HW 1 - ML Principles

Dharmik Patel - dsp187

September 2024

1 Question 1

1.1

Precision Rate, pr, is measured by

$$truePositives/(truePositives + falsePositives)$$

If we use pr to measure the empirical error on the training set, the circle classifier will focus on minimizing the number of false positives. So the circle will be smaller, making sure not to include any negative labels from the training set inside the circle. Here is an example of a binary circle classifier that is optimizing precision Figure 1:

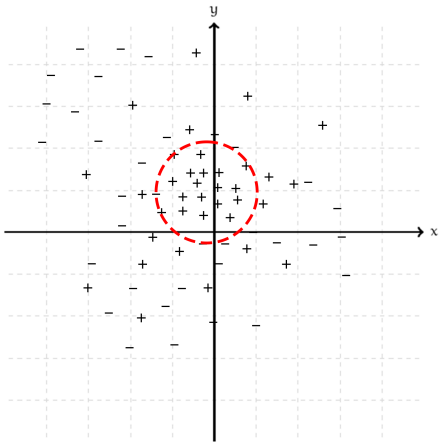

Figure 1: Circle classifier using pr to measure empirical error. *(not actually trained)*. As shown in the figure, the circle is tight around center cluster, which leads to false negatives because it excludes some positive labels from inside the circle

1.2

Recall Rate, rr, is measured by

$$truePositives/(truePositives + falseNegatives)$$

If we use rr to measure the empirical error on the training set, the circle classifier will focus on minimizing the number of false negatives. So the circle will be larger, making sure not to exclude most positive labels from the training set inside the circle. Here is an example of a binary circle classifier that is optimizing recall: Figure 2:

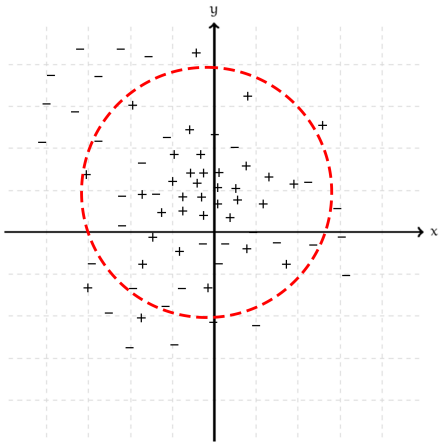

Figure 2: Circle classifier using rr to measure empirical error. *(not actually trained)*. As shown in the figure, the circle is large, as it tries to encompass all positive labels, irrespective of the negatives ones leading to false positives.

1.3

Both methods have some trade offs. If we used the method outlined in subsection 1.1 then the circle classifier will yield a small tight circle. Since the circle is small, it will exclude positive labels that are not within the circle, giving a lot of false negatives. This will reduce the coverage of the classifier.

If we used the method outlined in subsection 1.2 then the circle classifier will yield a larger circle. Since the circle is large, there is a chance of including negative labels in the circle, which would give a lot of false positives. This will cause the classifier to be less reliable/useful, because it would give a lot of positive predictions for negative labels , even thought it is predicting positive labels correctly.

The two methods are on opposite poles. I think if you use the Accuracy rate, given that the training data has similar amounts of positive and negative

labeled data, you would get a better circle classifier. That is because accuracy is just

$$correctPredictions/allPredictions$$

So if we try to maximize the number of correct predictions, then we will get a better model.

2 Question 2

2.1

$$\Omega_1 = \Omega_2 = \Omega_3 = ... = \Omega_M = \{L, R\}$$

At all sample space Ω_i where i is the level number, the ball only has 2 options: to either go left L or right R.

2.2

$$\Omega = \Omega_1 \times \Omega_2 ... \times \Omega_M = \{L^M, L^{M-1}R^1, L^{M-2}R^2, ..., L^1R^{M-1}, R^M\}$$

Since, $\Omega_1 = \Omega_2 = \Omega_3 = ... = \Omega_M = \{L, R\}$, we can say that:

$$\Omega = \Omega_1 \times \Omega_2 ... \times \Omega_M = \Omega_1^M = \{L, M\}^M$$

This is the Cartesian product of set Ω_1 taken M times. It can be written as:

$$\Omega = \Omega_1 \times \Omega_2 ... \times \Omega_M = \{(a_1, a_2, \ldots, a_M) a_i \in \Omega_1 \text{ for all } i = 1, 2, \ldots, M\}$$

This sample space Ω would encode all path that the ball could have taken to get to ground level G. An example element in Ω where $M = 3$ would be LRL, which would show that the ball took a left, right and then left at each respective level.

2.3

The location at L_G will tell us how many rights and lefts the ball took. It will not tell us the exact path/order of movement, because even if we know the number of rights and lefts, we cannot predict the exact order.

2.4

I would represent the location using the random variable X, which would be defined as X is the number of rights taken by the ball or the number of 'R's for a given value $\omega \in \Omega$.

2.5

$X \sim Bin(n, p)$ where $n = M$ is the number of levels/'trials' and at each level there is a $p = 0.5$ chance of going Left or Right. The pmf for any binomial distribution is given from:

$$f(x) = P(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$$

So the following plots the pmf of Binomial distribution when $M = 5, M = 10, M = 100$.

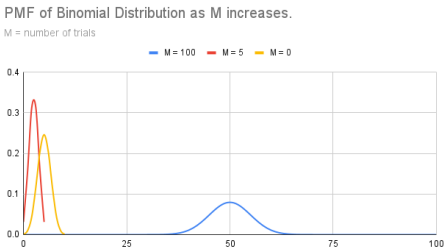

Figure 3: Made using google sheets: Link

The graph shows that as the number of trials increases, the distribution of X becomes a bell-shaped curve. This results in the pmf transitioning from a binomial shape for smaller M to a normal distribution for larger M. This illustrates the Central Limit Theorem because as the sample size increases, it will approach a normal distribution, irrespective of the starting distribution.

3 Question 3

Let

$$W = \{w, \neg w\} \tag{1}$$
$$D = \{d, \neg d\} \tag{2}$$

where W represents if the plant is watered or not, and D represents if the plant is dead or not. We are given that $P(\neg w) = .3$, $P(d|\neg w) = .8$, $P(d|w) = .2$

3.1

Find the probability of the plant not dying:

$$\begin{aligned} P(\neg d) &= 1 - P(d) && (3)\\ &= 1 - (P(d, w) + P(d, \neg w)) && (4)\\ &= 1 - (P(d|w)P(w) + P(d|\neg w)P(\neg w)) && (5)\\ &= 1 - (.2 * .7 + .8 * .3) && (6)\\ &= 1 - .38 && (7)\\ &= .62 && (8) \end{aligned}$$

3.2

Find the probability of the plant being dead if your friend did not water it. This probability is given in the question.

$$P(d|\neg w) = .8$$

3.3

Find the probability that your friend forgot to water, if the plant is dead.

$$\begin{aligned} P(\neg w|d) &= \frac{P(\neg w, d)}{P(d)} && (9)\\ &= \frac{P(d|\neg w)P(\neg w)}{P(d)} && (10)\\ &= \frac{.8 * .3}{.38} && (11)\\ &= .63 && (12) \end{aligned}$$

4 Question 4

Let lowercase d mean $D = +$ and let $\neg d$ mean $D = -$. When writing just D I mean that D can be d or $\neg d$.

4.1

Re-write the formula below given that G and B are conditionally independent given D. We are able to get to equation (15) because we can split the conditional probability into 2 separate conditional probabilities because G and B are

conditionally independent given D.

$$P(d|g,b) = \frac{P(d,g,b)}{P(g,b)} \tag{13}$$

$$= \frac{P(g,b|d)P(d)}{P(g,b)} \tag{14}$$

$$= \frac{P(g|d)P(b|d)P(d)}{P(g,b)} \tag{15}$$

To find $P(\neg d|g,b)$, you repeat the process above but replace all d with $\neg d$. So here is the following:

$$P(\neg d|g,b) = \frac{P(\neg d,g,b)}{P(g,b)} \tag{16}$$

$$= \frac{P(g,b|\neg d)P(\neg d)}{P(g,b)} \tag{17}$$

$$= \frac{P(g|\neg d)P(b|\neg d)P(\neg d)}{P(g,b)} \tag{18}$$

Given a glucose and blood pressure record (g,b), you can:

1. Calculate the prior probability $P(D)$ based on the training data.
2. Then calculate the likelihood by calculating $P(g|D)$ and $P(b|D)$. We can model the continuous variables G and B on normal distributions with the means and variances calculated from the data.
3. We can ignore the Marginal Likelihood, which is the denominator, since it is the same in both equations. So since we are comparing the final results, we do not need need to compute Marginal Likelihood.
   (a) If we had to compute Marginal Likelihood, then we would normalize the final 2 calculations.
4. Then we multiple the prior by the likelihood to get the posterior probability. The posterior is calculated for D=$\{d, \neg d\}$. Which ever one has the higher posterior, we predict that. So if $P(d|g,b) > P(\neg d|g,b)$, then we predict positive, and if not, then we predict negative.

4.2

When you run *dsp187_nb_train.py* you will see the mean and variances used. They are calculated manually instead of using built in functions like np.mean and np.var. Here is the output in figure 4. The maximum likelihood mean and variance are is used to make the Gaussian distribution.

4.3

Wrote the Naive Bayes classifier code in *dsp187_nb_cls.py*

```
(#, #) refers to (mean, variance) pairs
Given Postive Diabetes
        Glucose: (44.63, 81.16)
        Blood Pressure: (71.63, 40.76)
Given Negative Diabetes
        Glucose: (44.14, 13.40)
        Blood Pressure: (86.79, 21.14)
```

Figure 4: Output of *dsp187_nb_train.py*

4.4

I have gotten an accuracy rate of 0.923.

4.5

The standardization of data was not necessary when using a Naive Bayes classifier because the scale/magnitude of variables(glucose, and blood pressure) do not effect the posterior probabilities because to classify we are **comparing** the 2 posterior probabilities, so any offset/scaling that standardization would do, would not matter.

4.6

I do not think that the training data was reflected reality well. Because after seeing the prior distribution of $\{0.51125, 0.48875\}$, which means that 51% of people had diabetes while 49% of people did not. That prior distribution does not reflect reality well, because I do not think that half of the population has diabetes. But since we can not collect the data again, I would manually change the prior distribution to be move towards some figures found by a medical study about diabetes.

5 Question 5

$$X = 10000 \times 3 \text{ matrix}$$

$$E[X] = \begin{bmatrix} .2 \\ .3 \\ .1 \end{bmatrix} \quad COV[X, X] = \begin{bmatrix} 2.75 & 0.43 & 0 \\ 0.43 & 2.25 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

5.1

If $\vec{Y} = A\vec{X} + b$, then $\vec{Y}$ is just a linear transformation. That means that

$$E[Y] = E[AX + b] = A \begin{bmatrix} .2 \\ .3 \\ .1 \end{bmatrix} + b$$

$$COV[Y,Y] = COV[AX+b, AX+b] = COV[AX, AX] = A \begin{bmatrix} 2.75 & 0.43 & 0 \\ 0.43 & 2.25 & 0 \\ 0 & 0 & 1 \end{bmatrix} A^T$$

5.2

To whiten Y we must have

1. $COV[Y,Y] = I$ which means $A \cdot COV[X] \cdot A^T = I$
2. E[Y] $= 0$ which means $AE[X] + b = 0$

Let's solve for Part 1 first: The spectral decomposition of the $COV[X] = E\Lambda E^T$ where the E is filled with the the eigenvectors of $COV[X]$ and Λ is a diagonal matrix with its diagonal entries being the eigenvalues of $COV[X]$. Then $A = \Lambda^{-\frac{1}{2}} E^T$. So using numpy, we get

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 0.35239612 & -0.61250806 & 0 \\ -0.50065412 & -0.28804285 & 0 \end{bmatrix}$$

Then to solve Part 2: $AE[X] + b = 0$ can be written as $b = -AE[X]$. Solving this gives as

$$b = \begin{bmatrix} -0.1 \\ 0.11327319 \\ 0.18654368 \end{bmatrix}$$

These will insure that Y is whitened. These were calculated using numpy. All code can be see in question5.py.