Kerem Anar

ENGR 421: Intro. To Machine Learning

HW\_02 Report

Part 1)

Section 5.7 is read from the textbook.

Part 2)

I take the data by using np.genfromtxt() function as we did in previous lab sessions.

Part 3)

I divide the data and label into 2 parts as training and test. First 30000 data and label are assigned to data\_train and label\_train for training purpose. The remaining 5000 data and labels are assigned to data\_test and label\_test. Also, I record the class number in K ( = 5) and the number of pixels for each image data in N ( = 784).

Part 4)

Text

Description automatically generated

Figure : Mean calculation for each class.

My sample\_mean is 5x784 matrix. I took each image data (1x784) which are in the same class and calculated the average of each pixel values. I repeated it for other classes. Finally, I obtained the sample\_mean =

sample means:

[[254.99866667 254.98416667 254.85616667 ... 254.679 254.87816667

254.95933333]

[254.99733333 254.99733333 254.9965 ... 254.96883333 254.99216667

254.98866667]

[254.99933333 254.99933333 254.99233333 ... 251.52483333 254.4725

254.97483333]

[254.99666667 254.98983333 254.91416667 ... 252.39516667 254.44166667

254.93666667]

[254.999 254.98433333 254.93783333 ... 250.673 253.23333333

254.79083333]]

which is same as we could see in the pdf manual.

For deviation,

Text

Description automatically generated with low confidence

Figure : sample deviation calculation.

In each class, I took square the values which the sample mean values are subtracted from the data points which belong to same class, and then took the average of this. For example:

Mean (square (), …)

is for one class. Then I took the square root of this 1x784 array. Finally, I obtained sample deviations for each pixel value in same class. This is the biggest difference from the hw1. In this time, we obtained separate variance values for each pixel or element of a object in a class **independently**. Naïve Bayes means simpler version of Bayes. In previous homework, we considered the relationship between different properties (for this specific problem pixels), so we obtained a covariance matrix. If we want to construct a covariance matrix for this problem too, we should just put the square of deviations we found in diagonals of the matrix. Upper and lower triangle parts have to be zero because we do not consider the relationship between different pixels, they are independent for Naïve Bayes. Then, the sample\_deviations =

sample deviations:

[[ 0.09127736 0.25609108 1.31090756 ... 5.29826629 3.9117332

1.93959091]

[ 0.2065419 0.2065419 0.2163818 ... 1.04076669 0.47057267

0.70062226]

[ 0.05163547 0.04081939 0.16002465 ... 18.43665868 6.7881694

1.1061344 ]

[ 0.18436076 0.21617116 1.81046936 ... 15.67799977 6.34549162

1.79971911]

[ 0.04471018 0.64582342 3.03248555 ... 23.62576428 13.9167006

4.4727787 ]]

which is same with in pdf manual as we could see.

For class priorities,

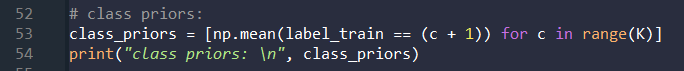


Figure : class priority calculation.

I simply sum like 1(.) (one) functions and divide by the total number of points as I did in previous hw and labs. We saw that the class priorities are equal:

class priors:

[0.2, 0.2, 0.2, 0.2, 0.2]

Part 5)

In Naive Bayes, we represent the classes as separate Gaussians whose means and deviations are what we calculated in the previous part.

p (x1 | c) = \*

p (x2 | c) = \*

…

p (x784 | c) = \*

P (x | c) = p(x1 | c) \* p(x2 | c) \*…\*p(x784 | c)

so, the probability density is the multiplication of gaussians.

P (c | x) = is the class estimation probability for the given x data. Denominator is same for all of them. So, we are interested in maximizing the numerator. P(c) is the class priority.

If we take loglikelihood of P(c|x), and maximize it, we could obtain the score function.

Score function = log ( p(x1 | c) \* p(x2 | c) \*…\*p(x784 | c) \* P(c) )

Text

Description automatically generated= [(-0.5 \* ) + ()] + [(-0.5 \* ) + ()] + … + [(-0.5 \* ) + ()] + log(P(c))

Figure : Score function in the code.

This function calculates score values of a data point for each class.

Then I assigned the class label by comparing the maximum of the score values. I tried to do label predictions by defining function for it like score, but I got some dimension errors. Probably I made some mistakes during assigning arrays in function parameters. Then, I made label prediction separately by hand in the code.

Text

Description automatically generated

Figure : Label estimation for training data

As in hw1, I used np.argmax() to obtain the index of the maximum element of the score values. I added them to a y\_predicted\_train array. Finally, I constructed the confusion matrix with it and label\_train values as the following:

Confussion matrix for training data:

y\_truth 1.0 2.0 3.0 4.0 5.0

y\_pred

1 3685 49 4 679 6

2 1430 5667 1140 1380 532

3 508 208 4670 2948 893

4 234 60 123 687 180

5 143 16 63 306 4389

Part 6)

In this part, I repeated the part 5 by using data\_test and label\_test instead of data\_train and label\_train.

The confusion matrix for the test data is the following:

Confussion matrix for test data:

y\_truth 1.0 2.0 3.0 4.0 5.0

y\_pred

1 597 6 0 114 1

2 237 955 188 267 81

3 92 25 785 462 167

4 34 11 16 109 29

5 40 3 11 48 722