Machine Learning: Exercise Set II

Georgios Manos
csd4333

20 October 2022

1 Probabilities

1.1 Theoretical

Maximum Likelihood Estimation for theta can be calculated as follows, since our N samples are independent and identically distributed:

$$\theta_{mle} = argmax_\theta P(x_1, x_2, ..., x_N|\theta) = argmax_\theta \prod_{i=1}^{N} P(x_i|\theta) = argmax_\theta \prod_{i=1}^{N} \theta^2(x_i+1)(1-\theta)^{x_i}$$

Essentially we are trying to find the θ value that maximizes $\prod_{i=1}^{N} \theta^2(x_i+1)(1-\theta)^{x_i}$. We can do so by recalling some basic calculus knowledge and find where the derivative of that expression w.r.t. to θ is 0.

The problem with that expression is that its hard to equate its derivative to 0, so instead we will maximize the log likelihood:

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

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

So, the expression describing MLE estimators for θ for N samples is:

$$L'(\theta) = \frac{d}{d\theta} \sum_{i=1}^{N} 2log(\theta) + log(x_i+1) + x_i log(1-\theta) = \sum_{i=1}^{N} \frac{2}{\theta} - \frac{x_i}{1-\theta} \implies$$

$$L'(\theta) = \frac{2N}{\theta} - \frac{\sum_{i=1}^{N} x_i}{1-\theta} = 0 \underset{\theta<1}{\implies} \frac{2N(1-\theta)}{\theta} = \sum_{i=1}^{N} x_i \implies \frac{1}{\theta} = 1 + \frac{1}{2N}\sum_{i=1}^{N} x_i$$

$$\underset{\theta>0}{\implies} \theta = \frac{1}{1+\frac{1}{2N}\sum_{i=1}^{N} x_i},\ \theta \in (0,1)$$

Finally, to make sure that the resulting θ_{mle} value is the local maximum, we need to assert that $\lim_{\theta\to\theta_{mle}^+} L(\theta) < 0$ and $\lim_{\theta\to\theta_{mle}^-} L(\theta) > 0$ respectively. To do that, we can use the 2nd derivative of $L(\theta)$ w.r.t. θ, and if it is negative then indeed that θ value is the maximum.

1.2 Need for Calculator

Using a calculator, the sum of those 15 samples is equal to 57,59. So, θ is:

$$\theta = \frac{1}{1 + \frac{57{,}59}{30}} \approx \frac{1}{2.919666} \approx 0.3425$$

2 Naive Bayes (Theoretical)

2.1 All Zero

First, lets start off with our beloved frequency tables, for each variable independently:

Frequency Table		U	
		0	1
x	0	2	2
	1	1	2

Frequency Table		U	
		0	1
y	0	1	2
	1	2	2

Frequency Table		U	
		0	1
z	0	0	3
	1	3	1

We have zeros, but for the purpose of this assignment we won't mess around with them this time, so I'll just ignore them.

The next step is to normalise the tables towards the respective number of data we have, so we end up with the following likelihood tables:

Frequency Table		U	
		0	1
x	0	2/3	2/4
	1	1/3	2/4

Frequency Table		U	
		0	1
y	0	1/3	2/4
	1	2/3	2/4

Frequency Table		U	
		0	1
z	0	0/3	3/4
	1	3/3	1/4

We have the following stats:

- #samples = 7
- #0 class = 3
- #1 class = 4
- $\implies P(U=0) = \frac{3}{7},\ P(U=1) = \frac{4}{7}$

Finally, the given probability using Naive Bayes comes out as:

$$P(U=0|x=0,y=0,z=1) = P(U=0)P(x=0|U=0)P(y=0|U=0)P(z=1|U=0)$$

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

The algorithm when classifying would actually stop right there and calculate the probability for the other class of U, then pick the class with the max probability. If we were to do all the maths on paper, to get the actual probability, we would need to divide this result with this probability and the probability for the other class of U:

$$P(U=1|x=0,y=0,z=1) = P(U=1)P(x=0|U=1)P(y=0|U=1)P(z=1|U=1)$$

$$\implies P(U=1|x=0,y=0,z=1) = \frac{4}{7}\frac{2}{4}\frac{2}{4}\frac{3}{4} = 0.10714$$

So:

$$P(U=0|x=0,y=0,z=1) \approx \frac{0.0952}{0.0952+0.10714} = \frac{0.0952}{0.20234} \approx 0.47048$$

2.2 Skipping Laplace

Like mentioned above, since we didn't have a case where $z=0$, we didn't have to use Laplace smoothing. If we had to predict on a scenario where $z=0$, such as $P(U=0|x=0,y=0,z=0)$, we would have the following problem:

$$P(U=0|x=0,y=0,z=1) = P(U=0)P(x=0|U=0)P(y=0|U=0)P(z=0|U=0)$$

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

which means its impossible, and we don't want to have impossible scenarios for our model. But more importantly, our classifier would fail when it would try to compare the probabilities with the other class:

$$P(U=1|x=0,y=0,z=0) = P(U=1)P(x=0|U=1)P(y=0|U=1)P(z=0|U=1)$$

$$\implies P(U=1|x=0,y=0,z=0) = \frac{4}{7}\frac{2}{4}\frac{2}{4}\frac{3}{4} = \frac{12}{112} \approx 0.10714$$

So:

$$\frac{P(U=1|x=0,y=0,z=0)}{P(U=0|x=0,y=0,z=0)} = \frac{0.10714}{0} \to \infty$$

2.3 Another Prediction

Since on this case, variables y and z can be anything, the predicted probability through Naive Bayes algorithm would be:

$$P(U = 1|x = 0) = P(U = 1)P(x = 0|U = 1) = \frac{4}{7}\frac{2}{4} = \frac{2}{7} \approx 0.2857$$

Again, to get the actual probability on paper we need to follow the same steps as above:

$$P(U = 0|x = 0) = P(U = 0)P(x = 0|U = 0) = \frac{3}{7}\frac{2}{3} \approx 0.2857$$

So the actual probability is:

$$P(U = 1|x = 0) = \frac{0.2857}{0.2857 + 0.2857} = 0.5$$

3 Naive Bayes Classifier (Programming)

3.1 Setting Up

To run the code, you need to have installed numpy, pandas and tqdm libraries.

Opening the file, you will see the configurations on top as global variables, from the data filepaths to the keys used in dictionaries, train set percentage, how many times to repeat training, if you want to print the 1st model in JSON-Like format and the smooth strength (L parameter).

3.2 Implementation Comments

I implemented all the requested functions. The model structure was a bit difficult to decide on, but I followed a solid structure that works both on categorical and continuous variables input. Pretty much it is a dictionary of dictionaries, hence the json-like formatted printing. The detailed javadoc strings explain the structure, but by simply running the code you can also see the format printed live. Finally, the parameter PLOT SMOOTH STR(ENGTH) EXP(ERIMENTS) corresponds to if you want to see the plots I added when explaining L parameter.

But enough with the code stuff, lets move to the results! ☺

3.3 Results

3.3.1 Categorical

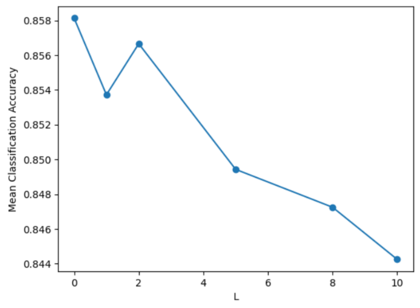
My implementation on the given discrete variable data gets a mean classification accuracy of 0.8537 on 100 runs. Given that the class balance of the training set was 0.5343 vs 0.4656 for class 0 and 1 respectively (in full set it is 0.555 for class 0 and 0.445 for class 1), this is a pretty good performance since its much better than random guessing on the majority class (0). Note that this occurs for MLE estimators, meaning that the Smoothing Strength was 0.

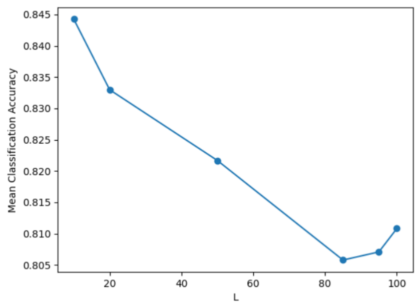
Toying around with the value of L, I got the following mean accuracy for 100 runs:

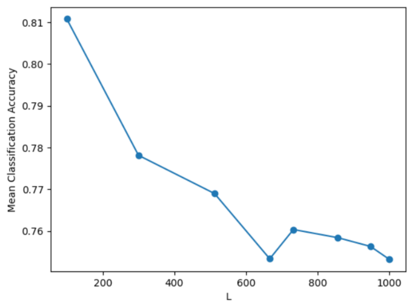
- $L = 1 \implies mean(acc) = 0.8566$
- $L = 10 \implies mean(acc) = 0.8442$
- $L = 100 \implies mean(acc) = 0.8108$
- $L = 1000 \implies mean(acc) = 0.7532$

Respectively, to show this increasing correlation, I made a couple of plots to see how the Mean Classification Accuracy drops while the Smoothing strength increases. The lines would probably be more straight downwards if we had more repeats as the mean accuracy resulting from the experiments varies for even the same L value, due to the stochasticity added from random sampling (train - test split). For the purpose of this experiment, I increased the number of repeats to 200.

The difference between the plots is the L range scale. On the first one $L \in [0, 10]$, then $L \in [10, 100]$ and finally $L \in [100, 1000]$.







Essentially, this happens as L is a paramteter that attempts to generalize the model by balancing out the probabilities, for potential new data. On small

L values, we don't see a significant impact on the average accuracy, while for bigger Ls we see a big drop. This happens due to the way this dataset is - if we had a few samples with different values than 0 and 1 in the whole dataset, we would probably see an improvement on the accuracy if they were caught in the test set instead of the training one.

3.3.2 Continuous

On continuous variables, the classifier's accuracy is at 0.943. This is also pretty good compared to the random choice, given that the class balance was 0.553 and 0.446 (on full dataset, 0.555 vs 0.445 for class 0 and 1 respectively). This is more straightforward as we have less parameters, so no fancy plots this time! ☺

4 Appendix

I love Paris!