Jhirsc21

February 4, 2024

1 Module 2 Lab - Probability

1.1 General Instructions

In this course, Labs are the chance to applying concepts and methods discussed in the module. They are a low stakes (pass/fail) opportunity for you to try your hand at *doing*. Please make sure you follow the general Lab instructions, described in the Syllabus. The summary is:

- Discussions should start as students work through the material, first Wednesday at the start of the new Module week.
- Labs are due by **Sunday**.
- Lab solutions are released Monday.

- Post Self Evaluation and Lab to Lab Group on Blackboard and Lab to Module on Blackboard on **Monday**.

The last part is important because the Problem Sets will require you to perform the same or similar tasks without guidance. Problem Sets are your opportunity to demonstrate that you understand how to apply the concepts and methods discussed in the relevant Modules and Labs.

1.2 Specific Instructions

1. For Blackboard submissions, if there are no accompanying files, you should submit *only* your notebook and it should be named using *only* your JHED id: fsmith79.ipynb for example if your JHED id were "fsmith79". If the assignment requires additional files, you should name the *folder/directory* your JHED id and put all items in that folder/directory, ZIP it up (only ZIP...no other compression), and submit it to Blackboard.

   - do **not** use absolute paths in your notebooks. All resources should located in the same directory as the rest of your assignments.
   - the directory **must** be named your JHED id and **only** your JHED id.
   - do **not** return files provided by us (data files, .py files)

2. Data Science is as much about what you write (communicating) as the code you execute (researching). In many places, you will be required to execute code and discuss both the purpose and the result. Additionally, Data Science is about reproducibility and transparency. This includes good communication with your team and possibly with yourself. Therefore, you must show **all** work.

3. Avail yourself of the Markdown/Codecell nature of the notebook. If you don't know about Markdown, look it up. Your notebooks should not look like ransom notes. Don't make

everything bold. Clearly indicate what question you are answering.

4. Submit a cleanly executed notebook. The first code cell should say `In [1]` and each successive code cell should increase by 1 throughout the notebook.

[1]:
```
from pprint import pprint
```

1.3 Manipulating and Interpreting Probability

Given the following *joint probability distribution*, $P(A, B)$, for A and B,

	a1	a2
b1	0.37	0.16
b2	0.23	?

Answer the following questions.

1. What is $P(A = a2, B = b2)$?

We know all possible probabilities in a joint probablity table must add up to 1 (from the `Second Axiom`). Therefore, P(A = a2, B = b2) is equal to `1 - (0.37 + 0.16 + 0.23) = 0.24`

2. If I observe events from this probability distribution, what is the probability of seeing (a1, b1) then (a2, b2)?

Assuming all outcomes in the table are equally likely to occur and are independent from each other, the probability of seeing (a1, b1) followed by (a2, b2) is `(1/4) * (1/4) = (1/16)`

3. Calculate the marginal probability distribution, $P(A)$.

The marginal probability distribution `P(A = a1) = 0.60`, while the marginal probability distribution of `P(A = a2) = 0.40`. Here we simply add each column pertaining to each outcome for A.

4. Calculate the marginal probability distribution, $P(B)$.

Similarly, `P(B = b1) = 0.53`, while `P(B = b2) = 0.47`.

5. Calculate the conditional probability distribution, $P(A|B)$.

In order to find conditional probabilities, we can look at each row of B, and normalize each outcome involving A over the marginal probability of `P(B = b1)` or `P(B = b2)`. We can say that `P(A = a1 | B = b1) = 0.37/0.53 =  0.70`, `P(A = a2 | B = b1) = 0.30` and `P(A = a1 | B = b2) = 0.49`, `P(A = a2 | B = b2) = 0.51`

6. Calculate the conditional probability distribution, $P(B|A)$.

Similarly, we can say that `P(B = b1 | A = a1) = 0.62`, `P(B = b2 | A = a1) = 0.38` and `P(B = b1 | A = a2) = 0.40`, `P(B = b2 | A = a2) = 0.60`

7. Does $P(A|B) = P(B|A)$? What do we call the belief that these are always equal?

P(A|B) does not equal P(B|A), as we can see each conditional probability above. The belief that these are always equal is called the `Inverse Probability Fallacy`.

8. Does $P(A) = P(A|B)$? What does that mean about the independence of A and B?

No, in this case, P(A) does not equal P(A|B). I don't think this tells us anything about whether A and B are independent. We need to also include the joint probabilities for that conclusion.

9. Using $P(A)$, $P(B|A)$, $P(B)$ from above, calculate,

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

Does it match your previous calculation for $P(A|B)$?

`P(A = a1 | B = b1) = (0.62)(0.60) / (0.53) = 0.70` `P(A = a2 | B = b1) = (0.40)(0.40) / (0.53) = 0.30` `P(A = a1 | B = b2) = (0.38)(0.60) / (0.47) = 0.49` `P(A = a2 | B = b2) = (0.60)(0.40) / (0.47) = 0.51` Yes these are the same answers as the previous calculations.

10. If we let A = H (some condition, characteristic, hypothesis) and B = D (some data, evidence, a test result), then how do we interpret each of the following: $P(H)$, $P(D)$, $P(H|D)$, $P(D|H)$, $P(H, D)$?

P(H) - Probability of our hypothesis. Our prior probability in Bayes Rule.

P(D) - Probability that we get this result. Our normalizer in Bayes Rule.

P(H|D) - Probability of the hypothesis given we know D. Posterior probability in Bayes Rule.

P(D|H) - Probability of our data given the hypothesis. The likelihood in Bayes Rule.

P(H,D) - The joint probability between the hypthesis and data.

1.4 Bayes Rule

Bayes Rule will be an important part of our toolset in the weeks to come, especially in terms of Bayesian Inference. Work through the following problems in Bayes Rule.

1.4.1 Problem 1 (Regular)

You might be interested in the relationship between alcoholism and liver disease, in which case "being an alcoholic" (or not) is a test (evidence for) for liver disease (or not).

Let `D` be the presence or absence of liver disease (`d` they have it; `~d`, "not d", they don't). Past data tells you that 10% of patients entering your clinic have liver disease. Let `A` be alcoholic (`a`) or not alcoholic (`~a`). 5% of the clinic's patients are alcoholics.

You know that among those patients diagnosed with liver disease, 7% are alcoholics and among those without liver disease, 95.2% are non-alcoholics.

1. What is Bayes Rule for this problem? (write it out symbolically)
2. From the above word problem, what values of Bayes Rule do you have? Which ones are missing?
3. Calculate the missing values.
4. Calculate the posterior probability *distributions* using Bayes Rule.
5. Describe what each individual posterior probability means *in words.*

answer here (again, either use Markdown here or change to a code cell if you want to program the answer. Use Markdown cells for comments. I suggest calculating the missing information then using the code from Fundamentals to calculate the answers.)

1. In general, Bayes Rule can be summarized as `P(B|A) = P(A|B) P(B) / P(A)` In this case, we would say that `P(D|A) = P(A|D) P(D) / P(A)` In other words, `P(patient has liver disease | patient is an alcoholic) = P(patient is an alcoholic | patient has liver disease) * P(patient has liver disease) / P(patient is an alcoholic)`
2. The info we have is `P(D = d), P(A = a), P(A = a | D = d)`, and `P(A = ~a | D = ~d)`. We are missing the joint probabilities, the marginal probabilities `P(A = ~a)`, `P(D = ~d)`, and the conditional probabilities `P(A = ~a | D = d)`, `P(A = a | D = ~d)`, and the four where D is conditional given A.
3. We know that `P(D = ~d) = 0.90` and `P(A = ~a) = 0.95`, since the marginal probabilities need to equal one for D or A, respectively.

We know `P(A = a | D = d) = 0.07 = P(A = a, D = d) / P(D = d)`, so we can find `P(A = a, D = d)` as `0.07 * 0.10 = 0.007`. We can perform the similar calculations to find each of the four joint distributions, summarized in the table below:

	a	~a	
d	0.007	0.093	0.10
~d	0.0432	0.8568	0.90
	0.05	0.95	

Finally we can look at the conditional probabilities. `P(A = ~a | D = d) = 0.93` and `P(A = a | D = ~d) = 0.048` `P(D = d | A = a) = 0.14` `P(D = d | A = ~a) = 0.099` `P(D = ~d | A = a) = 0.864` `P(D = ~d | A = ~a) = 0.902`

4. The posterior probability distributions are given as `P(D|A) = P(A|D) P(D) / P(A)`. `P(D = d | A = a) = (0.07)(0.10) / (0.05) = 0.14` `P(D = ~d | A = a) = (0.048)(0.90) / (0.05) = 0.864` `P(D = d | A = ~a) = (0.93)(0.10) / (0.95) = 0.098` `P(D = ~d | A = ~a) = (0.952)(0.90) / (0.95) = 0.902`
5. `P(D = d | A = a)` is the probability a patient has liver disease, given they are an alcoholic. `P(D = ~d | A = a)` is the probability a patient does not have liver disease, given they are an alcoholic. `P(D = d | A = ~a)` is the probability a patient has liver disease, given they are not an alcoholic. `P(D = ~d | A = ~a)` is the probability a patient does not have liver disease, given they are not an alcoholic.

1.4.2 Problem 2 (Harder)

In a particular pain clinic, 10% of patients are prescribed narcotic pain killers. Overall, five percent of the clinic's patients are addicted to narcotics (including pain killers and illegal substances). Out of all the people prescribed pain pills, 8% are addicts. What is the relationship between addiction and pain pill prescriptions?

1. What is Bayes Rule for this problem? (write it out symbolically)
2. From the above word problem, what values of Bayes Rule do you have? Which ones are missing?
3. Calculate the missing values.
4. Calculate the posterior probability *distributions* using Bayes Rule.
5. Describe what each individual posterior probability means *in words*.

(Note: this problem is structured slightly differently than usual. You will need to use Total Probability and the Axioms of Probability as well as solving simultaneous equations to get the answer).

1. In general, Bayes Rule can be summarized as `P(B|A) = P(A|B) P(B) / P(A)` We can ask what the probability is that a patient is prescribed pain killers given they are an addict. In this case, we would say that

```
P(patient is prescribed pain killers | patient is addict = P(patient is addict
| patient is prescribed pain killers) * P(patient is prescribed pain killers) /
P(patient is addict)
```

2. If we say that `a` is addicts and `~a` is non-addicts, and `p` is prescribed pain killers, while `~p` is not prescribed pain killers, then the info we have is `P(P = p), P(A = a), P(A = a | P = p)`. We are missing the joint probabilities, the marginal probabilities `P(A = ~a)`, `P(P = ~p)`, and the conditional probabilities `P(A = a | P = ~p)`, `P(A = ~a | P = p)`, `P(A = ~a | D = ~p)` and the four where P is conditional given A.

3. We know that `P(P = p) = 0.10` and `P(A = a) = 0.05`, and `P(A = a | P = p) = 0.08`

We know `P(A = a | P = p) = 0.08 = P(A = a, D = d) / P(P = p)`, so we can find `P(A = a, D = d)` as `0.08 * 0.10 = 0.008`. We can perform the similar calculations to find each of the four joint distributions, summarized in the table below:

	p	~p	
a	0.008	0.042	0.05
~a	0.092	0.858	0.95
	0.10	0.90	

Finally we can look at the conditional probabilities. `P(A = a | P = ~p) = 0.047` and `P(A = ~a | P = p) = 0.92` `P(A = ~a | P = ~p) = 0.95` `P(P = p | A = a) = 0.16` `P(P = ~p | A = a) = 0.84` `P(P = p | A = ~a) = 0.0968` `P(P = ~p | A = ~a) = 0.903`

4. The posterior probability distributions are `P(P = p | A = a) = (0.08)(0.10) / (0.05) = 0.16  P(P = ~p | A = a) = (0.047)(0.90) / (0.05) = 0.846  P(P = p | A = ~a) = (0.92)(0.10) / (0.95) = 0.0968 P(P = ~p | A = ~a) = (0.95)(0.90) / (0.95) = 0.90`

5. `P(P = p | A = a)` is the probability a patient is prescribed pain killers, given they are an addict. `P(P = ~p | A = a)` is the probability a patient is not prescribed pain killers, given they are an addict. `P(P = p | A = ~a)` is the probability a patient is prescribed pain killers, given they are not an addict. `P(P = ~p | A = ~a)` is the probability a patient is not prescribed pain killers, given they are not an addict.

1.5 Titanic

[13]:
```
import pandas as pd
from pandasql import sqldf
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
```

[14]:

```
pysqldf = lambda q: sqldf(q, globals())
warnings.filterwarnings('ignore')
```

Make sure you worked through the Titanic case study. This is a continuation of that analysis. *Feel free to copy code blocks from the case study as you see fit*

We start by loading the data:

[3]:

```
titanic = pd.read_csv("https://raw.githubusercontent.com/
  ↪fundamentals-of-data-science/datasets/master/titanic.csv")
```

Copying the `summarize category` function here

[7]:

```
from pandas.core.algorithms import value_counts
from pandas.core.reshape.concat import concat
```

[8]:

```
def summarize_category(series):
    res_regu = value_counts(series)
    res_norm = value_counts(series, normalize=True)
    result = concat([res_regu, res_norm], axis=1, keys=['Count', 'Frequency'])
    result = result.sort_index()
    return result
```

1.6 Conditional Probabilities

1. Calculate $P(survived|parch)$

(Remember…every "calculation" includes discuss/code/discuss. In this case, describe what the conditional probability is, what you expect to see, calculate it, and then discuss the results relative to your hypothesized values).

[4]:

```
titanic.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1310 entries, 0 to 1309
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   pclass     1309 non-null   float64
 1   survived   1309 non-null   float64
 2   name       1309 non-null   object
 3   sex        1309 non-null   object
 4   age        1046 non-null   float64
 5   sibsp      1309 non-null   float64
 6   parch      1309 non-null   float64
 7   ticket     1309 non-null   object
 8   fare       1308 non-null   float64
 9   cabin      295 non-null    object
 10  embarked   1307 non-null   object
 11  boat       486 non-null    object
 12  body        121 non-null    float64
 13  home.dest   745 non-null    object
dtypes: float64(7), object(7)
memory usage: 143.4+ KB
```

We can look at a summary of the survivors and parch using the `summarize_category` function created above.

```
[16]: parch_counts = summarize_category(titanic['parch'])

      parch_counts
```

```
[16]:        Count  Frequency
      parch
      0.0     1002   0.765470
      1.0      170   0.129870
      2.0      113   0.086325
      3.0        8   0.006112
      4.0        6   0.004584
      5.0        6   0.004584
      6.0        2   0.001528
      9.0        2   0.001528
```

```
[17]: survivor_counts = summarize_category(titanic['survived'])
      survivor_counts
```

```
[17]:           Count  Frequency
      survived
      0.0         809   0.618029
      1.0         500   0.381971
```

We see that the number of passengers who survived is 500 of 1309, while the number of single passengers was 1002 out of 1309 total. We are interested in the probability that a passenger survived, given they were a parent/child on board (did not board as a single family member). From our output above, we know that the marginal probabilities a passenger was single was 77%, while passengers with parents/children was about 23%. Similarly, the probability that a passenger survived was 38%, and did not survive was 62%. We can find the conditional probabilities using the `crosstab` method with pandas.

```
[19]: pd.crosstab(titanic['parch'], titanic['survived'], normalize='index')
```

```
[19]: survived       0.0       1.0
      parch
      0.0       0.664671  0.335329
      1.0       0.411765  0.588235
      2.0       0.495575  0.504425
      3.0       0.375000  0.625000
      4.0       0.833333  0.166667
      5.0       0.833333  0.166667
```

```
6.0        1.000000  0.000000
9.0        1.000000  0.000000
```

We can start with `P(survived = 1.0 | parch = 0.0) = 33.5%` and `P(survived = 0.0 | parch =0.0) = 66.4%`. From here we can find the joint probabilities as well.

Using a table as we did for the previous problems will help us visualize the data. We can start with the passengers who were single and survived - the joint probability is `P(survived | single) = P(survived, single) / P(single)`. Rearranging to find `P(survived, single) = 0.335 * 0.765 = 0.256`. From here, we can fill in the rest of the joint probabilities easily:

```
|         | survived   | ~survived |
|----     |------      |------     |
|  parch |  0.126      | 0.108     |  0.234
| ~parch |  0.256      | 0.509     |  0.765
            0.382        0.618
```

Finally, we can calculate `P(survived | parch) = 0.126 / 0.234 = 0.538, or 54%`.

2. Calculate $P(survived|fare)$

Similar to above, we can create a table to show the joint probabilities between passengers who survived and fare. Because there is a range of fare, we will need to discretize.

[22]:
```
pd.DataFrame(titanic['fare'].describe())
```

[22]:
```
              fare
count  1308.000000
mean     33.295479
std      51.758668
min       0.000000
25%       7.895800
50%      14.454200
75%      31.275000
max     512.329200
```

[24]:
```
titanic['fare_range'] = (titanic['fare'] // 100) * 100
summarize_category(titanic['fare_range'])
```

[24]:
```
            Count  Frequency
fare_range
0.0          1224   0.935780
100.0          46   0.035168
200.0          34   0.025994
500.0           4   0.003058
```

Now we can look at the conditional probabilities.

[25]:
```
pd.crosstab(titanic['fare_range'], titanic['survived'], normalize='index')
```

```
[25]: survived          0.0       1.0
      fare_range
      0.0          0.640523  0.359477
      100.0        0.260870  0.739130
      200.0        0.352941  0.647059
      500.0        0.000000  1.000000
```

So for example, to calculate our joint probability `P(survived, fare = 0.0)`, we would simly multiply `P(survived | fare=0.0)* P(fare=0.0) = 0.359 * 0.936 = 0.336`. Following this pattern, our table then becomes:

```
|            | survived   | ~survived |
|----        |------      |------     |
| fare 0.0   |   0.336    |  0.6      |  0.936
| fare 100.0 |   0.026    |  0.009    |  0.035
| fare 200.0 |   0.0168   |  0.0082   |  0.025
| fare 500.0 |   0.003    |  0.00     |  0.003
               0.382       0.618
```

From here we can easily calculate any of the conditional probabilities. For example, `P(survived = 1.0 | fare = 100.0) = 0.026 / 0.035 = 0.743`. Note there will be some rounding errors here.

1.7 Naive Bayes Classifier

```
[26]: from sklearn.naive_bayes import CategoricalNB
      from sklearn.preprocessing import OrdinalEncoder
```

1. Calculate the Naive Bayes Classifier for $P(survived|pclass, sex, decade, parch, sibsp)$ and make 5 predictions.

(Remember…discuss/code/discuss. This is especially true for the predictions…when you make up each passenger, do you expect them to survive or perish?)

We know that for Naive Bayes Classifier, we can calculate each prior probability for the target variable, as we assume each feature is independent. First we discretize the age (as decade):

```
[28]: titanic['decade'] = (titanic['age'] // 10) * 10
      summarize_category(titanic['decade'])
```

```
[28]:         Count  Frequency
      decade
      0.0        82   0.078394
      10.0      143   0.136711
      20.0      344   0.328872
      30.0      232   0.221797
      40.0      135   0.129063
      50.0       70   0.066922
      60.0       32   0.030593
      70.0        7   0.006692
      80.0        1   0.000956
```

Copying the conditional probability function here

[44]:

```
def conditional_probability(df, target, givens, cell="index"):
    """
    calculates a simple conditional probability (only one target variable) based off of:
    https://stackoverflow.com/questions/54040923/change-order-of-pandas-multiindex

    P(target|givens...)

    df: the DataFrame to use for the calculation
    target: the string name of the target variable
    givens: a string or List of strings that represent the "givens"
    cell: a column that is neither target nor givens to "count". Should be a column without NA.

    The default assumes you have added a column: df["index"] = df.index to your DataFrame.
    """
    if isinstance(givens, str):
        givens = [givens]
    print(f"P({target}|{', '.join(givens)})")
    columns = [target] + givens
    # handling multiple targets would require a more sophisticated join.
    result = (df.groupby(columns).count() / df.groupby(givens).count())[cell]
    # this makes sure the target is always the column
    result = result.reorder_levels(givens + [target]).sort_index()
    # this flattens the hiearchical index and should fill in missing values.
    result = result.unstack(fill_value=0.0)
    return pd.DataFrame(result)
```

We can calculate the conditional probability of survival.

[93]:

```
survived_bayes = conditional_probability(titanic, 'survived', ['pclass', 'sex', 'decade', 'parch', 'sibsp'], 'name')
survived_bayes
```

```
P(survived|pclass, sex, decade, parch, sibsp)
```

[93]:

```
survived                                0.0  1.0
pclass sex    decade parch sibsp
1.0    female 0.0    2.0   1.0          1.0  0.0
              10.0   2.0   0.0          0.0  1.0
                           1.0          0.0  1.0
                           2.0          0.0  1.0
                     0.0   0.0          0.0  1.0
…                                    …   …
```

```
3.0      male    40.0    6.0    1.0     1.0   0.0
                 50.0    0.0    0.0     1.0   0.0
                                1.0     1.0   0.0
                 60.0    0.0    0.0     1.0   0.0
                 70.0    0.0    0.0     1.0   0.0

[230 rows x 2 columns]
```

Using the scikit-learn tools to create pseudocounts for missing data, as in Fundamentals, chapter 4.

[61]:
```
clf = CategoricalNB()
encoder = OrdinalEncoder()
with_age = titanic[titanic['age'].notnull()]

encoder.fit(with_age[['pclass', 'sex', 'decade', 'parch', 'sibsp']])
encoder.categories_
```

[61]:
```
[array([1., 2., 3.]),
 array(['female', 'male'], dtype=object),
 array([ 0., 10., 20., 30., 40., 50., 60., 70., 80.]),
 array([0., 1., 2., 3., 4., 5., 6.]),
 array([0., 1., 2., 3., 4., 5., 8.])]
```

Now we can fit the classifier with the specified features.

[62]:
```
clf.fit(encoder.transform(with_age[['pclass', 'sex', 'decade', 'parch',␣
 ↪'sibsp']]), with_age['survived'])
```

[62]:
```
CategoricalNB()
```

Now time to make the predictions. We need 5 values to pass into the prediction, one for each of `passenger class`, `sex`, `decade`, `parch` and `sibsp`.

[63]:
```
clf.predict(encoder.transform([(1, 'female', 30, 3, 5)]))
```

[63]:
```
array([1.])
```

We can first calculate the naive Bayes classifier. Since there are a lot of categories for `parch` and for `sibsp`, I did not include every single one, but just a few is enough to get an idea of how the naive Bayes classifier works.

We can create 5 predictions to determine the probability a passenger will survive based on the above features. The predictions are as follows

[123]:
```
survived_bayes.loc[(3, 'male', 40, 0, 2)]
```

[123]:
```
survived
0.0      1.0
1.0      0.0
```

```
Name: (3.0, male, 40.0, 0.0, 2.0), dtype: float64
```

[37]:

```
from tabulate import tabulate
```

[124]:

```
table = []
for tup in [(1, 'male', 20, 0, 0), (1, 'female', 60, 0, 0), (3, 'male', 30, 1,␣
 ↪0),
            (1, 'female', 30, 2, 1), (3, 'male', 40, 0, 2)]:
    result = clf.predict_proba(encoder.transform([tup]))
    entry = survived_bayes.loc[tup]
    not_survived, survived = entry.values[0], entry.values[1]
    row = list(tup) + list(result[0]) + [not_survived, survived]
    table.append(row)

print(tabulate(table, headers=['pclass', 'sex', 'decade', 'parch', 'sibsp',
                                'NBC not survived', 'NBC survived', 'Empirical␣
 ↪not survived',
                                'Empirical survived']))
```

```
  pclass  sex       decade    parch    sibsp    NBC not survived    NBC survived
Empirical not survived    Empirical survived
--------  ------  --------  -------  -------  ------------------  --------------
------------------------  --------------------
       1  male          20        0        0           0.72867         0.27133
0.583333                 0.416667
       1  female        60        0        0           0.220754        0.779246
0                        1
       3  male          30        1        0           0.797931        0.202069
0                        1
       1  female        30        2        1           0.0397851       0.960215
0                        1
       3  male          40        0        2           0.90019         0.0998097
1                        0
```

I know the empirical data doesn't make much sense, I was having trouble with the loc method in pandas, even though I believe my conditional probability function is working properly. Here we can see the Naive Bayes classifier predictions for each of 5 different predictions.