11. Bayes theorem and data frames in Pandas LING 471

Learning outcomes

- Calculate **conditional probabilities**
- Use **Bayes theorem** to choose between equivalent formulations of conditional probability
- Describe Bayes theorem in terms of "beliefs"
- Describe **data frames**, **rows**, and **columns**
- Perform basic data frame creation and manipulation with the pandas Python library

Motivating examples

Motivating medical scenario

- Suppose
  - 1% of people have cancer
  - 80% of tests detect cancer correctly, and 20% of tests fail to detect it (false negative)
  - 9.6% of tests detect it when it is not there (false positive), and 90.4% of tests correctly return negative
- Question: If you get a positive result, what is the probability you actually have cancer?
  - Hint: it's not 80%...

Motivating linguistic scenario (easier)

- You have made an auto suggest feature for a keyboard
- The user has entered "to the" and you need to suggest the next word
- In your corpus, "to the X" appears in 3854 out of 1,640,329 trigrams
- Of these "to the other" appears 33 times, and "to the wrong" appears 27 times
- What is the probability the user enters "other"? What about "wrong"?

We will return to the previous examples shortly!

Bayes theorem

Bayes theorem

- Basically, a theorem in math is a statement that is provably true
- The theorem named after Thomas Bayes ("Bayes theorem") is that
  - $P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)}$
- Since it is a theorem, it must be provable
  - We will do this on the next slide
  - I promise it is straightforward! 🙂

Deriving Bayes theorem

We have defined conditional probability for A and B as

$$P(B|A) = \frac{P(A \cap B)}{P(A)}\,. \tag{1}$$

We need to show that it is equivalent to state

$$P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)}\,. \tag{2}$$

By definition, we know that

$$P(A|B) = \frac{P(A \cap B)}{P(B)}\,. \tag{3}$$

If we rearrange (3), we can see

$$P(A \cap B) = P(A|B) \cdot P(B)\,. \tag{4}$$

If we substitute the result from (4) into our original statement in (1), we have

$$P(B|A) = \frac{P(A|B) \cdot P(B)}{P(A)}\,. \tag{5}$$

We can see that (5) is identical to (2), so we are done and have proved Bayes theorem.

The composition of Bayes theorem

- The different parts of the probability statement represent different concepts in probability
- $P(A)$ – This is your **prior probability**, the overall probability of A occurring ($P(B)$ is analogous)
  - What you assume **before** you see the data
- $P(A|B)$ – This is your likelihood, the probability (density) associated with B
- $P(B|A)$ – This is your **posterior probability**, the probability you have **after** you have observed the data

So, what does Bayes theorem mean?

- It gives us a way to **update** our **belief** about an event after seeing data
- We generally have assumptions about the world (prior beliefs/probabilities)
- Each time we observe an event, we adjust our assumptions
  - The adjusted assumption is our posterior belief/probability

Choosing a form of conditional probability

- We have two formulations of conditional probability
- How do we choose which one to use?
  - It depends on the task
- Sometimes, one is easier to use (maybe you have P(A|B) and not P(A and B)
- Sometimes, one is conceptually better to use
  - "Updating beliefs" is easier with P(A|B)*P(A)

Returning to our medical example

- We want to determine P("have cancer" | "positive test")
- Take some time and work it out
- Use Python as your calculator
  - Or just use the one built into your operating system… ☹

Medical example solution

- P("have cancer" | "positive test") =
  P("have cancer" and "positive test") / P("positive test")
- We don't have P("have cancer" and "positive test"), though…
- We do, however, have
  - P("positive test" | "cancer") = 0.8
  - P("cancer") = 0.01, P("no cancer") = 0.99
  - P("positive test") = "false positive" + "true positive" =
    0.99 * 0.096 + 0.01* 0.8 = 0.096 + 0.008 = 0.10304
- Evaluate: (0.01 * 0.8) / 0.10304 = 0.0776 = 7.8%

Medical example solution

- We can also show it on a table, assuming a sample of 10,000
  - 1% of people have cancer
  - 80% of tests detect cancer correctly, and 20% of tests fail to detect it (false negative)
  - 9.6% of tests detect it when it is not there (false positive), and 90.4% of tests correctly return negative

	Has cancer	No cancer	Total
Test positive	80	955	1,030
Test negative	20	8,950	8,970
Total	100	9,900	10,000

Returning to our linguistic example

- We want to determine P(other | to the)
- We also want to determine P(wrong | to the)
- Take some time and calculate these probabilities
- Look at the “words.txt” and “trigrams.txt” files on Canvas for data

Linguistic example solution

- $P(\text{other}|\text{to the}) = {}^{P(\text{other} \cap \text{to the})}/_{P(\text{to the})}$
  - $P(\text{other} \cap \text{to the}) = {}^{33}/_{1{,}640{,}329}$
  - $P(\text{to the}) = {}^{3854}/_{1{,}640{,}329}$
  - $P(\text{other}|\text{to the}) = \frac{{}^{33}/_{1640329}}{{}^{3854}/_{1640329}} = \frac{33}{1640329} \cdot \frac{1640329}{3854} = \frac{33}{3854} = 0.86\%$
- $P(\text{wrong}|\text{to the}) = {}^{P(\text{wrong}|\text{to the})}/_{P(\text{to the})}$
  - $P(\text{wrong} \cap \text{to the}) = {}^{27}/_{1640329}$
  - $P(\text{wrong}|\text{to the}) = \frac{27 \cdot 1640329}{1640329 \cdot 3854} = \frac{27}{3854} = 0.70\%$

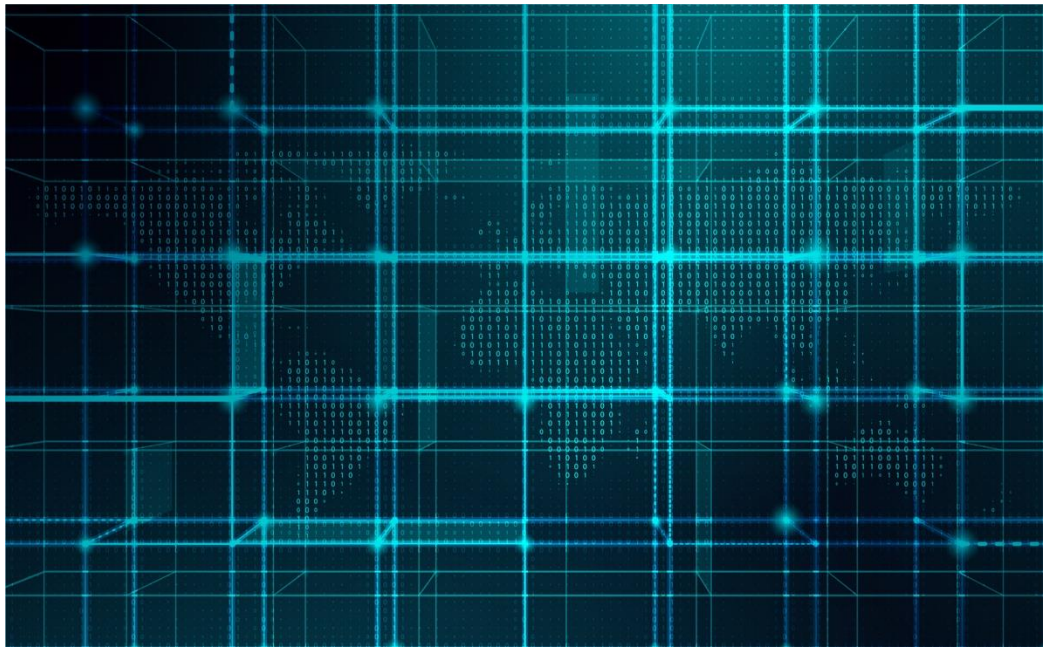
Now try this in Python

- Using the skeleton code in trigrams.py, replace the break statement inside the loop with code that updates the three counters (n_to_the, n_to_the_other, n_to_the_wrong).
- Hints: strip each line first, then think about what string methods let you check whether a line starts with a prefix or exactly equals a string.
- Once your counters are correct, the print statements will give you P(other | to the) and P(wrong | to the).
- Check: do your answers match the ones we computed on the board?

Data frames

Data formats

- When we are analyzing data, some formats are more convenient than others
  - See: the format for our programming assignments
- One very common format is called a **data frame**
  - Basically, a table



Data frames

Column1	Column2	Column3
1	1.4	"One"
2	1.8	"Two"

- Ubiquitous data format across statistics and machine learning
  - And all fields where these fields are applied (like linguistics…)
- Stores data as a table with **column names**
  - Row names are optional
- Each column must have **one** specific type
  - But **not** all columns need the same type

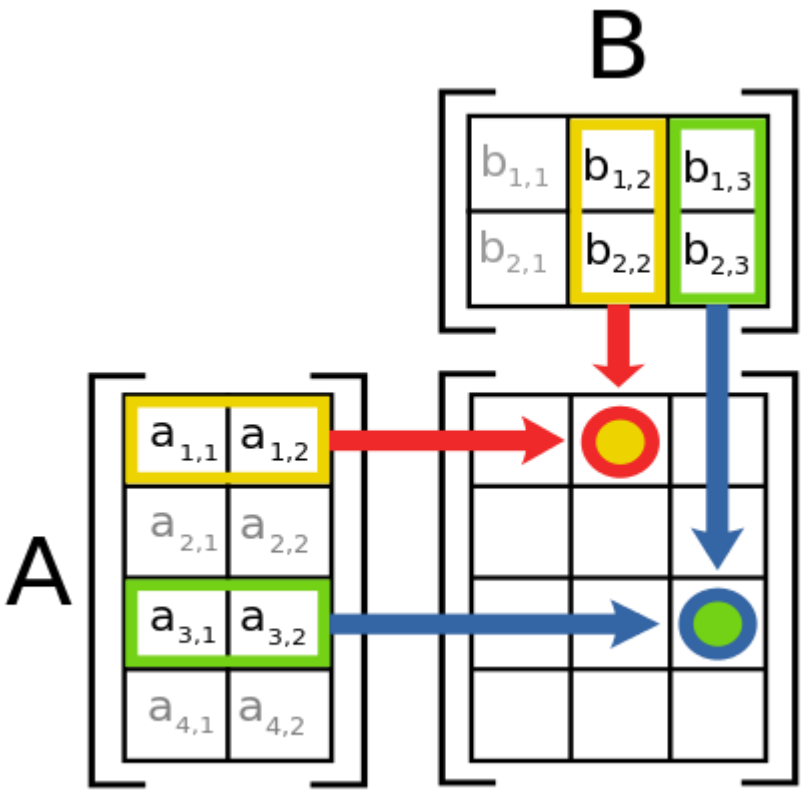
Data in a table

- Each row is considered an **observation**
- An observation is a **single measurement**
- Rows group related measurements together
- Columns contains all measurements of one variable

Row_name	Subject	Height_cm	Age_y
Observation_1	1aiA	160	18.5
Observation_2	2biiB	180	78.9

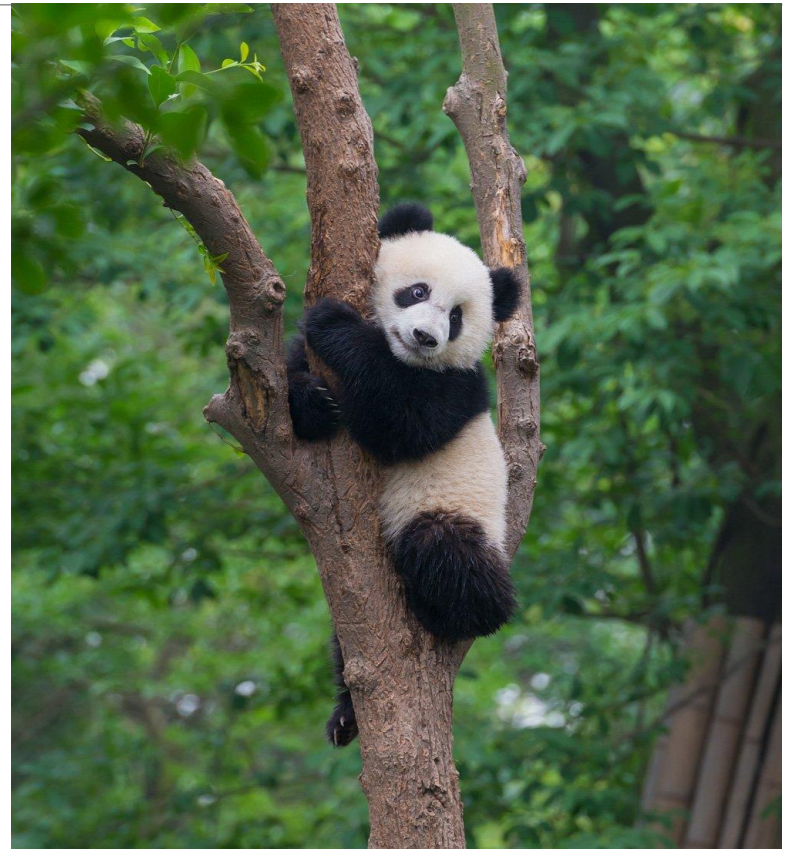
Why else do we use tables?

- **A LOT** of machine learning, statistics, AI, computation, etc. uses **matrix and vector multiplication**
- A row or column can be treated as a **row or column vector**
- An entire table can be treated as a **matrix**



Cool. How do we use data frames in Python? Pandas!

- pandas package in Python provides data frame functionality
  - Needs to be installed
- Used prevalently in Python-based ML
  - But still somehow doesn't always talk to other packages nicely…



Pandas demo

- Download demo files from GitHub and follow along