Nonparametric Classification

Prof. Richard Zanibbi

What to do when feature distributions (likelihoods) are not ‘normal’

Don’t Panic!

While they may be suboptimal, LDC and QDC may still be applied, even though the assumption of normality is violated. Often, this can yield usable results.

An Alternative: Estimate feature *density* locally.

Rather than assume a parametric form for the distribution, estimate distributions using local estimates of the *density* around each training sample.

Density Estimation: $\hat{p}(\mathbf{x}|\omega_i)$

Density Estimation:

Defining a probability density function (pdf) using the neighboring samples around each training sample.

Probability **x** is in feature space region *R*:

$$p^* = P(\mathbf{x} \in R) = \int_R p(\mathbf{u}) d\mathbf{u}$$

Probability *k* of *N* samples from the unknown distribution lie in *R (binomial distribution):*

$$p_k = \binom{N}{k} (p^*)^k (1 - p^*)^{N-k} \quad \text{where} \quad p^* \approx \frac{k}{N}$$

R·I·T

If we assume **x** lies in R and that p(**u**) is constant in region *R*:

$$p^* \approx p(\mathbf{x}) \int_R d\mathbf{u} = p(\mathbf{x}) V_R$$

where V_R is the volume of region *R* in R^n.

Using the previous two equations, we obtain:

$$p(\mathbf{x}) \approx \frac{k}{N V_R}$$

As N tends to infinity and the region volume shrinks to a point (volume approaches 0), this produces the **exact value for p(x).**

Multinomial (Histogram) Methods

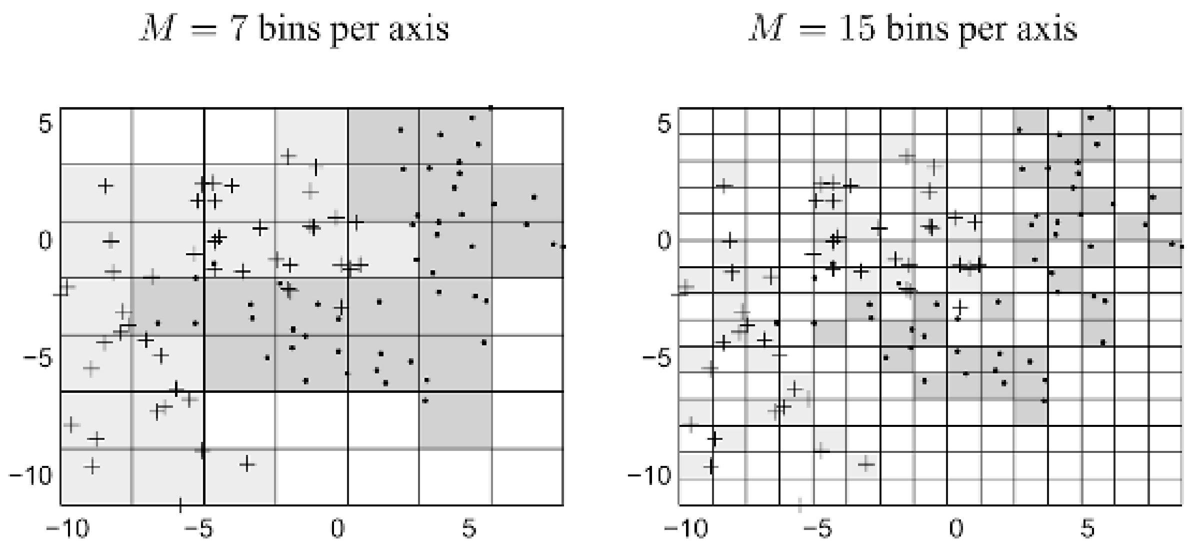


Fig. 2.2 *Classification regions found by a histogram classifier on the banana data. Plotted also is the training set.*

Divide feature space into equal bins/hypercubes (fix N,V_R)

hist(**x**): Assign class with most training samples in associated feature bin (region)

Estimated Priors:

$$\hat{P}(\omega_i) = \frac{N_i}{N}$$

Estimated Posteriors (approx. of Bayesian Classifier):

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \approx \frac{\frac{m_i}{N_i V_B}\frac{N_i}{N}}{\frac{m}{N V_B}}$$, therefore: $$P(\omega_i|\mathbf{x}) \approx \frac{m_i}{m}$$

where *m* is the number of points in a bin.

Notes on Histogram Classifier

Curse of Dimensionality

Total number of bins grows exponentially with feature space dimensions (M bins per dimension: M^n)

- Number of samples needed to prevent empty or sparse bins that lead to poor estimates of the posterior probabilities

Sensitivity to Bin Size

More bins: noisier estimate. Fewer bins: coarser estimate.

R·I·T

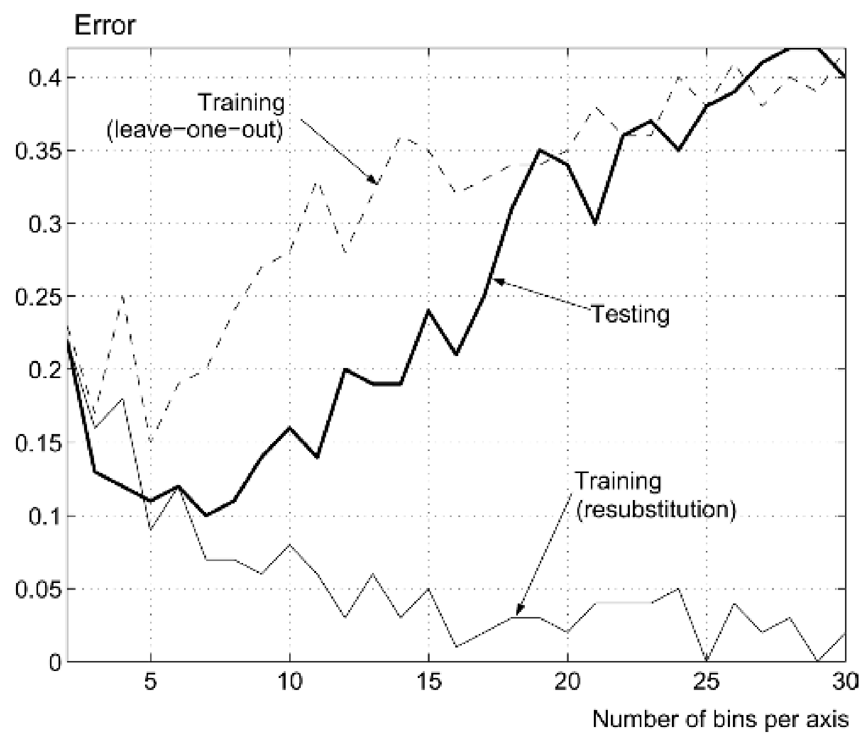


Fig. 2.3 *Resubstitution, leave-one-out and testing error rates of the histogram classifier versus the number of bins per axis.*

*Ties (white regions) broken randomly.

Evaluated using:

Training data (resub.)

Leave-one-out

Separate Test Set

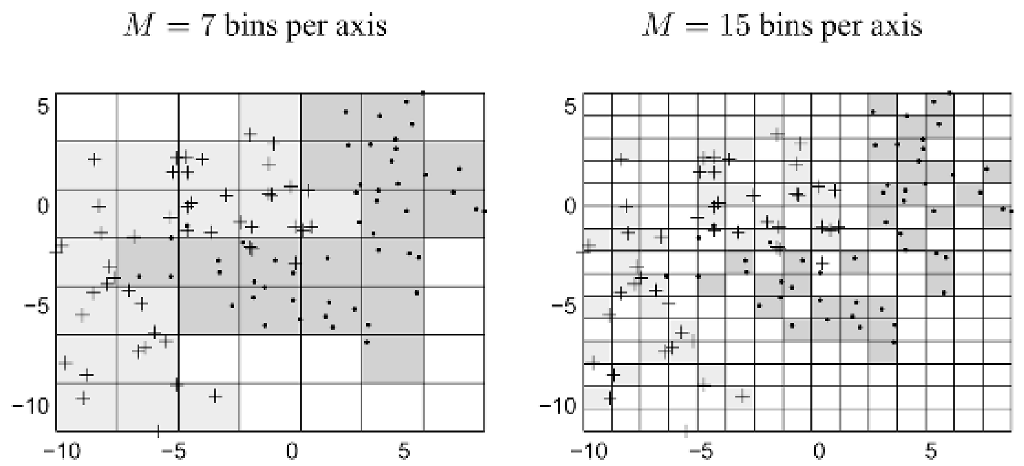


Fig. 2.2 *Classification regions found by a histogram classifier on the banana data. Plotted also is the training set.*

TABLE 2.1 Error Rates (in %) for Two Multinomial Classifiers for the Banana Data.

M	Resubstitution	Leave-one-out	Testing
7	7	20	10
15	5	35	26

Parzen Windows

Estimate density using a *kernel function* situated at each training sample.

Kernel function = ‘Parzen Window’

- Must peak at the origin, be nonnegative, have integral one over R^n (continuous, real-valued feature space)

Simplest kernel function: hyperbox with volume one

$$K(\mathbf{t}) = \begin{cases} 1, & \text{if } |t_i| \leq \frac{1}{2}, \forall i = 1, \ldots, n \\ 0, & \text{otherwise} \end{cases}$$

Parzen Windows Cont'd

Density estimate using kernel function K (k is number of windows containing **x**):

$$p(\mathbf{x}) \approx \frac{k}{N} = \frac{1}{N}\sum_{j=1}^{N} K(\mathbf{Z}_j - \mathbf{x})$$

Introduce *smoothing parameter* h to kernel fn, to avoid sparse density estimates:

$$\int_{\Re^n} \frac{1}{h^n} K\left(\frac{\mathbf{x}-\mathbf{z}_j}{h}\right) d\mathbf{x} = 1$$

Common kernel: Multidimensional Gaussian:

- S is covariance matrix (shape of kernel)

$$\frac{1}{h^n}\, K_G\left(\frac{\mathbf{x}-\mathbf{z}_k}{h}\right) = \frac{1}{h^n\,(2\pi)^{n/2}\sqrt{|S|}} \exp\left[-\frac{1}{2h^2}(\mathbf{x}-\mathbf{z}_k)^T S^{-1}(\mathbf{x}-\mathbf{z}_k)\right]$$

R·I·T

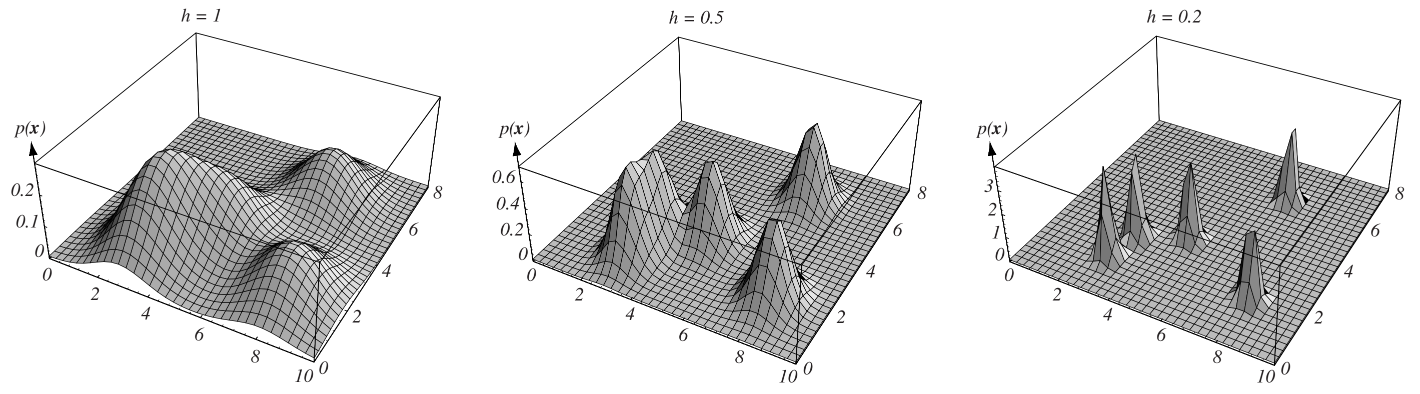


FIGURE 4.4. Three Parzen-window density estimates based on the same set of five samples, using the window functions in Fig. 4.3. As before, the vertical axes have been scaled to show the structure of each distribution. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

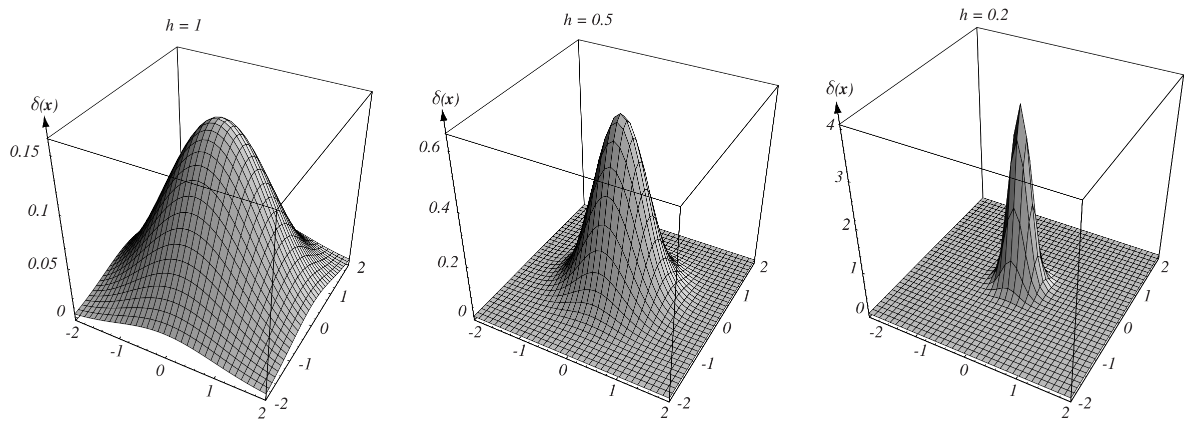


FIGURE 4.3. Examples of two-dimensional circularly symmetric normal Parzen windows for three different values of h. Note that because the $\delta(\mathbf{x})$ are normalized, different vertical scales must be used to show their structure. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

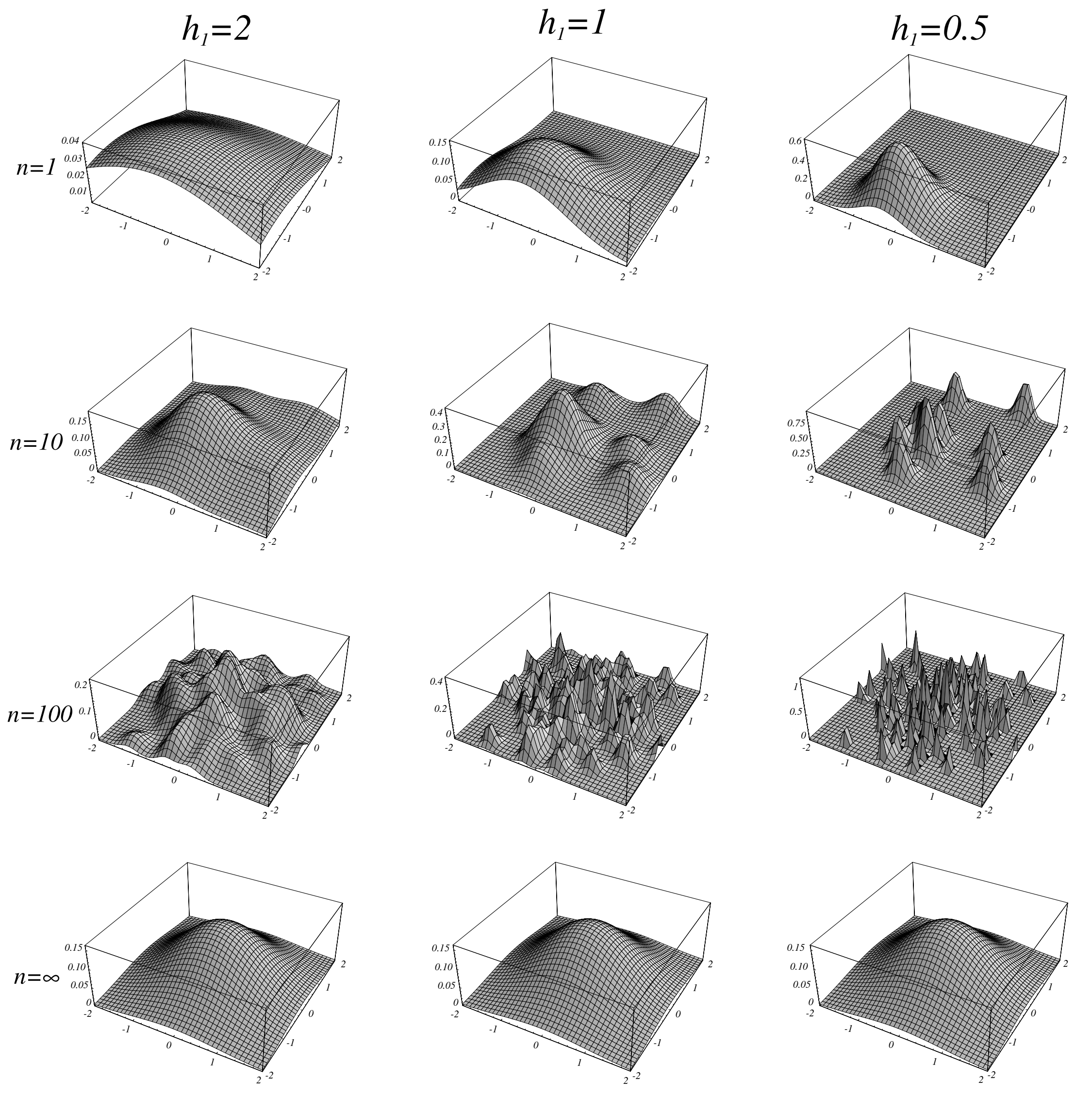


FIGURE 4.6. Parzen-window estimates of a bivariate normal density using different window widths and numbers of samples. The vertical axes have been scaled to best show the structure in each graph. Note particularly that the $n = \infty$ estimates are the same (and match the true distribution), regardless of window width. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Parzen Windows, Cont'd

Class-conditional density estimates (left); restriction on *h* for unbiased asymptotic estimate (*h* fn of class i train samples, right):

$$\hat{p}(\mathbf{x}|\omega_i) = \frac{1}{N_i} \sum_{l(\mathbf{z}_j)=\omega_i} \frac{1}{h^n} K\Big(\frac{\mathbf{x}-\mathbf{z}_j}{h}\Big) \qquad \lim_{N_i \to \infty} h(N_i) = 0$$

Estimates for posterior probability (using Ni/N for prior probability estimates):

$$\hat{P}(\omega_i|\mathbf{x}) = \frac{1}{Np(\mathbf{x})} \sum_{l(\mathbf{z}_j)=\omega_i} \frac{1}{h^n} K\Big(\frac{\mathbf{x}-\mathbf{z}_j}{h}\Big)$$

$$= C(\mathbf{x}, h, N) \sum K\Big(\frac{\mathbf{x}-\mathbf{z}_j}{h}\Big)$$

Asymptotically under the above assumption, we obtain a Bayesian Classifier.

R·I·T

Parzen Classifier

Discriminant functions:

$$g_i(\mathbf{x}) = \sum_{l(\mathbf{z}_j)=\omega_i} K\left(\frac{\mathbf{x}-\mathbf{z}_j}{h}\right)$$

Gaussian Kernel:

$$g_i(\mathbf{x}) = \sum_{l(\mathbf{z}_j)=\omega_i} \exp\left\{-\frac{1}{2h^2}[d_M(\mathbf{x},\mathbf{z}_j)]^2\right\}$$

$$[d_M(\mathbf{x},\mathbf{z}_j)]^2 = (\mathbf{x}-\mathbf{z}_j)^T S^{-1}(\mathbf{x}-\mathbf{z}_j)$$

(Squared Mahalanobis Distance)

Properties:

- With unbiased estimators (appropriate *h* values), produces an optimal Bayesian Classifier; in practice h varied over an interval, one producing smallest error chosen
- Density estimates require *every* training sample for *every* classifier input.
- Parallel implementation: Probabilistic Neural Nets (see DHS, Ch. 4)

R·I·T

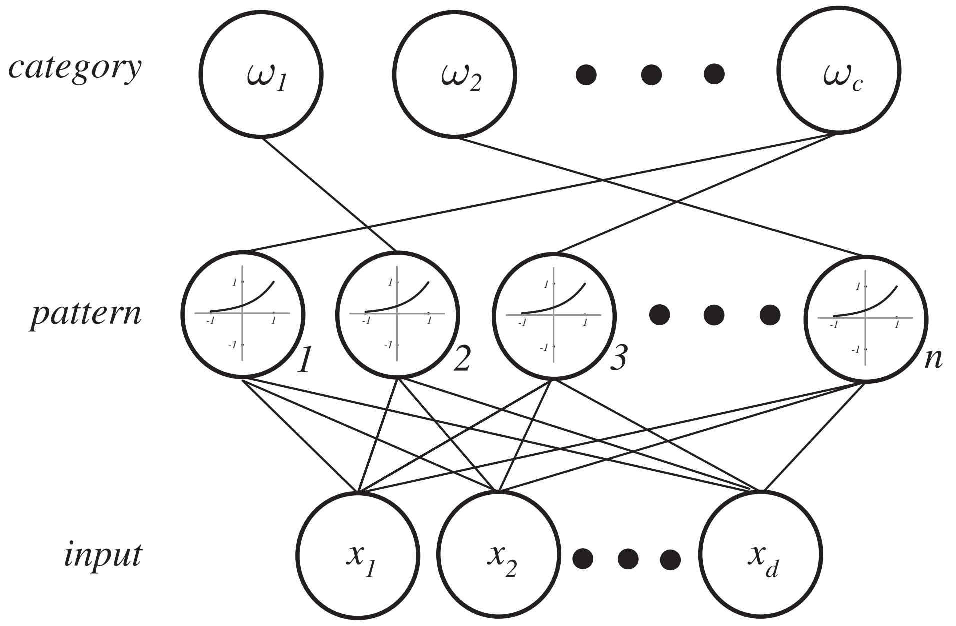


FIGURE 4.9. A probabilistic neural network (PNN) consists of d input units, n pattern units, and c category units. Each pattern unit forms the inner product of its weight vector and the normalized pattern vector $\mathbf{x}$ to form $z = \mathbf{w}^t\mathbf{x}$, and then it emits $\exp[(z-1)/\sigma^2]$. Each category unit sums such contributions from the pattern unit connected to it. This ensures that the activity in each of the category units represents the Parzen-window density estimate using a circularly symmetric Gaussian window of covariance $\sigma^2\mathbf{I}$, where $\mathbf{I}$ is the $d \times d$ identity matrix. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

k-Nearest Neighbor

Recall our general formula for nonparametric density estimation:

$$p(\mathbf{x}) \approx \frac{k}{NV_R}$$

Multinomial and Parzen Density Estimates:

Fix N and V_R, estimate k

k-Nearest Neighbor Density Estimation:

Fix *k* and N, allow V_R to vary.

R·I·T

k-NN Density Est./Classification

Nearest-Neighbor Posterior Estimation:

$$P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{p(\mathbf{x})} \approx \frac{\frac{k_i}{N_i V_R}\frac{N_i}{N}}{\frac{k}{N V_R}}$$

therefore, $P(\omega_i|\mathbf{x}) \approx \frac{k_i}{k}$

(also discriminant for k-NN classifier)

Notes

- Regions and region volume vary by training sample and training set.
- Region volume does not affect classifier output.
- k-NN classifier is Bayes-optimal when N approaches infinity, and region volume approaches 0 (equivalently, when k approaches infinity and k/N approaches 0)
- For 1-NN (single nearest neighbor), as N approaches infinity the error rate is bounded above by twice the Bayes error rate.

Density Estimation Using k-NN

Note: region volume used in estimation

$$p(\mathbf{x}) \approx \frac{k}{NV_R}$$

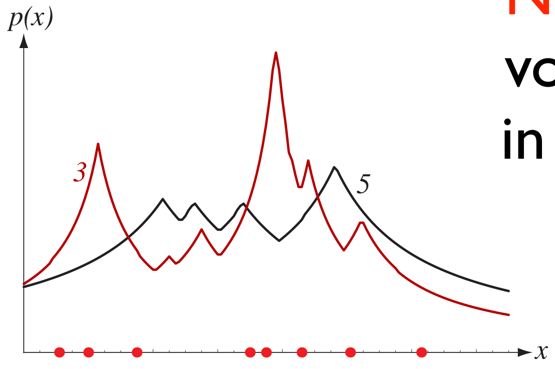


FIGURE 4.10. Eight points in one dimension and the *k*-nearest-neighbor density estimates, for $k = 3$ and 5. Note especially that the discontinuities in the slopes in the estimates generally lie *away* from the positions of the prototype points. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

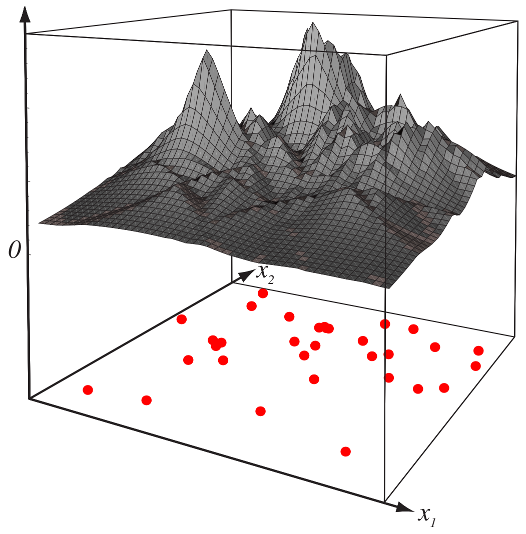
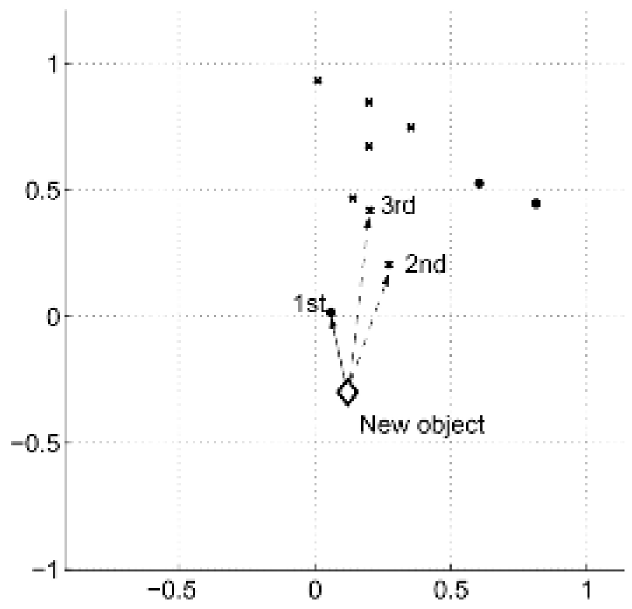


FIGURE 4.11. The *k*-nearest-neighbor estimate of a two-dimensional density for $k = 5$. Notice how such a finite *n* estimate can be quite "jagged," and notice that discontinuities in the slopes generally occur along lines away from the positions of the points themselves. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

k-NN Classification



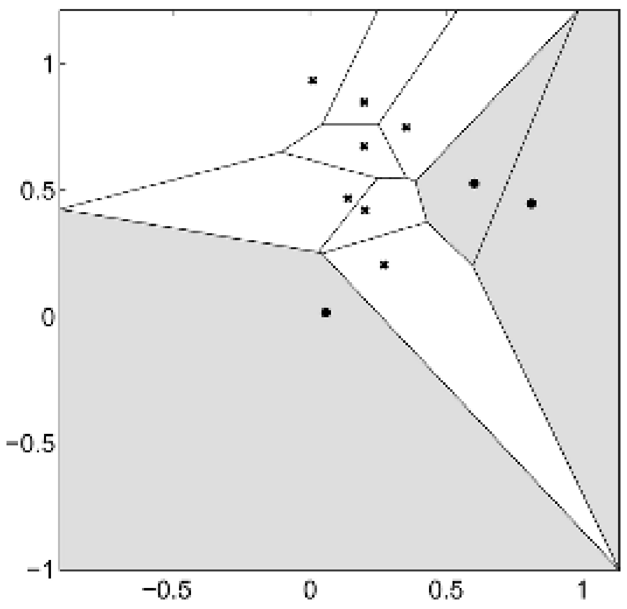


Fig. 2.4 *Illustration of the 1-nn and 3-nn classification rules and Voronoi diagrams.*

1-nn: a dot is closest, assign 'dot'
3-nn: dot and two x's are neighbors, assign 'x'

Voronoi diagram illustrates regions (Voronoi bins) closest to each training sample:

$$V(\mathbf{z}_j) = \left\{ \mathbf{x} \middle| \mathbf{x} \in \Re^n,\ d(\mathbf{z}_j, \mathbf{x}) = \min_{\mathbf{z}_k \in \mathbf{Z}} d(\mathbf{z}_k, \mathbf{x}) \right\}$$

More Voronoi Diagrams

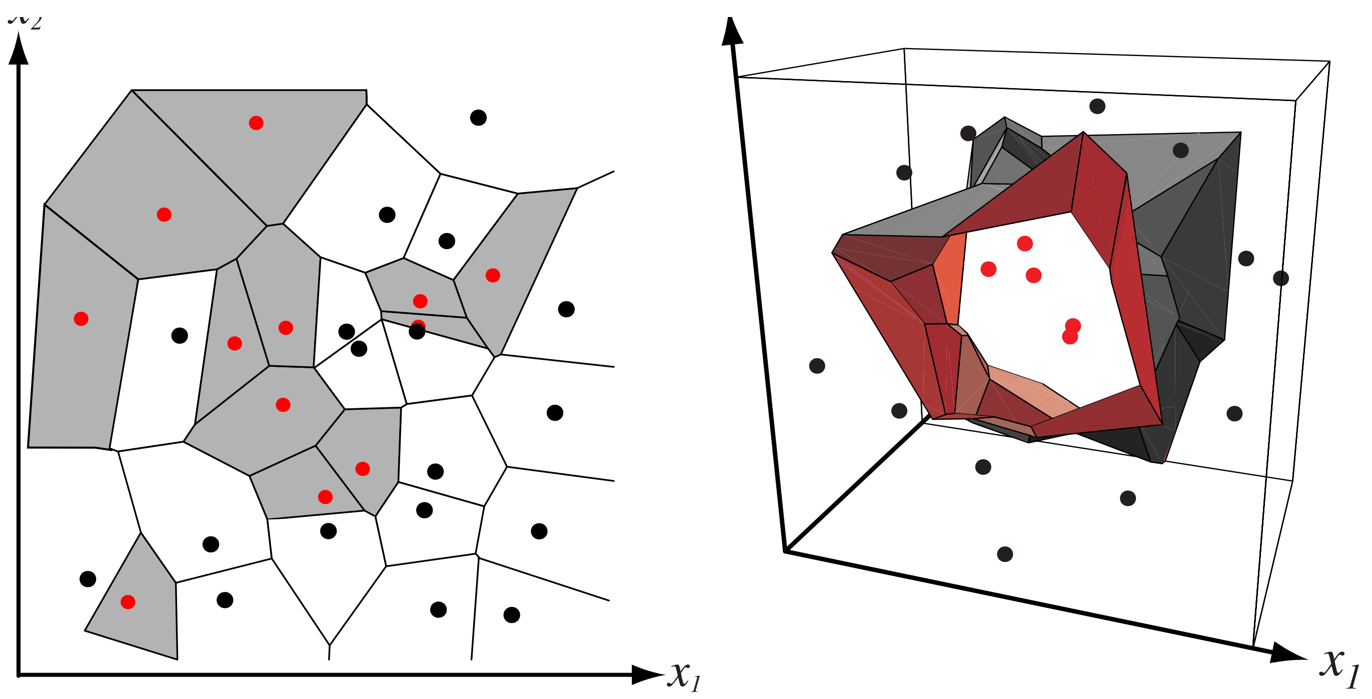


FIGURE 4.13. In two dimensions, the nearest-neighbor algorithm leads to a partitioning of the input space into Voronoi cells, each labeled by the category of the training point it contains. In three dimensions, the cells are three-dimensional, and the decision boundary resembles the surface of a crystal. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

More Notes on k-NN

Benefits

- Simple, elegant, nice theoretical properties.
- Unlike multinomial (histogram) classifier, entire feature space mapped to a class (see Voronoi diagrams)

Drawbacks

- Expensive: in general, must compare all instances with every input (*pre-structuring sometime used),* unless all comparisons parallelized (see DHS)
- Intolerant to noisy or redundant features
- Sensitive to distance metric
- No natural distance measure for nominal features
- Cannot handle missing feature values
- Little information about structure of data (local method)

R·I·T

Modifying k-NN Classifiers

Through:

1. Using a different value of k
2. Changing the distance metric (in R^n)
    - Note: asymptotic properties hold, provided a valid distance metric is used. Four required properties for a metric:
        - Nonnegative, reflexive: $d(a,b) = 0$ iff $a = b$, symmetric ($d(a,b) = d(b,a)$), triangle inequality: $d(a,b) + d(b,c) \geq d(a,c)$
        - Examples: Euclidean distance (L_2), Manhattan Distance (L_1)
3. Editing the training set Z (Prototype editing)

Prototype Editing and Extraction

Ideally

No objects have the same feature vector but a different class

- In this case, resubstitution error for training set Z is 0.

Motivations for Smaller Training Sets

Reduces both space and time requirements for classification.

Modifying the Training Set

Goal: define the smallest possible reference set V from Z to produce the highest possible 1-NN (k-NN) accuracy). Approaches:

- Prototype selection/editing (V as subset of Z, all available data)
- Prototype extraction (generate new samples from Z for V)

Prototype Editing

Strategies

- *Condensing:* Find smallest subset of Z (V) that produces 0 resubstitution error for Z. Tends to retain (many) points near the decision boundary.
- *Error-Based Editing:* Find subset V with low (not necessarily 0) resubstitution error for Z that generalizes well. Tends to retain points toward the center of class regions rather than decision boundaries.
- *Size/Accuracy Criterion:* search over possible subsets of Z to minimize a resubstitution error criterion, e.g.:

$$\max_{S \subseteq \mathbf{Z}} J(S) = \max_{S \subseteq \mathbf{Z}} \left\{ \text{Accuracy}(S) - \alpha \frac{|S|}{|\mathbf{Z}|} \right\}$$

(alpha is a constant weight)

Example Editing Methods

Hart's Method (Condensing)

Start with one sample from Z in V. For each subsequent sample, if misclassified by 1-NN on current set, add to V. Iterate over Z until no instances are misclassified.

Wilson's Method (Error-Editing)

Run 3-NN on Z, delete all misclassified samples to produce V.

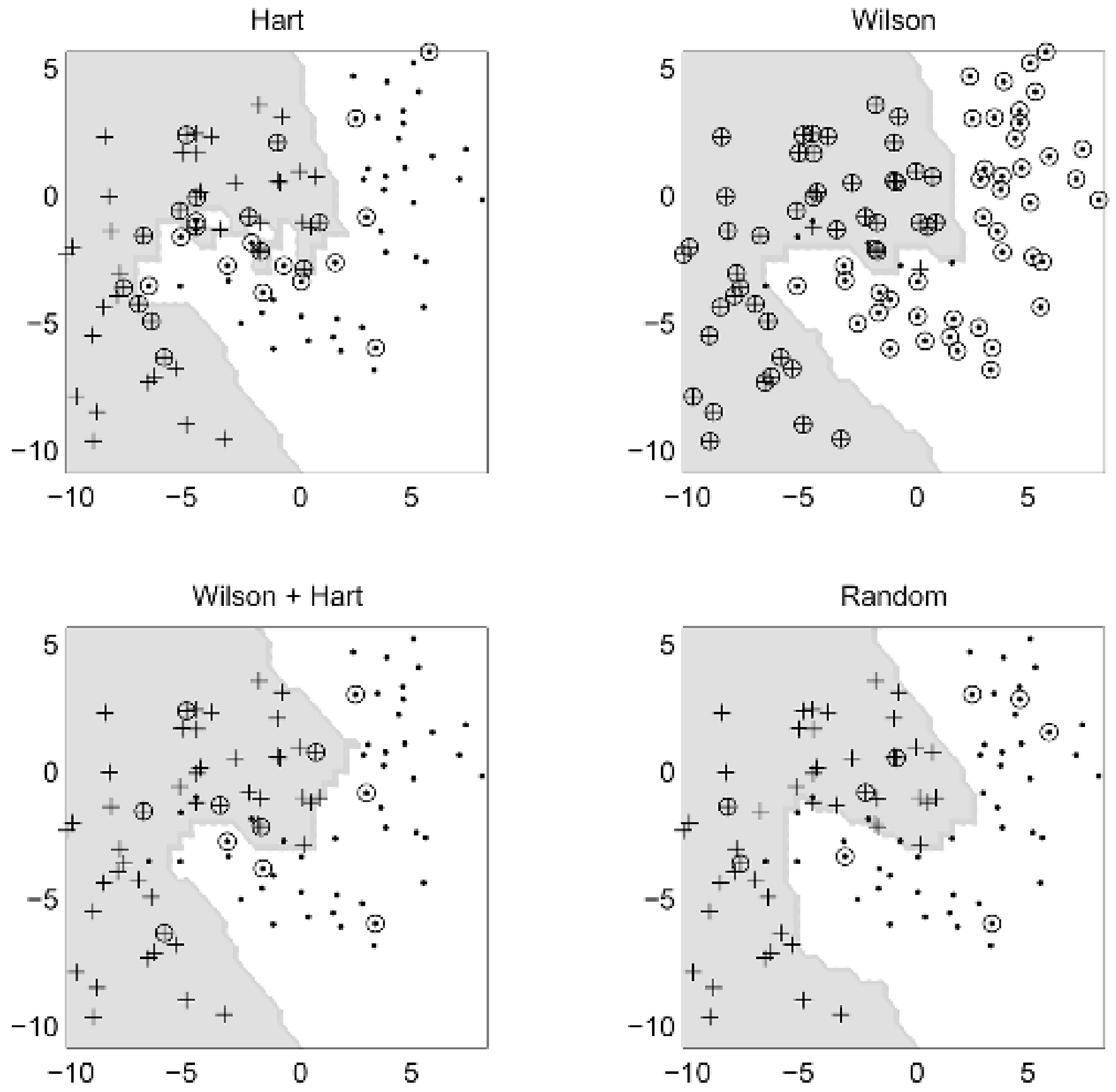
Random Editing (for comparison)

Choose |V|. Sample |V| elements from Z *T* times. Return the set with the smallest substitution error (for Z).

R·I·T

TABLE 2.2 Error Rates (in %) for 1-nn and Four Editing Methods for the Banana Data.

	Whole set	Hart	Wilson	Wilson + Hart	Random
Training error (%)	0	0	7	5	4
Testing error (%)	10	10	8	8	6
No. of prototypes	100	28	92	13	10



ig. 2.5 *Training data and classification regions for the banana data set obtained by the 1-nn lassifier and four prototype selection methods. The prototypes retained by the respective nethod are encircled.*

Protype Extraction

Idea

Allow training samples to be inferred at points in feature space, rather than being directly from Z.

Approaches

- Competitive learning: LVQ (learning vector quantization, a form of neural network model)
- Using gradient descent to tune sample locations.
- Bootstrap: variation of random editing, where new samples are inferred as the mean of its k-nearest neighbors

Example Prototype Extraction Methods

Chang (competitive learning)

Tries to find consistent prototype set (zero resubstitution error).

- Initially, V = Z.
- Iteratively finds two symbols ('parents') closest *within* a class, and replaces them by their average ('child').
- If this produces no error, keep the 'child,' otherwise mark symbol pair as ineligible.
- Repeat until all remaining 'parent' candidates produce a 'child' leading to an error.

Example Extractions, Cont'd

Bootstrap (random method)

Extend random editing to include a vector of parameters $(v_1, ..., v_i)$ for the number of elements to sample from each class. For chosen *k*, at each trial *T*:

- Sample |V| elements, according to $(v_1, ..., v_i) = |V|$
- Replace each selected sample by the mean of its k-nearest neighbors from the *same class*
- Return the prototype set with the lowest resubstitution error out of the T trials.

Example: Prototype Extraction

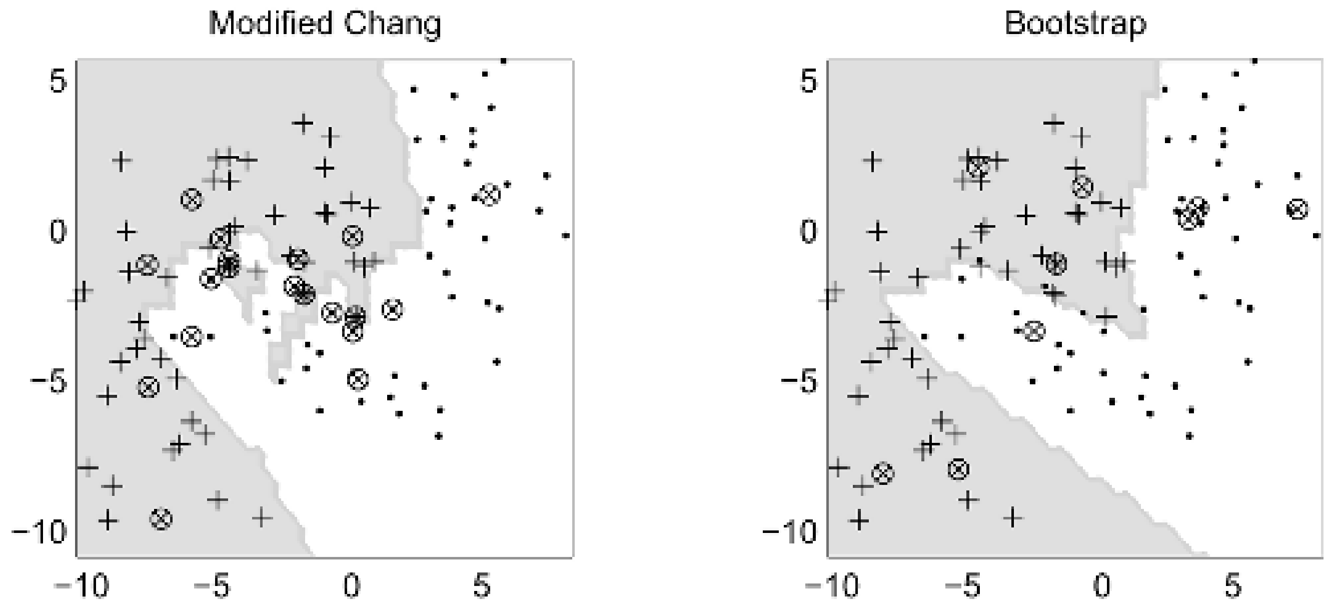


Fig. 2.6 *Training data and classification regions for the banana data set obtained by the 1-nn classifier and two prototype extraction methods. The prototypes retained by the respective method are encircled.*

Chang: finds 19 prototypes

Bootstrap: k=5, T = 1000, V = 10 (5,5)

- Train Error: Chang 0%, Bootstrap 3%
- Test Error: Chang 16%, Bootstrap 12%

R·I·T

Caveat

Theoretically, prototype extraction can produce better results than editing

- Locations where samples *might* be placed is unrestricted (e.g. can theoretically produce boundaries matching the underlying distribution)
- ...but there is no theoretical justification for the extraction methods described.

R·I·T

Incorporating Distances into k-NN

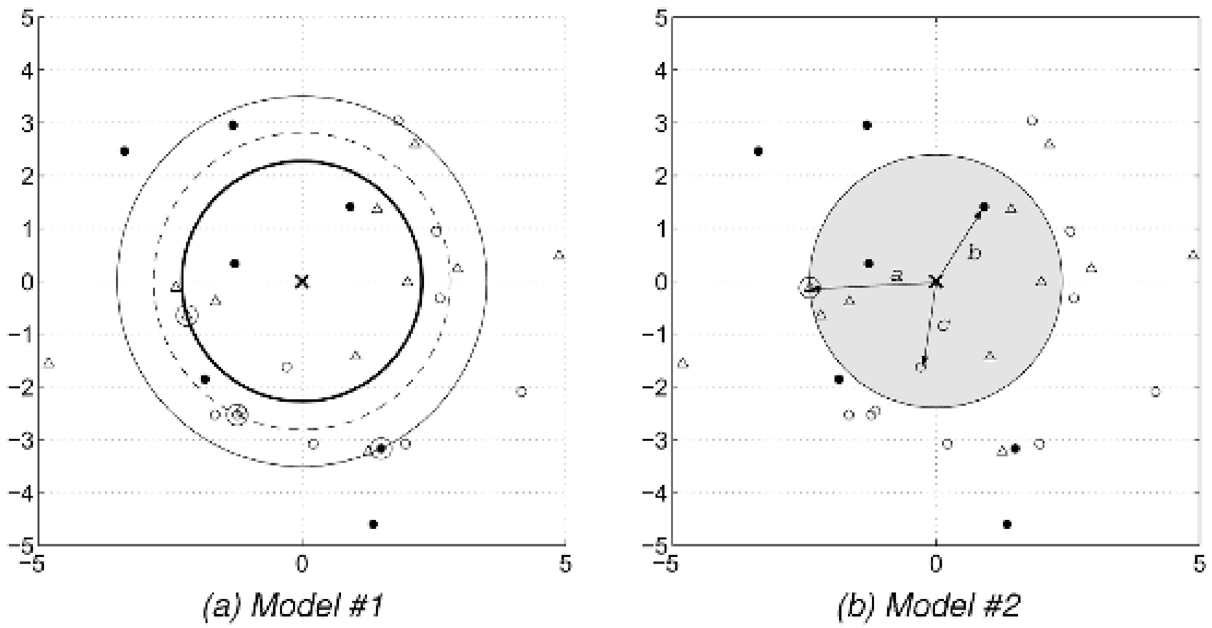


Fig. 2.7 *Illustration of k-nn Theoretical models no. 1 and no. 2.*

Model 1: Select class with smallest hypersphere containing exactly *k* instances

Model 2: For given k, select class based on k_i/a^2, where a is the radius of the hypersphere containing the ki samples.

Quick Aside: Data Structures

A number of metric data structures may be used to accelerate NN classification, or similarity comparison in general:

- k-d Trees (partitions features space)
- R-trees (partitions data set)
- M-trees (sample/distance-based indexing)

Ref: Foundations of Multidimensional and Metric Data Structures by Hanan Samet (2006))