Amin Fadaee

Pattern Recognition HW4

Python 3.5, Libraries: Numpy, Matplotlib, Math Documentation: HTML, Bootstrap 4, jqMath

Problem 1

As we know, the formula for estimating the density at point x in the area R, given a set of n training points denoted by D is given by the following formula:

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

where k is the number of points in R and V is the volume of R.

Parzen window method of estimation fixes the window size R (so that V is constant) and estimates the density at x using the following formula:

$$p_\phi(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h^d}\phi\left(\frac{x-x_i}{h}\right)$$

where d is the dimension of the points, h is the size parameter of the window and ϕ is the window function which computes the number of points in the region R. Window function is different based on the window shape. In the case of a hypercube shape the window function is:

$$\phi\left(\frac{x-x_i}{h}\right) = \begin{cases} 1 & |x-x_i| < \frac{h}{2} : j = 1, \dots, d \\ 0 & otherwise \end{cases}$$

In this problem, as $d = 1$ and $h = 1$ the mentioned formulas would become:

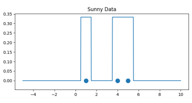
$$p_\phi(x) = \frac{1}{n}\sum_{i=1}^{n}\phi\left(\frac{x-x_i}{h}\right)$$

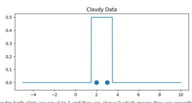
$$\phi\left(\frac{x-x_i}{h}\right) = \begin{cases} 1 & |x-x_i| < \frac{1}{2} \\ 0 & otherwise \end{cases}$$

The function `phi` in `Problem 1.py` implements this formula.

To derive the plots, the data for each class has been separated into 2 sets and the $p(x) = \frac{k}{nV}$ value has been calculated for all x in range -5 to 10 with steps of 0.01.

Here are the plots:





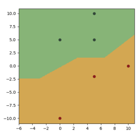
As can be seen the area under both plots are equal to 1 and they are above 0 which means they are correctly correspondent with probability distributions. Also the reason that all the rises have equal height is due to the fact that any parzen window with size 1 would at most has one point in it.

Problem 2

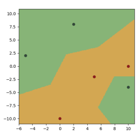
The file `Problem 2.py` contains a simple implementation of nearest neighbor classification for 2 or 3 classes named `nearest_neighbor` and the function `plot_decision_boundary` plots the regions of decision boundaries based on the classification using matplotlib's `pcolormesh`

Here are the results

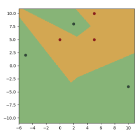
1. Class w_1 (red) and w_2 (green)



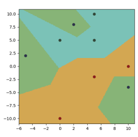
2. Class w_1 (red) and w_3 (green)



3. Class w_2 (red) and w_3 (green)



4. Class w_1 (red), w_2 (green), w_3 (blue)



As can be seen in each plot in an arbitrary point in the space, the color is corresponding to the color of the nearest neighbor to that point.

Problem 3

In Parzen window method, we are not limited to rectngular windows. We can use a general window ϕ as long as the resulting $p_\phi(x)$ is a legitimate density. In this case we are no longer counting the number of samples inside R, instead we are counting the weighted average of potentially every single sample point

In this problem we are going to use the *Spherical Gaussian* given by the following formula:

$$e^{\frac{-(x-x_i)^t(x-x_i)}{2h^2}}$$

The function `spherical_gaussian` implements this density.

In classifiers based on Parzen-window estimation, we estimate the densities for each category and classify a test point by the label corresponding to the maximum posteriori.

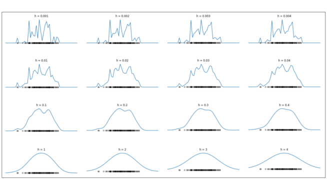
As the priors and the total probability is equal in all classes, we need only the $p(x \mid w_i)$s to compare for all classes and decide the proper class for an instance, the function `p_x` has been implemented to find the conditional probability based on data of a class at an arbitrary point x and based on a specified h. This function has been utilized to obtain the class for each of the instances and here are the results:

For $h = 1$:
- Class of (0.5, 1.0, 0.0) is 2
- Class of (0.31, 1.51, -0.5) is 2
- Class of (-0.3, 0.44, -0.1) is 2

For $h = 0.1$:
- Class of (0.5, 1.0, 0.0) is 2
- Class of (0.31, 1.51, -0.5) is 2
- Class of (-0.3, 0.44, -0.1) is 2

Problem 4

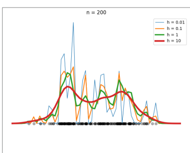
100 Samples were drawn from the distribution $N(0, 1)$ using `numpy.random.normal` and the function `gaussian_window` in `Problem 4.py` has been implemented to derive the parzen gaussian estimate at point x. Then for 16 different values of h the estimation has been plotted using all x in the interval -4 to 4 with steps of 0.15 in length. Here are the results:

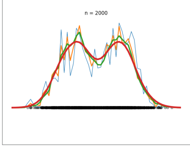


As can be seen, for small values of h the estimation is fluctuating and noisy. by increasing the Parzen size h, the estimation gets more smooth and get closer to the real density, however, for the higher hs (3 and 4) the result looks oversmoothed and out of focus.

Problem 5

The same window used in the previous problem has been utilized to obtain the following plots (The densities have been normalized for better aesthetics in the plots):





The first plot depicts the density for $h = (0.01, 0.1, 1, 10)$ with 100 samples for each of the distributions and the second plots depicts the same density but with 2000 samples in total.

As can be seen, in the same manner as the previous problem, in both plots, the density becomes smoother when h is increase, also having 10 times more samples, the second plots has tremendously smoother densities than the first plot and this is due to the fact that it can be shown that parzen window performs better as the number of samples are increased and it will ultimately converge to the exact density as number of samples goes to infinity

In case of the bimodality of the plots it can be seen that with small number of samples, the estimation only succeed to capture the bimodality in the highest h, whereas in the second plot, all the hs (even the lowest one) captures the general trend of two maximums.