sum_{j=1}^{n'} \beta_j k(., x'_j)$ (with $n' \in \mathbb{N}$, $\beta_j \in \mathbb{R}$ and $x'_j \in \mathcal{X}$) as

$$\langle f, g \rangle := \sum_{i=1}^{n} \sum_{j=1}^{n'} \alpha_i \beta_j k(x_i, x'_j). \tag{1.14}$$

To see that this is well-defined although it contains the expansion coefficients, note that $\langle f, g \rangle = \sum_{j=1}^{n'} \beta_j f(x'_j)$. The latter, however, does not depend on the particular expansion of f. Similarly, for g, note that $\langle f, g \rangle = \sum_{i=1}^{n} \alpha_i g(x_i)$. This also shows that $\langle \cdot, \cdot \rangle$ is bilinear. It is symmetric, as $\langle f, g \rangle = \langle g, f \rangle$. Moreover, it is positive definite, since positive definiteness of k implies that for any function f, written as (1.13), we have

$$\langle f, f \rangle = \sum_{i,j=1}^{n} \alpha_i \alpha_j k(x_i, x_j) \geq 0. \tag{1.15}$$

Next, note that given functions $f_1, \ldots, f_p$, and coefficients $\gamma_1, \ldots, \gamma_p \in \mathbb{R}$, we have

$$\sum_{i,j=1}^{p} \gamma_i \gamma_j \langle f_i, f_j \rangle = \left\langle \sum_{i=1}^{p} \gamma_i f_i, \sum_{j=1}^{p} \gamma_j f_j \right\rangle \geq 0. \tag{1.16}$$

Here, the left-hand equality follows from the bilinearity of $\langle \cdot, \cdot \rangle$, and the right-hand inequality from (1.15).

By (1.16), $\langle \cdot, \cdot \rangle$ is a positive definite kernel, defined on our vector space of functions. For the last step in proving that it even is a dot product, we note that by (1.14), for all functions (1.13),

$$\langle k(., x), f \rangle = f(x), \tag{1.17}$$

and in particular

$$\langle k(.,x), k(.,x')\rangle = k(x,x'). \tag{1.18}$$

reproducing kernel

By virtue of these properties, k is called a *reproducing kernel* (Aronszajn, 1950) .

Due to (1.17) and proposition 4, we have

$$|f(x)|^2 = |\langle k(.,x), f\rangle|^2 \leq k(x,x) \cdot \langle f, f\rangle. \tag{1.19}$$

By this inequality, $\langle f, f\rangle = 0$ implies $f = 0$, which is the last property that was left to prove in order to establish that $\langle ., .\rangle$ is a dot product.

Skipping some details, we add that one can complete the space of functions (1.13) in the norm corresponding to the dot product, and thus get a Hilbert space H, called a *reproducing kernel Hilbert space (RKHS)*.

reproducing kernel Hilbert space(RKHS)

One can define an RKHS as a Hilbert space $\mathcal{H}$ of functions on a set $\mathcal{X}$ with the property that for all $x \in \mathcal{X}$ and $f \in \mathcal{H}$, the point evaluations $f \mapsto f(x)$ are continuous linear functionals (in particular, all point values $f(x)$ are well-defined, which already distinguishes RKHSs from many L_2 Hilbert spaces). From the point evaluation functional, one can then construct the reproducing kernel using the Riesz representation theorem. The Moore-Aronszajn theorem (Aronszajn, 1950) states that for every positive definite kernel on $\mathcal{X} \times \mathcal{X}$, there exists a unique RKHS and vice versa.

There is an analogue of the kernel trick for distances rather than dot products, i.e., dissimilarities rather than similarities. This leads to the larger class of *conditionally positive definite kernels*. Those kernels are defined just like positive definite ones, with the one difference being that their Gram matrices need to satisfy (1.9) only subject to

$$\sum_{i=1}^{n} c_i = 0. \tag{1.20}$$

Interestingly, it turns out that many kernel algorithms, including SVMs and kernel principal component analysis PCA (see section 1.3.2), can be applied also with this larger class of kernels, due to their being translation invariant in feature space (Schölkopf and Smola, 2002; Hein et al., 2005).

We conclude this section with a note on terminology. In the early years of kernel machine learning research, it was not the notion of positive definite kernels that was being used. Instead, researchers considered kernels satisfying the conditions of Mercer's theorem (Mercer, 1909); see e.g. Vapnik (1998) and Cristianini and Shawe-Taylor (2000). However, while all such kernels do satisfy (1.3), the converse is not true. Since (1.3) is what we are interested in, positive definite kernels are thus the right class of kernels to consider.

1.2.3 Constructing Kernels

constructing new kernels

In the following we demonstrate how to assemble new kernel functions from existing ones using elementary operations preserving positive definiteness. The following proposition will serve us as the main working horse:

Proposition 5 *Below, $k_1, k_2, \ldots$ are arbitrary positive definite kernels on $\mathcal{X} \times \mathcal{X}$, where $\mathcal{X}$ is a nonempty set.*
(i) The set of positive definite kernels is a closed convex cone, i.e., (a) if $\alpha_1, \alpha_2 \geq 0$, then $\alpha_1 k_1 + \alpha_2 k_2$ is positive definite.
(ii) The pointwise product $k_1 k_2$ is positive definite.
(iii) Assume that for $i = 1, 2$, k_i is a positive definite kernel on $\mathcal{X}_i \times \mathcal{X}_i$, where $\mathcal{X}_i$ is a nonempty set. Then the tensor product $k_1 \otimes k_2$ and the direct sum $k_1 \oplus k_2$ are positive definite kernels on $(\mathcal{X}_1 \times \mathcal{X}_2) \times (\mathcal{X}_1 \times \mathcal{X}_2)$.
(iv) If $k(x, x') := \lim_{n\to\infty} k_n(x, x')$ exists for all x, x', then k is positive definite.
(v) The function $k(x, x') := f(x)f(x')$ is a valid positive definite kernel for any function f.

Let us use this proposition now to construct new kernel functions.

1.2.3.1 Polynomial Kernels

From proposition 5 it is clear that homogeneous polynomial kernels $k(x, x') = \langle x, x' \rangle^p$ are positive definite for $p \in \mathbb{N}$ and $x, x' \in \mathbb{R}^d$. By direct calculation we can derive the corresponding feature map (Poggio, 1975):

$$\langle x, x' \rangle^p = \left\langle \sum_{j=1}^{d} [x]_j, [x']_j \right\rangle^p = \sum_{j \in [d]^p} [x]_{j_1} \cdot \cdots \cdot [x]_{j_p} \cdot [x']_{j_1} \cdot \cdots \cdot [x']_{j_p} = \langle C_p(x), C_p(x') \rangle \,, \tag{1.21}$$

where C_p maps $x \in \mathbb{R}^d$ to the vector $C_p(x)$ whose entries are all possible pth-degree ordered products of the entries of x. The polynomial kernel of degree p thus computes a dot product in the space spanned by all monomials of degree p in the input coordinates. Other useful kernels include the inhomogeneous polynomial,

$$k(x, x') = (\langle x, x' \rangle + c)^p \text{ where } p \in \mathbb{N} \text{ and } c \geq 0, \tag{1.22}$$

which computes all monomials up to degree p.

1.2.3.2 Gaussian Kernel

Using the infinite Taylor expansion of the exponential function $e^z = \sum_{i=1}^{\infty} \frac{1}{i!} z^i$, it follows from propostion 5(iv) that

$$e^{\gamma \langle x, x' \rangle}$$

is a kernel function for any $x, x' \in \mathcal{X}$ and $\gamma \in \mathbb{R}$. Therefore, it follows immediately that the widely used Gaussian function $e^{-\gamma||x-x'||^2}$ with $\gamma > 0$ is a valid kernel function. This can be seen as rewriting the Gaussian function as

$$e^{-\gamma||x-x'||^2} = e^{-\gamma\langle x,x\rangle} e^{2\gamma\langle x,x'\rangle} e^{-\gamma\langle x',x'\rangle},$$

and using proposition 5(ii).

We see that the Gaussian kernel corresponds to a mapping into $\mathcal{C}^\infty$, i.e. the space of continuous functions. However, the feature map is *normalized*, i.e. $||\Phi(x)||^2 = k(x,x) = 1$ for any $x \in \mathcal{X}$. Moreover, as $k(x,x') > 0$ for all $x, x' \in \mathcal{X}$, all mapped points lie inside the same orthant in feature space.

1.2.3.3 Spline Kernels

It is possible to obtain spline functions as a result of kernel expansions (Smola, 1996; Vapnik et al., 1997) simply by noting that convolution of an even number of indicator functions yields a positive kernel function. Denote by I_X the indicator (or characteristic) function on the set X, and denote by $\otimes$ the convolution operation, $(f \otimes g)(x) := \int_{\mathbb{R}^d} f(x')g(x'-x)dx'$. Then the B-spline kernels are given by

$$k(x,x') = B_{2p+1}(x-x') \text{ where } p \in \mathbb{N} \text{ with } B_{i+1} := B_i \otimes B_0. \tag{1.23}$$

Here B_0 is the characteristic function on the unit ball[3] in $\mathbb{R}^d$. From the definition of (1.23) it is obvious that for odd m we may write B_m as the inner product between functions $B_{m/2}$. Moreover, note that for even m, B_m is not a kernel.

1.2.4 The Representer Theorem

From kernels, we now move to functions that can be expressed in terms of kernel expansions. The representer theorem (Kimeldorf and Wahba, 1971; Cox and O'Sullivan, 1990) shows that solutions of a large class of optimization problems can be expressed as kernel expansions over the sample points. We present a slightly more general version of the theorem with a simple proof (Schölkopf et al., 2001). As above, $\mathcal{H}$ is the RKHS associated with the kernel k.

Theorem 6 (Representer Theorem) *Denote by $\Omega : [0,\infty) \to \mathbb{R}$ a strictly monotonic increasing function, by $\mathcal{X}$ a set, and by $c : (\mathcal{X} \times \mathbb{R}^2)^n \to \mathbb{R} \cup \{\infty\}$ an arbitrary loss function. Then each minimizer $f \in \mathcal{H}$ of the regularized risk functional*

$$c\left((x_1, y_1, f(x_1)), \ldots, (x_n, y_n, f(x_n))\right) + \Omega\left(\|f\|_{\mathcal{H}}^2\right) \tag{1.24}$$

3. Note that in $\mathbb{R}$ one typically uses $\xi_{\left[-\frac{1}{2},\frac{1}{2}\right]}$.

admits a representation of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x). \tag{1.25}$$

Proof We decompose any $f \in \mathcal{H}$ into a part contained in the span of the kernel functions $k(x_1, \cdot), \cdots, k(x_n, \cdot)$, and one in the orthogonal complement:

$$f(x) = f_{\|}(x) + f_{\perp}(x) = \sum_{i=1}^{n} \alpha_i k(x_i, x) + f_{\perp}(x). \tag{1.26}$$

Here $\alpha_i \in \mathbb{R}$ and $f_{\perp} \in \mathcal{H}$ with $\langle f_{\perp}, k(x_i, \cdot)\rangle_{\mathcal{H}} = 0$ for all $i \in [n] := \{1, \ldots, n\}$. By (1.17) we may write $f(x_j)$ (for all $j \in [n]$) as

$$f(x_j) = \langle f(\cdot), k(x_j, .)\rangle = \sum_{i=1}^{n} \alpha_i k(x_i, x_j) + \langle f_{\perp}(\cdot), k(x_j, .)\rangle_{\mathcal{H}} = \sum_{i=1}^{n} \alpha_i k(x_i, x_j). \tag{1.27}$$

Second, for all $f_{\perp}$,

$$\Omega(\|f\|_{\mathcal{H}}^2) = \Omega\left(\left\|\sum_{i}^{n} \alpha_i k(x_i, \cdot)\right\|_{\mathcal{H}}^2 + \|f_{\perp}\|_{\mathcal{H}}^2\right) \geq \Omega\left(\left\|\sum_{i}^{n} \alpha_i k(x_i, \cdot)\right\|_{\mathcal{H}}^2\right). \tag{1.28}$$

Thus for any fixed $\alpha_i \in \mathbb{R}$ the risk functional (1.24) is minimized for $f_{\perp} = 0$. Since this also has to hold for the solution, the theorem holds. ∎

Monotonicity of Ω does not prevent the regularized risk functional (1.24) from having multiple local minima. To ensure a global minimum, we would need to require convexity. If we discard the strictness of the monotonicity, then it no longer follows that each minimizer of the regularized risk admits an expansion (1.25); it still follows, however, that there is always another solution that is as good, and that *does* admit the expansion.

The significance of the representer theorem is that although we might be trying to solve an optimization problem in an infinite-dimensional space $\mathcal{H}$, containing linear combinations of kernels centered on *arbitrary* points of $\mathcal{X}$, it states that the solution lies in the span of n particular kernels — those centered on the training points. We will encounter (1.25) again further below, where it is called the *support vector expansion*. For suitable choices of loss functions, many of the α_i often equal zero.

1.3 Operating in Reproducing Kernel Hilbert Spaces

We have seen that kernels correspond to an inner product in some possibly high-dimensional feature space. Since direct computation in these spaces is computationally infeasible one might argue that sometimes the application of kernels is rather limited. However, in this section we demonstrate for some cases that direct opera-

tion in feature space is possible. Subsequently we introduce kernel PCA which can extract features corresponding to principal components in this high-dimensional feature space.

1.3.1 Direct Operations in RKHS

1.3.1.1 Translation

Consider the modified feature map $\tilde{\Phi}(x) = \Phi(x) + \Gamma$, with $\Gamma \in \mathcal{H}$. This feature map corresponds to a translation in feature space. The dot product $\left\langle \tilde{\Phi}(x), \tilde{\Phi}(x') \right\rangle$ yields for this case the terms

$$\langle \Phi(x), \Phi(x') \rangle + \langle \Phi(x), \Gamma \rangle + \langle \Gamma, \Phi(x') \rangle + \langle \Gamma, \Gamma \rangle ,$$

which cannot always be evaluated. However, let us restrict the translation Γ to be in the span of the functions $\Phi(x_1), \cdots, \Phi(x_n) \in \mathcal{H}$ with $\{x_1, \ldots, x_n\} \in \mathcal{X}^n$. Thus if $\Gamma = \sum_{i=1}^n \alpha_i \Phi(x_i), \alpha_i \in \mathbb{R}$, then the dot product between translated feature maps can be evaluated in terms of the kernel functions solely. Thus we obtain for our modified feature map

$$\left\langle \tilde{\Phi}(x), \tilde{\Phi}(x') \right\rangle = k(x, x') + \sum_{i=1}^n \alpha_i k(x_i, x) + \sum_{i=1}^n \alpha_i k(x_i, x') + \sum_{i,j=1}^n \alpha_i \alpha_j k(x_i, x_j). \tag{1.29}$$

1.3.1.2 Centering

As a concrete application for a translation operation consider the case that we would like to *center* a set of points in the RKHS. Thus we would like to have a feature map $\tilde{\Phi}$ such that $\frac{1}{n} \sum_{i=1}^n \tilde{\Phi}(x_i) = 0$. Using $\tilde{\Phi}(x) = \Phi(x) + \Gamma$ with $\Gamma = -\sum_{i=1}^n \frac{1}{n} \Phi(x_i)$ this can be obtained immediately utilizing (1.29). The kernel matrix $\tilde{K}$ of the centered feature map $\tilde{\Phi}$ can then be expressed directly in terms of matrix operations by

$$\tilde{K}_{ij} = (K - 1_m K - K 1_m + 1_m K 1_m)_{ij} ,$$

where $1_m \in \mathbb{R}^{m \times m}$ is the constant matrix with all entries equal to $1/m$, and K is the kernel matrix evaluated using Φ.

1.3.1.3 Computing Distances

An essential tool for structured prediction is the problem of computing distances between two objects. For example, to assess the quality of a prediction we would like to measure the distance between predicted object and true object. Since kernel functions can be interpreted as dot products (see (1.3)) they provide an elegant way to measure distances between arbitrary objects. Consider two objects $x_1, x_2 \in \mathcal{X}$,

such as two-word sequences or two automata. Assume we have a kernel function k on such objects; we can use their distance in the RKHS, i.e.,

$$d(x_1, x_2) = ||\Phi(x_1) - \Phi(x_2)||_{\mathcal{H}} = \sqrt{k(x_1, x_1) + k(x_2, x_2) - 2k(x_1, x_2)}.$$

Here, we have utilized the fact that the dot product in $\mathcal{H}$ can be evaluated by kernel functions and thus define the distance between the objects to be the distance between the images of the feature map Φ.

1.3.1.4 Subspace Projections

Another elementary operation which can be performed in a Hilbert space is the one-dimensional orthogonal *projection*. Given two points Ψ, Γ in the RKHS $\mathcal{H}$ we project the point Ψ to the subspace spanned by the point Γ, obtaining

$$\Psi' = \frac{\langle \Gamma, \Psi \rangle}{||\Gamma||^2} \Gamma. \tag{1.30}$$

Considering the case that Ψ and Γ are given by kernel expansions, we see immediately that any dot product with the projected point Ψ' can be expressed with kernel functions only. Using such a projection operation in RKHS, it is straightforward to define a *deflation* procedure:

$$\Psi' = \Psi - \frac{\langle \Gamma, \Psi \rangle}{||\Gamma||^2} \Gamma. \tag{1.31}$$

Using projection and deflation operations, one can perform e.g. the Gram-Schmidt orthogonalization procedure for the construction of orthogonal bases. This was used for example in information retrieval (Cristianini et al., 2001) and computer vision (Wolf and Shashua, 2003). An alternative application of deflation and subspace projection in RKHS was introduced by Rosipal and Trejo (2002) in the context of *subspace regression*.

1.3.2 Kernel Principal Component Analysis

A standard method for feature extraction is the method of principal component analysis (PCA), which aims to identify principal axes in the input. The principal axes are recovered as the eigenvectors of the empirical estimate of the covariance matrix $C_{emp} = \mathbb{E}_{emp}\left[(\mathbf{x} - \mathbb{E}_{emp}[\mathbf{x}])(\mathbf{x} - \mathbb{E}_{emp}[\mathbf{x}])^\top\right]$. In contrast to PCA, kernel PCA introduced by Schölkopf et al. (1998) tries to identify principal components of variables which are nonlinearly related to input variables, i.e. principal axis in some feature space $\mathcal{H}$. To this end, given some training set $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of size n, one considers the eigenvectors $\mathbf{v} \in \mathcal{H}$ of the empirical covariance operator in feature space:

covariance in feature space

$$\mathbf{C}_{emp} = \mathbb{E}_{emp}\left[(\Phi(\mathbf{x}) - \mathbb{E}_{emp}[\Phi(\mathbf{x})])(\Phi(\mathbf{x}) - \mathbb{E}_{emp}[\Phi(\mathbf{x})])^\top\right].$$

Although this operator and thus its eigenvectors $\mathbf{v}$ cannot be calculated directly, they can be retrieved in terms of kernel evaluations only. To see this, note that even in the case of a high-dimensional feature space $\mathcal{H}$, a finite training set $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ of size n when mapped to this feature space spans a subspace $E \subset \mathcal{H}$ whose dimension is at most n. Thus, there are at most n principal axes $(\mathbf{v}_1, \ldots, \mathbf{v}_n) \in E^n$ with nonzero eigenvalues. It can be shown that these principal axes can be expressed as linear combinations of the training points $\mathbf{v}_j = \sum_{i=1}^{N} \alpha_i^j \Phi(\mathbf{x}_i), 1 \leq j \leq n$, where the coefficients $\alpha^j \in \mathbb{R}^n$ are obtained as eigenvectors of the kernel matrix evaluated on the training set. If one retains all principal components, kernel PCA can be considered as a basis transform in E, leaving the dot product of training points invariant. To see this, let $(\mathbf{v}_1, \ldots, \mathbf{v}_n) \in E^n$ be the principal axes of $\{\Phi(\mathbf{x}_1), \ldots, \Phi(\mathbf{x}_n)\}$. The kernel PCA map $\phi_n : \mathcal{X} \to \mathbb{R}^n$ is defined coordinatewise as

$$[\phi_n]_p(\mathbf{x}) = \Phi(\mathbf{x}) \cdot \mathbf{v}_p, \quad 1 \leq p \leq n.$$

Note that by definition, for all i and j, $\Phi(\mathbf{x}_i)$ and $\Phi(\mathbf{x}_j)$ lie in E and thus

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) = \phi_n(\mathbf{x}_i) \cdot \phi_n(\mathbf{x}_j). \tag{1.32}$$

The kernel PCA map is especially useful if one has structured data and one wants to use an algorithm which is not readily expressed in dot products.

1.4 Kernels for Structured Data

We have seen several instances of positive definite kernels, and now intend to describe some kernel functions which are particularly well suited to operate on data domains other than real vector spaces. We start with the simplest data domain: sets.

1.4.1 Set Kernels

Assume that we have given a finite alphabet Σ, i.e. a collection of symbols which we call characters. Furthermore let us denote by $\mathcal{P}(\Sigma)$ the power set of Σ. Then, we define a *set kernel* to be any valid kernel function k which takes two sets $A \in \mathcal{P}(\Sigma)$ and $B \in \mathcal{P}(\Sigma)$ as arguments. As a concrete example, consider the following kernel:

$$k(A, B) = \sum_{x \in A,\ y \in B} 1_{x=y},$$

kernels for text

where $1_{x=y}$ denotes a comparison. This kernel measures the size of the intersection of two sets and is widely used e.g. in text classification where it is referred to as the *sparse vector kernel*. Considering a text document as a set of words, the sparse vector kernel measures the similarity of text document via the number of common

words. Such a kernel was used e.g. in Joachims (1998) for text categorization using SVMs.

The feature map corresponding to the set kernel can be interpreted as a *representation by its parts*. Each singleton $x_i \in \Sigma, 1 \leq i \leq |\Sigma|$, i.e. all sets of cardinality 1, is mapped to the vertex e_i of the unit simplex in $\mathbb{R}^{|\Sigma|}$. Each set A with $|A| > 1$ is then the average of the vertex coordinates, i.e.,

$$\Phi(A) = \sum_{x \in A} \Phi(x) = \sum_{x_i \in \Sigma, x \in A} 1_{x=x_i} e_i.$$

Set kernels are in general very efficient to evaluate as long as the alphabet is finite since the feature map yields a sparse vector in $\mathbb{R}^{|\Sigma|}$. For example, in text classification each dimension corresponds to a specific word, and a component is set to a constant whenever the related word occurs in the text. This is also known as the *bag-of-words* representation. Using an efficient sparse representation, the dot product between two such vectors can be computed quickly.

1.4.2 Rational Kernels

One of the shortcomings of set kernels in applications such as natural language applications is that any relation among the set elements such as, e.g., word order in a document, is completely ignored. However, in many applications one considers data with a more sequential nature such as word sequences in text classification, temporal utterance order in speech recognition, or chains of amino acids in protein analysis. In these cases the data are of sequential nature and can consist of variable-length sequences over some basic alphabet Σ. In the following we review kernels which were introduced to deal with such data types and which belong to the general class of *rational* kernels.

Rational kernels are in principle similarity measures over *sets* of sequences. Since sets of sequences can be compactly represented by automata, rational kernels can be considered as kernels for weighted automata. For a discussion on automata theory see e.g. Hopcroft et al. (2000). In particular, since sequences can be considered as very simple automata, rational kernels automatically implement kernels for sequences. At the heart of a rational kernel is the concept of *weighted transducers* which can be considered as a representation of a binary relation between sequences; see e.g. Mohri et al. (2002) and Cortes et al. (2004).

kernels for automata

Definition 7 (Weighted Transducer) *Given a semiring $K = (\mathbb{K}, \oplus, \otimes)$, a weighted finite-state transducer (WFST) T over $\mathbb{K}$ is given by an* input *alphabet Σ, an output alphabet Ω, a finite set of* states *S, a finite set of* transitions *$E \subseteq S \times (\Sigma \cup \{\epsilon\}) \times (\Omega \cup \{\epsilon\}) \times \mathbb{K} \times S$, a set of* initial *states $S_0 \in S$, a set of* final *states $S_\infty \subseteq S$, and a weight function $w : S \to \mathbb{K}$.*

In our further discussion we restrict the output alphabet Ω to be equal to the input alphabet, i.e. $\Omega = \Sigma$. We call a sequence of transitions $h = e_1, \ldots, e_n \subset E$ a *path*, where the ith transition is denoted by $\pi_i(h)$. By $\pi_0(h)$ and $\pi_\infty(h)$ we denote starting

and termination states of a path h respectively. Given two sequences $x, y \in \Sigma^*$, we call a path h *successful* if it starts at an initial state, i.e. $\pi_0(h) \in S_0$, terminates in a final state, i.e. $\pi_\infty(h) \in S_\infty$, and concatenating the input and output symbols associated with the traversed transitions equals the sequences x and y. There might be more than a single successful path and we will denote the set of all successful paths depending on the pair (x, y) by $\Pi(x, y)$. Furthermore, for each transition $\pi_i[h] \in E$ we denote by $w(\pi_i[h]) \in \mathbb{K}$ the weight associated with the particular transition $\pi_i[h]$. A transducer is called *regulated* if the weight of any sequence input-output pair $(x, y) \in \Sigma^* \times \Sigma^*$ calculated by

$$[\![T]\!](x, y) := \bigoplus_{h \in \Pi(x,y)} w(\pi_0[h]) \otimes \bigotimes_{i=1}^{|h|} w(\pi_i[h]) \otimes w(\pi_\infty[h]) \tag{1.33}$$

is well-defined and in $\mathbb{K}$.

The interpretation of the weights $w(h)$ and in particular $[\![T]\!](x, y)$ depends on how they are manipulated algebraically and on the underlying semiring $\mathbb{K}$. As a concrete example for the representation of binary relations, let us consider the positive semiring $(\mathbb{K}, \oplus, \otimes, \mathbf{0}, \mathbf{1}) = (\mathbb{R}_+, +, \times, 0, 1)$ which is also called the probability or real semiring. A binary relation between two sequences $x, y \in \Sigma^*$ is e.g. the conditional probability $[\![T]\!](x, y) = P(y|x)$. Let x_i denote the ith element of the sequence x. We can calculate the conditional probability as

$$P(y|x) = \sum_{h \in \Pi(x,y)} \prod_{i=0} P(y_i|\pi_i[h], x_i) \times P(y_\infty|\pi_\infty(h), x_\infty),$$

where the sum is over all successful paths h and $w(\pi_i[h])) := P(y_i|\pi_i(h), x_i)$ denotes the probability of performing the transition $\pi_i(h)$ and observing (x_i, y_i) as input and outpout symbols. However, reconsidering the example with the *tropical* semiring $(\mathbb{K}, \oplus, \otimes, \mathbf{0}, \mathbf{1}) = (R \cup \{\infty, -\infty\}, \min, +, +\infty, 0)$ we obtain

$$[\![T]\!](x, y) = \max_{h \in \Pi(x,y)} \sum_{i=0} w(\pi_i[h]) + w(\pi_\infty[h]),$$

which is also known as the Viterbi approximation if the weights are negative log-probabilities, i.e. $w(\pi_\infty[h]) = -\log P(y_i|\pi_i[h], x_i)$. It is also possible to perform algebraic operations on transducers directly. Let T_1, T_2 be two weighted transducers, then a fundamental operation is *composition*.

Definition 8 (Composition) *Given two transducers* $T_1 = \{\Sigma, \Omega, S^1, E^1, S_0^1, S_\infty^1, w^1\}$ *and* $T_2 = \{\Omega, \Delta, S^2, E^2, S_0^2, S_\infty^2, w^2\}$*, the* composition $T_1 \circ T_2$ *is defined as transducer* $R = \{\Sigma, \Delta, S, E, S_0, S_\infty, w\}$ *such that*

$$S = S^1 \times S^2, \quad S_0 = S_0^1 \times S_0^2, \quad S_\infty = S_\infty^1 \times S_\infty^2$$

and each transition $e \in E$ *satisfies*

$$\forall e : (p, p') \xrightarrow{a:c/w} (q, q') \quad \Rightarrow \quad \exists \{p \xrightarrow{a:b/w_1} q,\ p' \xrightarrow{b:c/w_2} q'\},$$

with $w = w_1 \otimes w_2$.

For example, if the transducer T_1 models the conditional probabilities of a label given a feature observation $P(y|\phi(x))$ and another T_2 transducer models the conditional probabilities of a feature given an actual input $P(\phi(x)|x)$, then the transducer obtained by a composition $R = T_1 \circ T_2$ represents $P(y|x)$. In this sense, a composition can be interpreted as a matrix operation for transducers which is apparent if one considers the weights of the composed transducer:

$$[\![T_1 \circ T_2]\!](x, y) = \sum_{z \in \Omega} [\![T_1]\!](x, z)[\![T_2]\!](z, y).$$

Finally, let us introduce the *inverse* transducer T^{-1} that is obtained by swapping all input and output symbols on every transition of a transducer T. We are now ready to introduce the concept of rational kernels.

Definition 9 (Rational Kernel) *A kernel k over the alphabet Σ^* is called* rational *if it can be expressed as weight computation over a transducer T, i.e. $k(x, x') = \Psi([\![T]\!](x, x'))$ for some function $\Psi : \mathbb{K} \to \mathbb{R}$. The kernel is said to be defined by the pair (T, Ψ).*

kernel evaluation by transducers

Unfortunately, not any transducer gives rise to a positive definite kernel. However, from proposition 5(v) and from the definition it follows directly that any transducer $S := T \circ T^{-1}$ is a valid kernel since

$$k(x, y) = \sum_z [\![T]\!](x, z)[\![T]\!](x', z) = [\![S]\!](x, x').$$

The strength of rational kernels is their compact representation by means of transducers. This allows an easy and modular design of novel application-specific similarity measures for sequences. Let us give an example for a rational kernel.

1.4.2.1 n-gram Kernels

An n-gram is a block of n adjacent characters from an alphabet Σ. Hence, the number of distinct n-grams in a text is less than or equal to $|\Sigma|^n$. This shows that the space of all possible n-grams can be very high even for moderate values of n. The basic idea behind the *n-gram kernel* is to compare sequences by means of the subsequences they contain:

$$k(x, x') = \sum_{s \in \Sigma^n} \#(s \in x)\#(s \in x'), \tag{1.34}$$

where $\#(s \in x)$ denotes the number of occurrences of s in x. In this sense, the more subsequences two sequences share, the more similar they are. Vishwanathan and Smola (2004) proved that this class of kernels can be computed in $O(|x| + |x'|)$ time and memory by means of a special suited data structure allowing one to find a compact representation of all subsequences of x in only $O(|x|)$ time and space.

Furthermore, the authors show that the function $f(x) = \langle w, \Phi(x) \rangle$ can be computed in $O(|x|)$ time if preprocessing linear in the size of the expansion w is carried out. Cortes et al. (2004) showed that this kernel can be implemented by a transducer kernel by explicitly constructing a transducer that counts the number of occurrences of n symbol blocks; see e.g figure 1.2. One then can rewrite (1.34) as

$$k(x, x') = [\![T \circ T^{-1}]\!](x, x'). \tag{1.35}$$

In the same manner, one can design transducers that can compute similarities incorporating various costs as, for example, for gaps and mismatches; see Cortes et al. (2004).

1.4.3 Convolution Kernels

One of the first instances of kernel functions on structured data was *convolutional kernels* introduced by Haussler (1999). The key idea is that one may take a structured object and split it up into parts. Suppose that the object $x \in \mathcal{X}$ consists of substructures $x_p \in \mathcal{X}_p$ where $1 \leq p \leq r$ and r denotes the number of overall substructures. Given then the set $\mathcal{P}(\mathcal{X})$ of all possible substructures $\bigotimes_{i=1}^{r} \mathcal{X}_i$, one can define a *relation* R between a subset of $\mathcal{P}$ and the composite object x. As an example consider the relation "part-of" between subsequences and sequences. If there are only a finite number of subsets, the relation R is called finite. Given a finite relation R, let $R^{-1}(x)$ define the set of all possible decompositions of x into its substructures: $R^{-1}(x) = \{z \in \mathcal{P}(\mathcal{X}) : R(z, x)\}$. In this case, Haussler (1999) showed that the so-called R-convolution given as

representation by parts

$$k(x, y) = \sum_{x' \in R^{-1}(x)} \sum_{y' \in R^{-1}(y)} \prod_{i=1}^{r} k_i(x'_i, y'_i) \tag{1.36}$$

is a valid kernel with k_i being a positive definite kernel on $\mathcal{X}_i$.The idea of decomposing a structured object into parts can be applied recursively so that one only requires to construct kernels k_i over the "atomic" parts $\mathcal{X}_i$.

Convolution kernels are very general and were successfully applied in the context of natural language processing (Collins and Duffy, 2002; Lodhi et al., 2000). However, in general the definition of R and in particular R^{-1} for a specific problem is quite difficult.

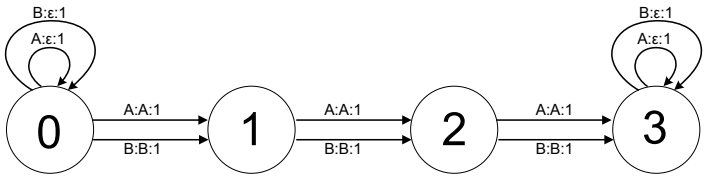

Figure 1.2 A transducer that can be used for calculation of 3-grams for a binary alphabet.

1.4.4 Kernels Based on Local Information

Sometimes it is easier to describe the local neighborhood than to construct a kernel for the overall data structure. Such a neighborhood of a data item might be defined by any item that differs only by the presence or absence of a single property. For example, when considering English words, neighbors of a word can be defined as any other word that would be obtained by misspelling. Given a set of data items, all information about neighbor relations can be represented by e.g. a *neighbor graph*. A vertex in such a neighbor graph would correspond to a data item and two vertices are connected whenever they satisfy some neighbor rule. For example, in the case of English words, a neighbor rule could be that two words are neighbors whenever their edit distance is smaller than some apriori defined threshold. Kondor and Lafferty (2002) utilize such neighbor graphs to construct global similarity measures by using a *diffusion process* analogy. To this end, the authors define a diffusion process by using the so-called *graph Laplacian*, L being a square matrix and where each entry encodes information on how to propagate the information from vertex to vertex. In particular, if A denotes the binary adjacency matrix of the neighbor graph, the graph Laplacian is given by $L = A - D$, where D is a diagonal matrix and each diagonal D_{ii} is the vertex degree of the ith data item. The resulting kernel matrix K is then obtained as the matrix exponential of βL with $\beta < 1$ being a propagation parameter:

similarities due to a diffusion process

$$K = e^{-\beta L} := \lim_{n\to\infty} \left(\mathbf{1} - \frac{\beta}{n} L\right)^n .$$

Such diffusion kernels were successfully applied to such diverse applications as text-categorization, as e.g. in Kandola et al. (2002); gene-function prediction by Vert and Kanehisa (2002); and semisupervised learning, as e.g. in Zhou et al. (2004).

Even if it is possible to define a kernel function for the whole instance space, sometimes it might be advantageous to take into account information from local structure of the data. Recall the Gaussian kernel and polynomial kernels. When applied to an image, it makes no difference whether one uses as x the image or a version of x where all locations of the pixels have been permuted. This indicates that the function space on $\mathcal{X}$ induced by k does not take advantage of the *locality* properties of the data. By taking advantage of the local structure, estimates can be improved. On biological sequences one may assign more weight to the entries of the sequence close to the location where estimates should occur, as was performed e.g. by Zien et al. (2000). In other words, one replaces $\langle x, x' \rangle$ by $x^\top \Omega x$, where $\Omega \succeq 0$ is a diagonal matrix with largest terms at the location which needs to be classified.

In contrast, for images, local interactions between image patches need to be considered. One way is to use the *pyramidal* kernel introduced in Schölkopf (1997) and DeCoste and Schölkopf (2002), which was inspired by the pyramidal cells of the brain: It takes inner products between corresponding image patches, then raises the latter to some power p_1, and finally raises their sum to another power p_2. This

means that mainly short-range interactions are considered and that the long-range interactions are taken with respect to short-range groups.

1.4.5 Tree and Graph Kernels

We now discuss similarity measures on more structured objects such as trees and graphs.

1.4.5.1 Kernels on Trees

For trees Collins and Duffy (2002) propose a decomposition method which maps a tree x into its set of subtrees. The kernel between two trees x, x' is then computed by taking a weighted sum of all terms between both trees and is based on the convolutional kernel (see section 1.4.3). In particular, Collins and Duffy (2002) show an $O(|x| \cdot |x'|)$ algorithm to compute this expression, where $|x|$ is the number of nodes of the tree. When restricting the sum to all proper rooted subtrees it is possible to reduce the time of computation to $O(|x| + |x'|)$ time by means of a tree to sequence conversion (Vishwanathan and Smola, 2004).

1.4.5.2 Kernels on Graphs

A *labeled graph* G is described by a finite set of vertices V, a finite set of edges E, two sets of symbols which we denote by Σ and Ω, and two functions $\mathsf{v} : V \to \Sigma$ and $\mathsf{e} : E \to \Omega$ which assign each vertex and edge a label from the sets Σ, Ω respectively. For directed graphs, the set of edges is a subset of the Cartesian product of the ordered set of vertices with itself, i.e. $E \subseteq V \times V$ such that $(v_i, v_j) \in E$ if and only if vertex v_i is connected to vertex v_j. One might hope that a kernel for a labeled graph can be similarly constructed using some decomposition approach similar to the case of trees. Unfortunately, due to the existence of cycles, graphs cannot be as easily serialized, which prohibits, for example, the use of transducer kernels for graph comparison. A workaround is to artificially construct *walks*, i.e. eventually repetitive sequences of vertex and edge labels. Let us denote by $W(G)$ the set of all possible walks in a graph G of arbitrary length. Then, using an appropriate sequence kernel k_h, a valid kernel for two graphs G_1, G_2 would take the form

graph kernels based on paths

$$k_G(G_1, G_2) = \sum_{h \in W(G_1)} \sum_{h' \in W(G_2)} k_h(h, h'). \tag{1.37}$$

Unfortunately, this kernel can only be evaluated if the graph is acyclic since otherwise the sets $P(G_1), P(G_2)$ are not finite. However, one can restrict the set of all walks $W(G)$ to the set of all *paths* $P(G) \subset W(G)$, i.e. nonrepetitive sequences of vertex and edge labels. Borgwardt and Kriegel (2005) show that computation of this so-called *all-path kernel* is NP-complete. As an alternative, for graphs where each edge is assigned to a cost instead of a general label they propose to further restrict the set of paths. They propose to choose the subset of paths which appear in

path kernels are intractable

shortest-path graph kernel

an all-pairs shortest-path transformed version of the original graph. Thus for each graph G_i which has to be compared, the authors build a new completely connected graph $\hat{G}_i$ of the same size. In contrast to the original graph each edge in $\hat{G}_i$ between nodes v_i and v_j corresponds to the length of the shortest path from v_i to v_j in the original graph G_i. The new kernel function between the transformed graphs is then calculated by comparing all walks of length 1, i.e.,

$$k_{\hat{G}}(G_1, G_2) = \sum_{\substack{h \in W(\hat{G}_1) \\ |h|=1}} \sum_{\substack{h' \in W(\hat{G}_2) \\ |h'|=1}} k_h(h, h'). \tag{1.38}$$

Since algorithms for determining all-pairs shortest paths as, for example, Floyd-Warshall, are of cubic order and comparing all walks of length 1 is of fourth order, the all-pairs shortest-path kernel in (1.38) can be evaluated in $O(|V|^4)$ complexity.

comparing random walks

An alternative approach proposed by Kashima et al. (2003) is to compare two graphs by measuring the similarity of the *probability distributions* of random walks on the two graphs. The authors propose to consider a walk h as a hidden variable and the kernel as a *marginalized kernel* where marginalization is over h, i.e.,

$$k_{RG}(G_1, G_2) = \mathbb{E}[k_G(G_1, G_2)] = \sum_{h \in W(G_1)} \sum_{h' \in W(G_2)} k_h(h, h') p(h|G_1) p(h|G_2), \tag{1.39}$$

where the conditional distributions $p(h|G_1), p(h'|G_2)$ in (1.39) for the random walk h, h' are defined as start, transition, and termination probability distribution over the vertices in V. Note that this marginalized graph kernel can be interpreted as a *randomized version* of (1.37).

By using the dot product of the two probability distributions as kernel, the induced feature space $\mathcal{H}$ is infinite-dimensional, with one dimension for every possible label sequence. Nevertheless, the authors developed an algorithm for how to calculate (1.39) explicitly with $O(|V|^6)$ complexity.

1.4.6 Kernels from Generative Models

In their quest to make density estimates directly accessible to kernel methods Jaakkola and Haussler (1999a,b) designed kernels which work directly on probability density estimates $p(x|\theta)$. Denote by

$$U_\theta(x) := \partial_\theta - \log p(x|\theta) \tag{1.40}$$

$$I := \mathbf{E}_x \left[U_\theta(x) U_\theta^\top(x) \right] \tag{1.41}$$

Fisher information

the Fisher scores and the Fisher information matrix respectively. Note that for maximum likelihood estimators $\mathbf{E}_x [U_\theta(x)] = 0$ and therefore I is the covariance of $U_\theta(x)$. The Fisher kernel is defined as

$$k(x, x') := U_\theta^\top(x) I^{-1} U_\theta(x') \text{ or } k(x, x') := U_\theta^\top(x) U_\theta(x') \tag{1.42}$$

depending on whether we study the normalized or the unnormalized kernel respectively. It is a versatile tool to reengineer existing density estimators for the purpose of discriminative estimation.

In addition to that, it has several attractive theoretical properties: Oliver et al. (2000) show that estimation using the normalized Fisher kernel corresponds to an estimation subject to a regularization on the $L_2(p(\cdot|\theta))$ norm.

Moreover, in the context of exponential families (see section 3.6 for a more detailed discussion) where $p(x|\theta) = \exp(\langle\phi(x), \theta\rangle - g(\theta))$, we have

$$k(x, x') = [\phi(x) - \partial_\theta g(\theta)]\,[\phi(x') - \partial_\theta g(\theta)] \tag{1.43}$$

for the unnormalized Fisher kernel. This means that up to centering by $\partial_\theta g(\theta)$ the Fisher kernel is identical to the kernel arising from the inner product of the sufficient statistics $\phi(x)$. This is not a coincidence and is often encountered when working with nonparametric exponential families. A short description of exponential families is given further below in section 3.6. Moreover, note that the centering is immaterial, as can be seen in lemma 13.

1.5 An Example of a Structured Prediction Algorithm Using Kernels

In this section we introduce concepts for structured prediction based on kernel functions. The basic idea is based on the property that kernel methods embed any data type into a linear space and thus can be used to transform the targets to a new representation more amenable to prediction using existing technqiues. However, since one is interested in predictions of the original type one has to solve an additional *reconstruction* problem that is independent of the learning problem and therefore might be solved more easily. The first algorithm following this recipe was kernel dependency estimation(KDE) introduced by Weston et al. (2002) and which we discuss next.

kernel dependency estimation

Given n pairs of data items $D_n = \{(x_i, y_i)\}_{i=1}^n \subset \mathcal{X} \times \mathcal{Y}$ one is interested in learning a mapping $t_Z : \mathcal{X} \to \mathcal{Y}$. As a first step in KDE one constructs a linear embedding of the targets only. For example, Weston et al. (2002) propose kernel PCA using a kernel function on $\mathcal{Y}$, i.e. $k_y(y_1, y_2) : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Note that this kernel function gives rise to a feature map ϕ_y into a RKHS $\mathcal{H}_y$ and allows application of the kernel PCA map (see section 1.3.2). The new vectorial representation of the outputs can then be used to learn a map $T_{\mathcal{H}}$ from the input space $\mathcal{X}$ to the vectorial representation of the outputs, i.e. $\mathbb{R}^n$. This new learning problem using the transformed output is a standard multivariate regression problem and was solved for example in Weston et al. (2002) with kernel ridge regression using a kernel for $\mathcal{X}$.

kernel for the outputs

Finally, for a given new input point x^* and its predicted representation $T_{\mathcal{H}}(x^*)$, one has to *reconstruct* the output element $y^* \in \mathcal{Y}$ that matches the predicted representation best, i.e.

$$y^* = \arg\min_{y \in \mathcal{Y}} \; ||\phi_y(y) - T_{\mathcal{H}}(x^*)||^2_{\mathcal{H}_y}. \tag{1.44}$$

pre-image/decoding problem

The problem (1.44) is known as the *pre-image* problem or alternatively as the *decoding* problem and has wide applications in kernel methods. , We summarize all feature maps used in KDE in figure 1.3 where we denote by $\Gamma : \mathcal{H}_y \to \mathcal{Y}$ the pre-image map which is given by (1.44). In chapter 8, we see an application of KDE to the task of string prediction where the authors design a pre-image map based on n-gram kernels.

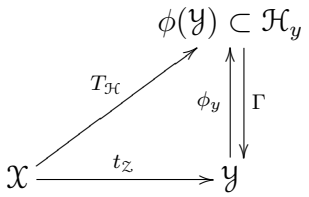

Figure 1.3 Mappings between original sets $\mathcal{X}, \mathcal{Y}$ and corresponding feature spaces $\mathcal{H}_y$ in kernel dependency estimation.

1.6 Conclusion

Kernels can be used for decorrelation of nontrivial structures between points in Euclidean space. Furthermore, they can be used to embed complex data types into linear spaces leading straightforward to distance and similarity measures among instances of arbitrary type. Finally, kernel functions *encapsulate* the data from the algorithm and thus allow use of the same algorithm on different data types without changing the implementation. Thus, whenever a learning algorithm can be expressed in kernels it can be utilized for arbitrary data types by exchanging the kernel function. This reduces the effort of using existing inference algorithms for novel application fields to introducing a novel specifically designed kernel function.

2 Discriminative Models

2.1 Introduction

In this chapter we consider the following problem: Given a set of data points $Z := \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ drawn from some data distribution $P(x, y)$, can we find a function $f(x) = \sigma(\langle w, x \rangle + b)$ such that $f(x) = y$ for all $(x, y) \in Z$, and $\mathbf{E}_{\text{emp}}[f(x) \neq y]$ is minimized. This problem is hard because of two reasons:

- Minimization of the empirical risk with respect to (w, b) is NP-hard (Minsky and Papert, 1969). In fact, Ben-David et al. (2003) show that even approximately minimizing the empirical risk is NP-hard, not only for linear function classes but also for spheres and other simple geometrical objects. This means that even if the statistical challenges could be solved, we still would be saddled with a formidable algorithmic problem.
- The indicator function $I_{\{f(x) \neq y\}}$ is discontinuous and even small changes in f may lead to large changes in both empirical and expected risk. Properties of such functions can be captured by the VC-dimension (Vapnik and Chervonenkis, 1971), that is, the maximum number of observations which can be labeled in an arbitrary fashion by functions of the class. Necessary and sufficient conditions for estimation can be stated in these terms (Vapnik and Chervonenkis, 1991). However, much tighter bounds can be obtained by using the scale of the class, too (Alon et al., 1993; Bartlett et al., 1996; Williamson et al., 2001). In fact, there exist function classes parameterized by a scalar which have infinite VC-dimension (Vapnik, 1995).

Given the difficulty arising from minimizing the empirical risk of misclassification, we now discuss algorithms which minimize an upper bound on the empirical risk, while providing good computational properties and consistency of the estimators. A common theme underlying all such algorithms is the notion of margin maximization. In other words, these algorithms map the input data points into a high-dimensional feature space using the so-called *kernel trick* discussed in the previous chapter, and maximize the separation between data points of different classes. In this chapter, we begin by studying the perceptron algorithm and its variants. We provide a unifying exposition of common loss functions used in these algorithms. Then we move on to support vector machine (SVM) algorithms and discuss how to obtain convergence rates for large-margin algorithms.

2.2 Online Large-Margin Algorithms

2.2.1 Perceptron and Friends

Let $\mathcal{X}$ be the space of observations, and $\mathcal{Y}$ the space of labels. Let, $\{(\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$ be a sequence of data points. The perceptron algorithm proposed by Rosenblatt (1962) is arguably the simplest online learning algorithm which is used to learn a separating hyperplane between two classes $\mathcal{Y} := \{\pm 1\}$. In its most basic form, it proceeds as follows. Start with the initial weight vector $w_0 = 0$. At step t, if the training example (x_t, y_t) is classified correctly, i.e., if $y_t(\langle x_t, w_t \rangle) \geq 0$, then set $w_{t+1} = w_t$; otherwise set $w_{t+1} = w_t + \eta y_t x_t$ (here, $\eta > 0$ is a learning rate). Repeat until all data points in the class are correctly classified. Novikoff's theorem shows that this procedure terminates, provided that the training set is separable with nonzero margin:

Theorem 10 (Novikoff (1962)) *Let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a dataset containing at least one data point labeled $+1$, and one data point labeled -1, and $R = \max_i ||x_i||_2$. Assume that there exists a weight vector w^* such that $||w^*||_2 = 1$, and $y_i(\langle w^*, x_t \rangle) \geq \gamma$ for all i, then the number of mistakes made by the perceptron is at most $(R/\gamma)^2$.*

Collins (2002b) introduced a version of the perceptron algorithm which generalizes to multiclass problems. Let, $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^d$ be a feature map which takes into account both the input as well as the labels. Then the algorithm proceeds as follows. Start with the initial weight vector $w_0 = 0$. At step t, predict with

perceptron for multiclass

$$z_t = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle \phi(x_t, y), w_t \rangle .$$

If $z_t = y_t$, then set $w_{t+1} = w_t$; otherwise set $w_{t+1} = w_t + \eta(\phi(x_t, y_t) - \phi(x_t, z_t))$. As before, $\eta > 0$ is a learning rate. A theorem analogous to the Novikoff theorem exists for this modified perceptron algorithm:

Theorem 11 (Collins (2002b)) *Let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a nontrivial dataset, and $R = \max_i \max_y ||\phi(x_i, y_i) - \phi(x_i, y)||_2$. Assume that there exists a weight vector w^* such that $||w^*||_2 = 1$, and $\min_{y \neq y_t} \langle w^*, \phi(x_t, y_t) \rangle - \langle w^*, \phi(x_t, y) \rangle \geq \gamma$ for all t, then the number of mistakes made by the modified perceptron is at most $(R/\gamma)^2$.*

In fact, a modified version of the above theorem also holds for the case when the data are not separable.

We now proceed to derive a general framework for online learning using large-margin algorithms and show that the above two perceptron algorithms can be viewed as special cases.

2.2.2 General Online Large-Margin Algorithms

As before, let $\mathcal{X}$ be the space of observations, and $\mathcal{Y}$ the space of labels. Given a sequence $\{(\boldsymbol{x}_i, y_i) | \boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}\}$ of examples and a loss function $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}$,

regularized risk

large-margin online algorithms aim to minimize the regularized risk

$$J(f) = \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, f) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2,$$

where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) of functions on $\mathcal{X}$. Its defining kernel satisfies the reproducing property i.e., $\langle f, k(x, \cdot) \rangle_{\mathcal{H}} = f(x)$ for all $f \in \mathcal{H}$. Let $\phi : \mathcal{X} \to \mathcal{H}$ be the corresponding feature map of the kernel $k(\cdot, \cdot)$; then we predict the label of $x \in \mathcal{X}$ as $\operatorname{sgn}(\langle w, \phi(x) \rangle)$. Finally, we make the assumption that l

derivatives in $\mathcal{H}$

only depends on f via its evaluations at $f(x_i)$ and that l is piecewise differentiable.

By the reproducing property of $\mathcal{H}$ we can compute derivatives of the evaluation functional. That is,

$$g := \partial_f f(x) = \partial_f \langle f, k(x, \cdot) \rangle_{\mathcal{H}} = k(x, \cdot).$$

Since l depends on f only via its evaluations we can see that $\partial_f l(x, y, f) \in \mathcal{H}$. Using the stochastic approximation of $J(f)$,

$$J_t(f) := l(x_t, y_t, f) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2$$

and setting

$$g_t := \partial_f J_t(f_t) = \partial_f l(x_t, y_t, f_t) + \lambda f_t \,,$$

we obtain the following simple update rule:

$$f_{t+1} \leftarrow f_t - \eta_t g_t,$$

where η_t is the step size at time t. This algorithm, also known as NORMA (Kivinen et al., 2004), is summarized in algorithm 2.1.

Algorithm 2.1 Online learning

1. Initialize $f_0 = 0$
2. **Repeat**
   (a) Draw data sample (x_t, y_t)
   (b) Predict $f_t(x_t)$ and incur loss $l(x_t, y_t, f_t)$
   (c) Update $f_{t+1} \leftarrow f_t - \eta_t g_t$

Observe that, so far, our discussion of the online update algorithm is independent of the particular loss function used. In other words, to apply our method to a new

setting we simply need to compute the corresponding loss function and its gradient. We discuss particular examples of loss functions and their gradients in section 2.4. But, before that, we turn our attention to the perceptron algorithms discussed above.

In order to derive the perceptron as a special case, set $\mathcal{H} = \mathbb{R}^d$ with the Euclidean dot product, $\eta_t = \eta$, and the loss function

$$l(x, y, f) = \max(0, -y \langle x, f \rangle).$$

It is easy to check that

$$g = \partial_f l(x, y, f) = \begin{cases} 0 \text{ if } y \langle x, f \rangle \geq 0 \\ -yx \text{ otherwise,} \end{cases}$$

and hence algorithm 2.1 reduced to the perceptron algorithm.

As for the modified perceptron algorithm, just set $\mathcal{H} = \mathbb{R}^d$ with the Euclidean dot product, $\eta_t = \eta$, and the loss function

$$l(x, y, f) = \max(0, \max_{\tilde{y} \neq y} \langle \phi(x, \tilde{y}), f \rangle - \langle \phi(x, y), f \rangle).$$

Observe that the feature map ϕ now depends on both x and y. This and other extensions to multiclass algorithms will be discussed in more detail in section 2.4. But for now it suffices to observe that

$$g = \partial_f l(x, y, f) = \begin{cases} 0 \text{ if } \langle \phi(x, y), f \rangle \geq \max_{\tilde{y} \neq y} \langle \phi(x, \tilde{y}), f \rangle \\ \max_{\tilde{y} \neq y} \{\phi(x, \tilde{y}) - \phi(x, y)\} \text{ otherwise,} \end{cases}$$

and we recover the modified perceptron algorithm from algorithm 2.1.

2.3 Support Vector Estimation

Until now we concentrated on online learning algorithms. Now we turn our attention to batch algorithms which predict with a hypothesis that is computed after seeing all data points.

2.3.1 Support Vector Classification

Assume that $Z := \{(x_1, y_1), \ldots, (x_n, y_n)\} \subseteq \mathcal{X} \times \mathcal{Y}$ is separable, i.e. there exists a linear function $f(x)$ such that $\operatorname{sgn} y f(x) = 1$ on Z. In this case, the task of finding a large-margin separating hyperplane can be viewed as one of solving (Vapnik and Lerner, 1963)

maximally separating hyperplane

$$\underset{w,b}{\text{minimize}} \ \tfrac{1}{2} \|w\|^2 \ \text{ subject to } \ y_i \left(\langle w, x \rangle + b \right) \geq 1. \tag{2.1}$$

Note that $\|w\|^{-1} f(x_i)$ is the distance of the point x_i to the hyperplane $H(w,b) := \{x | \langle w, x\rangle + b = 0\}$. The condition $y_i f(x_i) \geq 1$ implies that the margin of separation is at least $2\|w\|^{-1}$. The bound becomes exact if equality is attained for some $y_i = 1$ and $y_j = -1$. Consequently minimizing $\|w\|$ subject to the constraints maximizes the margin of separation. Eq. (2.1) is a quadratic program which can be solved efficiently (Luenberger, 1984; Fletcher, 1989; Boyd and Vandenberghe, 2004; Nocedal and Wright, 1999).

Mangasarian (1965) devised a similar optimization scheme using $\|w\|_1$ instead of $\|w\|_2$ in the objective function of (2.1). The result is a *linear* program. In general, one may show (Smola et al., 2000) that minimizing the ℓ_p norm of w leads to the maximizing of the margin of separation in the ℓ_q norm where $\frac{1}{p} + \frac{1}{q} = 1$. The ℓ_1 norm leads to sparse approximation schemes (see also Chen et al. (1999)), whereas the ℓ_2 norm can be extended to Hilbert spaces and kernels.

nonseparable problem

To deal with nonseparable problems, i.e. cases when (2.1) is infeasible, we need to relax the constraints of the optimization problem. Bennett and Mangasarian (1992) and Cortes and Vapnik (1995) impose a linear penalty on the violation of the large-margin constraints to obtain:

$$\underset{w,b,\xi}{\text{minimize}} \ \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \text{ subject to } y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i \text{ and } \xi_i \geq 0. \quad (2.2)$$

Eq.(2.2) is a quadratic program which is always feasible (e.g. $w, b = 0$ and $\xi_i = 1$ satisfy the constraints). $C > 0$ is a regularization constant trading off the violation of the constraints vs. maximizing the overall margin.

Whenever the dimensionality of $\mathcal{X}$ exceeds n, direct optimization of (2.2) is computationally inefficient. This is particularly true if we map from $\mathcal{X}$ into an RKHS. To address these problems one may solve the problem in dual space as follows. The Lagrange function of (2.2) is given by

Lagrange function

$$L(w,b,\xi,\alpha,\eta) = \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i\left(1 - \xi_i - y_i\left(\langle w, x_i\rangle + b\right)\right) - \sum_{i=1}^{n}\eta_i\xi_i,$$

where $\alpha_i, \eta_i \geq 0$ for all $i \in [n]$. To compute the dual of L we need to identify the first-order conditions in w, b. They are given by

$$\partial_w L = w - \sum_{i=1}^{n}\alpha_i y_i x_i = 0, \quad \partial_b L = -\sum_{i=1}^{n}\alpha_i y_i = 0 \ \text{ and } \ \partial_{\xi_i} L = C - \alpha_i + \eta_i = 0. \quad (2.3)$$

This translates into $w = \sum_{i=1}^{n}\alpha_i y_i x_i$, the linear constraint $\sum_{i=1}^{n}\alpha_i y_i = 0$, and the box-constraint $\alpha_i \in [0, C]$ arising from $\eta_i \geq 0$. Substituting (2.3) into L yields the Wolfe dual (Wolfe, 1961):

dual problem

$$\underset{\alpha}{\text{minimize}} \ \tfrac{1}{2}\alpha^\top Q\alpha - \alpha^\top 1 \text{ subject to } \alpha^\top y = 0 \text{ and } \alpha_i \in [0, C]. \quad (2.4)$$

$Q \in \mathbb{R}^{n \times n}$ is the matrix of inner products $Q_{ij} := y_i y_j \langle x_i, x_j \rangle$. Clearly this can be extended to feature maps and kernels easily via $K_{ij} := y_i y_j \langle \Phi(x_i), \Phi(x_j) \rangle = y_i y_j k(x_i, x_j)$. Note that w lies in the span of the x_i. This is an instance of the representer theorem (see section 1.2.4). The Karush-Kuhn-Tucker (KKT) conditions (Karush, 1939; Kuhn and Tucker, 1951; Boser et al., 1992; Cortes and Vapnik, 1995) require that at optimality $\alpha_i(y_i f(x_i) - 1) = 0$. This means that only those x_i may appear in the expansion (2.3) for which $y_i f(x_i) \leq 1$, as otherwise $\alpha_i = 0$. The x_i are commonly referred to as support vectors, (SVs)..

Note that $\sum_{i=1}^n \xi_i$ is an upper bound on the empirical risk, as $y_i f(x_i) \leq 0$ implies $\xi_i \geq 1$ (see also lemma 12). The number of misclassified points x_i itself depends on the configuration of the data and the value of C. The result of Ben-David et al. (2003) suggests that finding even an approximate minimum classification error solution is difficult. That said, it is possible to modify (2.2) such that a desired target number of observations violates $y_i f(x_i) \geq \rho$ for some $\rho \in \mathbb{R}$ by making the threshold itself a variable of the optimization problem (Schölkopf et al., 2000). This leads to the following optimization problem (ν-SV classification):

ν-SV classification

$$\underset{w,b,\xi}{\text{minimize}} \ \tfrac{1}{2}\|w\|^2 + \sum_{i=1}^n \xi_i - n\nu\rho \text{ subject to } y_i\left(\langle w, x_i \rangle + b\right) \geq \rho - \xi_i \text{ and } \xi_i \geq 0. \tag{2.5}$$

The dual of (2.5) is essentially identical to (2.4) with the exception of an additional constraint:

$$\underset{\alpha}{\text{minimize}} \ \tfrac{1}{2}\alpha^\top Q \alpha \text{ subject to } \alpha^\top y = 0 \text{ and } \alpha^\top 1 = n\nu \text{ and } \alpha_i \in [0, 1]. \tag{2.6}$$

One can show that for every C there exists a ν such that the solution of (2.6) is a multiple of the solution of (2.4). Schölkopf et al. (2000) prove that solving (2.6) for which $\rho > 0$ satisfies:

1. ν is an upper bound on the fraction of margin errors.
2. ν is a lower bound on the fraction of SVs.

Moreover, under mild conditions with probability 1, asymptotically, ν equals both the fraction of SVs and the fraction of errors.

This statement implies that whenever the data are sufficiently well separable (that is, $\rho > 0$), ν-SV classification finds a solution with a fraction of at most ν margin errors. Also note that for $\nu = 1$, all $\alpha_i = 1$, that is, f becomes an affine copy of the Parzen windows classifier (1.6).

2.3.2 Estimating the Support of a Density

We now extend the notion of linear separation to that of estimating the support of a density (Schölkopf et al., 2001; Tax and Duin, 1999). Denote by $X = \{x_1, \ldots, x_n\} \subseteq \mathcal{X}$ the sample drawn i.i.d. from $\Pr(x)$. Let $\mathcal{C}$ be a class of measurable subsets of $\mathcal{X}$

and let λ be a real-valued function defined on $\mathcal{C}$. The *quantile function* (Einmal and Mason, 1992) with respect to $(\Pr, \lambda, \mathcal{C})$ is defined as

$$U(\mu) = \inf \{\lambda(C) | \Pr(C) \geq \mu, C \in \mathcal{C}\} \text{ where } \mu \in (0, 1].$$

We denote by $C_\lambda(\mu)$ and $C_\lambda^m(\mu)$ the (not necessarily unique) $C \in \mathcal{C}$ that attain the infimum (when it is achievable) on $\Pr(x)$ and on the empirical measure given by X respectively. A common choice of λ is the Lebesgue measure, in which case $C_\lambda(\mu)$ is the minimum volume set $C \in \mathcal{C}$ that contains at least a fraction μ of the probability mass.

Support estimation requires us to find some $C_\lambda^m(\mu)$ such that $|\Pr(C_\lambda^m(\mu)) - \mu|$ is small. This is where the complexity tradeoff enters: On the one hand, we want to use a rich class $\mathcal{C}$ to capture all possible distributions; on the other hand large classes lead to large deviations between μ and $\Pr(C_\lambda^m(\mu))$. Therefore, we have to consider classes of sets which are suitably restricted. This can be achieved using an SVM regularizer.

In the case where $\mu < 1$, it seems the first work was reported in Sager (1979) and Hartigan (1987), in which $\mathcal{X} = \mathbb{R}^2$, with $\mathcal{C}$ being the class of closed convex sets in $\mathcal{X}$. Nolan (1991) considered higher dimensions, with $\mathcal{C}$ being the class of ellipsoids. Tsybakov (1997) studied an estimator based on piecewise polynomial approximation of $C_\lambda(\mu)$ and showed it attains the asymptotically minimax rate for certain classes of densities. Polonik (1997) studied the estimation of $C_\lambda(\mu)$ by $C_\lambda^m(\mu)$. He derived asymptotic rates of convergence in terms of various measures of richness of $\mathcal{C}$. More information on minimum volume estimators can be found in that work, and in Schölkopf et al. (2001).

learning the support

SV support estimation[1] relates to previous work as follows: set $\lambda(C_w) = \|w\|^2$, where $C_w = \{x | f_w(x) \geq \rho\}$, and (w, ρ) are respectively a weight vector and an offset with $f_w(x) = \langle w, x \rangle$. Stated as a convex optimization problem we want to separate the data from the origin with maximum margin via:

$$\underset{w,\xi,\rho}{\text{minimize}} \ \tfrac{1}{2}\|w\|^2 + \sum_{i=1}^{n} \xi_i - n\nu\rho \text{ subject to } \langle w, x_i \rangle \geq \rho - \xi_i \text{ and } \xi_i \geq 0. \tag{2.7}$$

Here, $\nu \in (0, 1]$ plays the same role as in (2.5), controlling the number of observations x_i for which $f(x_i) \leq \rho$. Since nonzero slack variables ξ_i are penalized in the objective function, if w and ρ solve this problem, then the decision function $f(x)$ will attain or exceed ρ for at least $1 - \nu$ instances x_i contained in X while the regularization term $\|w\|$ will still be small. The dual of (2.7) yields:

$$\underset{\alpha}{\text{minimize}} \ \tfrac{1}{2}\alpha^\top K \alpha \text{ subject to } \alpha^\top 1 = \nu n \text{ and } \alpha_i \in [0, 1]. \tag{2.8}$$

To compare (2.8) to a Parzen windows estimator assume that k is such that it can be normalized as a density in input space, such as a Gaussian. Using $\nu = 1$ in (2.8)

1. Note that this is also known as one-class SVM.

the constraints automatically imply $\alpha_i = 1$. Thus f reduces to a Parzen windows estimate of the underlying density. For $\nu < 1$, the equality constraint (2.8) still ensures that f is a thresholded density, now depending only on a *subset* of X — those which are important for the decision $f(x) \leq \rho$ to be taken.

2.4 Margin-Based Loss Functions

In the previous sections we implicitly assumed that $\mathcal{Y} = \{\pm 1\}$. But many estimation problems cannot be easily written as binary classification problems. We need to make three key changes in order to tackle these problems. First, in a departure from tradition, but keeping in line with Collins (2002b), Altun et al. (2004d), Tsochantaridis et al. (2004), and Cai and Hofmann (2004), we need to let our kernel depend on the labels as well as the observations. In other words, we minimize a regularized risk

$$J(f) = \frac{1}{m} \sum_{i=1}^{m} l(x_i, y_i, f) + \frac{\lambda}{2} \|f\|_{\mathcal{H}}^2, \tag{2.9}$$

where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) of functions on both $\mathcal{X} \times \mathcal{Y}$. Its defining kernel is denoted by $k : (\mathcal{X} \times \mathcal{Y})^2 \to \mathbb{R}$, and the corresponding feature map by $\phi : \mathcal{X} \times \mathcal{Y} \to \mathcal{H}$. Second, we predict the label of $x \in \mathcal{X}$ as

$$\operatorname*{argmax}_{y \in \mathcal{Y}} f(x, y) = \operatorname*{argmax}_{y \in \mathcal{Y}} \langle w, \phi(x, y) \rangle ,$$

and finally we need to modify the loss function in order to deal with structured output spaces. While the online variants minimize a stochastic approximation of the above risk, the batch algorithms predict with the best hypothesis after observing the whole dataset.

Also, observe that the perceptron algorithms did not enforce a margin constraint as a part of their loss. In other words, they simply required that the data points be well classified. On the other hand, large-margin classifiers not only require a point to be well classified but also enforce a margin constraint on the loss function.

In this section, we discuss some commonly used loss functions and put them in perspective. Later, we specialize the general recipe described above to multicategory classification, ranking, and ordinal regression. Since the online update depends on it, we will state the gradient of all loss functions we present below, and give their kernel expansion coefficients.

2.4.0.1 Loss Functions on Unstructured Ouput Domains

Binary classification uses the hinge or soft-margin loss (Bennett and Mangasarian, 1992; Cortes and Vapnik, 1995),

$$l(x, y, f) = \max(0, \rho - yf(x)), \tag{2.10}$$

where $\rho > 0$, and $\mathcal{H}$ is defined on $\mathcal{X}$ alone. We have

$$\partial_f l(x, y, f) = \begin{cases} 0 \text{ if } yf(x) \geq \rho \\ -yk(x, \cdot) \text{ otherwise} \end{cases}. \tag{2.11}$$

Multiclass classification employs a definition of the margin arising from log-likelihood ratios (Crammer and Singer, 2000). This leads to

$$l(x, y, f) = \max(0, \rho + \max_{\tilde{y} \neq y} f(x, \tilde{y}) - f(x, y)) \tag{2.12}$$

$$\partial_f l(x, y, f) = \begin{cases} 0 \text{ if } f(x, y) \geq \rho + f(x, y^*) \\ k((x, y^*), \cdot) - k((x, y), \cdot) \text{ otherwise} \end{cases}. \tag{2.13}$$

Here we defined $\rho > 0$, and y^* to be the maximizer of the $\max_{\tilde{y} \neq y}$ operation. If several y^* exist we pick one of them arbitrarily, e.g. by dictionary order.

Logistic regression works by minimizing the negative log-likelihood. This loss function is used in Gaussian process classification (MacKay, 1998). For binary classification this yields

$$l(x, y, f) = \log(1 + \exp(-yf(x))) \tag{2.14}$$

$$\partial_f l(x, y, f) = -yk(x, \cdot)\frac{1}{1 + \exp(yf(x))}. \tag{2.15}$$

Again the RKHS $\mathcal{H}$ is defined on $\mathcal{X}$ only.

Multiclass logistic regression works similarly to the example above. The only difference is that the log-likelihood arises from a conditionally multinomial model (MacKay, 1998). This means that

$$l(x, y, f) = -f(x, y) + \log \sum_{\tilde{y} \in \mathcal{Y}} \exp f(x, \tilde{y}) \tag{2.16}$$

$$\partial_f l(x, y, f) = \sum_{\tilde{y} \in \mathcal{Y}} k((x, \tilde{y}), \cdot)[p(\tilde{y}|x, f) - \delta_{y,\tilde{y}}], \tag{2.17}$$

$$\text{where we used} \quad p(y|x, f) = \frac{e^{f(x,y)}}{\sum_{\tilde{y} \in \mathcal{Y}} e^{f(x,\tilde{y})}}. \tag{2.18}$$

Novelty detection uses a trimmed version of the log-likelihood as a loss function. In practice this means that labels are ignored and the one-class margin needs to exceed 1 (Schölkopf et al., 2001). This leads to

$$l(x, y, f) = \max(0, \rho - f(x)) \tag{2.19}$$

$$\partial_f l(x, y, f) = \begin{cases} 0 \text{ if } f(x) \geq \rho \\ -k(x, \cdot) \text{ otherwise} \end{cases} . \tag{2.20}$$

2.4.0.2 Loss Functions on Structured Label Domains

class hierarchies

In many applications the output domain has an inherent structure. For example, document categorization deals with the problem of assigning a set of documents to a set of predefined topic hierarchies or taxonomies. Consider a typical taxonomy shown in figure 2.1 which is based on a subset of the open directory project.[2] If a document describing CDROMs is classified under hard disk drives (HDD), intuitively the loss should be smaller than when the same document is classified under Cables. Roughly speaking, the value of the loss function should depend on the length of the shortest path connecting the actual label to the predicted label, i.e., the loss function should respect the structure of the output space (Tsochantaridis et al., 2004).

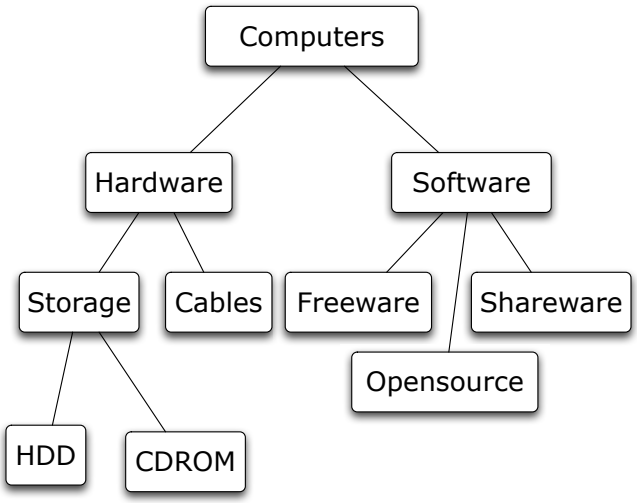

Figure 2.1 A taxonomy based on the open directory project.

To formalize our intuition, we need to introduce some notation. A weighted graph $G = (V, E)$ is defined by a set of nodes V and edges $E \subseteq V \times V$, such that each

2. `http://www.dmoz.org/`.

edge $(v_i, v_j) \in E$ is assigned a nonnegative weight $w(v_i, v_j) \in \mathbb{R}^+$. A path from $v_1 \in V$ to $v_n \in V$ is a sequence of nodes $v_1 v_2 \dots v_n$ such that $(v_i, v_{i+1}) \in E$. The weight of a path is the sum of the weights on the edges. For an undirected graph, $(v_i, v_j) \in E \implies (v_j, v_i) \in E \wedge w(v_i, v_j) = w(v_j, v_i)$.

A graph is said to be connected if every pair of nodes in the graph is connected by a path. In the sequel we will deal exclusively with connected graphs, and let $\Delta_G(v_i, v_j)$ denote the weight of the shortest (i.e., minimum weight) path from v_i to v_j. If the output labels are nodes in a graph G, the following loss function takes the structure of G into account:

$$l(x, y, f) = \max\{0, \max_{\tilde{y} \neq y} [\Delta_G(\tilde{y}, y) + f(x, \tilde{y})] - f(x, y)\}. \tag{2.21}$$

This loss requires that the output labels $\tilde{y}$ which are "far away" from the actual label y (on the graph) must be classified with a larger margin while nearby labels are allowed to be classified with a smaller margin. More general notions of distance, including kernels on the nodes of the graph, can also be used here instead of the shortest path $\Delta_G(\tilde{y}, y)$.

Analogous to (2.17), by defining y^* to be the maximizer of the $\max_{\tilde{y} \neq y}$ operation we can write the gradient of the loss as

$$\partial_f l(x, y, f) = \begin{cases} 0 \text{ if } f(x, y) \geq \Delta_G(y, y^*) + f(x, y^*) \\ k((x, y^*), \cdot) - k((x, y), \cdot) \text{ otherwise} \end{cases}. \tag{2.22}$$

The multiclass loss (2.12) is a special case of graph-based loss (2.21): consider a simple two-level tree in which each label is a child of the root node, and every edge has a weight of $\frac{\rho}{2}$. In this graph, any two labels $y \neq \tilde{y}$ will have $\Delta_G(y, \tilde{y}) = \rho$, and thus (2.21) reduces to (2.12). In the sequel, we will use $\Delta(y, \tilde{y})$ (without the subscript G) to denote the desired margin of separation between y and $\tilde{y}$.

2.4.1 Multicategory Classification, Ranking, and Ordinal Regression

Key to deriving convex optimization problems using the generalized risk function (2.9) for various common tasks is the following lemma:

Lemma 12 *Let $f : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ and assume that $\Delta(y, \tilde{y}) \geq 0$ with $\Delta(y, y) = 0$. Moreover let $\xi \geq 0$ such that $f(x, y) - f(x, \tilde{y}) \geq \Delta(y, \tilde{y}) - \xi$ for all $\tilde{y} \in \mathcal{Y}$. In this case $\xi \geq \Delta(y, \operatorname{argmax}_{\tilde{y} \in \mathcal{Y}} f(x, \tilde{y}))$.*

Proof Denote by $y^* := \operatorname{argmax}_{\tilde{y} \in \mathcal{Y}} f(x, \tilde{y})$. By assumption we have $\xi \geq \Delta(y, y^*) + f(x, y^*) - f(x, y)$. Since $f(x, y^*) \geq f(x, \tilde{y})$ for all $\tilde{y} \in \mathcal{Y}$ the inequality holds. ∎

The construction of the estimator was suggested in Taskar et al. (2004b) and Tsochantaridis et al. (2004), and a special instance of the above lemma is given by Joachims (2005). We now can derive the following optimization problem from (2.9) (Tsochantaridis et al., 2004):

$$\underset{w,\xi}{\text{minimize}} \ \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{n}\xi_i \tag{2.23a}$$

$$\text{s.t. } \langle w, \phi(x_i, y_i) - \phi(x_i, y)\rangle \geq \Delta(y_i, y) - \xi_i \text{ for all } y \in \mathcal{Y}. \tag{2.23b}$$

This is a convex optimization problem which can be solved efficiently if the constraints can be evaluated without high computational cost. One typically employs column-generation methods (Hettich and Kortanek, 1993; Rätsch, 2001; Bennett et al., 2000; Tsochantaridis et al., 2004; Fletcher, 1989) which identify one violated constraint at a time to find an approximate minimum of the optimization problem.

To describe the flexibility of the framework set out by (2.23) we give several examples of its application.

- Binary classification can be recovered by setting $\Phi(x, y) = y\Phi(x)$, in which case the constraint of (2.23) reduces to $2y_i \langle \Phi(x_i), w\rangle \geq 1 - \xi_i$. Ignoring constant offsets and a scaling factor of 2, this is exactly the standard SVM optimization problem.
- Multicategory classification problems (Crammer and Singer, 2000; Collins, 2002b; Allwein et al., 2000; Rätsch et al., 2002a) can be encoded via $\mathcal{Y} = [N]$, where N is the number of classes, $[N] := \{1, \dots N\}$, and $\Delta(y, y') = 1 - \delta_{y,y'}$. In other words, the loss is 1 whenever we predict the wrong class and is 0 for correct classification. Corresponding kernels are typically chosen to be $\delta_{y,y'}k(x, x')$.
- We can deal with joint labeling problems by setting $\mathcal{Y} = \{\pm 1\}^n$. In other words, the error measure does not depend on a single observation but on an entire set of labels. Joachims (2005) shows that the so-called F_1 score (van Rijsbergen, 1979) used in document retrieval and the area under the receiver operating characteristic (ROC) curve (Bamber, 1975; Gribskov and Robinson, 1996) fall into this category of problems. Moreover, Joachims (2005) derives an $O(n^2)$ method for evaluating the inequality constraint over $\mathcal{Y}$.
- Multilabel estimation problems deal with the situation where we want to find the best subset of labels $\mathcal{Y} \subseteq 2^{[N]}$ which correspond to some observation x. The problem is described in Elisseeff and Weston (2001), where the authors devise a ranking scheme such that $f(x, i) > f(x, j)$ if label $i \in y$ and $j \notin y$. It is a special case of a general ranking approach described next.

Note that (2.23) is invariant under translations $\phi(x, y) \leftarrow \phi(x, y) + \phi_0$ where ϕ_0 is constant, as $\phi(x_i, y_i) - \phi(x_i, y)$ remains unchanged. In practice this means that transformations $k(x, y, x', y') \leftarrow k(x, y, x', y') + \langle \phi_0, \phi(x, y)\rangle + \langle \phi_0, \phi(x', y')\rangle + \|\phi_0\|^2$ do not affect the outcome of the estimation process. Since ϕ_0 was arbitrary, we have the following lemma:

Lemma 13 *Let $\mathcal{H}$ be an RKHS on $\mathcal{X} \times \mathcal{Y}$ with kernel k. Moreover, let $g \in \mathcal{H}$. Then the function $k(x, y, x', y') + f(x, y) + f(x', y') + \|g\|^2_{\mathcal{H}}$ is a kernel and it yields the same estimates as k.*

We need a slight extension to deal with general ranking problems. Denote by $\mathcal{Y} = \mathcal{G}[N]$ the set of all directed graphs on N vertices which do not contain loops of less than three nodes. Here an edge $(i, j) \in y$ indicates that i is preferred to j with respect to the observation x. Our goal is to find some function $f : \mathcal{X} \times [N] \to \mathbb{R}$ which imposes a total order on $[N]$ (for a given x) by virtue of the function values $f(x, i)$ such that the total order and y are in good agreement.

More specifically, Dekel et al. (2003), Crammer (2005), and Crammer and Singer (2005) propose a decomposition algorithm $\mathcal{A}$ for the graphs y such that the estimation error is given by the number of subgraphs of y which are in disagreement with the total order imposed by f. As an example, multiclass classification can be viewed as a graph y where the correct label i is at the root of a directed graph and all incorrect labels are its children. Multilabel classification can be seen as a bipartite graph where the correct labels only contain outgoing arcs and the incorrect labels only incoming ones.

This setting leads to a form similar to (2.23) except for the fact that we now have constraints over each subgraph $G \in \mathcal{A}(y)$. We solve

$$\begin{aligned}&\underset{w,\xi}{\text{minimize}} \ \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{n} |\mathcal{A}(y_i)|^{-1} \sum_{G \in \mathcal{A}(y_i)} \xi_{iG} \\ &\text{subject to } \langle w, \Phi(x_i, u) - \Phi(x_i, v)\rangle \geq 1 - \xi_{iG} \text{ and } \xi_{iG} \geq 0 \text{ for all } (u, v) \in G \in \mathcal{A}(y_i).\end{aligned}$$

That is, we test for all $(u, v) \in G$ whether the ranking imposed by the subgraph $G \in y_i$ is satisfied.

Finally, ordinal regression problems which perform ranking not over labels y but rather over observations x were studied by Herbrich et al. (2000) and Chapelle and Harchaoui (2005) in the context of ordinal regression and conjoint analysis respectively. In ordinal regression x is preferred to x' if $f(x) > f(x')$ and hence one minimizes an optimization problem akin to (2.23), with constraint $\langle w, \Phi(x_i) - \Phi(x_j)\rangle \geq 1 - \xi_{ij}$. In conjoint analysis the same operation is carried out for $\Phi(x, u)$, where u is the user under consideration. Similar models were also studied by Basilico and Hofmann (2004).

2.5 Margins and Uniform Convergence Bounds

So far we motivated the algorithms by means of practicality and the fact that $0-1$ loss functions yield hard-to-control estimators. We now follow up on the analysis by providing uniform convergence bounds for large-margin classifiers. We focus on the case of scalar-valued functions applied to classification for two reasons: The derivation is well established and it can be presented in a concise fashion. Secondly, the derivation of corresponding bounds for the vectorial case is by and large still an open problem. Preliminary results exist, such as the bounds by Collins (2002b) for the case of perceptrons; Taskar et al. (2004b), who derive capacity bounds in terms of covering numbers by an explicit covering construction; and Bartlett and

Mendelson (2002), who give Gaussian average bounds for vectorial functions. We believe that the scaling behavior of these bounds in the number of classes $|\mathcal{Y}|$ is currently not optimal, when applied to the problems of type (2.23).

Our analysis is based on the following ideas: firstly the $0-1$ loss is upper-bounded by some function $\psi(yf(x))$ which can be minimized, such as the soft-margin function $\max(0, 1 - yf(x))$ of the previous section. Secondly we prove that the empirical average of the ψ-loss is concentrated close to its expectation. This will be achieved by means of Rademacher averages. Thirdly we show that under rather general conditions the minimization of the ψ-loss is consistent with the minimization of the expected risk. Finally, we combine these bounds to obtain rates of convergence which only depend on the Rademacher average and the approximation properties of the function class under consideration.

concentration

2.5.1 Margins and Empirical Risk

Unless stated otherwise $\mathbf{E}[\cdot]$ denotes the expectation with respect to all random variables of the argument. Subscripts, such as $\mathbf{E}_X[\cdot]$, indicate that the expectation is taken over X. We will omit them wherever obvious. Finally we will refer to $\mathbf{E}_{\text{emp}}[\cdot]$ as the empirical average with respect to an n-sample.

While the sign of $yf(x)$ can be used to assess the accuracy of a binary classifier we saw that for algorithmic reasons one rather optimizes a (smooth function of) $yf(x)$ directly. In the following we assume that the binary loss $\chi(\xi) = \frac{1}{2}(1 - \operatorname{sgn}\xi)$ is majorized by some function $\psi(\xi) \geq \chi(\xi)$, e.g. via the construction of lemma 12. Consequently $\mathbf{E}[\chi(yf(x))] \leq \mathbf{E}[\psi(yf(x))]$ and likewise $\mathbf{E}_{\text{emp}}[\chi(yf(x))] \leq \mathbf{E}_{\text{emp}}[\psi(yf(x))]$. The hope is (as will be shown in section 2.5.3) that minimizing the upper bound leads to consistent estimators.

There is a long-standing tradition of minimizing $yf(x)$ rather than the number of misclassifications. $yf(x)$ is known as "margin" (based on the geometrical reasoning) in the context of SVMs (Vapnik and Lerner, 1963; Mangasarian, 1965), as "stability" in the context of neural networks (Krauth and Mézard, 1987; Ruján, 1993), and as the "edge" in the context of arcing (Breiman, 1999). One may show (Makovoz, 1996; Barron, 1993; Herbrich and Williamson, 2002) that functions f in an RKHS achieving a large margin can be approximated by another function f' achieving almost the same empirical error using a much smaller number of kernel functions.

Note that by default, uniform convergence bounds are expressed in terms of minimization of the empirical risk average with respect to a *fixed* function class $\mathcal{F}$, e.g. Vapnik and Chervonenkis (1971). This is very much unlike what is done in practice: in SVM (2.23) the sum of empirical risk and a regularizer is minimized. However, one may check that minimizing $\mathbf{E}_{\text{emp}}[\psi(yf(x))]$ subject to $\|w\|^2 \leq W$ is equivalent to minimizing $\mathbf{E}_{\text{emp}}[\psi(yf(x))] + \lambda\|w\|^2$ for suitably chosen values of λ. The equivalence is immediate by using Lagrange multipliers. For numerical reasons, however, the second formulation is much more convenient (Tikhonov, 1963; Morozov, 1984), as it acts as a regularizer. Finally, for the design of adaptive

estimators, so-called luckiness results exist, which provide risk bounds in a data-dependent fashion (Shawe-Taylor et al., 1998; Herbrich and Williamson, 2002).

2.5.2 Uniform Convergence and Rademacher Averages

The next step is to bound the deviation $\mathbf{E}_{\text{emp}}[\psi(yf(x))] - \mathbf{E}[\psi(yf(x))]$ by means of Rademacher averages. For details see Boucheron et al. (2005), Mendelson (2003), Bartlett et al. (2002), and Koltchinskii (2001). Denote by $g : \mathcal{X}^n \to \mathbb{R}$ a function of n variables and let $c > 0$ such that $|g(x_1, \ldots, x_n) - g(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)| \leq c$ for all $x_1, \ldots, x_n, x_i' \in \mathcal{X}$ and for all $i \in [n]$, then (McDiarmid, 1989)

$$\Pr\left\{\mathbf{E}\left[g(x_1, \ldots, x_n)\right] - g(x_1, \ldots, x_n) > \epsilon\right\} \leq \exp\left(-2\epsilon^2/nc^2\right). \tag{2.24}$$

Assume that $f(x) \in [0, B]$ for all $f \in \mathcal{F}$ and let $g(x_1, \ldots, x_n) := \sup_{f\in\mathcal{F}} |\mathbf{E}_{\text{emp}}[f(x)] - \mathbf{E}[f(x)]|$. Then it follows that $c \leq \frac{B}{n}$. Solving (2.24) for g we obtain that with probability at least $1 - \delta$,

$$\sup_{f\in\mathcal{F}} \mathbf{E}\left[f(x)\right] - \mathbf{E}_{\text{emp}}\left[f(x)\right] \leq \mathbf{E}\left[\sup_{f\in\mathcal{F}} \mathbf{E}\left[f(x)\right] - \mathbf{E}_{\text{emp}}\left[f(x)\right]\right] + B\sqrt{-\frac{\log\delta}{2n}}. \tag{2.25}$$

This means that with high probability the largest deviation between the sample average and its expectation is concentrated around its mean and within an $O(n^{-\frac{1}{2}})$ term. The expectation can be bounded by a classical symmetrization argument (Vapnik and Chervonenkis, 1971) as follows:

$$\begin{aligned}
&\mathbf{E}_X\left[\sup_{f\in\mathcal{F}} \mathbf{E}[f(x')] - \mathbf{E}_{\text{emp}}[f(x)]\right] \;\leq\; \mathbf{E}_{X,X'}\left[\sup_{f\in\mathcal{F}} \mathbf{E}_{\text{emp}}[f(x')] - \mathbf{E}_{\text{emp}}[f(x)]\right] \\
=\;& \mathbf{E}_{X,X',\sigma}\left[\sup_{f\in\mathcal{F}} \mathbf{E}_{\text{emp}}[\sigma f(x')] - \mathbf{E}_{\text{emp}}[\sigma f(x)]\right] \leq 2\mathbf{E}_{X,\sigma}\left[\sup_{f\in\mathcal{F}} \mathbf{E}_{\text{emp}}[\sigma f(x)]\right].
\end{aligned}$$

The first inequality follows from the convexity of the argument of the expectation; the second equality follows from the fact that x_i and x_i' are drawn i.i.d. from the same distribution; hence we may swap terms. Here σ_i are independent ± 1-valued zero-mean Rademacher random variables. The final term $\mathbf{E}_{X,\sigma}\left[\sup_{f\in\mathcal{F}} \mathbf{E}_{\text{emp}}[\sigma f(x)]\right] := R_n\left[\mathcal{F}\right]$ is referred as the *Rademacher average* (Mendelson, 2001; Bartlett and Mendelson, 2002; Koltchinskii, 2001) of $\mathcal{F}$ w.r.t. sample size n.

Radermacher averages for linear functions

For linear function classes $R_n[\mathcal{F}]$ takes on a particularly nice form. We begin with $\mathcal{F} := \{f | f(x) = \langle x, w \rangle \text{ and } \|w\| \leq 1\}$. It follows that $\sup_{\|w\| \leq 1} \sum_{i=1}^n \sigma_i \langle w, x_i \rangle = \|\sum_{i=1}^n \sigma_i x_i\|$. Hence

$$\begin{aligned} nR_n[\mathcal{F}] &= \mathbf{E}_{X,\sigma} \left\| \sum_{i=1}^n \sigma_i x_i \right\| \\ &\leq \mathbf{E}_X \left[\mathbf{E}_\sigma \left[\left\| \sum_{i=1}^n \sigma_i x_i \right\|^2 \right] \right]^{\frac{1}{2}} \\ &= \mathbf{E}_X \left[\sum_{i=1}^n \|x_i\|^2 \right]^{\frac{1}{2}} \leq \sqrt{n \mathbf{E}\left[\|x\|^2 \right]}. \end{aligned} \tag{2.26}$$

Here the first inequality is a consequence of Jensen's inequality, the second equality follows from the fact that σ_i are i.i.d. zero-mean random variables, and the last step again is a result of Jensen's inequality. Corresponding tight lower bounds by a factor of $1/\sqrt{2}$ exist and they are a result of the Khintchine-Kahane inequality (Kahane, 1968).

Note that (2.26) allows us to bound $R_n[\mathcal{F}] \leq n^{-\frac{1}{2}} r$ where r is the average length of the sample. An extension to kernel functions is straightforward: by design of the inner product we have $r = \sqrt{\mathbf{E}_x[k(x,x)]}$. Note that this bound is *independent* of the dimensionality of the data but rather only depends on the expected length of the data. Moreover r is the trace of the integral operator with kernel $k(x, x')$ and probability measure on $\mathcal{X}$.

Since we are computing $\mathbf{E}_{\text{emp}}[\psi(yf(x))]$ we are interested in the Rademacher complexity of $\psi \circ \mathcal{F}$. Bartlett and Mendelson (2002) show that $R_n[\psi \circ \mathcal{F}] \leq L R_n[\mathcal{F}]$ for any Lipschitz continuous function ψ with Lipschitz constant L and with $\psi(0) = 0$. Secondly, for $\{yb \text{ where } |b| \leq B\}$ the Rademacher average can be bounded by $B\sqrt{2 \log 2/n}$, as follows from (Boucheron et al., 2005, eq. (4)). This takes care of the offset b. For sums of function classes $\mathcal{F}$ and $\mathcal{G}$ we have $R_n[\mathcal{F} + \mathcal{G}] \leq R_n[\mathcal{F}] + R_n[\mathcal{G}]$. This means that for linear functions with $\|w\| \leq W$, $|b| \leq B$, and ψ Lipschitz continuous with constant L, we have $R_n \leq \frac{L}{\sqrt{n}}(Wr + B\sqrt{2 \log 2})$.

2.5.3 Upper Bounds and Convex Functions

We briefly discuss consistency of minimization of the surrogate loss function $\psi : \mathbb{R} \to [0, \infty)$ about which assume that it is convex and that $\psi \geq \chi$ (Jordan et al., 2003; Zhang, 2004). Examples of such functions are the soft-margin loss $\max(0, 1 - \gamma\xi)$, which we discussed in section 2.3, and the boosting loss $e^{-\xi}$, which is commonly used in AdaBoost (Schapire et al., 1998; Rätsch et al., 2001).

Denote by f_χ^* the minimizer of the expected risk and let f_ψ^* be the minimizer of $\mathbf{E}[\psi(yf(x)]$ with respect to f. Then, under rather general conditions on ψ (Zhang, 2004), for all f the following inequality holds:

$$\mathbf{E}[\chi(yf(x))] - \mathbf{E}\left[\chi(yf_\chi^*(x))\right] \leq c\left(\mathbf{E}[\psi(yf(x))] - \mathbf{E}\left[\psi(yf_\psi^*(x))\right]\right)^s. \tag{2.27}$$

In particular we have $c = 4$ and $s = 1$ for soft-margin loss, whereas for boosting and logistic regression $c = \sqrt{8}$ and $s = \frac{1}{2}$. Note that (2.27) implies that the minimizer of the ψ loss is consistent, i.e. $\mathbf{E}[\chi(yf_\psi(x))] = \mathbf{E}[\chi(yf_\chi(x))]$.

2.5.4 Rates of Convergence

We now have all tools at our disposal to obtain rates of convergence to the minimizer of the expected risk which depend only on the complexity of the function class and its approximation properties in terms of the ψ-loss. Denote by $f_{\psi,\mathcal{F}}^*$ the minimizer of $\mathbf{E}[\psi(yf(x))]$ restricted to $\mathcal{F}$, let $f_{\psi,\mathcal{F}}^n$ be the minimizer of the empirical ψ-risk, and let $\delta(\mathcal{F},\psi) := \mathbf{E}\left[yf_{\psi,\mathcal{F}}^*(x)\right] - \mathbf{E}\left[yf_\psi^*(x)\right]$ be the approximation error due to the restriction of f to $\mathcal{F}$. Then a simple telescope sum yields

$$\begin{aligned}\mathbf{E}\left[\chi(yf_{\psi,\mathcal{F}}^n)\right] \leq & \mathbf{E}\left[\chi(yf_\chi^*)\right] + 4\left[\mathbf{E}\left[\psi(yf_{\psi,\mathcal{F}}^n)\right] - \mathbf{E}_{\text{emp}}\left[\psi(yf_{\psi,\mathcal{F}}^n)\right]\right] \\ & + 4\left[\mathbf{E}_{\text{emp}}\left[\psi(yf_{\psi,\mathcal{F}}^*)\right] - \mathbf{E}\left[\psi(yf_{\psi,\mathcal{F}}^*)\right]\right] + \delta(\mathcal{F},\psi) \\ \leq & \mathbf{E}\left[\chi(yf_\chi^*)\right] + \delta(\mathcal{F},\psi) + 4\frac{RW\gamma}{\sqrt{n}}\left[\sqrt{-2\log\delta} + r/R + \sqrt{8\log 2}\right].\end{aligned} \tag{2.28}$$

Here γ is the effective margin of the soft-margin loss $\max(0, 1-\gamma yf(x))$, W is an upper bound on $\|w\|$, $R \geq \|x\|$, r is the average radius, as defined in the previous section, and we assumed that b is bounded by the largest value of $\langle w, x\rangle$. A similar reasoning for logistic and exponential loss is given in Boucheron et al. (2005).

Note that we get an $O(1/\sqrt{n})$ rate of convergence regardless of the dimensionality of x. Moreover, note that the rate is dominated by $RW\gamma$, that is, the classical radius-margin bound (Vapnik, 1995). Here R is the radius of an enclosing sphere for the data and $1/(W\gamma)$ is an upper bound on the radius of the data — the soft-margin loss becomes active only for $yf(x) \leq \gamma$.

2.5.5 Localization and Noise Conditions

In many cases it is possible to obtain better rates of convergence than $O(1/\sqrt{n})$ by exploiting information about the magnitude of the error of misclassification and about the variance of f on $\mathcal{X}$. Such bounds use Bernstein-type inequalities and they lead to localized Rademacher averages (Bartlett et al., 2002; Mendelson, 2003; Boucheron et al., 2005).

Basically the slow $O(1/\sqrt{n})$ rates arise whenever the region around the Bayes optimal decision boundary is large. In this case, determining this region produces

the slow rate, whereas the well-determined region could be estimated at an $O(1/n)$ rate.

Tsybakov's noise condition (Tsybakov, 2003) requires that there exist $\beta, \gamma \geq 0$ such that

$$\Pr\left\{\left|\Pr\{y = 1|x\} - \tfrac{1}{2}\right| \leq t\right\} \leq \beta t^{\gamma} \text{ for all } t \geq 0. \tag{2.29}$$

Note that for $\gamma = \infty$ the condition implies that there exists some s such that $\left|\Pr\{y = 1|x\} - \frac{1}{2}\right| \geq s > 0$ almost surely. This is also known as Massart's noise condition.

The key benefit of (2.29) is that it implies a relationship between variance and expected value of classification loss. More specifically, for $\alpha = \frac{\gamma}{1+\gamma}$ and $g : \mathcal{X} \to \mathcal{Y}$ we have

$$\mathbf{E}\left[\left[\{g(x) \neq y\} - \{g^*(x) \neq y\}\right]^2\right] \leq c\left[\mathbf{E}\left[\{g(x) \neq y\} - \{g^*(x) \neq y\}\right]\right]^{\alpha}. \tag{2.30}$$

Here $g^*(x) := \operatorname{argmax}_y \Pr(y|x)$ denotes the Bayes optimal classifier. This is sufficient to obtain faster rates for finite sets of classifiers. For more complex function classes localization is used. See, e.g., Boucheron et al. (2005) and Bartlett et al. (2002) for more details.

2.6 Conclusion

In this chapter we reviewed some online and batch discriminative models. In particular, we focused on methods employing the kernel trick in conjunction with a large margin loss. We showed how these algorithms can naturally be extended to structured output spaces. We also showed how various loss functions used in the literature are related. Furthermore, we showed statistical properties of the estimators, as, e.g., convergence rates using Rademacher averages and related concepts.