Assignment 2

Type	Homework
Materials	
Reviewed	☐
Date	@September 8, 2022
Topic	General

1: Independent Events and Bayes Theorem

1.

We aim to prove that

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\neg A)P(\neg A)}$$

First, from the definition of conditional probability, $P(A|B) = \frac{P(A \cap B)}{P(B)}$ and $P(B|A) = \frac{P(B \cap A)}{P(A)}$. Note that the numerator of both expressions is the same, as the intersection of sets is commutative. Therefore,

$$P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

Using the right two terms above:

$$P(A|B)P(B) = P(B|A)P(A)$$
$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

The denominator of the above statement can be transformed. $A \cup \neg A$contains all events, as the union of an event with its compliment contains all events.

Therefore, $P(B) = P(A \cap B) + P(\neg A \cap B)$. From the definition of conditional probability, this can be written as $P(B|A)P(A) + P(B|\neg A)P(\neg A)$. If we replace $P(B)$ in the denominator of the above expression, we are left with

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\neg A)P(\neg A)}$$

This is what we aimed to prove. ■

2a

If X is independent of Y, then $P(B \cap A) = P(B)P(A)$. In this example, we find the chance of each event by summing all of the probabilities of outcomes included that event.

$P(X \cap Y) = 0.1 + 0.175 = 0.275$

$P(X) = 0.05 + 0.1 + 0.1 + 0.175 = 0.425$

$P(Y) = 0.2 + 0.1 + 0.175 + 0.175 = 0.65$

$P(X)P(Y) = 0.27625 \neq 0.275 = P(X \cap Y)$

Therefore, X and Y are not independent.

2b

To find conditional independence of X and Y, given Z, we use the same process as above, while summing only the outcomes where Z=1. We also divide each by the probability that Z occurs, which is $P(Z) = 0.1 + 0.1 + 0.175 + 0.175 = 0.55$

$P(X \cap Y|Z) = \frac{P(X \cap Y)}{P(Z)} = \frac{0.175}{0.55} = 0.31818$

$P(X|Z) = \frac{0.1+0.175}{0.55} = \frac{0.275}{0.55} = 0.5$

$P(Y|Z) = \frac{0.175+0.175}{0.55} = \frac{0.35}{0.55} = 0.6363636$

$P(X|Z)P(Y|Z) = 0.31818 = P(X \cap Y|Z)$

Therefore, X is conditionally independent of Y given Z.

2c

First, we calculate that $P(Z = 0) = 0.45$, which we get by summing the probability of all outcomes where Z = 0.

$P(X \neq Y|Z = 0) =$
$P(X = 0, Y = 1|Z = 0) + P(X = 1, Y = 0|Z = 0)$

Now, we compute the two probabilities.

$P(X = 0, Y = 1|Z = 0) = \frac{P(X' \cap Y \cap Z)}{P(Z=0)} = \frac{0.2}{0.45} = \frac{4}{9}$

$P(X = 1, Y = 0|Z = 0) = \frac{P(X \cap Y' \cap Z)}{P(Z=0)} = \frac{0.05}{0.45} = \frac{1}{9}$

Therefore, $P(X \neq Y|Z = 0) = \frac{4}{9} + \frac{1}{9} = \frac{5}{9}$

2: Maximum Likelihood Estimation

1

The log likelihood function is
$l(\hat{\theta}) = \log L(\hat{\theta}) = \log(P(X_1, ..., X_n|\hat{\theta})$

Because we are told that a value is drawn from a single Bernoulli distribution with parameter θ, and we know that X_i can only have the values of 0 or 1, then we can define

$$P(X_i|\hat{\theta}) = \hat{\theta}^{X_i}(1-\hat{\theta})^{1-X_i}$$

The likelihood of all of the values being drawn is the product of all of the probabilities of each event occurring, as we consider the events independent.

$$L(\hat{\theta}) = P(X_1, ..., X_n|\hat{\theta}) = \prod_{i=1}^{n} \hat{\theta}^{X_i}(1-\hat{\theta})^{1-X_i}$$

Therefore, the log likelihood function is

$$l(\hat{\theta}) = \log\left(\prod_{i=1}^{n} \hat{\theta}^{X_i}(1-\hat{\theta})^{1-X_i}\right)$$

While the above equation is correct, we can simplify further using log properties.

$$l(\hat{\theta}) = \sum_{i=1}^{n} \log(\hat{\theta}^{X_i}(1-\hat{\theta})^{1-X_i})$$

$$= \sum_{i=1}^{n} \Big[X_i \log(\hat{\theta}) + (1-X_i)\log(1-\hat{\theta})\Big]$$

Because the log likelihood is a sum, the order of the random variables will not matter, as addition is commutative.

2

In order to find the maximum likelihood estimate, we first find the derivative of the above function.

$$l'(\hat{\theta}) = \frac{\delta}{\delta\hat{\theta}} \left(\sum_{i=1}^{n} X_i \log(\hat{\theta}) + (1 - X_i) \log(1 - \hat{\theta}) \right)$$

Now, we calculate the partial derivative on the right-hand side.

$$\begin{aligned}
l'(\hat{\theta}) &= \sum_{i=1}^{n} X_i \cdot \frac{1}{\hat{\theta}} - (1 - X_i) \frac{1}{(1 - \hat{\theta})} \\
&= \sum_{i=1}^{n} \frac{X_i}{\hat{\theta}} - \frac{1 - X_i}{1 - \hat{\theta}} \\
&= \sum_{i=1}^{n} \frac{X_i(1 - \hat{\theta})}{\hat{\theta}(1 - \hat{\theta})} - \frac{\hat{\theta}(1 - X_i)}{\hat{\theta}(1 - \hat{\theta})} \\
&= \sum_{i=1}^{n} \frac{X_i(1 - \hat{\theta}) - \hat{\theta}(1 - X_i)}{\hat{\theta}(1 - \hat{\theta})} \\
&= \sum_{i=1}^{n} \frac{X_i - X_i\hat{\theta} - \hat{\theta} + \hat{\theta}X_i}{\hat{\theta}(1 - \hat{\theta})} \\
&= \sum_{i=1}^{n} \frac{X_i - \hat{\theta}}{\hat{\theta}(1 - \hat{\theta})}
\end{aligned}$$

Using this notation, we can make an important observation. X_i is always either the value 0 or 1. When X_i is 1, the term after the sum will be $\frac{1}{\hat{\theta}}$, and when X_i is 0, the term will be $-\frac{1}{1-\hat{\theta}}$. This observation makes it much easier to compute the best value of $\hat{\theta}$. The sample data has 6 1s, and 4 0s. Therefore, the sum is

$$6 \cdot \frac{1}{\hat{\theta}} - 4 \cdot \frac{1}{1 - \hat{\theta}}$$

We want to find the value of $\hat{\theta}$ when this expression = 0

$$
\begin{aligned}
\frac{6}{\hat{\theta}} - \frac{4}{1-\hat{\theta}} &= 0 \\
\frac{6(1-\hat{\theta}) - 4\hat{\theta}}{\hat{\theta}(1-\hat{\theta})} &= 0 \\
\frac{6 - 6\hat{\theta} - 4\hat{\theta}}{\hat{\theta}(1-\hat{\theta})} &= 0 \\
\frac{6 - 10\hat{\theta}}{\hat{\theta}(1-\hat{\theta})} &= 0
\end{aligned}
$$

This function is zero when the numerator is 0 and the denominator is not zero. Therefore, the function is zero at $\hat{\theta} = 0.6$.

Because $l'(0.6) = 0$, it must be a local maximum or minimum for $l(\hat{\theta})$. Recall that $\hat{\theta}$ must be between 0 and 1. Therefore, the maximum and minimum for $l(\hat{\theta})$ must fall at either 0, 1, or 0.6. We will now calculate these values. (Done programmatically to save time)

```python
SAMPLES = np.array((0, 1, 0, 1, 1, 0, 0, 1, 1, 1))
def likelihood(sample, theta):
    """Return the likelihood of a sample given a probability theta."""
    prod = np.prod(theta**sample * (1-theta)**(1-sample))
    return np.log(prod)

print(f"0: {likelihood(SAMPLES, 0)}")
print(f"0.6: {likelihood(SAMPLES, 0.6)}")
print(f"1: {likelihood(SAMPLES, 1)}")
```

[15] ✓ 0.3s Python

```
0: -inf
0.6: -6.730116670092564
1: -inf

C:\Users\matth\AppData\Local\Temp\ipykernel_10432\2755395218.py:5:
RuntimeWarning: divide by zero encountered in log
  return np.log(prod)
```

From these results, we can see that a maximum occurs at 0.6. Therefore, the maximum likelihood estimate $\hat{\theta}^{\text{MLE}} = 0.6$.

3

As before, the log likelihood function is
$l(\hat{\theta}) = \log L(\hat{\theta}) = \log(P(X_1, ..., X_n|\hat{\theta})$

where $P(Y_1, ..., Y_m|\hat{\theta})$ is the probability of seeing the m i.i.d. random variables $Y_1, ..., Y_m$ when drawn from a Binomial distribution with $B(n, \theta)$.

First, we look to define $P(Y_1, ..., Y_m|\hat{\theta})$. Because it is drawn from a Binomial distribution,

$$P(Y_i = k) = \frac{n!}{k!(n-k)!} \cdot \hat{\theta}^k (1-\hat{\theta})^{n-k}$$

Because we are told that $Y_i = k$, we can make that substitution in the right-hand side.

$$P(Y_i = k) = \frac{n!}{Y_i!(n-Y_i)!} \cdot \hat{\theta}^{Y_i} (1-\hat{\theta})^{n-Y_i}$$

Now, since we assume the random variables to be independent, $P(Y_1, ...Y_m|\hat{\theta})$ is the product of $P(Y_i|\hat{\theta})$ for all i from 1 to m.

$$P(Y_1, ...Y_m|\hat{\theta}) = \prod_{i=1}^{m} \frac{n!}{Y_i!(n-Y_i)!} \cdot \hat{\theta}^{Y_i} (1-\hat{\theta})^{n-Y_i}$$

Now, we can create an expression for the log likelihood.

$$l(\hat{\theta}) = \log L(\hat{\theta}) = \log(P(Y_1, ..., Y_m|\hat{\theta}))$$
$$= \log \left[\prod_{i=1}^{m} \frac{n!}{Y_i!(n-Y_i)!} \cdot \hat{\theta}^{Y_i} \cdot (1-\hat{\theta})^{n-Y_i} \right]$$

Now, we simplify using log properties.

The product becomes a sum, and the terms multiplied together becomes sums.

$$l(\hat{\theta}) = \sum_{i=1}^{m} \log \left(\frac{n!}{Y_i!(n-Y_i)!} \right) + \log(\hat{\theta}^{Y_i}) + \log(1-\hat{\theta})^{n-Y_i}$$

Now, exponents are moved before the logs, and the fraction on the left is changed to subtraction

$$l(\hat{\theta}) = \sum_{i=1}^{m} \log(n!) - \log(Y_i!(n-Y_i)!) + Y_i \log(\hat{\theta}) + (n-Y_i) \log(1-\hat{\theta})$$
$$l(\hat{\theta}) = \sum_{i=1}^{m} \log(n!) - \log(Y_i!) - \log((n-Y_i)!) + Y_i \log(\hat{\theta}) + (n-Y_i) \log(1-\hat{\theta})$$

4

To find the maximum likelihood estimate, we need to find a value of $\hat{\theta}$ that maximizes $l(\hat{\theta})$. To do that, we find the first derivative $l'(\hat{\theta})$, then find the values of $\hat{\theta}$ where $l'(\hat{\theta}) = 0$. This give local extrema, one of which is likely the maximum.

$$l'(\hat{\theta}) = \frac{\delta}{\delta\hat{\theta}} \left[\sum_{i=1}^{m} \log(n!) - \log(Y_i!) - \log((n - Y_i)!) + Y_i \log(\hat{\theta}) + (n - Y_i) \log(1 - \hat{\theta}) \right]$$

When we differentiate with respect to $\hat{\theta}$, all terms without a $\hat{\theta}$ drop out. The sum remains the same due to normal rules of differentiation.

$$l'(\hat{\theta}) = \frac{\delta}{\delta\hat{\theta}} \left[\sum_{i=1}^{m} Y_i \log(\hat{\theta}) + (n - Y_i) \log(1 - \hat{\theta}) \right]$$

$$\begin{aligned} l'(\hat{\theta}) &= \sum_{i=1}^{m} \frac{Y_i}{\hat{\theta}} - \frac{(n - Y_i)}{(1 - \hat{\theta})} \\ &= \sum_{i=1}^{m} \frac{Y_i - Y_i\hat{\theta}}{\hat{\theta}(1 - \hat{\theta})} - \frac{n\hat{\theta} - Y_i\hat{\theta}}{\hat{\theta}(1 - \hat{\theta})} \\ &= \sum_{i=1}^{m} \frac{Y_i - n\hat{\theta}}{\hat{\theta}(1 - \hat{\theta})} \end{aligned}$$

For this problem, we know $n = 5$, so we can make that substitution. We now look to determine what $\hat{\theta}$ makes this expression 0.

$$\sum_{i=1}^{m} \frac{Y_i - 5\hat{\theta}}{\hat{\theta}(1 - \hat{\theta})} = 0$$

We only have 2 Y values, so it is easy to write the whole left-hand side without the sigma.

$$\frac{3 - 5\hat{\theta}}{\hat{\theta}(1 - \hat{\theta})} + \frac{3 - 5\hat{\theta}}{\hat{\theta}(1 - \hat{\theta})} = 2 \cdot \frac{3 - 5\hat{\theta}}{\hat{\theta}(1 - \hat{\theta})} = 0$$

We can easily find the solution to this equation by finding values of $\hat{\theta}$ where the numerator is equal to 0 and the denominator is not equal to 0. This occurs at $\hat{\theta} = 0.6$.

Because $l'(0.6) = 0$, it must be a local maximum or minimum for $l(\hat{\theta})$. Recall that $\hat{\theta}$ must be between 0 and 1. Therefore, the maximum and minimum for $l(\hat{\theta})$ must fall at either 0, 1, or 0.6.

We will now calculate these values. (Done programmatically to save time)

```
SAMPLES = np.array((3,3), dtype = int)
def likelihood(sample, theta, n=5):
    """Return the likelihood of a sample given a probability theta."""
    choose_term = sp.special.factorial(n) / (sp.special.factorial(sample) * sp.special.factorial(n-sample))
    prod = np.prod(choose_term * theta**sample * (1-theta)**(n-sample))
    return np.log(prod)
print(f"0: {likelihood(SAMPLES, 0)}")
print(f"0.6: {likelihood(SAMPLES, 0.6)}")
print(f"1: {likelihood(SAMPLES, 1)}")
```

[9] ✓ 0.1s

```
0: -inf
0.6: -2.124946484104473
1: -inf

C:\Users\matth\AppData\Local\Temp\ipykernel_33156\4089486007.py:6: RuntimeWarning: divide by zero encountered in log
  return np.log(prod)
```

From these results, we can see that a maximum occurs at 0.6. Therefore, the maximum likelihood estimate $\hat{\theta}^{\text{MLE}} = 0.6$.

5

My log likelihood function for part 1 and part 3 were very similar. This is unsurprising, as the definition for binomial distribution contains the definition for the Bernoulli distribution with the addition of the “choose” term. After the logarithm was applied to both, they both ended up as a sum, and the last two terms were almost identical, but part 3 had $(n - Y_i)$ and part 1 had $(1 - X_i)$. This is purely a feature of Y_i being able to be any number between 0 and n, whereas X_i was either 1 or 0.

In parts 2 and 4, I was left with the same value for $\hat{\theta}^{MLE}$. This was in large part due to the derivative with respect to $\hat{\theta}$ being almost equal for both values. In Part 4, the “choose” terms dropped out of the sum as they did not contain $\hat{\theta}$ in them. Again, the only difference between the two is the extra n in the numerator of Part 4 (due to Y_i having values between 0 and n and X_i having values 0 or 1). Both parts having the same answer does make logical sense. In part 2, the samples are (0, 1, 0, 1, 1, 0, 0, 1, 1, 1), and we are told in part 4 that “Y_1 and Y_2 resulted in (0, 1, 0, 1, 1) and (0, 0, 1, 1, 1) respectively.” When you concatenate the values from part 4, you get the same samples as part 1. It is clear that the same parameter for the Bernoulli distribution would be the most likely estimate for both.

This also provides some extra evidence to the claim that order does not matter for Part 1. In the calculation for part 4, the order of values pulled from the Bernoulli distribution was never used, only the fact that 3 of the 5 were 1. Y_1 and Y_2 could have come from (1, 1, 1, 0, 0) and (1, 1, 1, 0, 0) respectively, and the same answer would have been reached.

3: Implementing Naive Bayes

In order to implement Naive Bayes, I used scikit learn's `MultiNomialNB` and `ComplementNB` . The full documentation for the classifiers can be found at https://scikit-learn.org/stable/modules/naive_bayes.html .

All classifiers make the naive assumption of conditional independence between each pair of features given the label. Both use MAP under-the-hood. Additionally, both models take a parameter α that smooths the data to account for features not present in the data. I chose to set $\alpha = 1$

I selected `MultinomialNB` specifically because the documentation indicated that it performs well on data represented as word vector counts.

I then tried `ComplementNB` as it is a variation of `MultinomialNB` that uses the complement and argmin to calculate the weights. The documentation also said that this method regularly outperforms the above method, so it was worth a try.

See below for screenshots of how each model performed. The source itself will also be attached as a jupyter notebook.

Here are the results for MultinomialNB

```
model = MultinomialNB(alpha=1)

train_time_start = time.time()
model = model.fit(X_train, Y_train)
train_time_end = time.time()
print(f"training took {train_time_end - train_time_start:.4f} seconds")

print(f"training accuracy is {model.score(X_train, Y_train):.5f}")
print(f"test accuracy is {model.score(X_test, Y_test):.5f}")
```

✓ 0.4s

```
training took 0.1112 seconds
training accuracy is 0.96930
test accuracy is 0.98228
```

Here are the results for ComplementNB:

```
    model = ComplementNB(alpha=1)

    train_time_start = time.time()
    model = model.fit(X_train, Y_train)
    train_time_end = time.time()
    print(f"training took {train_time_end - train_time_start:.4f} seconds")

    print(f"training accuracy is {model.score(X_train, Y_train):.5f}")
    print(f"test accuracy is {model.score(X_test, Y_test):.5f}")
[9]  ✓ 0.4s
...  training took 0.1214 seconds
     training accuracy is 0.96907
     test accuracy is 0.98228
```

As we can see, both models took similar (very small) amounts of time to train. They ended up with training accuracies within 0.01% of each other, and identical test accuracy. With an accuracy of above 98%, both classifiers are very good at labeling data.