EE 573 Pattern Recognition Project 4

Oğuzhan Sevim

I. INTRODUCTION

In this project, we design a classifier based on Bayesian Decision Rule. Due to some characteristics of the given dataset, feature reduction will first be applied. In the remaining sections, we will assume that the class conditional probability distribution of the samples are not know. In order to estimate the likelihoods $p(x|w_c)$, we will use 2 non–parametric techniques: Parzen Window and K–Nearest Neighbor estimation methods. Then, by combining them with Bayesian decision rule, we will separately make classifications on training and test sets.

A. Dataset

The given dataset consists of $n = 1315$ samples where each sample has $d = 500$ features. The given dataset is classified into $C = 8$ different classes. For each class c, where $c = 1,\dots,8$, we portion the randomly selected 75% of the class c data as D_c and the remaining 25% as T_c. Since distribution of D_c sets are used for future predictions, they can be considered as training data. Therefore, in the remaining of this report, D_c will be referred as training sets, where we will call T_c as test sets.

For the covariance matrix of class c to be nonsingular (invertible), $n_c > d$ should be satisfied, where n_c denotes the number of samples in class c. Since each class in the dataset consists of few hundreds of samples, it is not feasible to continue with the given dataset. Instead, a feature reduction method (e.g., PCA) is required. In the remaining part of the project, feature size $d = 500$ will be reduced to $d' = 50$.

II. TASK 1: PARZEN WINDOW ESTIMATION

In this section, we try to estimate the likelihoods by Parzen Window method. Then, by using the estimated likelihoods, we derive decision rules for each class c. By using the resulting decision rule, the tests samples will be classified.

For the Parzen Window method, we use the Gaussian Kernel for the window function which is given as

$$\phi(u) = \frac{1}{\sqrt{2\pi}} e^{-u^2/2}. \quad (1)$$

Then, using $h_n = h_1/\sqrt{n}$, the likelihood estimation for a given sample x will be calculated by

$$\begin{aligned} p_n(x|w_c) &= \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_n}\phi\left(\frac{x-x_{c,i}}{h_n}\right) \quad (2)\\ &= \frac{1}{h_1\sqrt{n}}\sum_{i=1}^{n}\frac{1}{\sqrt{2\pi}}e^{-(x-x_{c,i})^T(x-x_{c,i})/2} \\ &= \frac{1}{h_1\sqrt{n}}\frac{1}{\sqrt{2\pi}}\sum_{i=1}^{n}e^{-(x-x_{c,i})^T(x-x_{c,i})/2}. \end{aligned}$$

Here, n is the number of samples which is used to estimate the likelihood density function. For each classes, two different values, $n = 10$ and $n = 50$, are used. Also, h_1 parameters are taken as $h_1 = 0.5$ and $h_1 = 1$.

Also, the prior probability for each class c is assumed to be known by

$$P(w_c) = \frac{|D_c|}{\sum_{j=1}^{C}|D_j|}. \quad (3)$$

A. Decision Rule

After estimating the likelihood, we now can derive the decision rule. For a given sample x, by the Bayes Rule, we decide w_c if

$$P(w_c|x) > P(w_j|x), \forall j \neq c. \quad (4)$$

Also, by the Bayes formula, the posterior probabilities can be expressed as

$$P(w_c|x) = \frac{p(x|w_c)P(w_c)}{p(x)}. \quad (5)$$

By using (3), posteriors can be rewritten by

$$P(w_c|x) = \frac{1}{p(x)\sum_{j=1}^{C}|D_j|}p(x|w_c)|D_c|. \quad (6)$$

Here, the first fractional term is just a constant independent of w_c. Therefore, the decision rule can be simplified as follows: Decide w_c if

$$p(x|w_c)|D_c| > p(x|w_j)|D_j|, \forall j \neq c. \quad (7)$$

After replacing (3) into (7), and simplifying the resulting decision rule by taking the natural logarithm of both sides, we get the following classification rule: Decide w_c if

$$ln(D_c)\sum_{i=1}^{n}e^{-(x-x_{c,i})^T(x-x_{c,i})/2} > ln(D_j)\sum_{i=1}^{n}e^{-(x-x_{j,i})^T(x-x_{j,i})/2} \quad (8)$$

Please note that there is no window size parameter h_1 dependency in the decision rule (8). Therefore, we expect to see no difference when we change h_1.

In order to evaluate the performance of the classification algorithm, precision and recall metrics are used. These metrics are given in Tables I, II, and III for the test (25%) set. Since the randomly chosen n values will be classified correctly, the training (75%) set is not classified.

The corresponding code for this section can be found in task1.m file.

TABLE I
METRICS ON TEST SET FOR $n = 10$ AND $h_1 = 0.5 or 1$

Class	**c1**	**c2**	**c3**	**c4**	**c5**	**c6**	**c7**	**c8**
Prec.	0.71	0.34	0.70	0.33	0.16	0.58	0.40	0.41
Recall	0.40	0.29	0.73	0.18	0.37	0.47	0.33	0.68

TABLE II
METRICS ON TEST SET FOR $n = 30$ AND $h_1 = 0.5 or 1$

Class	c1	c2	c3	c4	c5	c6	c7	c8
Prec.	0.48	0.45	0.75	0.43	0.24	0.64	0.35	0.48
Recall	0.50	0.48	0.80	0.33	0.23	0.60	0.30	0.68

TABLE III
METRICS ON TEST SET FOR $n = 50$ AND $h_1 = 0.5 or 1$

Class	c1	c2	c3	c4	c5	c6	c7	c8
Prec.	0.60	0.49	0.74	0.40	0.21	0.64	0.38	0.53
Recall	0.60	0.50	0.82	0.31	0.23	0.60	0.40	0.61

As we increase n, I would expect a general increase in the performance metrics. If we compare $n = 10$ and $n = 50$ cases, 11 metrics out of 16 increases with number of samples. But, it seems like a weak result.

III. Task 2: K-Nearest Neighbor Estimation

In this part, we are trying to calculate the posterior probability $P(w_c|x)$ for each class c by using K–nearest Neighbor rule. Given n class c samples, the likelihood of sample x for class c can be calculated as:

$$P(x|w_c) = \frac{k/n}{V_k}. \tag{9}$$

Here, n is the number of samples in the given class c. Also, V_k is the smallest volume centered at x which contains k samples. By putting (9) into (7), the decision rule can be written as follows: Decide w_c if

$$\frac{|D_c|}{V_{k,c}} > \frac{|D_j|}{V_{k,j}}, \forall j \neq c. \tag{10}$$

This rule works as follows: We are given a test point x. For each class, which consists of n samples, we look for the closest k points to point x. Then, out of k points, we determine the one that is farthest from x. If we denote this point as $x_{c,far}$, the volume can be approximated as $V_{k,c} = |x - x_{c,far}|^d$ where d is the dimension of a sample. Then, the final decision rule can be expressed as: Decide w_c if

$$\frac{|D_c|}{|x - x_{c,far}|^d} > \frac{|D_j|}{|x - x_{j,far}|^d}, \forall j \neq c. \tag{11}$$

Performance of the classification rule is measured by precicion and recall metrics. The results are given in Tables IV, V, and VI.

TABLE IV
METRICS ON TEST SET FOR $k = 5$ AND $n = 16$

Class	c1	c2	c3	c4	c5	c6	c7	c8
Prec.	0.61	0.26	0.71	0.32	0.17	0.63	0.43	0.62
Recall	0.37	0.10	0.83	0.41	0.30	0.57	0.53	0.54

TABLE V
METRICS ON TEST SET FOR $k = 10$ AND $n = 50$

Class	c1	c2	c3	c4	c5	c6	c7	c8
Prec.	0.46	0.43	0.70	0.42	0.18	0.59	0.43	0.65
Recall	0.43	0.39	0.83	0.49	0.10	0.53	0.53	0.54

TABLE VI
METRICS ON TEST SET FOR $k = 10$ AND $n = 80$

Class	c1	c2	c3	c4	c5	c6	c7	c8
Prec.	0.50	0.43	0.74	0.42	0.13	0.59	0.45	0.64
Recall	0.50	0.39	0.83	0.39	0.13	0.53	0.57	0.57

After seperating the dataset as training and test data, some classes have less than 100 data points in the training set. Therefore, the simulations are carried out for maximum of $n = 80$ cases.

For higher n values, the density estimation becomes more accurate for a given class. Therefore, as we increase k and n, I would expect a general increase in the performance metrics. If we compare $n = 10$ and $n = 50$ cases, 10 metrics out of 16 increases with number of samples. But, I think it is a weak result.

The corresponding code for this section can be found in task2.m file.

IV. Comparison of Estimators

Since they have the same $n = 50$ values, we may try to make a comparison by looking at Tables III and V. For the classes $c = 1, 2, 3, 5, 6$ Parzen Window method performs better. For the remaining classes, K-Nearest Neighbor rule gives better results. Therefore, the best estimation rule may depend on the class distribution and data set.