Statistical Pattern Recognition
Assignment 4

Ali Gholami

Department of Computer Engineering & Information Technology
Amirkabir University of Technology

https://aligholamee.github.io
aligholami7596@gmail.com

Abstract

In statistics, kernel density estimation (KDE) is a non-parametric way to estimate the probability density function of a random variable. Kernel density estimation is a fundamental data smoothing problem where inferences about the population are made, based on a finite data sample. In some fields such as signal processing and econometrics it is also termed the Parzen-Rosenblatt window method, after Emanuel Parzen and Murray Rosenblatt, who are usually credited with independently creating it in its current form. (Summarization from Wikipedia)

Keywords. *KNN Classifier, Kernel Density Estimation.*

1 Parzen Windows with Gaussian Kernels

Generate 100 random points from each of the following two distributions: $N(20, 5)$ and $N(35, 5)$. Write a program that employs Parzen window technique with a Gaussian kernel to estimate density using all 200 points. Note that this density conforms to a single bimodal distribution (a Gaussian Mixture Model).

(a) Plot the estimated density function for each of the following window widths: h = 0.01, 0.1, 1, 10. [Note: You can estimate the density at discrete values of x in the [0, 55] interval with a step-size of 1.]

(b) Repeat the above after generating (i) 500 training points from each of the two distributions, (ii) 1,000 training points from each of the two distributions; and (iii) 2,000 training points from each of the two distribution.

(c) Discuss how the estimated density changes as a function of the window width and the number of training points.

Solution

(a) The following items are generated by the Python file under the directory: *src/1*.

Suppose $h = 0.01$. The results is given in figure 1.1.

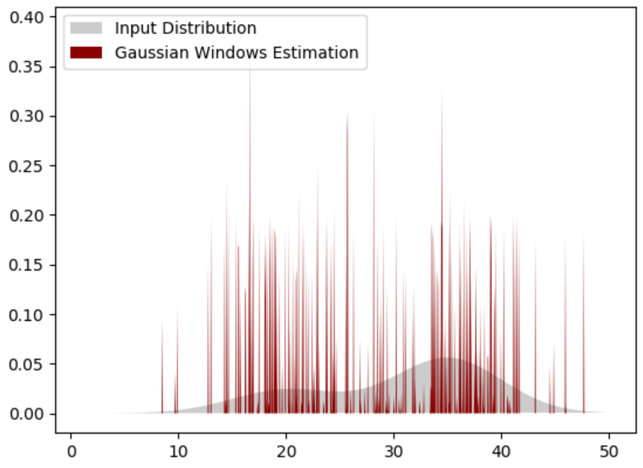


Figure 1.1: Density Estimation with Gaussian Kernels: $h = 0.01$.

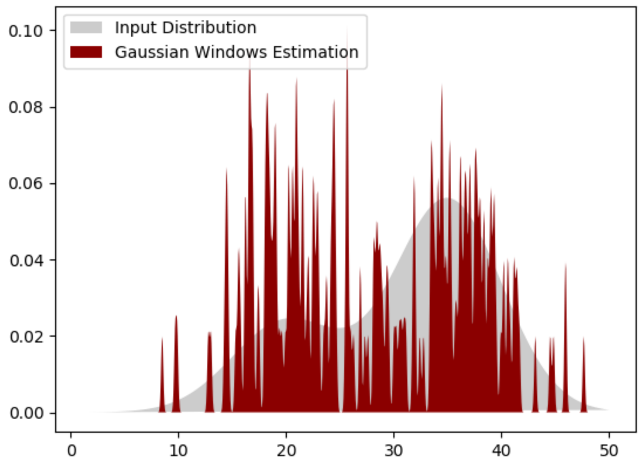


Figure 1.2: Density Estimation with Gaussian Kernels: $h = 0.1$.

Suppose $h = 0.1$. Plotted densities are given in figure 1.2.

Suppose $h = 1$. The results is given in figure 1.3.

Suppose $h = 10$. The results is given in figure 1.4.

(b) Here is the repeated procedure for 500 number of points for $h = 1$. Results is given in figure 1.5. It is obvious that the accuracy of estimation is higher in this case.

Results for 1000 number of samples is provided in figure 1.6.

Final results for 2000 samples is provided in figure 1.7.

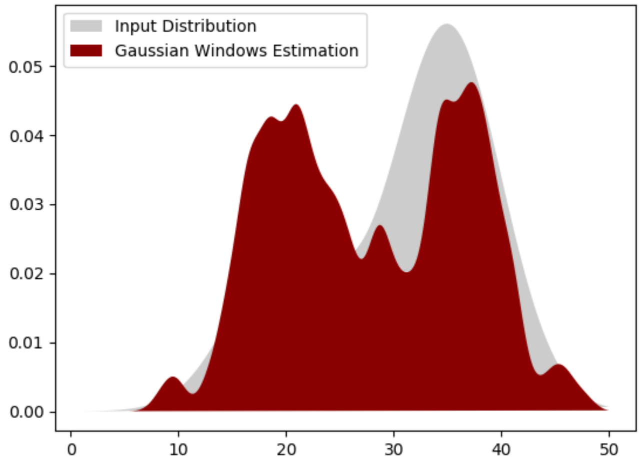


Figure 1.3: Density Estimation with Gaussian Kernels: $h = 1$.

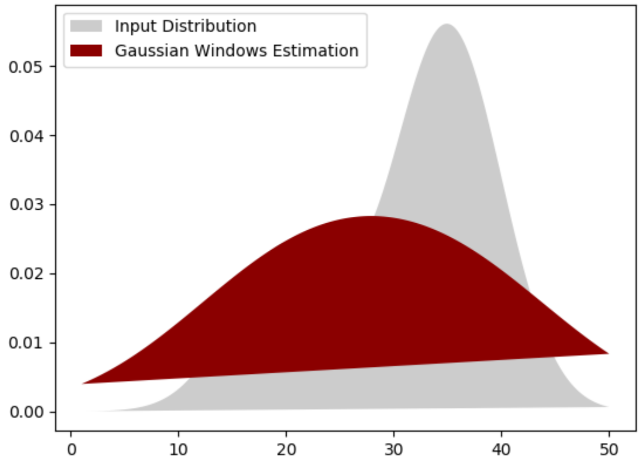


Figure 1.4: Density Estimation with Gaussian Kernels: $h = 10$.

(c) For any choice of kernel, the bandwidth h is a smoothing parameter, and controls how smooth the fit is by controlling the size of the neighbourhood around the reference, x0.

If h is large, we consider a large neighborhood, and vice versa.

In the Gaussian kernel case, varying h has the same effect as varying the variance of a Gaussian. Small h leads to a thinner, more peaked Gaussian, whereas larger h leads to a fatter Gaussian, in the extreme case, closer and closer to a flat line.

On the other hand, when the number of samples increase, the accuracy of the estimation

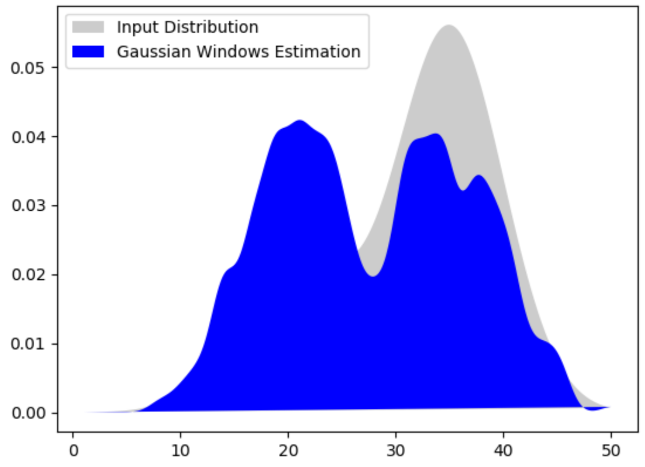


Figure 1.5: Density Estimation with Gaussian Kernels: $h = 1$, 500 Samples.

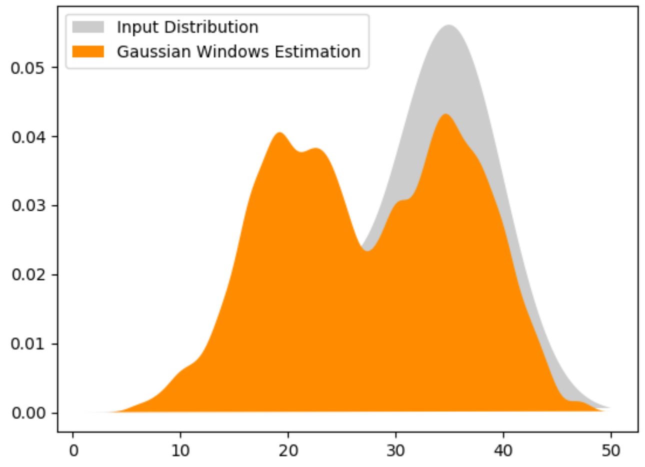


Figure 1.6: Density Estimation with Gaussian Kernels: $h = 1$, 1000 Samples.

goes high. This simply means that the bias of the estimation goes down.

2 Kernel Density Estimation & KNN Estimation

Given dataset $X = 2, 3, 4, 4, 4, 5, 5, 10, 11, 11, 11, 12, 14, 16$ use Parzen windows to estimate density $p(x)$ at $x = 5$ and $x = 12$; assuming $h = 4$ in the following conditions.

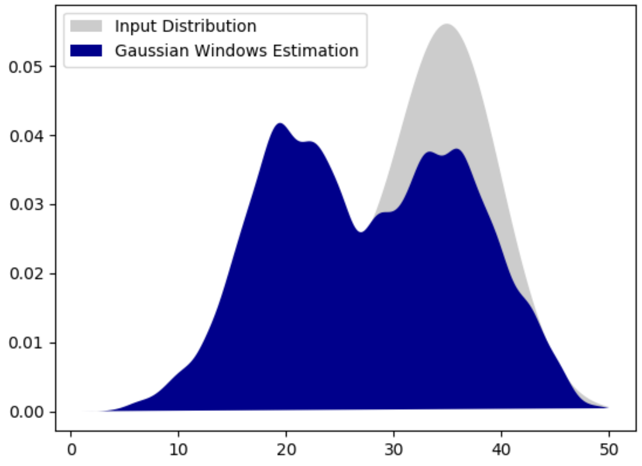


Figure 1.7: Density Estimation with Gaussian Kernels: $h = 1$, 2000 Samples.

(a) If you use standard kernel function

$$K(u) = \begin{cases} 1 & |u| \leq \frac{1}{2} \\ 0 & o.w. \end{cases}$$

(b) If you use Gaussian Kernel, $N(0, 1)$.

(c) For the same dataset and the same sample points, i.e. $x = 5$ and $x = 12$, estimate the density using KNN approach. Take $K = 5$ in your estimations.

Solution

(a) We'll use (1.1) to estimate the density using different kernels.

$$\hat{p}_{(x)} = \frac{1}{n * h^d} \sum_{i=1}^{k} \Phi(\frac{x - x_i}{h}) \tag{2.1}$$

In this case, we'll compute the distance of each of the given points from our dataset. For every distance less than 2 we'll consider the effect of that point in our estimation.

$$x = 5 \quad \rightarrow \quad \hat{p}_{(x)} = \frac{1}{14 * 4} * (6) = \frac{3}{28} = 0.107$$

Doing the same for the point $x = 12$ and we'll have the following results:

$$x = 12 \quad \rightarrow \quad \hat{p}_{(x)} = \frac{1}{14 * 4} * (6) = \frac{3}{28} = 0.107$$

(b) In this case, the value of items in the $\sum$ are no longer 1. They will have different outputs, specially on the center of the curve. This can lead to an smoother and more realistic estimation of the density. The Gaussian Kernel equation is given in (1.2).

$$\phi(u) = \frac{1}{\sqrt{2\pi}} \exp(\frac{-u^2}{2}) \tag{2.2}$$

$$x = 5 \quad \rightarrow \quad \hat{p}_{(x)} = \frac{1}{14 * 4} * \frac{1}{\sqrt{2\pi}}(\exp(-2) + 3\exp(\frac{-1}{2}) + 2)) = 0.028$$

$$x = 12 \quad \rightarrow \quad \hat{p}_{(x)} = \frac{1}{14 * 4} * \frac{1}{\sqrt{2\pi}}(2\exp(-2) + 3\exp(\frac{-1}{2}) + 1)) = 0.021$$

(c) In this case, we have to consider the window size as a dynamic variable. In case $h = 2$ then $x = 4$, $x = 4$, $x = 4$ and $x = 5$ will be in our window of estimation as well as the centered point $x = 5$ (K = 5). To estimate the density we'll have the following equation:

$$x = 5 \quad \rightarrow \quad \hat{p}_{(x)} = \frac{1}{14 * 2} * (5) = 0.178$$

Note that we have considered the kernel to be the same as part a. In the second case, choosing the $h = 2$ yields 5 points.

$$x = 12 \quad \rightarrow \quad \hat{p}_{(x)} = \frac{1}{14 * 2} * (5) = 0.178$$

3 Parzen Windows & KNN Estimation

Consider the following training set drawn from an unknown density $f(x)$:

$$X = \{0.01, 0.12, 0.19, 0.32, 0.41, 0.48\}$$

(a) Let $\phi(x) = N(0, 1)$. Find and sketch the Parzen Windows estimate for the values of h_n of 0.1 and 1.0.

$$\hat{f}_{n(x)} = \frac{1}{n * h_n} \sum_{i=1}^{k} \Phi(\frac{x - x_i}{h_n}) \tag{3.1}$$

(b) Find and sketch the 3-nearest neighbor estimate of $f(x)$.

Solution

(a) For each point in our dataset, we'll center the Gaussian Kernel and add the values of these kernels. Here are the computations in case that $h_n = 0.1$.

$$\hat{f}_{n(x=0.01)} = \frac{1}{6 * 0.1} * 1 = 1.66 \qquad \hat{f}_{n(x=0.12)} = \frac{1}{6 * 0.1} * 1 = 1.66$$

$$\hat{f}_{n(x=0.19)} = \frac{1}{6*0.1}*1 = 1.66 \qquad \hat{f}_{n(x=0.32)} = \frac{1}{6*0.1}*1 = 1.66$$
$$\hat{f}_{n(x=0.41)} = \frac{1}{6*0.1}*1 = 1.66 \qquad \hat{f}_{n(x=0.48)} = \frac{1}{6*0.1}*1 = 1.66$$

As denoted above, estimation will give 0 for all of the points in the dataset with $h_n = 0.1$. We can understand that, the window size has been chosen too small that no point is being considered by the estimator. Note that the $h = 0.1$ will be divided by 2 (Being symmetric) and the neighborhood of each given point in the dataset is checked with a diameter of 0.05. In this case, non of the windows contained other points. Since we have considered the point itself having an effect on the density (As in previous problem), we'll consider them here also. In this case, the density estimation plot is given in the figure 3.1.

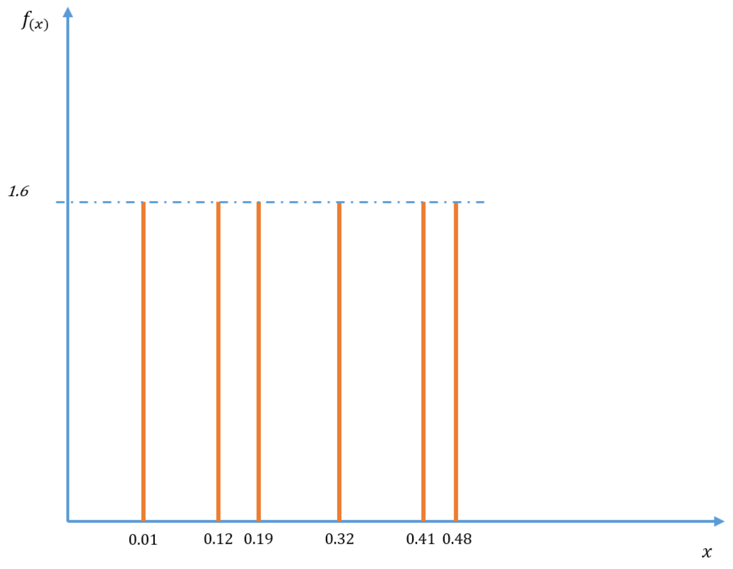


Figure 3.1: Kernel Estimation of f(x) with $h_n = 0.1$.

In the second case, the window size is $h_n = 1.0$. For each point in the dataset, if we put a window of size 1.0 on that point, all other points are included in the window. We can understand that in this case, the window size is too large. $f(x)$ estimation is given below.

$$\hat{f}_{n(x=0.01)} = \frac{1}{6*0.1}*(exp(0) + exp(\frac{-0.0121}{2}) + exp(\frac{-0.0324}{2}) + exp(\frac{-0.0961}{2}) +$$
$$exp(\frac{-0.16}{2}) + exp(\frac{-0.2209}{2}) = 0.95$$

This procedure is same for all of the given points in the dataset. In the case of window size of 0.1, the estimation results is given in the figure 3.2.

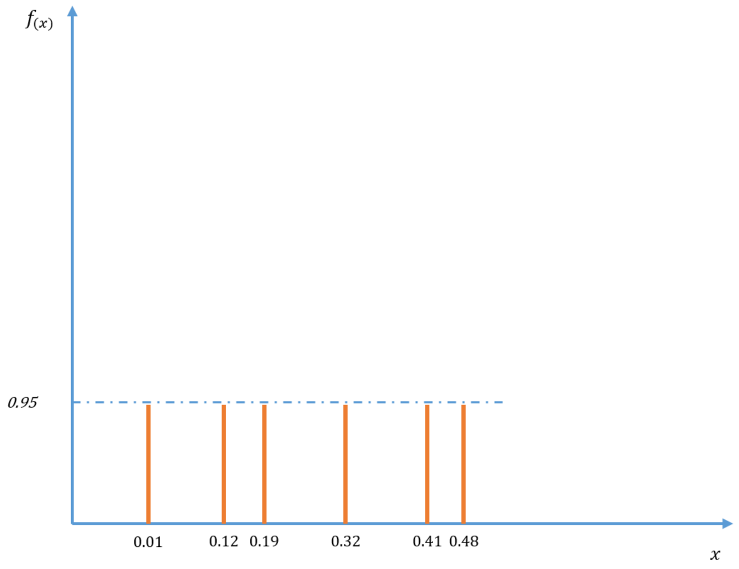


Figure 3.2: Kernel Estimation of f(x) with $h_n = 1$.

(b) In this section, we'll start from $h_n = 0$. We'll gradually increase the value of h_n until the window includes 3 samples. Starting from $x = 0.01$, suppose $h = 0.2$. Only 2 samples are included in the window. Increasing the window size to $h = 0.4$ will properly include 3 samples which is denoted by $k = 3$. Thus, for the first sample, the proper window size is $h_n = 0.4$. The estimated value will be:

$$\hat{f}_{n(x=0.01)} = \frac{1}{6 * 0.4} * (exp(0) + exp(\frac{-0.0121}{2}) + exp(\frac{-0.0324}{2})) = \frac{2.71}{2.4} = 1.12$$

$$\hat{f}_{n(x=0.12)} = \frac{1}{6 * 0.22} * (exp(\frac{-0.0121}{2}) + exp(0) + exp(\frac{-0.0049}{2})) = \frac{2.99}{1.32} = 1.24$$

Continuing this procedure for all of the points in the dataset yields the figure 3.3.

$$\hat{f}_{n(x=0.19)} = \frac{2.97}{1.56} = 1.9$$

$$\hat{f}_{n(x=0.32)} = \frac{2.97}{1.56} = 1.9$$

$$\hat{f}_{n(x=0.41)} = \frac{2.98}{1.08} = 2.75$$

$$\hat{f}_{n(x=0.48)} = \frac{2.96}{1.92} = 1.54$$

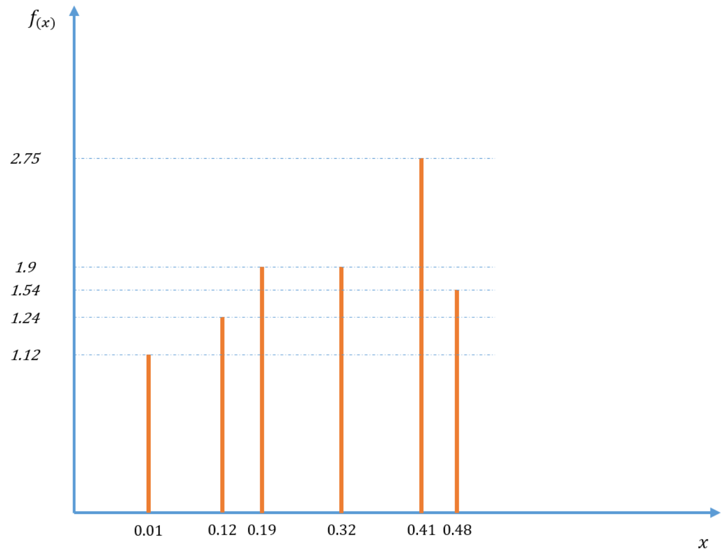
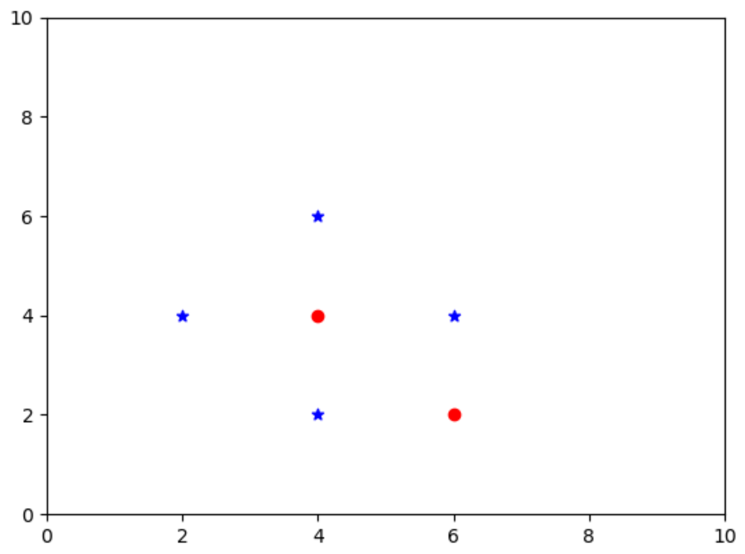


Figure 3.3: Kernel Estimation of f(x) with KNN Estimator with $K = 3$.

4 KNN Decision Boundary & Voronoi Cells

In the following questions you will consider a K-Nearest Neighbor classifier using Euclidean distance metric on a binary classification task.

(a) Sketch the 1-Nearest Neighbor decision boundary for this dataset.



(b) How would the point (8, 1) be classified using 1-NN?

Solution

(a) The decision boundary for 1-NN is given in figure 4.1.

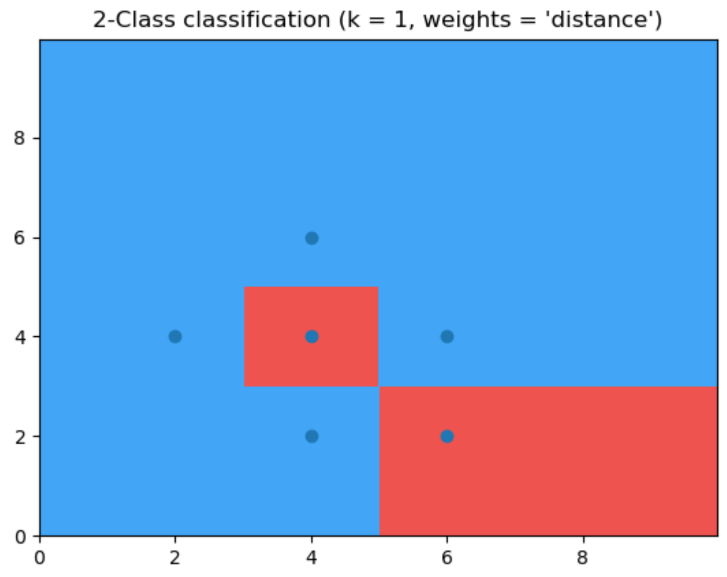


Figure 4.1: 1-NN Decision Boundaries & Voronoi Cells.

(b) It will be classified as red, as the regions are given in figure 4.1.

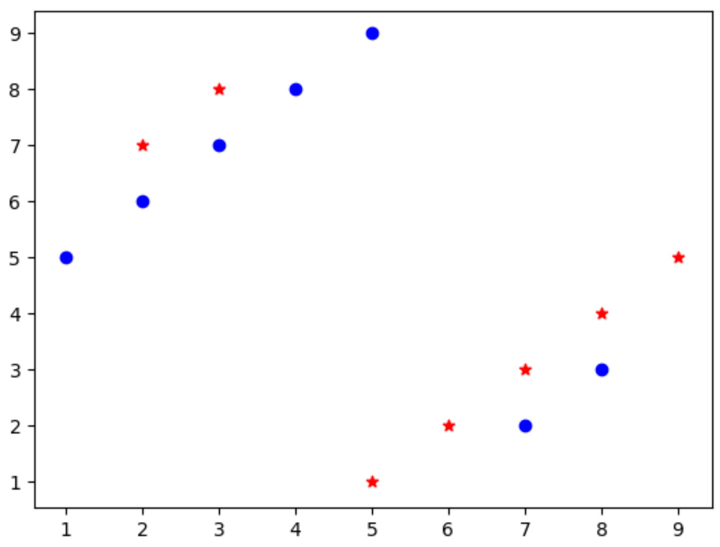
5 KNN Decision Boundary & KNN Error Rate

Consider a K-Nearest Neighbor classifier using Euclidean distance metric on a binary classification task.

(a) In the figure, sketch the 1-Nearest Neighbor decision boundary for this dataset.

(b) If you try to classify the entire training dataset using a K-NN classifier, what value of K will minimize the error for this dataset?

(c) What happens if you use a very large K on this dataset? Why might too small values of K also be bad?



Solution

(a) Here is the results plotted using Matplotlib in Python.

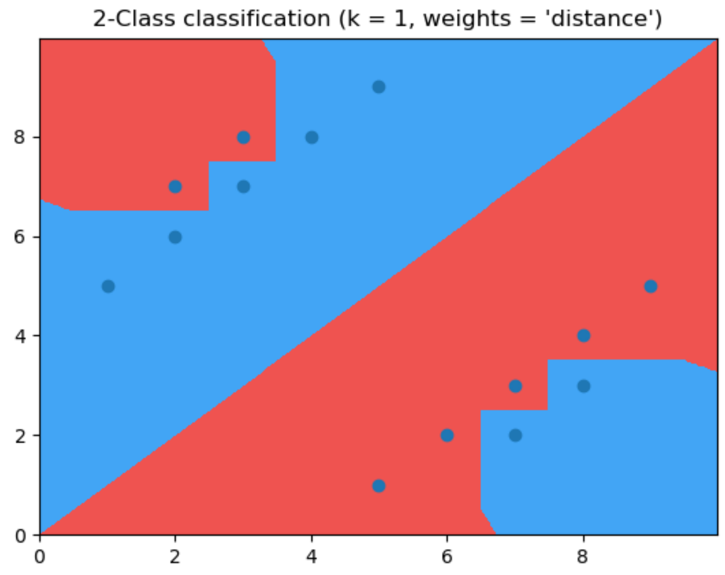


Figure 5.1: 1-NN Decision Boundaries & Voronoi Cells.

(b) K can be found using experiments. The code is provided in the *src* folder.

(c) Using large K's yields in a smaller degree of freedom. Thus, it prevents the classifier from overfitting. Too small values for K result in a higher degree of freedom. This results in an overfitted model.