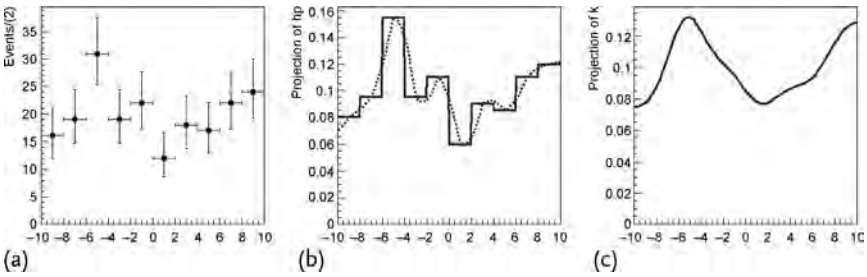

Figure 5.13 Example of the KEYS adaptive kernel estimation. (a) Input data; (b) histogram-based estimate with second order interpolation; (c) KEYS adaptive kernel estimate. (Reprinted with permission from Verkerke and Kirkby (2006), copyright 2006, W. Verkerke and D. Kirkby).

The iteration on the fixed kernel estimator nearly removes the dependence on our initial choice of w. The boundaries pose some complication in carrying this out.

There are packages for adaptive kernel estimation, for example, the KEYS ("Kernel Estimating Your Shapes") package (Cranmer, 2001). Figure 5.13 illustrates the use of this package.

5.10 Naive Bayes Classification

We anticipate the classification chapters of this book with the introduction of a simple yet often very satisfactory classification algorithm, the *naive Bayes* classifier. The problem is to correctly classify an observation $\boldsymbol{x}$ into one of K classes, C_k, $k = 1, \ldots, K$. We must learn how to do this using a dataset of N observations $\mathbf{X} = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ with known classes. We assume that observation $\boldsymbol{x}$ is sampled from the same distribution as the "training set" $\mathbf{X}$. For example, the training set may be produced by a simulation where the true classes are known.

The idea of the naive Bayes algorithm is to use Bayes' theorem to form an estimate for the probability that the $\boldsymbol{x}$ belongs to class k. With such probability estimates for each class, the class with the highest probability is chosen. To apply Bayes' theorem, we need two ingredients: the marginal (prior) probability to be in class k, $P(C_k)$; and the probability to sample observation $\boldsymbol{x}$ given class k, $P(\boldsymbol{x}|C_k)$. That is,

$$P(C_k|\boldsymbol{x}) \propto P(\boldsymbol{x}|C_k)P(C_k)\,. \tag{5.34}$$

The normalization isn't needed as it is the same for all classes.

The training set is used to estimate $P(\boldsymbol{x}|C_k)$, by evaluating the local density for class C_k in the neighborhood of $\boldsymbol{x}$. In general, this may be very difficult to do precisely if there are many dimensions. So a great simplification is made in the naive

Table 5.1 Confusion matrices corresponding to Figure 5.14.

		Predicted class			
		(a)		(b)	
		Plus	**Dot**	**Plus**	**Dot**
Actual class	**Plus**	215	185	143	257
	Dot	40	1960	23	977

Bayes algorithm: Assume that the probability $P(\boldsymbol{x}|C_k)$ factorizes, for each class like

$$P(\boldsymbol{x}|C_k) = \prod_{d=1}^{D} P(x_d|C_k)\,. \tag{5.35}$$

That is, we assume that the variables in $\boldsymbol{x}$ are statistically independent when conditioned on class. This makes the density estimation problem much easier, since we have only to estimate D one-dimensional densities (for each class) instead of a D-dimensional density. Typically, this is implemented with D one-dimensional kernel density estimators.

In spite of the simplicity, the naive Bayes algorithm often performs rather well, even in some cases when the assumption of class-conditioned independence is not correct. Examples of naive Bayes classification are illustrated in Figure 5.14. Figure 5.14a shows separation for a class with a bimodal distribution from a class with a unimodal distribution, but for which the assumption of independence is strictly correct. Figure 5.14b shows an example of the trained class boundary for a dataset in which the assumption of independence within a class is not correct. The algorithm is clearly having trouble in this case. Other values for the smoothing parameter do not improve the performance in this example. The separation boundary of a more general classifier, a neural net, on this dataset may be seen for comparison in Figure 12.3.

Quantitative measures of performance of a classifier are developed in coming chapters. However, a simple picture is given by the *confusion matrices*, given in Table 5.1. These are computed by applying the classifiers as trained by the data in the two sides of Figure 5.14 to new datasets generated with the same distributions. We could also compute the matrices directly on the data used for training, but the results would tend to be biased towards overstating the performance.

5.11 Multivariate Kernel Estimation

Besides the curse of dimensionality, the multidimensional case introduces the complication of covariance. When using a product kernel, the local estimator has

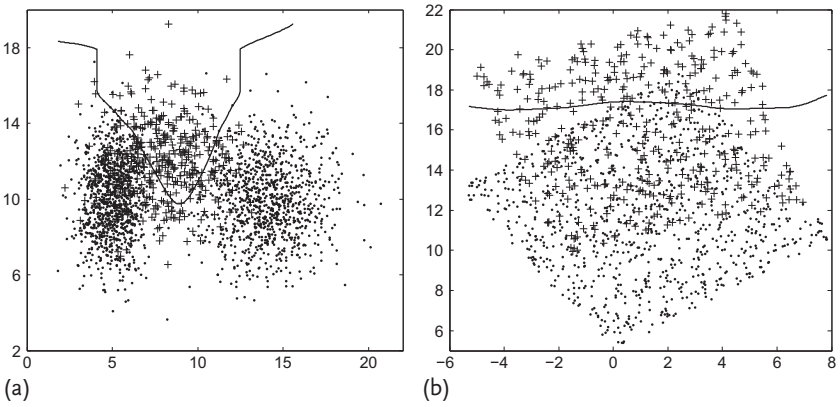

Figure 5.14 Naive Bayes classifier examples in two feature dimensions and two classes. The two classes are indicated with dots and pluses. The solid line shows the prediction boundary between the two classes as determined by the naive Bayes algorithm using the dataset in each figure. Training is performed with the MATLAB `NaiveBayes.fit` method, using default Gaussian kernel smoothing for the density estimation.

diagonal covariance matrix. In principle, we could apply a local linear transformation of the data to a coordinate system with diagonal covariance matrices. This amounts to a nonlinear transformation of the data in a global sense, and may not be straightforward. However, we can at least work in the system for which the overall covariance matrix of the data is diagonal.

If $\{\boldsymbol{\gamma}_n\}_{n=1}^N$ is the suitably diagonalized data, the product fixed kernel estimator in D dimensions is

$$\hat{p}_0(\boldsymbol{\gamma}) = \frac{1}{N}\sum_{n=1}^{N}\left[\prod_{d=1}^{D}\frac{1}{w_d}K\left(\frac{\gamma^{(d)}-\gamma_n^{(d)}}{w_d}\right)\right], \tag{5.36}$$

where $\gamma^{(d)}$ denotes the dth component of the vector $\boldsymbol{\gamma}$. The asymptotic, normal MISE-optimized smoothing parameters are now

$$w_d = \left(\frac{4}{D+2}\right)^{1/(D+4)}\sigma_d N^{-1/(D+4)}. \tag{5.37}$$

The corresponding adaptive kernel estimator follows the discussion as for the univariate case. An issue in the scaling for the adaptive bandwidth arises when the multivariate data is approximately sampled from a lower dimensionality than the dimension D.

Figure 5.15 shows an example in which the sampling distribution has diagonal covariance matrix (locally and globally).

Applying kernel estimation to this distribution yields the results in Figure 5.16, shown for two different smoothing parameters.

For comparison, Figure 5.17 shows an example in which the sampling distribution has nondiagonal covariance matrix. Applying the same kernel estimation to