3 Hydroacoustic Signal Classification Using Support Vector Machines

*Matthias Tuma, Christian Igel, and Mark Prior**

CONTENTS

3.1 INTRODUCTION

While the fundamental scientific principles underlying many of the Earth-monitoring systems in operation today have been well known for decades, sensor hardware and sensor deployment are

* The views expressed herein are solely those of the authors and do not necessarily reflect the views of the CTBTO Preparatory Commission.

steadily evolving further. Simultaneously, advances in data transmission and storage leave their own and distinct marks on the field. Today, vast amounts of raw data are routinely being collected and transferred. One key challenge on the receiving end is to reliably extract knowledge that is both meaningful and yet sufficiently condensed, in order to facilitate human (or other) interpretation of the information flow. This especially holds for systems performing real-time monitoring. In general, we obtain information about real-world processes in the form of nonlinear, noisy, multidimensional signals. Both the underlying phenomenon as well as all elements of the propagation path are generally too complex to be fully captured by a physical model. Therefore, best practice data processing and analysis rules have to be inferred from observations and validated empirically. Here, *machine learning* comes into play, which is a branch of computer science and applied statistics covering software that improves its performance on a given task based on sample data or experience. This chapter considers *supervised learning* for classification, which refers to automatically assigning signals to predefined categories. In the supervised learning scenario, the learning machine is provided with sample input–output pairs during a preparational training phase. The learner's task then is to infer from the training data a function that relates any admissible input (not only those seen during training) to a corresponding output. Such a function is called the learner's hypothesis. We demand that all examples used for training as well as later in operations be generated independently of each other by the same probability distribution. Relying on this assumption, the learner should choose a hypothesis which with high confidence will perform accurately on unseen test data. One central challenge in any supervised learning task is to settle a trade-off between on the one hand well fitting the hypothesis to the training examples, and on the other hand hedging against *overfitting*, which occurs when overly adapting to misleading peculiarities of the training data. Section 3.2 formally introduces the supervised learning task, regularized risk minimization, and support vector machines (SVMs) for classification. We introduce SVMs step by step from linear classification to the full kernelized case. This chapter emphasizes aspects deemed relevant to practitioners and at the same time strives to include a reasonably systematic overview over regularized risk minimization, SVM classification, and SVM model selection. The application of these concepts is illustrated through a classification task in acoustic remote sensing. We learn to discriminate signals stemming from the verification network for the Comprehensive Nuclear-Test-Ban Treaty (CTBT). The CTBT's permanently installed International Monitoring System (IMS) consists of several hundred geophysical sensors and relies on four different monitoring technologies. In particular, we are concerned with distinguishing explosive-like and nonexplosive signals recorded by the IMS underwater sensor network. Section 3.3 provides background on the IMS as well as preprocessing routines relevant to the application. We then approach the actual problem of CTBT hydroacoustic signal classification by drawing on the generic concepts established earlier. Section 3.4 concludes this chapter.

3.2 SUPERVISED LEARNING AND SUPPORT VECTOR MACHINE CLASSIFICATION

We next formalize the supervised learning and classification task. In the standard setting, we have obtained training data S from the same data-generating process to which we intend to apply the trained learning machine. The data S consists of N exemplary input–output pairs $(x_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $1 \le i \le N$. The input and output domains $\mathcal{X}$ and $\mathcal{Y}$ can in general be any (nonempty) set. Given S, the output of a learning machine is a prediction function or hypothesis $h: \mathcal{X} \to \mathcal{Y}$ from the input set $\mathcal{X}$ to the output set $\mathcal{Y}$. For each possible input $x \in \mathcal{X}$ we present to h, it will yield its prediction $h(x) \in \mathcal{Y}$ of what output value should most likely be associated with x. It is not sufficient to learn the training data by heart. We rather want h to perform as well as possible for the entirety of input values in $\mathcal{X}$, and especially for those that occur often. When we

know the true label y of a sample x, we can compare it to the learner's prediction $h(x)$. Feeding both to a *loss function* L

$$L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_0^+,$$
$$\text{with the property: } L(h(x), y) = 0 \quad \text{for } h(x) = y, \tag{3.1}$$

assigns a "cost" to predicting $h(x)$ when the true label is y. Formally, the goal of supervised learning may now be expressed as finding a function h that minimizes the overall cost, or *risk* $\mathcal{R}_p$, when evaluated over the entire probability distribution $p(x, y)$ underlying our data-generating process:

$$h = \arg\min_{\hat{h} \in \mathcal{H}} \mathcal{R}_p(\hat{h}) = \arg\min_{\hat{h} \in \mathcal{H}} \int_{\mathcal{X} \times \mathcal{Y}} L(\hat{h}(x), y)\, \mathrm{d}p(x, y), \tag{3.2}$$

where $\mathcal{H}$ would ideally be the space of all (measurable) functions mapping from $\mathcal{X}$ to $\mathcal{Y}$. A hypothesis minimizing Equation 3.2 is called a *Bayes optimal* solution, the according risk the *Bayes risk* of p, and both depend on p and the loss function employed. Note that the Bayes risk is not necessarily zero. This is easy to see for finite $\mathcal{X}$. If an input pattern $x \in \mathcal{X}$ can belong to two different classes—$p(x, y_1) > 0$ and $p(x, y_2) > 0$ for different $y_1, y_2 \in \mathcal{Y}$—the best possible hypothesis still can map x only to one class and will inevitably make mistakes. In practice, nonzero Bayes risk frequently occurs in the case of noisy input signals, signals describing the underlying process incompletely, or uncertain labels.

Clearly, if we knew the underlying distribution p, we would have complete knowledge about the data-generating process. But p usually is unknown. Thus, for all practical purposes the overall risk $\mathcal{R}_p$ can neither be computed nor optimized directly, even if the integrals in Equation 3.2 were tractable. One step toward optimizing Equation 3.2 is to replace $\mathcal{R}_p$ by the equivalent quantity restricted to the training data. This defines the *empirical risk* $\mathcal{R}_S$ on S,

$$\mathcal{R}_S(\hat{h}) = \frac{1}{N} \sum_{i=1}^{N} L(\hat{h}(x_i), y_i), \tag{3.3}$$

which is simply the average loss on the training data. A minimizer of $\mathcal{R}_S$ is a hypothesis as consistent with the training examples as possible. Minimum empirical risk can, for example, be achieved by a function that reproduces the labels of all training examples and merely returns one single, arbitrary output for all other possible inputs. Evidently, this would be a poor prediction function, and a better objective than to just minimize Equation 3.3 over all functions is needed.

3.2.1 Regularized Risk Minimization

A hypothesis h should not solely reflect peculiarities of the given training data, but work well for examples previously unencountered. What kind of quantity can assist us—without knowing more about the data-generating distribution than S—in automatically deciding which hypothesis may have *overfitted* to the training data and which may *generalize* well to unseen data? It is not only intuitive to look for a simple hypothesis still yielding a reasonably low empirical risk, but a range of according theorems in statistical learning theory (e.g., [34,39]) formalize and justify this concept. These theorems typically provide bounds on the risk in the following form. If, of all functions in a certain function or hypothesis space $\mathcal{H}$, h is the minimizer of $\mathcal{R}_S$, then with probability of at least $1 - \delta$ it holds that $\mathcal{R}_p(h) \leq \mathcal{R}_S(h) + B(N, \delta, \mathcal{H})$. The function B bounds the extent to which the true risk might exceed the empirical risk. An increase in both the number of training examples N and the uncertainty δ leads to a decrease in B. Most importantly, B increases with the complexity of $\mathcal{H}$. We argue that any function $B(N,\delta,\mathcal{H})$ which permits inequalities like the above provides a measure

of the complexity of a function space $\mathcal{H}$—and therefore simplicity of a hypothesis class can be (nonuniquely) defined in precise ways. Such theorems confirm that if we permit the minimizer of $\mathcal{R}_S$ to stem from a highly complex class, this expressive power might be exploited for overfitting on the training data rather than be helpful in producing better hypotheses. We thus want to enforce the preference that if the learning machine draws from a more complex hypothesis class, it should have a very good justification in terms of—sufficiently—high associated decrease in training error. The hypothesis spaces considered in this chapter can be endowed with a norm $\| \|_{\mathcal{H}}$ serving as a measure of complexity of a hypothesis. Then we can express the aforementioned trade-off within the *regularized risk minimization* paradigm, in which h is found by minimizing the regularized risk $\mathcal{P}_S$,

$$\mathcal{P}_S(\hat{h}) = \|\hat{h}\|_{\mathcal{H}^2} + C\sum_{i=1}^{N} L(\hat{h}(x_i), y_i). \tag{3.4}$$

Here, $C \in \mathbb{R}^+$ is the so-called *regularization parameter* and balances the preference for low training error (right summand) against keeping the hypothesis simple (left summand), where complexity is assumed to correlate to the norm in $\mathcal{H}$, cf. Refs. [34,39].

3.2.2 Support Vector Machines

In general, a multitude of supervised learning algorithms exist, which may or may not fall into the framework of regularized risk minimization presented in the previous section. In this chapter, we focus on SVMs [7, 29], which are most commonly used for classification, but also applicable to regression and density estimation tasks. SVMs can be seen as composed of building blocks from originally different areas of research and, for example, are linked to functional analysis and convex optimization. Next, we introduce SVMs step by step as composition of concepts, namely straightforward linear classification seeking for large separating margins; allowing for margin violations in a regularized risk minimization framework; and nonlinear classification via kernel functions, which replace the scalar product in the original input space by a scalar product in another, unrealized feature space.

3.2.3 Linear Classification

From now on we assume, for clarity, that all inputs are represented by an m-dimensional, real-valued feature vector $x \in \mathbb{R}^m = \mathcal{X}$. Further, we restrict our considerations to two-class or binary classification and set $\mathcal{Y} = \{-1, 1\}$. A two-class hypothesis function h: $\mathcal{X} \to \{-1, 1\}$ with a linear decision boundary can be realized through an affine linear *decision function* f: $\mathcal{X} \to \mathbb{R}$, $f(x) = \langle x, w\rangle + b$, by taking the sign of f:

$$h(x) = \mathrm{sgn}(f(x)) = \mathrm{sgn}\Big(\big\langle x, w\big\rangle + b\Big). \tag{3.5}$$

Here, the weight vector $w \in \mathcal{X}$ lies in the same space as the input data, $b \in \mathbb{R}$ is a real-valued offset term, and sgn() is the standard signum function except for an argument of zero, in which case it returns +1. The decision boundary is the subspace of all points x for which $\langle x, w\rangle + b = 0$. For two-dimensional input, this subspace is a line, and a hyperplane of dimension $m - 1$ in general. The vector w is perpendicular to the decision boundary. With an offset or bias term of zero, $b = 0$, the decision surface passes through the origin. For $b \neq 0$, the decision surface is shifted from 0 by a distance of $\frac{b}{\|w\|}$.

The quantity $y_i(\langle w, x_i\rangle + b) = y_i f(x_i)$ is positive if the ith pattern (x_i, y_i) is classified correctly by the hypothesis $\mathrm{sgn}(f(x))$. We call this quantity the *functional margin* of (x_i, y_i) with respect to the linear decision boundary induced by (w, b). The *geometric margin* of (x_i, y_i) with respect to the linear decision boundary induced by (w, b) is given by $y_i f(x_i)/\|w\|$. The absolute value of the geometric margin is the distance of x_i from the decision boundary in the input space. A collection S of N two-class data points is called linearly separable if there exists a linear classifier (w, b) separating both classes

without error. This implies $y_i(\langle w, x_i\rangle + b)/\|w\| \geq \rho > 0$ for all $1 \leq i \leq N$. The largest ρ for which this holds true defines the geometric margin of the linear classifier (w, b) with respect to S. Its value is determined by the data point having the smallest margin. In the following, we will not explicitly distinguish between functional and geometric margin when the meaning is clear from the context.

3.2.4 Linear Support Vector Machines

We first introduce large-margin SVM classification for linearly separable data. Figure 3.1 shows a separable two-class problem in two dimensions, and in the upper left a "barely separating"

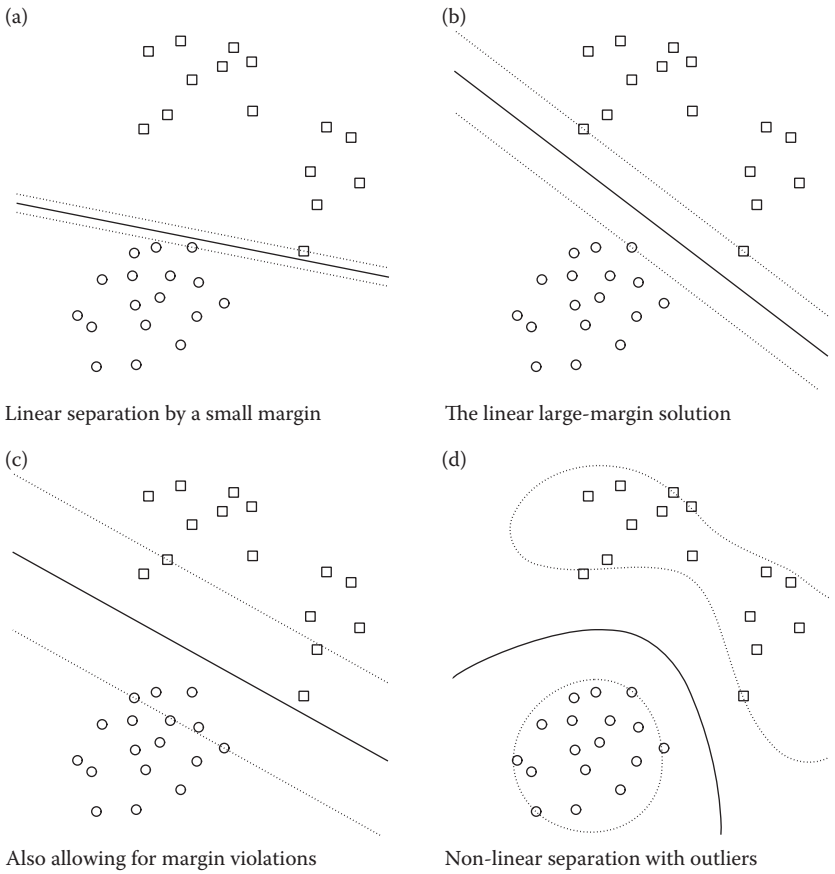

FIGURE 3.1 Example of linearly separable two-class data, and different hypotheses for a discrimination boundary between them. (a) A hypothesis (solid line) that might have been proposed by an algorithm not maximizing the margin (distance to dotted lines) between samples and the discrimination boundary. (b) A maximum-margin solution found by the linear hard-margin SVM of Equation 3.6. We can expect the solution in (a) to be less reliable when classifying data from the same generating distribution. (c) The hypothesis produced by a soft-margin SVM using $C > 0$ in Equation 3.7. Note that its different slope and position could not have been reached in Equation 3.6. (d) A curved decision surface obtained by solving the canonical, nonlinear SVM optimization problem (Equation 3.10), using $C > 0$ and a radial basis function kernel (cf. Section 3.2.5) with $\gamma > 0$. While (a) clearly is not a large-margin classifier, all three hypotheses in (b) through (d) are valid solutions to the full SVM optimization problem. Solution (b) can be seen as a special case of (c) with a strong preference against margin violations expressed through a large regularization parameter. Both linear solutions are further special cases of the nonlinear one, using the scalar product in the original feature space as kernel function. In other words, (b) through (d) have all been obtained by solving the SVM optimization problem (Equation 3.10) for different choices of the regularization parameter C and kernel function k.

hyperplane. Intuitively, we see that this hypothesis is quite vulnerable, as already little noise on samples close to the boundary would lead to their misclassification. Therefore, *hard margin* SVMs for separable training data S yield a hypothesis for which the smallest margin of a training data point—the distance between the decision hyperplane and the closest data point in S—is maximal. It can be shown that solving the optimization problem

$$\begin{aligned} &\text{minimize}_{w,b} && \frac{1}{2}\langle w, w\rangle \\ &\text{subject to} && y_i\left(\langle w, x_i\rangle + b\right) \geq 1, \quad 1 \leq i \leq N, \end{aligned} \tag{3.6}$$

leads to a decision function with a geometric margin of $\frac{1}{\|w\|}$, so that the objective of problem (3.6) exactly ensures a hypothesis with maximum margin [29]. The term "hard margin" refers to the fact that the constraints in Equation 3.6 strictly enforce a functional margin of at least one for each training pattern, which implies correct classification of all examples. Applied to the previous example in Figure 3.1, the large-margin objective (Equation 3.6) generates the hypothesis shown in the upper right. In practice, few datasets are linearly separable in the input space. But even for those, it may be beneficial to allow for misclassification of training patterns if in turn a more appealing hypothesis can be established with respect to the overall data. Obviously, if the Bayes risk is nonzero and the pool of training data sufficiently large, the Bayes optimal hypothesis makes errors on the training data. We thus want to allow for *margin violations*, that is, for training patterns (x_i, y_i) for which $y_i(\langle w, x_i\rangle + b) < 1$. This includes misclassified patterns, for which $y_i(\langle w, x_i\rangle + b) < 0$ indicates that they lie on the "wrong" side of the hyperplane. We, for these reasons, return to the concept of regularized risk minimization established in Section 3.2.1. To make usable an objective of the form of Equation 3.4, a loss function $L(h(x), y)$ has to be chosen. For classification we would ideally like to use the 0-1-loss, which returns 0 for correct classification, $h(x) = y$, and 1 otherwise. Incorporating the nonconvex 0-1-loss into Equation 3.6 would, however, complicate the optimization procedure by voiding convexity of the overall problem. For this reason, SVMs commonly rely on the *hinge loss* $L(f(x), y) = \max(0, 1 - yf(x))$ defined on the SVM decision function f as a convex *surrogate loss function*. The hinge loss also is the tightest convex upper bound on the 0-1-loss. Introducing the possibility for margin violations into Equation 3.6 and penalizing them by the hinge loss yields

$$\begin{aligned} &\text{minimize}_{\xi,w,b} && \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{N}\xi_i \\ &\text{subject to} && y_i\left(\langle w, x_i\rangle + b\right) \geq 1 - \xi_i, \quad 1 \leq i \leq N \\ & && \xi_i \geq 0, \quad 1 \leq i \leq N. \end{aligned} \tag{3.7}$$

This is the linear *soft margin* SVM optimization problem. Each variable $\xi_i \geq 0$ measures the margin violation of pattern (x_i, y_i). Their sum accounts for all violations of the separability paradigm by exactly the sum of the corresponding hinge losses, which in turn is penalized within the objective function. The lower left of Figure 3.1 shows the result of solving Equation 3.7 for $C > 0$. While the difference on the toy dataset is not drastic, it exemplifies the fact that Equation 3.7 allows for solutions not reachable through Equation 3.6. The hard-margin solution can however still be obtained by letting C tend to infinity. We are now only an additional step short of obtaining the canonical nonlinear SVM formulation as we will also employ in the experimental Section 3.3.

3.2.5 Kernelized Support Vector Machines

Solving Equation 3.7 will yield a regularized large-margin hypothesis, however always using a linear decision function. This can be a disadvantage, for example, when imagining a two-dimensional classification problem in which samples of one class all lie within a circle around the origin, and samples of the second class surround them in a ring-like fashion, as illustrated on the left of Figure 3.2. SVMs incorporate nonlinear hypotheses by implicitly transforming the input data to an unrealized, possibly high- or infinite-dimensional dot product space via kernel functions. In this feature space, different from the input space, SVMs perform linear classification. This, in general, gives rise to nonlinear decision surfaces in the original input space. From another angle, one might imagine subjecting the input data to a nonlinear transformation and only then solving Equation 3.7 for the transformed input. In the right of Figure 3.2, the result of carrying out such a nonlinear transformation from the input space into a higher-dimensional space is shown. If the new feature space, however, is of high dimension, carrying out computations dimension-wise is time consuming, and the transformation might be as well. Instead, we use the fact that a solution w of Equation 3.7 admits a representation of the form $w = \sum_i^N \alpha_i x_i$ with $\alpha \in \mathbb{R}^N$ [9,31]. In other words, it is guaranteed that the solution is a linear expansion of the training examples.

Substituting into Equation 3.7 yields

$$\begin{aligned} \text{minimize}_{\xi,\alpha,b} \quad & \frac{1}{2}\sum_{i,j=1}^{N}\alpha_i\alpha_j\langle x_i, x_j\rangle + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i\left(\sum_{j=1}^{N}\alpha_j\langle x_j, x_i\rangle + b\right) \geq 1-\xi_i, \quad 1 \leq i \leq N \\ & \xi_i \geq 0, \quad 1 \leq i \leq N. \end{aligned} \tag{3.8}$$

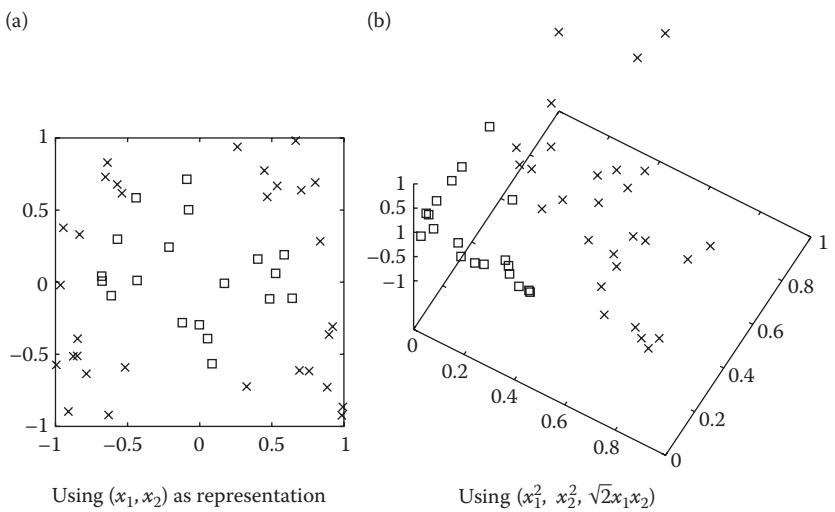

FIGURE 3.2 Example of how an embedding into another feature space can turn a linearly nonseparable dataset into a linearly separable one. Squares and crosses indicate examples of two different classes. (a) Samples are represented by their raw input coordinates $(x_1; x_2)$. (b) A nonlinear feature map ϕ: $\mathbb{R}^2 \to \mathbb{R}^3$ is used to change the representation from (x_1,x_2) to $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$.

Looking at Equation 3.8 we see that it is solely formulated in terms of scalar products between training examples. Where the examples mapped to another dot product space $\mathcal{F}$ by the map ϕ: $\mathbb{R}^m \to \mathcal{F}$ before solving Equation 3.8, each occurrence of $\langle x_i, x_j \rangle$ would simply have to be replaced by $\langle \phi(x_i), \phi(x_j) \rangle_F$. In order to allow for efficient computation of the dot product in $\mathcal{F}$, we can use a kernel function k: $\mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ with $k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle \mathcal{F}$. More importantly, one can proceed the other way around and specify a kernel k in order to solve Equation 3.8 in another dot product space. The theory of reproducing kernels [1,2,29] establishes the requirements k has to fulfill in order to be sure that substituting $k(x_i, x_j)$ for the scalar product will actually correspond to solving Equation 3.8 in some valid dot product space. The only condition on a symmetric function k is that k must be *positive definite*, in the sense that for every collection of points from $\mathcal{X}$, the matrix K of kernel entries between these points, $K_{ij} = k(x_i, x_j)$, must be positive definite. In detail, for a nonempty input set $\mathcal{X}$ and a function k: $\mathcal{X} \times \mathcal{X} \to \mathbb{R}$, the following holds:

$$\forall x, z \in \mathcal{X} : k(x,z) = k(z,x) \quad \text{and} \quad \forall n \in \mathbb{N}, x_1, \ldots, x_n \in \mathcal{X}, c_1, \ldots, c_n \in \mathbb{R} : \sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0$$
$$\Rightarrow \exists (\mathcal{F}, \phi : \mathcal{X} \to \mathcal{F}) \forall x, z \in \mathcal{X} : k(x,z) = \left\langle \phi(x), \phi(z) \right\rangle_{\mathcal{F}}. \tag{3.9}$$

Equation 3.9 states an equivalence between a kernel function k and a scalar product in some dot product space $\mathcal{F}$ as long as k is symmetric and positive definite. Subsequently, we list commonly used families of positive-definite kernels. Completing the kernelization of the SVM optimization problem, we state the final objective function which allows for both misclassified training examples and nonlinear decision surfaces using kernel functions:

$$\begin{aligned} &\text{minimize}_{\xi,\alpha,b} && \frac{1}{2} \sum_{i,j=1}^{N} \alpha_i \alpha_j k(x_i, x_j) + C \sum_{i=1}^{N} \xi_i \\ &\text{subject to} && y_i \left(\sum_{j=1}^{N} \alpha_j k(x_j, x_i) + b \right) \geq 1 - \xi_i, \quad 1 \leq i \leq N \\ & && \xi_i \geq 0, \quad 1 \leq i \leq N. \end{aligned} \tag{3.10}$$

This constitutes the canonical SVM optimization problem [7]. The decision function f of an SVM is linear in the kernel-induced feature space. In the original input space, the SVM's final hypothesis h takes the form

$$h(x) = \text{sgn}(f(x)) = \text{sgn}\left(\sum_{i=1}^{N} \alpha_i k(x, x_i) + b \right). \tag{3.11}$$

The lower right of Figure 3.1 shows a classifier obtained from Equation 3.10 using $C > 0$ and a radial basis function kernel of $\gamma > 0$ (see below). The solution would differ considerably for different values, and even more so for other kernel function families. The final SVM hypothesis does not depend on correctly classified training examples with a distance to the decision hyperplane larger than the safety margin. Their coefficients α_i will hence be zero. All samples with nonzero coefficients α_i will lie on the margin or violate it (i.e., have $y_i f(x_i) = 1$ or $y_i f(x_i) < 1$, respectively). These are called *support vectors*, and only they contribute to the sum in Equation 3.11. This *sparsity property* of SVMs reduces the computational burden when evaluating the hypothesis on unseen examples.

3.2.5.1 Kernel Functions

Question arises as how to choose a proper kernel function. We first introduce two families of kernels commonly used for real-valued input vectors.

Polynomial kernel. For a nonnegative, real-valued offset parameter c and a positive integer exponent d, the function $k(x, z) = (\langle x, z\rangle + c)^d$ is a kernel on real-valued input vectors $x, z \in \mathbb{R}^m$. For $c = 0$ and $d = 1$, it reduces to the standard scalar product.

Gaussian kernel. A general Gaussian kernel on real-valued input vectors $x, z \in \mathbb{R}^m$ is given by $k(x,z) = e^{-(x-z)^T Q(x-z)}$, where Q is a positive-definite $m \times m$ matrix. The most common variant is the radial basis function (RBF) kernel using a positive scaling of the identity matrix I, $Q = \gamma I$, $\gamma \in \mathbb{R}^+$, which introduces γ as single free parameter. A theoretically appealing property of RBF kernels is that they fulfill a necessary condition for an SVM to be *universally consistent* [35]: an SVM using an RBF kernel will under mild conditions converge to the Bayes optimal hypothesis as the collection of training examples grows. Another variant is the automatic relevance detection (ARD) kernel, for which $Q_{ij} = \delta_{ij}\gamma_i$, with Kronecker delta δ. The ARD kernel owes its name to the fact that learning values for the m positive parameters $\gamma_i \in \mathbb{R}^+$ can provide insight into the relevance of individual features for classification. As a drawback, the ARD kernel introduces as many free parameters as there are input space dimensions. However, efficient parameter optimization has been demonstrated for both the ARD and the general Gaussian kernel ([14,15], also see Section 3.2.7).

Individual application domains, for example in biology or natural language processing, can require the use of highly specific and task-tailored kernel functions. Their design and analysis often constitute an active area of research of its own (e.g., [13,30]). Positive-definite kernels further exhibit convenient closure properties in the sense that several operations between kernels again yield a valid kernel [2]. For two kernels k_1 and k_2, their product $k_1 \cdot k_2$ and sum $k_1 + k_2$ is positive definite. Similarly, all following operations on a kernel k retain positive definiteness: scaling by a positive constant to ak, $a \in \mathbb{R}^+$; taking k as exponent e^k; or normalizing k to $\overline{k}(x,z) = k(x,z)/\sqrt{k(x,x)k(z,z)}$. Further, closure under sum and product also imply closure under direct sum and direct product: let $k_a(x, z)$ be a kernel on $A \times A$ and $k_b(u, w)$ a kernel on $B \times B$. Then, kernels $k((x, u), (z, w))$ on $(A \times B) \times (A \times B)$ are both given by $k_a(x, z) \cdot k_b(u, w)$ and $k_a(x, z) + k_b(u, w)$. This, for example, is useful when working with combinations of features from different domains.

Convolution kernels. Not a standard choice of kernel function as such, we list Haussler's convolution kernel* [18] in preparation for an application in Section 3.3. Suppose the input space $\mathcal{X} = \mathbb{R}^m$ can be divided into a product of g subspaces. For simplicity, we assume that these subspaces are equivalent and write $\mathcal{X} = S^g$. This yields a partitioning of an input vector x into g subvectors $x^{(i)} \in S = \mathbb{R}^{\binom{m}{g}}$, $1 \le i \le g$, of equal length. Further, assume that we have g corresponding subkernels $k^{(i)}$: $S \times S \to \mathbb{R}$ defined on the subspace S. By the above composition rules we can construct a composite kernel on $\mathcal{X} \times \mathcal{X}$ by, for example, adding all subkernels, $k(x,z) = \sum_i k^{(i)}(x^{(i)}, z^{(i)})$, or multiplying them, $k(x, z) = \prod_i k^{(i)}(x^{(i)}, z^{(i)})$. The convolution or ANOVA kernel k_D of order D, $D \in \{1, \ldots, g\}$, generalizes from these two exemplary compositions by viewing both as sums over all possible monomials (multiples excluded) of degree D:

$$k_D(x,z) = \sum_{1 \le j_1 < \cdots < j_D \le g} \prod_{d=1}^{D} k^{(j_d)}(x^{(j_d)}, z^{(j_d)}). \tag{3.12}$$

Here, the sum runs over all possibilities to draw unique subsets of size D from $\{1, \ldots, g\}$. Clearly, the convolution kernel k_D is the direct sum kernel for $D = 1$ and the direct product kernel for $D = g$. In general, k_D yields the sum of all monomials (multiples excluded) of order D.

* The special form of convolution kernel as we consider here has also been studied in [28,39] as ANOVA kernels. We refer the reader to Haussler's overarching construction as the most systematic one.

3.2.6 Support Vector Machine Optimization

One of the canonical approaches to SVM optimization, that is, to solving problem 3.10, is to derive the corresponding *dual program* via Lagrange multipliers [7,39]. The resulting constrained quadratic optimization problem would be solvable using off-the-shelf methods, but highly efficient tailored methods have been derived. Decomposition methods for SVMs [21,25] are iterative algorithms that operate on a fixed subset of (usually two) variables per iteration. It is generally accepted that such solvers have a runtime complexity between quadratic and cubic in the number of training examples [21]. When working with large datasets, fast solvers specialized on linear kernels can be employed, at the cost of purely linear decision functions (e.g., [10,33]). Alternatives that also support nonlinear kernels are *online* SVM solvers (e.g., [3]), which usually aim for an approximate solution.

3.2.6.1 Support Vector Machine Execution

In general, the hypothesis put forth by an SVM is *sparse*: it only depends on the fraction of training examples that are support vectors. SVM execution can thus be faster than, for example, naive implementations of nearest-neighbor classification. It has, however, been shown that the number of support vectors itself grows linearly with the number of training examples if the Bayes risk is nonzero [36]. A number of different approaches have been proposed for approximating the final solution when classification speed is important (e.g., [24,32]).

3.2.7 Model Selection

Maybe the most important aspect in SVM usage is that of model selection—the process of choosing the regularization parameter C and a kernel function family as well as values for the kernel parameters. A multitude of methods for hyperparameter selection exist. Most of them optimize an approximation of, bound on, or heuristic substitute for the generalization error, that is, for Equation 3.2 using the 0-1-loss. We present the standard method, grid search on the cross-validation error, together with a gradient-based maximum-likelihood approach better suited for kernels with more than a few parameters.

3.2.7.1 Direct Search

One common estimator of the generalization error is the n-fold cross-validation error. By partitioning the available training data S into n different parts or *folds*, one obtains n possibilities for training a classifier on $n - 1$ parts of S. The average of the n validation errors on each remaining single evaluation fold is the n-fold cross-validation error (CV-n). It can be shown that the CV-n is a slightly biased estimator of the generalization error, and choices of $n = 5$ or $n = 10$ have proven reasonable in practice [17]. As the CV-n is not differentiable, the hyperparameter space is generally probed at multiple locations using some direct (zeroth-order) search heuristic.

Grid search. The most common SVM model selection procedure, grid search, defines a multidimensional grid of points in the hyperparameter search space, where the grid points may, for example, be spaced evenly on a linear or logarithmic scale. For each point on the grid, n SVMs are trained (each on a different union of $n - 1$ folds) using the corresponding hyperparameter vector. The parameter combination yielding lowest CV-n over the entire grid is in turn chosen to train the final classifier, now using all data in S. Due to the curse of dimensionality, CV-n gets prohibitive for more than a few free SVM hyperparameters. Sometimes variations such as nested grid search are employed, where the grid resolution iteratively increases while focusing around the previously best point. Beyond grid search, more elaborate direct search techniques, such as evolution strategies, have successfully been applied to SVM model selection [11,19].

3.2.7.2 Gradient-Based Model Selection

A lot of research has been devoted to developing and evaluating differentiable estimates of, bounds on, or heuristic substitutes for the generalization error (see e.g., [5]). Even if such an objective is not convex (i.e., does not prevent gradient-based optimizers from getting stuck in suboptimal local optima) gradient descent is still appealing. First, the directional information provided by the gradient guides the search. Second, the derivative of the objective might be faster to evaluate at a given point in the hyperparameter space than training n SVMs for CV-n. As a consequence, gradient-based approaches can have significant advantages over direct search, especially when the parameter space has more than a few dimensions.

Maximum-likelihood model selection. We present one recent gradient-based model selection algorithm which has been shown to outperform several established methods on a large benchmark set [15]. The classification error in general is a nondifferentiable quantity with respect to the parameters of a deterministic hypothesis. In contrast, assume a probabilistic classifier approximating the probability $P(y|x)$ of observing class $y \in \mathcal{Y}$ given input $x \in \mathcal{X}$ by some model $\hat{P}(y|x)$, where $\hat{P}$ depends smoothly on its parameters and the hyperparameters of the learning algorithm. A typical approach for learning these parameters is maximizing the *logarithmic likelihood function* $\mathcal{L} = \sum_{(x_i,y_i)\in S} \log \hat{P}(y_i|x_i)$ with respect to the adaptive parameters, which can be done using gradient ascent.

Learning a model of $P(y|x)$ is more general and therefore usually more difficult task than "just" learning a hypothesis for classification. While a perfect model gives the Bayes optimal hypothesis by $h(x) = \arg\max_{y\in\mathcal{Y}} P(y|x)$, a bad model leads to bad classification results. In the SVM framework, one therefore searches for a proper hypothesis directly without estimating $P(y|x)$. This at the same time prevents us from using the maximum-likelihood approach for model selection as described above. Therefore, Glasmachers and Igel [15] use a probabilistic interpretation of the output of an already-trained SVM solely for the purpose of model selection. They follow an approach by Platt [26], who proposed to estimate class membership probabilities $P(y = +1 \mid f(x))$ from SVM decision functions f by fitting a simple sigmoid $\sigma_{r,s}(f(x)) = 1/(1 + \exp(s \cdot f(x) + r))$ around f, where $s \in \mathbb{R}^-$ and $r \in \mathbb{R}$ are the scaling and offset parameter, respectively. This fitting can be done by gradient-based optimization of a cross-validation estimate of the likelihood. For an SVM decision function f, a sigmoid $\sigma_{r,s}$ around f, and validation data S', the likelihood is

$$\mathcal{L}(S', \sigma_{r,s}, f) = \sum_{\substack{(x',y')\in S' \\ y'=+1}} \log\sigma_{r,s}(f(x')) + \sum_{\substack{(x',y')\in S' \\ y'=-1}} \log(1 - \sigma_{r,s}(f(x'))). \tag{3.13}$$

In Ref. [15], this quantity is optimized with respect to the kernel parameters and the regularization parameter C of the SVM using gradient ascent. This requires the kernel function to depend smoothly on its parameters. The derivative of the SVM with respect to its hyperparameters and the kernel can be computed using the procedure proposed in Ref. [22]. It has to be stressed that the probabilistic interpretation of the SVM output is a heuristic, because the SVM (on purpose) does not aim at learning proper probabilities. However, the probabilistic interpretation is solely used to guide model selection. In practice, this maximum-likelihood approach to SVM model selection achieves state-of-the-art results and especially performs well when optimizing flexible kernels on small datasets [15].

The model selection algorithms described in this chapter are all available as part of the open-source machine learning library Shark [20].

3.2.8 Summary

To conclude our introduction to SVMs for classification, we emphasize the following properties of these powerful learning machines:

- *Convex optimization*. When all of an SVM's hyperparameters are fixed, SVM training corresponds to solving a convex quadratic optimization problem, which is free from suboptimal local extrema. In other words, SVMs always return some best solution possible given a fixed set of hyperparameters.
- *Consistency*. Under relatively mild conditions, standard SVMs are known to be universally consistent. As the number of training examples increases, the SVM solutions converge to the Bayes optimal hypothesis, which is the best classifier possible given the data-generating distribution [35].
- *Kernel trick*. As long as a kernel function can be defined between them, the input patterns may be arbitrary elements. For example, text documents, graphs, or trees can directly serve as input to an SVM if a valid kernel function between them is defined. Kernel-based algorithms in general can be seen as elegantly separating the general part of a learning machine from the problem-specific part. The kernel function (as well as the regularization parameter) allows for incorporation of domain-specific prior knowledge into the learning process, which is necessary to achieve well-generalizing hypotheses.
- *Model selection problem*. SVMs—like most other learning machines—are not parameter-free in the sense that they do not intrinsically determine all entities their behavior is influenced by. The regularization parameter and a kernel function with additional free parameters must be specified externally. However, there cannot be a "universal" learning machine that excels across all possible problems (e.g., [4]). That is, one has to incorporate prior knowledge to tailor a learning machine to some given task. For SVMs this includes choosing the SVM hyperparameters such as a proper family of kernel functions.
- *Multiclass classification*. There is no canonical extension of the binary SVM formulation (Equation 3.10) to multiclass problems with $|\mathcal{Y}| > 2$. Many application studies in the multiclass case rely on training and combining a number of binary machines. At the same time, several multiclass SVM formulations exist that solve the multicategory classification task in one joint optimization problem [8,23,40]. All these have different properties and require solvers distinct from those used for purely binary problems.
- *Training and execution time*. Training time in general is between quadratic and cubic in the number of training examples and additionally influenced by input dimension and SVM hyperparameters [21]. Evaluating an SVM on an unseen example benefits from the sparseness property and is linear in the number of support vectors times the number of operations per kernel function evaluation. If the Bayes risk is nonzero, the number of support vectors scales linearly with the number of training examples [36]. Several extensions or modifications have been proposed to reduce training and execution times. The latter can be achieved, for instance, by approximating the SVM solution after training [27,37].

In summary, consistency and convexity constitute convenient theoretical advantages that provide certain guarantees on the solution obtained by an SVM. In order for a practitioner to obtain truly meaningful results, however, some familiarity with the model selection problem and standard techniques to approach it are essential. While SVM learning for nonstandard domains—such as trees, graphs, or text—is well supported conceptually, this often gives rise to highly expertized fields of research in itself.

3.3 HYDROACOUSTIC SIGNAL CLASSIFICATION

We consider an application task of hydroacoustic signal classification using remote sensor data from the CTBT verification network.

3.3.1 Nuclear-Test-Ban Verification

International arms control treaties must be verifiable with high confidence. A verification regime is a set of technological and administrative measures that discourages attempts toward treaty violation from the start by making actual violations detectable with high probability. The CTBT is an international agreement banning all nuclear explosions. As of 2011, it awaits formal ratification by nine further states before it will enter into force. Since 1996, the *Preparatory Commission for the Comprehensive Nuclear-Test-Ban Treaty Organization* (CTBTO) is tasked with building up a global verification system for operation under the treaty. Nuclear weapons have been detonated underground, underwater, and in different altitudes of the atmosphere. While the latter two forms are banished under the Partial Nuclear Test Ban Treaty of 1963, the CTBT would also forbid, and hence require verification against, underground tests.

3.3.1.1 The International Monitoring System

At the heart of the CTBT's verification regime lies the *International Monitoring System* (IMS), a network of 321 geophysical monitoring stations positioned around the globe. Both the sensor technologies they employ and their locations have been defined by the treaty and its annexes. Any additional on-site inspection would be limited to an area of 1000 square kilometers through the treaty, which poses implicit constraints on the localization performance of the IMS. The IMS relies on four different monitoring technologies to achieve its goal [16,38]. In full operation, around 15 gigabyte of incoming data are expected each day, mostly transmitted in real time through a global VSAT satellite network. The largest subnetwork of 170 seismic stations, including arrays for enhanced detection capability, measures seismic energy traveling through Earth. Further, 60 infrasound stations record low-frequency pressure variations in the atmosphere, and 11 hydroacoustic stations measure energy transmitted through the world's oceans. These three sensor types constitute the so-called waveform technologies. While the analysis of waveform data can indicate that a detected event might not be of natural origin, the fourth sensor network, radionuclide measurements, can provide valuable evidence for a certain event being a nuclear explosion rather than an earthquake, chemical explosion, or other source not violating the treaty.

3.3.1.2 Hydroacoustic Monitoring

The hydroacoustic network's main purpose is to monitor the oceans for underwater nuclear tests. The main signature of a test would be water pressure waves generated by the underwater explosion. Other sources of underwater sound are natural events such as iceberg calving, suboceanic earthquakes, underwater volcanoes, and marine mammals. Man-made events include intentional or accidental chemical explosions (e.g., in military exercises or dynamite fishing), seismic air-gun surveys, and marine vessels. As a secondary use case, the hydroacoustic network can also contribute to processing of signals originating from the continents. Conceptually, global hydroacoustic monitoring relies on a natural phenomenon of underwater sound propagation. The deep sound channel (DSC) or sound fixing and ranging (SOFAR) channel is a certain depth region in the ocean around a minimum in the vertical sound speed profile. While local channel depths vary from close to the ocean's surface to around 1200 m below, the DSC as a whole exhibits waveguide properties for underwater sound on a global scale. Due to this phenomenon, the relatively small number of IMS hydroacoustic stations are sufficient for global ocean coverage. The IMS hydroacoustic network consists of 11 stations. Six of these use hydrophones placed underwater at the local deep sound channel axis, and five use seismometers residing near the coastline of steep-sloped oceanic islands. Figure 3.3 shows

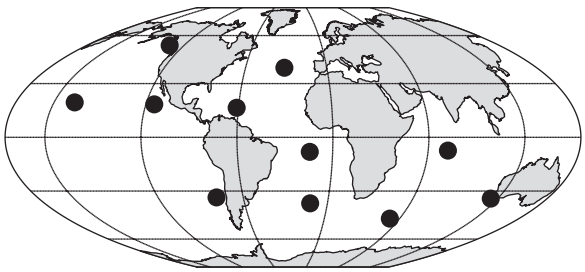

FIGURE 3.3 Locations of hydroacoustic stations defined by the treaty.

the locations of all hydroacoustic stations as defined by the treaty. Every on- and offshore station consists of several individual sensors, facing different sides of the island which hosts the communication infrastructure.

We classify signals recorded by IMS in-ocean hydrophones, learning to label them as either having explosive or nonexplosive signature. Information about sensor or source location is not taken into account, and also no assumption is made that the same signal might have been recorded by multiple sensors. As a consequence, we solve a classical classification task as if all signals had been obtained independently and identically distributed through a single sensor.

3.3.2 Preprocessing

We want to evaluate the baseline potential of SVM hydroacoustic signal classification while relying on as few as possible modifications of the existing processing pipeline. Hence, for representation, we use a set of precalculated features provided by the CTBTO's Provisional Technical Secretariat. The raw sensor data were sampled at 250 Hz and from the outset filtered into eight partially overlapping frequency bands between 1 and 100 Hz. Table 3.1 shows the filter band lower and upper frequency limits. In each band, detection and feature extraction are carried out independently, but according to the same algorithm. Within the continuous data stream, a detection algorithm for each band monitors if the ratio of a short-term average (10 s window) to long-term average (150 s window) exceeds a station-specific threshold. If detections in different frequency bands occur close enough in time, they are grouped into a signal, which is hence defined by one or more contemporaneous detections across frequency bands. For each signal a fixed set of 16 identically calculated features is extracted from every band with an associated detection. The union of all extracted features then serves as representation of the event which triggered the detections. Each event is thus represented by $\ell \cdot 16$ real numbers with $\ell \in \{1, \ldots, 8\}$. As a consequence, the feature sets of any two given signals can differ and need not even overlap. The 16 features extracted in every frequency band are listed in Table 3.2. They can be grouped into (i) time related, (ii) energy related, (iii) statistical moments, and (iv) cepstral features. The power cepstrum is a good indicator of periodicity in the signal's power spectrum, that is, the presence of harmonics. For this reason, cepstral features may be good indicators of the bubble pulses possibly accompanying underwater explosions. They are also in general considered to well separate the propagation Green's function from the source

TABLE 3.1
List of the Eight Frequency Bands Used for Feature Extraction

	Frequency Filter Bands (Hz)							
Lower limit	1	2	3	6	8	16	32	64
Upper limit	2	80	6	12	16	32	64	100

TABLE 3.2
Features Extracted from Each Band with a Detection

Temporal	Energy	Statistical	Cepstral Peak (2×)
Peak time	Peak level	Time spread	Position
Mean arrival	Total energy	Skewness	Level
Total duration	Average noise	Kurtosis	Variance
Zero crossing rate			

function. The cepstral features listed in Table 3.2 were calculated in two variants, once from a low-pass-filtered spectrum and once from a detrended spectrum.

3.3.3 Classifying Incomplete Data

The classification task described above falls into the group of missing data problems (e.g., [12]). For such it is common to consider a missingness matrix $R \in \{0, 1\}^{mN}$ of the same size as the matrix formed by all feature vectors $x_i \in S$. Each of its Boolean entries $R_{ij} \in \{0, 1\}$ indicates whether the corresponding feature x_{ij} is present in the sample x_i or not. The missingness matrix R of given training data S is then often interpreted as one specific instance or draw from an underlying *missingness distribution* p_R, just as S is assumed to be drawn from a data-generating distribution p. This presumes that the missing values actually existed and would have been observable, but were for some reason not obtained. Such missing data problems are often grouped according to three possible relations between these two generating distributions. If p_R is completely independent of p, the data is coined *missing completely at random* (MCAR). A situation where p_R depends on p, but is conditionally independent of all missing values, is termed as *missing at random* (MAR). For all other relations between p_R and p, one speaks of data *missing not at random* (MNAR). In addition, there are cases where a feature's absence is not simply due to nonobservation, but the entity to be observed was undefined or did not exist for some reason. This is referred to as *structural absence*. CTBT hydroacoustic data constitute a mixture of features being MNAR and structurally absent.

The most convenient way to deal with samples holding missing values is to eliminate them from the training data, or to discard all features which exhibit missing values. Since the missingness ratios in the hydroacoustic data are high across both features and samples, and incomplete test cases with possibly unencountered missingness pattern have to be classifiable, neither is a viable option. In general, deletion can significantly bias both classifier and results, and hence a wide range of more systematic approaches have been suggested, see, for example, Ref. [12] for a review. Among these, imputation techniques are the most common, where missing values are filled in according to some heuristic. Traditional imputation techniques include constant value imputation (e.g., imputing zero or the mean feature value), and guessing a good imputation value, possibly through regression or by using the value of another example that is similar in some sense. More elaborate techniques include, for example, multiple imputation, a Monte Carlo technique for which an imputation model first has to be specified. Then, multiple complete datasets are generated by sampling from the imputation model. Analysis is carried out on each of the imputed datasets and then combined into one final result. In maximum-likelihood-based approaches, an underlying model for the data-generating process is assumed and its parameters are estimated from the available data. Besides imputation, learning algorithms can also be modified to directly deal with incomplete input. For example, an elegant SVM variant by Chechik et al. [6] avoids imputation by altering the margin interpretation for samples with values that are structurally absent.

In practice, it is often tedious to identify among all possible approaches one that performs well on a given application problem. In addition, many of the above methods have been formulated for MAR and MCAR settings rather than MNAR or structural absence which are relevant for CTBT

hydroacoustic data. We here use a straightforward approach and impute a single value of zero for all missing values. For many features in Table 3.2, this would constitute a physically or statistically plausible continuation for the limit case of a zero-threshold detector. For example, peak and total energy as well as skewness of a nonpresent signal might be well represented by zero. For others, such as the average noise level, zero cannot be seen as a logical continuation and our choice is far from ideal. In addition to zero-imputation, we use the "flag" approach which was found to be a well-performing baseline method in Ref. [6]. For each of the eight frequency bands, a Boolean variable is added to the imputed feature set, indicating whether or not that band has been detected and its features extracted. We further extend this "flag" approach by using kernel functions which have a bipartite structure, with one subkernel operating on the Boolean missingness representation and another on the real-valued, imputed features.

3.3.3.1 Heterogeneous Kernel Functions

Let S_r denote the zero-imputed hydroacoustic training data and x_r one of its samples, a 128-dimensional real-valued vector. Then we write x_b for the corresponding missingness vector of eight Booleans indicating whether the features of each band are present or not. Thus, each sample $x = (x_b, x_r)$ is represented by a vector in a 136-dimensional joint feature space $\mathcal{X}$, which is the Cartesian product $\mathcal{X}_b \times \mathcal{X}_r$ of the space of the according Boolean and real-valued feature vectors. In order to allow the machine to incorporate information held by the Boolean and real values differently, we employ kernel functions with a *bipartite structure*, where one subkernel k_b operates on the Boolean features $x_b \in \mathcal{X}_b$ and another subkernel k_r on their real counterparts $x_r \in \mathcal{X}_r$. On $\mathcal{X}_r$ we employ a standard radial basis function (RBF) kernel $k(x_r, z_r) = e^{-\gamma\|x_r - z_r\|^2}$. For $\mathcal{X}_b$, a polynomial kernel $k(x_b, z_b) = (\langle x_b, z_b\rangle + c)^d$ is used. When combining the subkernels k_b and k_r into one joint kernel k through a function f,

$$k : (\mathcal{X}_b \times \mathcal{X}_r) \times (\mathcal{X}_b \times \mathcal{X}_r) \to \mathbb{R}, \quad k(x,z) = f(k_b(x_b, z_b), k_r(x_r, z_r)), \tag{3.14}$$

f must be of such a form that the overall kernel k remains positive definite. Recalling the kernel composition rules of Section 3.2.5, we consider two intuitive possibilities. First, a direct product kernel

$$k_p(x,z) = \left(\langle x_b, z_b\rangle + c\right)^d \cdot e^{-\gamma\|x_r - z_r\|^2}, \tag{3.15}$$

and a weighted direct sum kernel

$$k_s(x,z) = \left(\langle x_b, z_b\rangle + c\right)^d + we^{-\gamma\|x_r - z_r\|^2} \tag{3.16}$$

with weighting factor $w \in \mathbb{R}^+$. The structure of k_p and k_s stresses the similarity of features across all bands on the one hand and inherent differences between the Boolean and real-valued features on the other.

Independent of the combination of k_b and k_r, we might desire an overall kernel family that is more suitable for our hydroacoustic application task. Especially given that underwater signal propagation is frequency dependent, the overall kernel could better account for the fact that the input vector concatenates information from eight different frequency bands. Mirroring the steps above, we view $\mathcal{X}$ as the Cartesian product of the band-wise feature spaces: $\prod_{i=1}^{8}(\mathcal{X}_b^{(i)} \times \mathcal{X}_r^{(i)})$. For each band's subspace $\mathcal{X}^{(i)}$, a *band-wise bipartite direct product kernel* $k^{(i)}$ can in analogy to Equation 3.15 be defined as

$$k^{(i)}(x^{(i)}, z^{(i)}) = k_b^{(i)} \cdot k_r^{(i)} = \left(\langle x_b^{(i)}, z_b^{(i)}\rangle + c\right)^d \cdot e^{-\gamma^{(i)}\|x_r^{(i)} - z_r^{(i)}\|^2}. \tag{3.17}$$

Here, the polynomial kernel parameters c and d are chosen identical across all subkernels $k^{(i)}$. The RBF bandwidth parameters $\gamma^{(i)}$ are, however, allowed to vary from band to band since we expect different feature distributions across the frequency bands. Note that in each subkernel defined in Equation 3.17, the Boolean subkernel operates on two single Boolean values only. The corresponding scalar product can hence yield only one (if both samples have a detection in band i) or zero (if only one or none of them do). In the special case of $c = 0$, this has the effect of switching on or off the contribution of the overall subkernel: if two samples both have detections for band i, $k^{(i)}$ returns the RBF kernel evaluation between their real-valued features in band i. If only one or none hold detections in band i, $k^{(i)}$ returns zero. For this reason, and because it would again introduce more hyperparameters, we consider the direct product kernel (Equation 3.15) on the subspaces $\mathcal{X}^{(i)}$ rather than the weighted direct sum kernel (Equation 3.16). For combining all subkernels $k^{(i)}$ into one overall kernel, both addition and multiplication are feasible options. As seen in Section 3.2.5, convolution kernels allow for multiplication as well as addition of subkernels and also cover an intermediate range by varying the integer degree D as single free parameter. The convolution kernel k_D of degree D on the eight subkernels $k^{(i)}$ is

$$k_D(x,z) = \sum_{1 \le j_1 < \cdots < j_D \le 8} \prod_{d=1}^{D} k^{(j_d)}(x^{(j_d)}, z^{(j_d)}). \tag{3.18}$$

With Equations 3.15, 3.16, and 3.18, we have three candidate families of kernels for the imputed, Boolean-augmented data S on $\mathcal{X}$. All three kernels have as free parameters the real-valued polynomial offset $c \in \mathbb{R}_0^+$ and integer degree $d \in \mathrm{N}^+$. For Equations 3.15 and 3.16, the RBF kernels introduce the bandwidth $\gamma \in \mathbb{R}^+$ as a single parameter, while Equation 3.18 holds one RBF parameter $\gamma^{(i)} \in \mathbb{R}^+$ for each frequency band. An SVM using kernel (Equation 3.18) has 11 free kernel parameters plus the SVM regularization parameter C. In the following, we describe our experimental setup, including the model selection procedure to determine these SVM hyperparameters.

3.3.4 Experiments

In total, 778 expert-labeled samples were available for classifier training, validation, and testing. Of these, less than 5% had values for all 128 features, while 91% of samples held values for the most common frequency band between 6 and 12 Hz. For all experiments described below we obtained the test error as an average over five different splits into 80% training and 20% test data. Within each of these 80% of training data, we used another "inner" fivefold cross-validation procedure for SVM model selection. The best hyperparameters were used to re-train an SVM on the entire 80% before obtaining the test error on the remaining 20% of test data. For the direct sum and direct product kernel, we conducted simple grid search on the fivefold cross-validation error CV-5 to find the best values for c, d, (w), γ, and C. For the convolution kernel, grid search is far from feasible and we employed the maximum-likelihood-based approach described in Section 3.2.7.2. We optimized the real-valued parameters only and repeated this for different combinations of integer values for the kernel parameters d and D.

3.3.5 Results

We compared SVMs using the three candidate kernels (Equations 3.15, 3.16, and 3.18) on the zero-imputed and Boolean-augmented dataset S to two baseline methods which operated on the zero-imputed, but not Boolean-augmented dataset S_r only. Table 3.3 shows the results obtained. In the first column, *LDA* refers to a baseline linear discriminant analysis [17]. The SVM baseline classifier

TABLE 3.3
Average Classification Test Errors

	Classifier				
	LDA	***svm***	***svm-s***	***svm-p***	***svm-c***
Error (%)	5.2	4.9	4.9	4.8	4.3

svm used one single RBF kernel and was optimized by grid search as well. The three entries *svm-s*, *svm-p*, and *svm-c* all operated on the zero-imputed and Boolean-augmented training data S, and correspond to the candidate SVMs with a direct sum kernel (Equation 3.16), direct product kernel (Equation 3.15), and convolution kernel of degree one (Equation 3.18), respectively. In summary, all SVMs performed better than the linear approach, and the two bipartite kernels from Equations 3.15 and 3.16 were on par with the baseline RBF kernel not having access to the Boolean-encoded missingness pattern. Additionally, passing the Boolean indicators to the baseline *svm* did not influence its performance. The convolution kernel of degree one, which corresponds to summing up all bandwise subkernels, performed best among all approaches. With increasing degree however, error rates tended to increase as well. At the highest value of $D = 8$, which corresponds to multiplying all bandwise subkernels, the test error with 5.9% was higher than that of LDA.

It should be noted that SVMs allow to control the trade-off between sensitivity and specificity, or false-positive and false-negative rates, by penalizing positive and negative misclassification differently through two different values for the regularization parameter C. Table 3.3 should hence be seen as ranking the different approaches at some generic operation point rather than providing actual error rates for the practical application. It might be desirable to operate such a classifier at high sensitivity at the cost of more false alarms having to be rejected during human analyst review.

3.4 SUMMARY

We introduced SVMs as one specific approach to solving supervised classification tasks. Motivated by the concept of regularized risk minimization, we iteratively refined the SVM optimization problem from linear large-margin classification to the canonical kernelized case. Emphasizing several properties of SVMs relevant to practitioners, we in particular discussed the model selection problem and two approaches solve it: simple grid search and one gradient-based method for hyperparameter selection. In the second part of the chapter, we described an exemplary application task of classifying hydroacoustic signals recorded by the sensor network for verification of the CTBT. We combined information from different frequency bands via task-specific kernel functions also incorporating information about a sample's missingness pattern. This custom classifier, in combination with parameter optimization through a maximum-likelihood approach to model selection, showed improved performance over baseline linear methods as well as SVMs using standard kernels.

REFERENCES

1. Aronszajn, N., Theory of reproducing kernels. *Transactions of the American Mathematical Society* 68(3), 337–404, 1950.
2. Berg, C., Christensen, J., Ressel, P., *Harmonic Analysis on Semigroups: Theory of Positive Definite and Related Functions*. Berlin, Heidelberg: Springer, 1984.
3. Bordes, A., Ertekin, S., Weston, J., Bottou, L., Fast kernel classifiers with online and active learning. *Journal of Machine Learning Research* 6, 1579–1619, 2005.
4. Bousquet, O., Boucheron, S., Lugosi, G., Introduction to statistical learning theory. In: Bousquet, O., von Luxburg, U., Rätsch, G. (eds.) *Advanced Lectures in Machine Learning, LNCS*, vol. 3176, pp. 169–207. Berlin, Heidelberg: Springer, 2004.

5. Chapelle, O., Vapnik, V., Bousquet, O., Mukherjee, S., Choosing multiple parameters for support vector machines. *Machine Learning* 46(1), 131–159, 2002.
6. Chechik, G., Heitz, G., Elidan, G., Abbeel, P., Koller, D., Max-margin classification of data with absent features. *Journal of Machine Learning Research* 9, 1–21, 2008.
7. Cortes, C., Vapnik, V., Support-vector networks. *Machine Learning* 20(3), 273–297, 1995.
8. Crammer, K., Singer, Y., On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research* 2, 265–292, 2002.
9. De Vito, E., Rosasco, L., Caponnetto, A., Piana, M., Verri, A., Some properties of regularized kernel methods. *Journal of Machine Learning Research* 5, 1363–1390, 2004.
10. Fan, R.E., Chang, K.W., Hsieh, C.J., Wang, X.R., Lin, C.J., LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research* 9, 1871–1874, 2008.
11. Friedrichs, F., Igel, C., Evolutionary tuning of multiple SVM parameters. *Neurocomputing* 64(C), 107–117, 2005.
12. Garcia-Laencina, P., Sancho-Gomez, J., Figueiras-Vidal, A., Pattern classification with missing data: A review. *Neural Computing and Applications* 19(2), 263–282, 2010.
13. Gärtner, T., *Kernels for Structured Data*. Toh Tuck Link, Singapore; Hackensack, NJ; London: World Scientific Publishing, 2009.
14. Glasmachers, T., Igel, C., Gradient-based adaptation of general Gaussian kernels. *Neural Computation* 17(10), 2099–2105, 2005.
15. Glasmachers, T., Igel, C., Maximum likelihood model selection for 1-norm soft margin SVMs with multiple parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32, 1522–1528, 2010.
16. Hafemeister, D., Progress in CTBT monitoring since its 1999 senate defeat. *Science and Global Security* 15, 151–183, 2007.
17. Hastie, T., Tibshirani, R., Friedman, J.H., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer, 2003.
18. Haussler, D., Convolution kernels on discrete structures. Technical Report, UCS-CRL-99–10, University of California at Santa Cruz., 1999.
19. Igel, C., Evolutionary kernel learning. In: Sammut, C., Webb, G.I. (eds.) *Encyclopedia of Machine Learning*. Berlin: Springer, 2010.
20. Igel, C., Glasmachers, T., Heidrich-Meisner, V., Shark. *Journal of Machine Learning Research* 9, 993–996, 2008.
21. Joachims, T., Making large-scale SVM learning practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) *Advances in Kernel Methods—Support Vector Learning*, pp. 169–184. Cambridge, MA: MIT Press, 1998.
22. Keerthi, S.S., Sindhwani, V., Chapelle, O., An efficient method for gradient-based adaptation of hyperparameters in SVM models. In: Schölkopf, B., Platt, J.C., Hoffman, T. (eds.) *Advances in Neural Information Processing Systems 19 (NIPS)*, pp. 673–680. Cambridge, MA: MIT Press, 2006.
23. Lee, Y., Lin, Y., Wahba, G., Multicategory support vector machines: Theory and application to the classification of microarray data and satellite radiance data. *Journal of the American Statistical Association* 99(465), 67–82, 2004.
24. Maji, S., Berg, A.C., Malik, J., Classification using intersection kernel support vector machines is efficient. *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2008.
25. Platt, J., Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C., Smola, A. (eds.) *Advances in Kernel Methods—Support Vector Learning*, pp. 185–208. Cambridge, MA: MIT Press, 1998.
26. Platt, J., Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Smola, A.J., Bartlett, P. L., Schölkopf, B., Schuurmans, D. (eds.) *Advances in Large Margin Classifiers*, pp. 61–74. Cambridge, MA: MIT Press, 2000.
27. Romdhani, S., Torr, P., Schölkopf, B., Blake, A., Efficient face detection by a cascaded support vector machine expansion. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 460(2051), 3283–3297, 2004.
28. Saunders, C., Gammerman, A., Vovk, V., Ridge regression learning algorithm in dual variables. In: *Proceedings of the Fifteenth International Conference on Machine Learning*, pp. 515–521. San Francisco, CA: Morgan Kaufmann, 1998.
29. Schölkopf, B., Smola, A.J., *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.
30. Schölkopf, B., Tsuda, K., Vert, J., *Kernel Methods in Computational Biology*. Cambridge, MA: MIT Press, 2004.

31. Schölkopf, B., Herbrich, R., Smola, A., A generalized representer theorem. In: Helmbold, D., Williamson, B. (eds.) *Computational Learning Theory, LNCS*, vol. 2111, pp. 416–426. Berlin, Heidelberg: Springer, 2001.
32. Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Müller, K.R., Rätsch, G., Smola, A., Input space versus feature space in kernel-based methods. *IEEE Transactions on Neural Networks* 10, 1000–1017, 1999.
33. Shalev-Shwartz, S., Singer, Y., Srebro, N., Cotter, A., Pegasos: Primal estimated sub-gradient solver for SVM. *Mathematical Programming* 127, 3–30, 2011.
34. Shawe-Taylor, J., Cristianini, N., *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press, 2004.
35. Steinwart, I., Support vector machines are universally consistent. *Journal of Complexity* 18, 768–791, 2002.
36. Steinwart, I., Sparseness of support vector machines—Some asymptotically sharp bounds. In: Thrun, S., Saul, L., Schölkopf, B. (eds.) *Advances in Neural Information Processing Systems* 16. pp. 1069–1076. Cambridge, MA: MIT Press, 2004.
37. Suttorp, T., Igel, C., Resilient simplification of kernel classifiers. In: Marques de Sá , J., Alexandre, L.A., Duch, W., Mandic, D (eds.) *Proceedings of the 17th International Conference on Artificial Neural Networks (ICANN). LNCS*, vol. 4668, pp. 139–148. Berlin, Heidelberg: Springer, 2007.
38. Thunborg, A. (ed.), *Science for Security. Verifying the Comprehensive Nuclear-Test-Ban Treaty*. Preparatory Commission for the CTBTO, Vienna, Austria, 2009.
39. Vapnik, V., *Statistical Learning Theory*. New York: Wiley, 1998.
40. Weston, J., Watkins, C., Support vector machines for multi-class pattern recognition. In: *Proceedings of the 7th European Symposium in Artificial Neural Networks (ESANN)*, pp. 219–224, 1999.