Pattern Recognition
and Applications Lab

Part 3:

Elements of Non-parametric Techniques: the k-Nearest Neighbor (kNN) Classifier

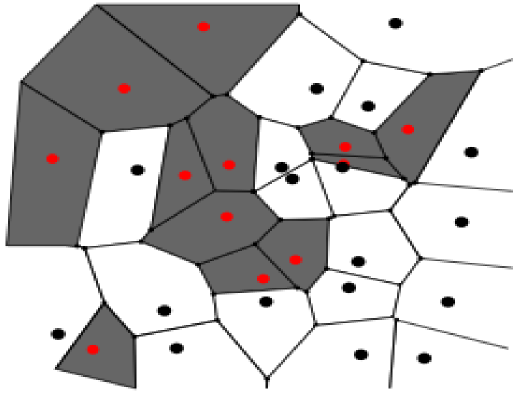
Battista Biggio
battista.biggio@unica.it

Introduction

- In Part 2, we assumed that the forms of the probability density functions were known
  - However, this assumption cannot be made in some pattern recognition applications
- In this chapter, we examine a nonparametric method that can be used with arbitrary distributions, without assuming knowledge of the underlying probability densities
- We discuss the k-Nearest Neighbor (kNN) pattern classifier, which allows:
  - estimating the density function $p(\mathbf{x} | \omega_j)$
  - estimating the posterior probability $P(\omega_j | \mathbf{x})$

The k-Nearest Neighbor (kNN) Method

- Nonparametric classification is often associated with the notion of **prototype**
- We can think of a prototype as a representative element from a class
- The class label assigned to an example is based on the similarity of this example to one or more prototypes
- Similarity is defined in a geometrical sense, i.e., based on a certain distance
  - The smaller the distance, the higher the similarity between the input **x** and the prototype(s)



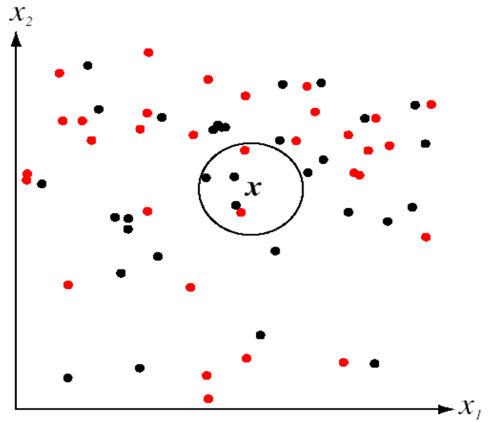
The k-Nearest Neighbor (kNN) Method

- *kNN is one of the most theoretically elegant and simple classification techniques* (L. Kuncheva, 2004)

- Let D be a labeled training set containing n points, called prototypes

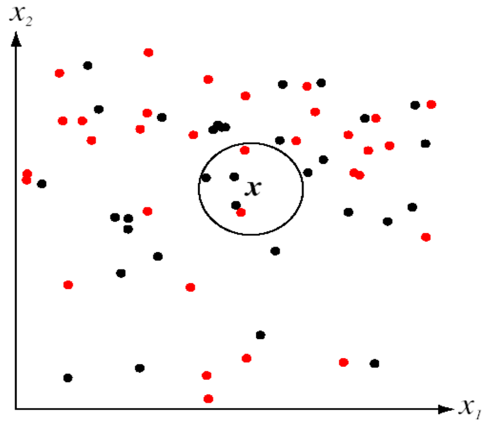
- Each prototype belongs to one of the "c" classes, i.e., $\mathbf{x}_i \, \varepsilon \, \omega_j \;\; j=1,\dots,c$

$$\mathbf{D} = [\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n]$$
$$\mathbf{x}_i = (\mathbf{x}_{i1}, \mathbf{x}_{i2}, \dots, \mathbf{x}_{id}) \;\; i=1,\dots,n$$



The k-Nearest Neighbor (kNN) Method

- To classify an input **x**, the k nearest prototypes are retrieved from **D** together with their class labels
- The input **x** is labeled to the most represented class label amongst the k nearest neighbors
- In the figure:
  - **x** is the pattern to be classified
  - we consider a "region" *R* of the feature space containing the k nearest prototypes to **x**
  - we classify **x** as belonging to the most represented class label amongst the k nearest neighbors within the region R.



The k-Nearest Neighbor (kNN) Method

- It can be shown that the kNN method estimates the posterior probabilities as:

$$\hat{P}(\omega_i|\boldsymbol{x}) = \frac{k_i}{k}$$

- k_i is the number of nearest neighbors belonging to the class ω_i within region *R*
- *k* is the number of the **k nearest prototypes** of **x** within region *R*
- The minimum (Bayes) error classifier using the approximations above will assign **x** to the class with the highest posterior probability
  - i.e., the class most represented among the k nearest neighbors of **x**

How to Select the Right Value of the Hyperparameter *k*

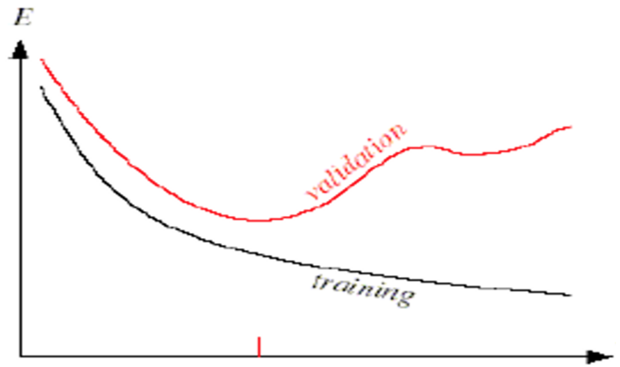
- One simple rule of thumb (heuristic rule) is to select *k* as $k = \sqrt{n}$
  - n is the number of examples in our training set D

- Note that this rule imposes that the number *k* of nearest neighbors of the pattern **x** that fall within the region R is smaller than the total number *n* of training set examples

- In binary (two-class) classification problems or problems with an **even** number of classes, it is helpful to choose *k* to be an odd number as this avoids **tie-breaks**

How to Select the Right Value of the Hyperparameter *k*

- **Cross-validation** is an experimental method for selecting the hyperparameter k
  - *We will discuss it in greater detail in Part 6*
- It consists of subdividing the initial data set D into three subsets:
  - training set, validation set, and test set
- We use the training set as the set containing the "prototypes"
- **Simple method:** we evaluate error E using different values of the k parameter with the validation set (more on this later)



The k-Nearest Neighbor (kNN) Method

- In the next slides, we see the theoretical concepts behind the k-NN method and how one can arrive at the approximation below:

$$\hat{P}\,(\omega_i|\boldsymbol{x})=\frac{k_i}{k}$$

Density Estimation for the kNN Method

- The basic idea underlying many non-parametric methods can be illustrated as follows
- The probability ***P*** that a vector **x** will fall in a region $\mathcal{R}$ of the feature space is:

$$P = \int_{\mathcal{R}} p(\mathbf{x}')\,d\mathbf{x}'$$

- if *R* is a small region, *P* can be regarded as a smoothed or averaged version of the density function p(**x**)
- We can thus estimate this smoothed value of p(**x**) with P

Density Estimation

- Suppose that n samples $\mathbf{x}_1,\ldots,\mathbf{x}_n$ are drawn independently and identically distributed (i.i.d.) according to the probability law p(**x**).

- If we know that **k** samples out of these **n** fall in *R*, then ***P*** can be estimated as ***P*** = k/n

- **Proof.** The probability that **k** out of **n** samples fall in *R* is given by the binomial law:

$$P_k = \binom{n}{k} P^k (1 - P)^{n-k}$$

- We know that the expected value of the binomial distribution for *k* is : $e(k) = nP$

Density Estimation

- If we consider the ratio k/n as argument of the binomial distribution, we can rewrite the expected value for k/n as:

$$\varepsilon(k/n) = P$$

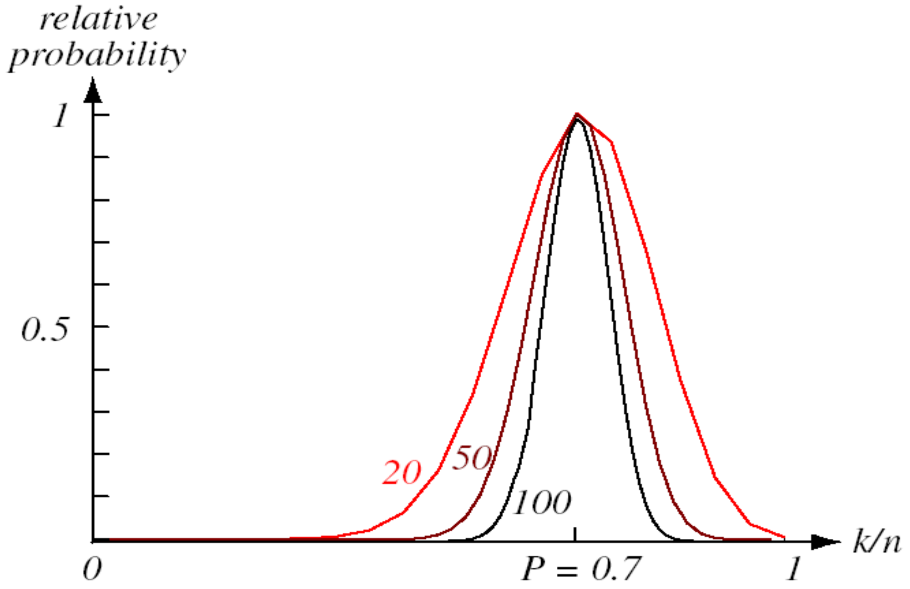
- We know that the variance of the binomial law is var(k/n)=P(1-P)/n
- If the number of samples n increases, the limit when it goes to infinity is:

$$n \to +\infty \;\to\; \varepsilon(k/n) = P \text{ and } \mathrm{var}(\mathrm{k/n})=0$$

- Therefore, we can say that k/n is an asymptotically unbiased estimator of P
- We expect that the ratio k/n will be a very good estimate for the probability P, and hence for the smoothed density function if n is very large
  - Indeed, this estimate is especially accurate when n is very large (see next Figure)

Density Estimation: Binomial Distribution

- The plot shows the probability P_k of finding k samples in a region where the averaged probability is P=0.7, as a function of k/n
- Each curve is labeled by the total number of patterns n
- **For large n, such binomial distributions peak strongly at k/n = P** (here chosen to be 0.7)



Density Estimation

- If we now assume that p(**x**) is continuous and that **the region R is so small** that p(**x**) does not vary appreciably within it, we can write:

$$\int_{\Re} p(\mathbf{x}')d\mathbf{x}' \cong p(\mathbf{x})V = P$$

- where x is a point within R and V is the volume enclosed by R.

- Combining these two estimates, we get the following expression for p(x):

$$\int_{\Re} p(\mathbf{x}')d\mathbf{x}' \cong p(\mathbf{x})V \cong k/n \;\rightarrow\; p(\mathbf{x}) \cong \frac{k/n}{V}$$

Density Estimation

C. Bishop, Pattern Recognition and Machine Learning, pp. 121-122, 2006

$$p(\boldsymbol{x}) = \frac{k}{nV}$$

- The validity of the above estimate depends on two ***contradictory*** assumptions:
  1. the region R has to be small enough that the density is approximately constant over the region
  2. and yet sufficiently large (*in relation to the value of that density*) that the number k of points falling inside the region is sufficient for the binomial distribution to be sharply peaked

- We can exploit this result in two different ways
  – Fix k and determine the value of V from the data, which gives rise to the kNN method
  – Fix V and determine k from data, giving rise to kernel density estimators (Parzen windows)

- It can be shown that both the kNN density estimator and the kernel density estimator converge to the true probability density in the limit of infinite samples, provided that V shrinks suitably with n and k grows with n (Duda and Hart, 1973).

Density Estimation: Contradictory Assumptions

$$\int_{\Re} p(\mathbf{x}')d\mathbf{x}' \cong p(\mathbf{x})V \cong k/n \;\rightarrow\; p(\mathbf{x}) \cong \frac{k/n}{V}$$

- **Bias:** If we fix the volume V and take more and more training samples, the ratio k/n will converge (in probability) as desired, but we have only obtained an estimate of the space-averaged value of p(**x**):

$$\frac{P}{V} = \frac{\int_{\Re} p(\mathbf{x}')d\mathbf{x}'}{\int_{\Re} d\mathbf{x}'}$$

- **Variance:** if we want to obtain p(**x**) rather than just a smoothed version of it, V has to tend to zero

Density Estimation

- From a practical standpoint, we note that the number of samples is always limited. Thus, the volume V can not be allowed to become arbitrarily small
  - One will have to accept a certain amount of *variance* in the ratio k/n and a certain amount of *bias*, i.e., averaging of the density p(x)

- From a theoretical standpoint, it is interesting to consider what happens in the limit of infinite samples
  - In practice, you have to ensure that, while the region shrinks (to reduce bias), the number of samples k within that region tends to infinity but at a lower rate than n (to reduce variance)

Density Estimation: Convergence

- To estimate the density at **x**, we form a sequence of regions R_1, R_2,... R_n containing **x** while varying the total number of samples *n*
- Let V_n be the volume of R_n, k_n be the number of samples falling in R_n, and p_n(**x**) be the *n*th estimate for p(**x**):

$$p_n(\mathbf{x}) = \frac{k_n / n}{V_n}$$

- If $p_n(x)$ is to converge to p(x), three conditions appear to be required (as number of samples goes to infinity):

$$\lim_{n \text{®} \text{¥}} V_n = 0$$

$$\lim_{n \text{®} \text{¥}} k_n = \text{¥}$$

$$\lim_{n \text{®} \text{¥}} \frac{k_n}{n} = 0$$

Density Estimation: Convergence

- The first condition assures us that the space averaged P/V will converge to p(**x**), provided that the regions shrink uniformly and that p(·) is continuous at **x**.

- The second condition, which only makes sense if p(**x**)≠ 0, assures us that the frequency ratio will converge (in probability) to the probability P

- The third condition is necessary if p_n(**x**) is to converge at all. It also says that although many samples will eventually fall within the small region R_n, they will form a negligibly small fraction of the total number of samples.

$$\lim_{n \to \infty} V_n = 0$$

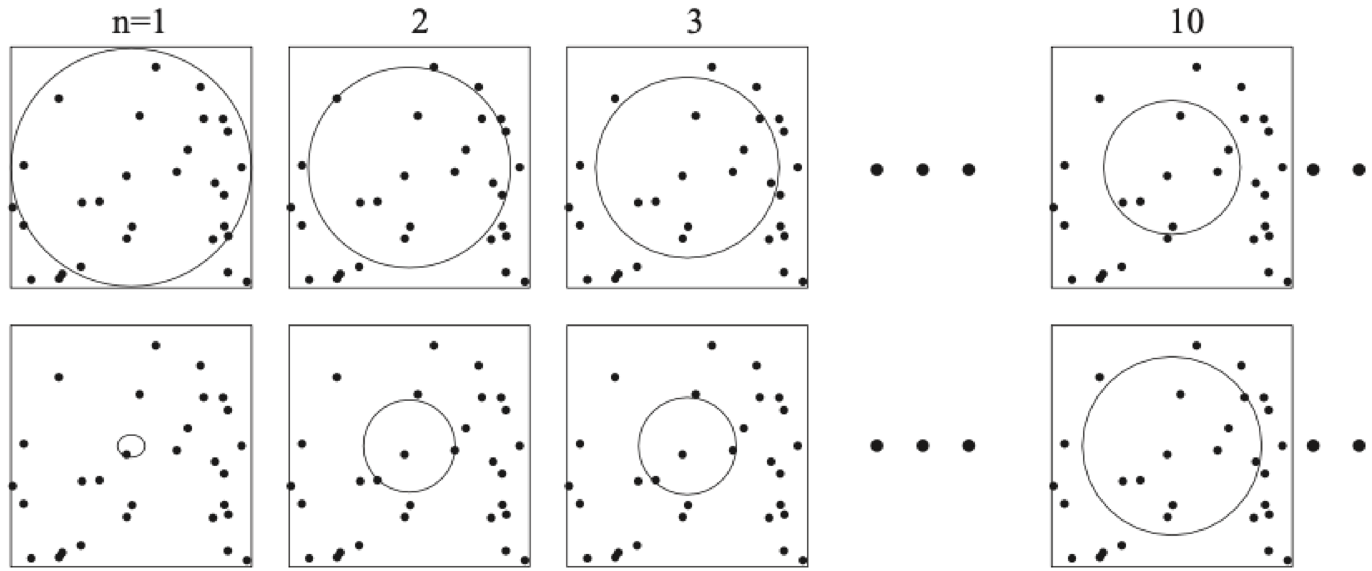
$$\lim_{n \to \infty} k_n = \infty$$

$$\lim_{n \to \infty} \frac{k_n}{n} = 0$$

Density Estimation: Convergence

- There are two common ways of obtaining sequences of regions that satisfy these conditions
- One is to shrink an initial region by specifying the volume V_n as some function of n, such as $V_n = 1/\sqrt{n}$. It then must be shown that the random variables k_n and k_n/n behave properly, or more to the point, that $p_n(x)$ converges to $p(x)$
  - This is the **Parzen-window / kernel density estimation (KDE)** method
- The second method is to specify k_n as some function of n, such as $k_n = \sqrt{n}$. Here, the volume V_n is grown until it encloses k_n neighbors of x
  - This is the **kNN** method
- Both these methods converge in the asymptotic regime of infinite samples, but it is hard to give guarantees in the finite-sample regime

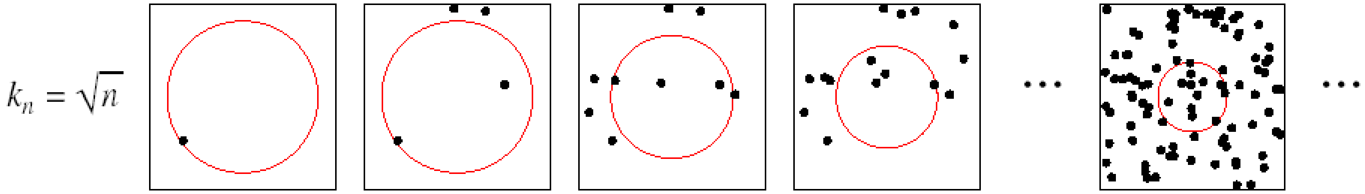
Density Estimation: KDE and kNN



k-Nearest Neighbor Method: Value of k

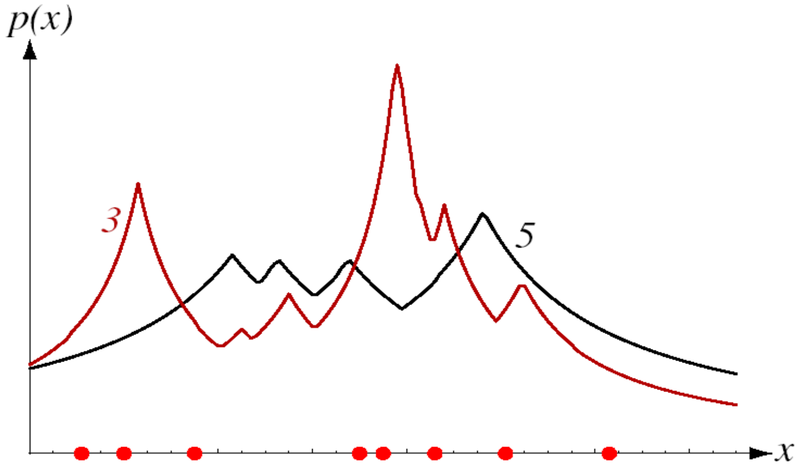
➤ **kNN non-parametric method**

- This method specifies k_n as some function of n, such as $k_n = \sqrt{n}$. Here, the volume Vn grows until it encloses k_n neighbors of x.
- **Key concept:** the region (volume) is specified considering the number k of the samples that fall into it.



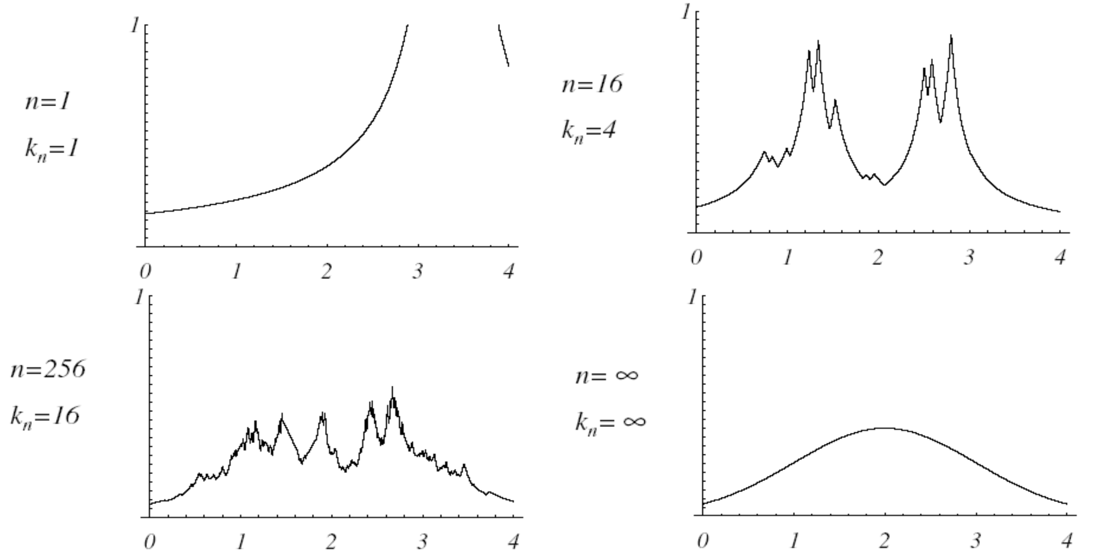
An Example of Density Estimation with kNN

- Eight points in one dimension and the kNN density estimates for k=3 and k=5.
- The discontinuities in the slopes in the estimates generally occur away from the positions of the points themselves



An Example of Density Estimation with kNN

- Several kNN estimates of a Gaussian distribution
- Note how the finite n estimates can be quite "spiky"



Kernel Density Estimation

- Recall that:

$$p(\boldsymbol{x}) = \frac{k}{nV}$$

- The validity of the above estimate depends on two ***contradictory*** assumptions:
  1. the region R has to be small enough that the density is approximately constant over the region
  2. and yet sufficiently large (*in relation to the value of that density*) that the number k of points falling inside the region is sufficient for the binomial distribution to be sharply peaked

- We can exploit this result in two different ways
  - Fix k and determine the value of V from the data, which gives rise to the kNN method
  - **Fix V and determine k from data, giving rise to kernel density estimators (Parzen windows)**

Kernel Density Estimation

C. Bishop, PRML, pp. 123-124, 2006

- Let's consider R to be a small hypercube centered on the origin
  - This is referred to as a *kernel* function or Parzen window

$$k(\mathbf{u}) = \begin{cases} 1, & |u_i| \leqslant 1/2, \qquad i = 1, \ldots, D, \\ 0, & \text{otherwise} \end{cases}$$

- The quantity $k\left(\frac{x - x_n}{h}\right)$ will be one if the data point x_n lies inside a cube of side h centered on x, and zero otherwise

- The total number of data points lying inside this cube will therefore be

$$K = \sum_{n=1}^{N} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right)$$

Kernel Density Estimation

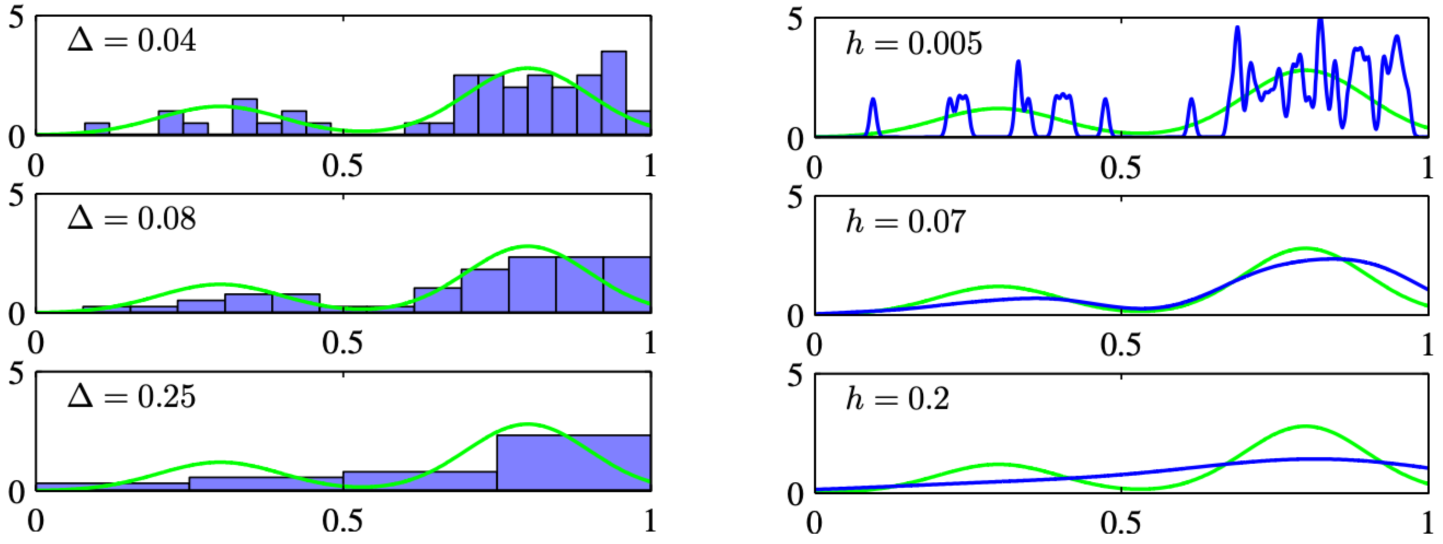
- We can thus estimate p(x) as

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{h^D} k\left(\frac{\mathbf{x} - \mathbf{x}_n}{h}\right)$$

- If we consider a Gaussian kernel instead of a hypercube:

$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{1/2}} \exp\left\{-\frac{\|\mathbf{x} - \mathbf{x}_n\|^2}{2h^2}\right\}$$

Kernel Density Estimation: 1D Examples



We have discussed density estimation so far.
How about classification?

From Density Estimation to Classification

- We can use the kNN or KDE estimators to get an estimate of the likelihood $p(\mathbf{x}|y)$
- Then estimate priors to compute the posterior probabilities, using the Bayes' theorem
- … and decide for the class exhibiting the maximum support (MAP criterion)

The k-Nearest Neighbor (kNN) Method

- In the previous slides, we have shown that:

$$p_n(\mathbf{x}) = \frac{k_n / n}{V_n}$$

- It is easy to show that the kNN method estimates the posterior probabilities as

$$P(\omega_i|\boldsymbol{x}) = \frac{k_i}{k}$$

- where k_i is the number of nearest neighbors belonging to the class ω_i within the region R

The k-Nearest Neighbor (kNN) Method

- To arrive at this classification method, we fix k_n and n and allow for a variable V_n. Assuming Euclidean distance, let R be the region containing exactly k_n of the elements of the training set D. We know that the unconditional p.d.f. can be approximated as

$$p_n(\mathbf{x}) = \frac{k_n / n}{V_n}$$

- Denoting by k_i the number of elements in R from class ω_i, the class-conditional pdf for ω_i, i=1,...,c, can be approximated in R, as

$$p(\mathbf{x} / w_i) = \frac{k_i / n_i}{V_n}$$

The k-Nearest Neighbor (kNN) Method

- The posterior probabilities are obtained as

$$P_n(w_i \mid \mathbf{x}) = \frac{p_n(\mathbf{x} / w_i) P(w_i)}{p_n(x)} = \frac{\frac{k_i/n_i}{V} \frac{n_i}{n}}{\frac{k/n}{V}} = \frac{k_i}{k}$$

- The minimum error (Bayes) classifier using the approximations above will assign **x** to the class with the highest posterior probability, i.e., the class most represented among the k nearest neighbors of **x**

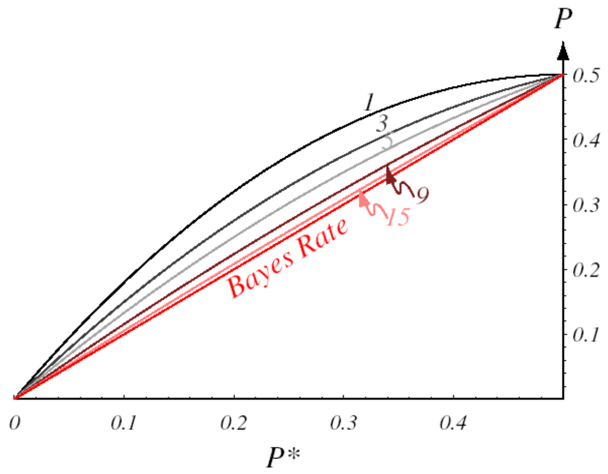
- The region R and the volume V are specific for each **x**
    - The kNN classification rule, however, assigns the class label using only the numbers k_i, so the winning label does not depend on V

The k-Nearest Neighbor (kNN) Method

- kNN is Bayes-optimal when:

$$\lim_{n\text{®}\,\text{¥}} k_n = \text{¥}$$

$$\lim_{n\text{®}\,\text{¥}} \frac{k_n}{n} = 0$$



- The error rate for the kNN rule for a two-category problem. Each curve is labeled by k; when k=∞, the estimated probabilities match the true probabilities, and thus, the error rate is equal to the Bayes rate, i.e., P = P_{Bayes}

The Nearest-Neighbor Rule (1NN)

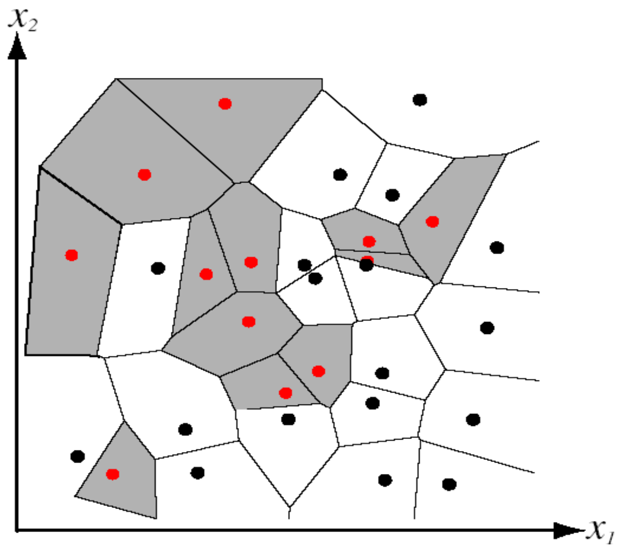
- Let be $D^n=\{x_1,..,x_n\}$ a training set of samples belonging to the "c" classes $\omega_1,..,\omega_c$ and be **x**'ε D^n the nearest sample to the unknown sample **x**
- Decision rule: we assign **x** to the class of **x**'
- The nearest-neighbor rule is a sub-optimal procedure
  - that usually leads to an error rate greater than the minimum possible, the Bayes rate.
  - However, with infinite prototypes, the error rate is never worse than twice the Bayes rate.

The Nearest-Neighbor Rule (1NN)

- Let be ω' the true class of **x'**. If "n" is very large, we can assume that **x'** is very close to **x**, so that P(ω'|**x'**)= P(ω_i|**x**)
  - Here, the nearest-neighbor decision rule is in good agreement with the MAP rule.
- In general, if we specify ω_m(**x**) by $P(w_m \mid \mathbf{x}) = \max_i P(w_i \mid \mathbf{x})$
- The MAP rule assigns the pattern to the class ω_m
  - If **x**' has been assigned to the class **j**, then we should assume that ω_m(**x'**)= ω_j
- This rule partitions the feature space into cells that contain the closest points to a given training point **x**'
  - All points in such a cell are thus labeled as the same class of the training point — a so-called **Voronoi tesselation** of the space (see next slide)

1NN and Voronoi Tesselation: A 2D Example

- In two dimensions, the nearest-neighbor algorithm **with k=1 (1-NN)** partitions the input space into Voronoi cells, each labeled by the category of the training point it contains



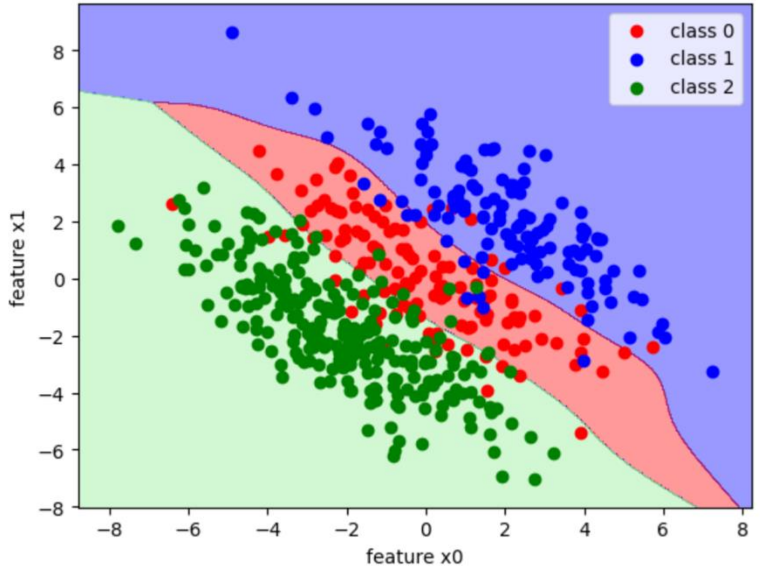
The Nearest-Neighbor Rule: Final Remarks

- If $P(\omega_m|\mathbf{x})\cong 1$, the nearest-neighbor rule is close to the optimal rule, as it is very unlikely that the posterior probability changes abruptly.
  - When the minimum probability of error is small, the nearest-neighbor rule's probability of error is also small.

- If $P(\omega_m|\mathbf{x})\cong 1/c$, the classes have the same probabilities, and the nearest-neighbor rule is likely suboptimal
  - In this case, the error probability is about $1-1/c$ for both methods.

Exercise (Laboratory)

- Consider the exercise solved with the Gaussian classifier in Part 2
- Solve it implementing a KDE classifier that estimates the likelihood of each class p(x | y) with KDE
- Is it any different? Which one is more accurate?
- Test the two models on the moon dataset. *Which one is more accurate? Why?*



Open in Colab https://github.com/unica-ml/ml/blob/master/notebooks/kde_classifier.ipynb

References

- Sections 4.1, 4.2, 4.3, 4.4, 4.5, Pattern Classification, R. O. Duda, P. E. Hart, D. G. Stork, John Wiley & Sons, 2000
- Chris Bishop, Machine Learning and Pattern Recognition, 2006.
- L. Kuncheva, Combining pattern classifiers, Wiley, 2004.