outliers.

Combinatorial Problem

Given $\nu \in (0, 1]$, the resulting algorithm computes (8.6) subject to (8.7), and thereby constructs a region R such that for $OL = \{i : x_i \notin R\}$, we have $\frac{|OL|}{m} \leq \nu$. The "$\leq$" is *sharp* in the sense that if we multiply the solution $\mathbf{w}$ by $(1 - \epsilon)$ (with $\epsilon > 0$), it becomes a "$>$." The algorithm does *not* solve the following *combinatorial* problem, however: given $\nu \in (0, 1]$, compute

$$\underset{\mathbf{w}\in\mathcal{H}, OL\subset[m]}{\text{minimize}} \quad \frac{1}{2}\|\mathbf{w}\|^2,$$

$$\text{subject to} \quad \langle \mathbf{w}, \Phi(x_i)\rangle \geq 1 \text{ for } i \in [m] \setminus OL \text{ and } \frac{|OL|}{m} = \nu. \tag{8.38}$$

Ben-David et al. [31] analyze a problem related to (8.38): they consider a sphere (which for some feature spaces is equivalent to a half-space, as shown in Section 8.3), fix its radius, and attempt to find its center such that it encloses as many points as possible. They prove that it is already NP hard to approximate the maximal number to within a factor smaller than 3/418.

Kernel-Based Vector Quantization

We conclude this section by mentioning another kernel-based algorithm that has recently been proposed for the use on unlabelled data [541]. This algorithm applies to vector quantization, a standard process which finds a codebook such that the training set can be approximated by elements of the codebook with small error. Vector quantization is briefly described in Example 17.2 below; for further detail, see [195].

Given some metric d, the kernel-based approach of [541] uses a kernel that indicates whether two points lie within a distance $R \geq 0$ of each other,

$$k(x, x') = I_{\{(x,x')\in\mathcal{X}\times\mathcal{X}:d(x,x')\leq R\}}. \tag{8.39}$$

Let Φ_m be the empirical kernel map (2.56) with respect to the training set. The main idea is that if we can find a vector $\boldsymbol{\alpha} \in \mathbb{R}^m$ such that

$$\boldsymbol{\alpha}^\top \Phi_m(x_i) > 0 \tag{8.40}$$

holds true for all $i = 1, \ldots, m$, then each point x_i lies within a distance R of some point x_j which has a positive weight $\alpha_j > 0$. To see this, note that otherwise all nonzero components of $\boldsymbol{\alpha}$ would get multiplied by components of Φ_m which are 0, and the dot product in (8.40) would equal 0.

To perform vector quantization, we can thus use optimization techniques, which produce a vector $\boldsymbol{\alpha}$ that satisfies (8.40) while being sparse. As in Section 7.7, this can be done using linear programming techniques. Once optimization is complete, the nonzero entries of $\boldsymbol{\alpha}$ indicate the codebook vectors.

8.7 Experiments

We apply the method to artificial and real-world data. Figure 8.6 shows a comparison with a Parzen windows estimator on a 2-D problem, along with a family of