CS577 - Machine Learning – Assignment 2

Alexandros Angelakis csdp1363, `angelakis@csd.uoc.gr`

Exercise 1 - Probabilities (Theoretical)

Let the random variable X follow the distribution

$$f(x;\theta) = \theta^2(x+1)(1-\theta)^x \quad ,x = 0,1,2,\ldots,\theta \in [0,1]$$

a.

We are trying to find the expression describing the MLE estimators for θ, for N independent identically distributed (i.i.d) samples. The likelihood function $L(\theta)$ is, by definition:

$$L(\theta) = f(x_1,x_2,\ldots,x_N|\theta) = \prod_{i=1}^{N} f(x_i|\theta) = \prod_{i=1}^{N} \theta^2(x_i+1)(1-\theta)^{x_i}$$

In order to find the MLE estimator for θ, we need to calculate the derivative of the expression described above. This is hard, so the easiest way is to take the logarithm of this expression and then calculate the derivative with respect to θ:

$$LL(\theta) = \log(\sum_{i=1}^{N} \theta^2(x_i+1)(1-\theta)^{x_i}) = \sum_{i=1}^{N}(\log\theta^2 + \log(x_i+1) + \log(1-\theta)^{x_i}) = \sum_{i=1}^{N}(2\log\theta + \log(x_i+1) + x_i\log(1-\theta))$$
$$= \sum_{i=1}^{N} 2\log\theta + \sum_{i=1}^{N}\log(x_i+1) + \sum_{i=1}^{N} x_i\log(1-\theta) = 2N\log\theta + \sum_{i=1}^{N}\log(x_i+1) + \log(1-\theta)\sum_{i=1}^{N} x_i$$

Now, we take the derivative of the expression with respect to θ and set it equal to zero:

$$\frac{\partial LL(\theta)}{\partial\theta} = 0 \longleftrightarrow \frac{2N}{\theta} - \frac{\sum_{i=1}^{N} x_i}{1-\theta} = 0 \longleftrightarrow \frac{2N}{\theta} = \frac{\sum_{i=1}^{N} x_i}{1-\theta} \longleftrightarrow 2N(1-\theta) = \theta\sum_{i=1}^{N} x_i \longleftrightarrow$$
$$\longleftrightarrow 2N - 2N\theta = \theta\sum_{i=1}^{N} x_i \longleftrightarrow 2N = \theta(\sum_{i=1}^{N} x_i + 2N) \longleftrightarrow \theta = \frac{2N}{\sum_{i=1}^{N} x_i + 2N}$$

Thus, $\hat{\theta}_{\text{MLE}} = \frac{2N}{\sum_{i=1}^{N} x_i + 2N}$. To be exactly sure that this value is indeed the maximum, we need to check the sign of the second derivative at $\hat{\theta}$. If $\frac{\partial^2 L(\theta|x)}{\partial\theta^2} < 0$, then the log-likelihood function is concave at $\hat{\theta}$, indicating that $\hat{\theta}$ is a maximum. The first derivative is:

$$\frac{\partial LL(\theta)}{\partial\theta} = \frac{2N}{\theta} - \frac{\sum_{i=1}^{N} x_i}{1-\theta}$$

so the second derivative is:

$$\frac{\partial^2 LL(\theta)}{\partial\theta^2} = -\frac{2N}{\theta^2} - \frac{\sum_{i=1}^{N} x_i}{(1-\theta)^2}$$

The term $-\frac{2N}{\theta^2}$ is negative for $\theta \in (0,1]$, since $2N > 0$ and $\theta^2 > 0$ for $\theta \neq 0$.
The term $-\frac{\sum_{i=1}^{N} x_i}{1-\theta}$ is also negative for $\theta \in [0,1)$, since $\sum_{i=1}^{N} x_i \geq 0$ and $(1-\theta)^2 > 0$ for $\theta \neq 1$.
Thus, for $\theta \in (0,1)$, the second derivative is negative, which indicates that the log-likelihood function is concave, and hence, the value of θ we found corresponds to a maximum (at $\theta = 0$ the first term would tend to $-\infty$, making $\theta = 0$ a local minimum and at $\theta = 1$, the second term would tend to $-\infty$, making $\theta = 1$ a local minimum. Thus, $\theta = 0$ and $\theta = 1$ are not the maximum points in general).

b.

We are given 15 samples: [3.2, 1.4, 2.2, 7, 0.5, 3.3, 9, 0.15, 2, 3.21, 6.13, 5.5, 1.8, 1.2, 11] and we want to calculate the value of θ using the formula calculated in the first step. We have to first calculate the $\sum_{i=1}^{N} x_i$, which is equal to 57.59. So

$$\theta = \frac{30}{57.59 + 30} = \frac{30}{87.59} \approx 0.3425$$

Exercise 2 - Naïve Bayes (Theoretical)

Let's consider the Table 1 presenting a dataset composed of 7 samples, each presenting three Boolean variables x, y and z, and a corresponding Boolean classification U.

x	**y**	**z**	**U**
1	1	1	0
0	1	1	0
0	0	1	0
1	0	0	1
0	0	1	1
0	1	0	1
1	1	0	1

Table 1: Training dataset

In order to train a Naïve Bayes classifier, we will construct the frequency tables of our training data:

Frequency Table

	U	
x	0	1
0	2	2
1	1	2

Frequency Table

	U	
y	0	1
0	1	2
1	2	2

Frequency Table

	U	
z	0	1
0	0	3
1	3	1

# samples = 7, #0 = 3, #1 = 4

We can see that there is a zero in the frequency table of **z**. In this case, we should apply the Laplace trick by adding 1 to cell of all the frequency tables. However, for the purposes of this exercise, we will ignore this issue.

Now, let's construct the likelihood tables by normalizing the frequency tables towards the respective number of data we have:

Likelihood Table

	U	
x	0	1
0	$\frac{2}{3}$	$\frac{2}{4}$
1	$\frac{1}{3}$	$\frac{2}{4}$

Likelihood Table

	U	
y	0	1
0	$\frac{1}{3}$	$\frac{2}{4}$
1	$\frac{2}{3}$	$\frac{2}{4}$

Likelihood Table

	U	
z	0	1
0	$\frac{0}{3}$	$\frac{3}{4}$
1	$\frac{3}{3}$	$\frac{1}{4}$

And the prior probabilities are: $P(0) = \frac{3}{7}$ and $P(1) = \frac{4}{7}$

Now, our Naïve Bayes Classifier (NBC) is ready for classification.

a.

The given predicted probability will be

$$\begin{aligned} P(U = 0|x = 0, y = 0, z = 1) &= P(x = 0, y = 0, z = 1|U = 0)P(U = 0) \\ &= P(x = 0|U = 0)P(y = 0|U = 0)P(z = 1|U = 0)P(U = 0) \\ &= \frac{2}{3}\frac{1}{3}\frac{3}{3}\frac{3}{7} = \frac{2}{21} \approx 0.0952 \end{aligned}$$

This is the only probability Naïve Bayes Classifier wants for $U = 0$ in order to classify the specific combination of x, y and z. Now let's calculate the probability of this exact combination classified in $U = 1$:

$$\begin{aligned}
P(U = 1|x = 0, y = 0, z = 1) &= P(x = 0, y = 0, z = 1|U = 1)P(U = 1) \\
&= P(x = 0|U = 1)P(y = 0|U = 1)P(z = 1|U = 1)P(U = 1) \\
&= \frac{2}{4}\frac{2}{4}\frac{1}{4}\frac{4}{7} = \frac{1}{28} \approx 0.0357
\end{aligned}$$

We can see that $P(U = 0|x = 0, y = 0, z = 1) > P(U = 1|x = 0, y = 0, z = 1)$, so the combination $x = 0, y = 0, z = 1$ will be classified to class $U = 0$.

However, in order to calculate the final probability $P(U = 0|x = 0, y = 0, z = 1)$, we have to divide it by the total probability. So:

$$P(U = 0|x = 0, y = 0, z = 1) \approx \frac{0.0952}{0.0952 + 0.0357} \approx \frac{0.0952}{0.1309} \approx 0.7272$$

b.

We didn't need to exploit the Laplace trick to solve question a. because the combination of $x = 0, y = 0, z = 1$ appears for both outcomes of U (0 and 1). Therefore, no zero-probability event occurs for this combination of our variables.
However, if we try to solve for $P(U = 0|x = 0, y = 0, z = 0)$, the Naïve Bayes Classifier formula fails. This is because the probability will be zero, and when attempting to classify which class the combination belongs to, it will result in division by zero, leading to infinite values.

$$\begin{aligned}
P(U = 0|x = 0, y = 0, z = 0) &= P(x = 0, y = 0, z = 0|U = 0)P(U = 0) \\
&= P(x = 0|U = 0)P(y = 0|U = 0)P(z = 0|U = 0)P(U = 0) \\
&= \frac{2}{3}\frac{1}{3}\frac{0}{3}\frac{3}{7} = 0
\end{aligned}$$

c.

Using the probabilities obtained during the Bayes Classifier training, the predicted probability $P(U = 1|x = 0)$ will be:

$$P(U = 1|x = 0) = \frac{P(x = 0|U = 1)P(U = 1)}{P(x = 0|U = 1)P(U = 1) + P(x = 0|U = 0)P(U = 0)} = \frac{\frac{2}{4}\frac{4}{7}}{\frac{2}{4}\frac{4}{7} + \frac{2}{3}\frac{3}{7}} = \frac{\frac{2}{7}}{\frac{4}{7}} = \frac{1}{2}$$

Exercise 3 - Naïve Bayes Classifier (Programming)

a.

I implemented the training function of the Naïve Bayes Classifier (NBC) for categorical data by breaking it into small steps. First, I calculate the prior probabilities, then the frequency tables, and finally the likelihood tables. The model is represented as a Python dictionary that includes the likelihood table matrices (a dictionary containing the likelihood matrices for each feature) of and the prior probability vector.
If the data is continuous, the training process becomes much simpler since we only need to calculate the means and variances for each feature of every class. In that case, the model is a Python dictionary consisting of the mean matrix, the standard deviation matrix, and the prior probability vector.

b.

The prediction function of the Naïve Bayes Classifier (NBC) is quite straightforward. I used the formulas discussed in the recitation for both categorical and continuous data to calculate the probabilities and assign each sample in the test set to the class with the highest probability.

c. d.(BONUS)

In my experiments, I used different values of the hyperparameter L (the strength of smoothing) as powers of two:

$$L = [0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]$$

I evaluated my classifier using the accuracy metric, averaging the results over 500 runs. I also randomly split each dataset into two parts for each run: 75% of the samples were used as the training set, and the remaining 25% as the test set. In Figure 1, we can see how L affects the mean accuracy of the classifier trained on categorical data.

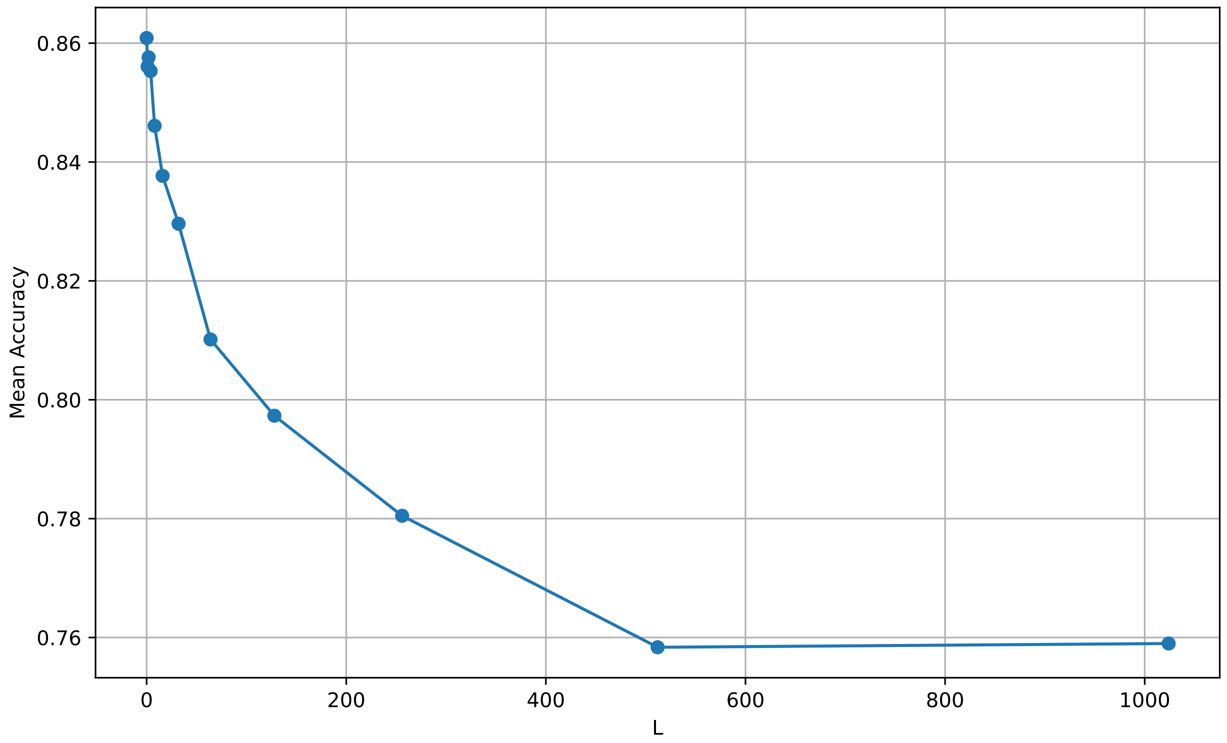

Figure 1: Mean Accuracy vs L averaged over 500 runs with categorical data.

When $L = 0$, meaning we are essentially using Maximum Likelihood Estimation (MLE) to estimate our probabilities, the mean accuracy reaches its highest value, around 86%. This accuracy is quite good, considering that the dataset is not heavily imbalanced. Specifically, 53.5% of the samples in the training set belong to class 0, while 46.5% belong to class 1. The baseline accuracy for an imbalanced dataset is often simply the accuracy achieved by predicting the majority class for all instances. In our example of a binary classification, the baseline accuracy would be 53.5%. As L increases, the mean accuracy decreases. That means that when $L = 0$, the model fits the data closely and performs well. As L increases, the model becomes more generalized, which can degrade performance as it no longer captures the nuances in the data, thus decreasing accuracy.
On the other hand, when working with continuous data the strength of smoothing does not affect the mean accuracy that much. In Figure 2 we can see how L affects the mean accuracy of the classifier trained on continuous data.

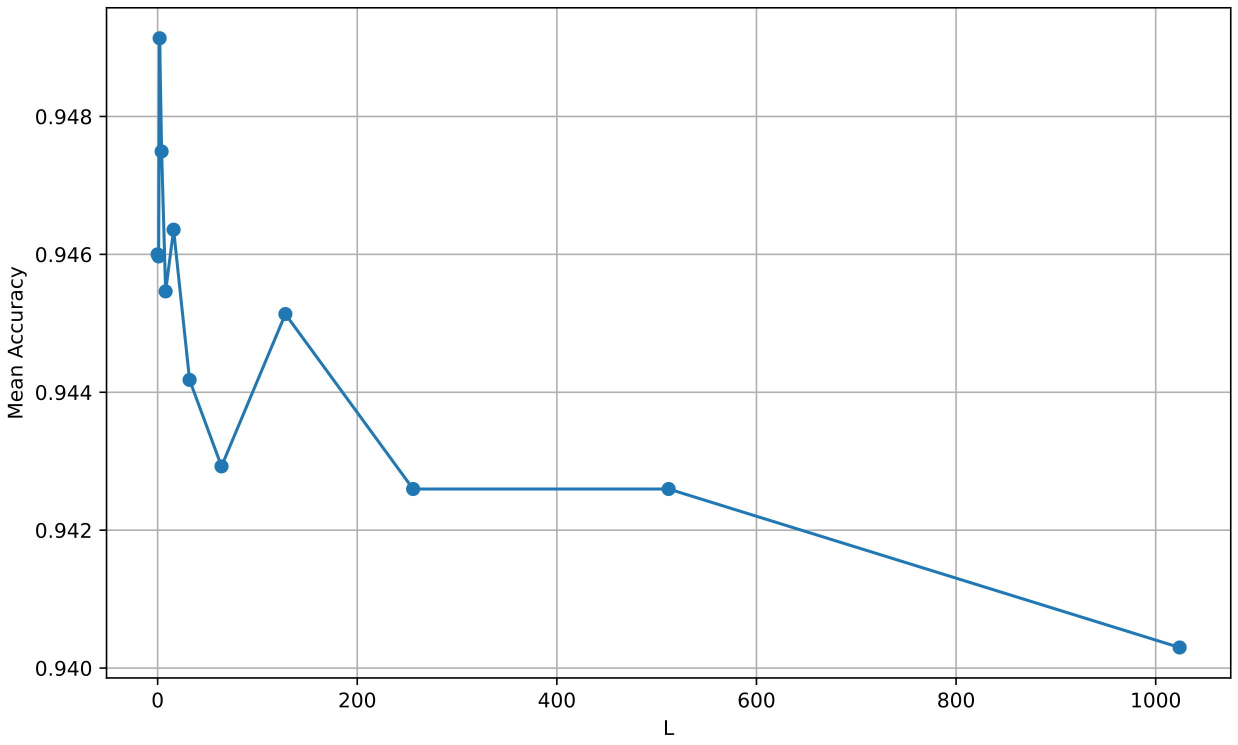

Figure 2: Mean Accuracy vs L averaged over 500 runs with continuous data.

Although we can see that the mean accuracy decreases as L increases, the decrease is relatively small, only about 0.007. My guess is that when the data are continuous, using a smoothing term does not significantly improve the model's fit to the data. In this case, it becomes unnecessary and could be omitted from the implementation. However, for the sake of this exercise, I did not modify my code and continued including L in my formulas. The highest mean accuracy occurs when L is small, reaching approximately 95%. The same applies to this dataset as well, since it is not heavily imbalanced, as I mentioned earlier.