Assignment 0: Linear Algebra, Probability, and Python Warmup (UNGRADED)

Goals The main goal of this assignment is for you to assess whether you have adequate preparation for the course. It's fine to not be familiar with every concept here. However, if you find yourself struggling with much of this assignment, you should ask the course staff whether this course is appropriate for you given your background. This assignment is designed to take around 2 hours.

This assignment is ungraded! If you wish to discuss any of it, feel free to ask the course staff.

1 Linear Algebra

Q1 For each of the following matrices, give the answer or write "undefined" if the operation is invalid. You do not need to show work.

a) $\begin{bmatrix} 1 & 2 & 4 \\ 3 & 4 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 2 \end{bmatrix}$ b) $\begin{bmatrix} 1 & 2 & 4 \\ 3 & 4 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 5 \\ 2 \end{bmatrix}$ c) $\begin{bmatrix} 1 & 2 & 4 \\ 3 & 4 & 2 \end{bmatrix} \begin{bmatrix} 4 & 2 \end{bmatrix}$ d) $\begin{bmatrix} 6 \\ 2 \\ 4 \end{bmatrix}^\top \begin{bmatrix} 5 \\ 2 \\ 1 \end{bmatrix}$

a) ND b) $\begin{bmatrix} 22 \\ 36 \end{bmatrix}$ c) ND d) 38

Q2 Write a matrix operation capturing the following computation. Your answer should be a mathematical expression involving the vectors/matrices A, B, and C. Your math expression does not need to account for initialization of A, B, and C; it only needs to return the same value as sum given the same inputs.

```
A = np.rand(3)
B = np.rand(3,2)
C = np.rand(2)
sum = 0.0
for i in range(0,3):
  for j in range(0,2):
    sum += A[i] * B[i,j] * C[j]
return sum
```

If you assume A and C are row vectors: $ABC^\top$ or $(ABC^\top)^\top$
If you assume A and C are column vectors: $A^\top BC$ or $(A^\top BC)^\top$

2 Probability

Q3 Consider the following joint distribution:

$P(X,Y)$	$Y=1$	$Y=2$	$Y=3$
$X=1$	0.1	0.2	0.2
$X=2$	0.05	0.1	0.1
$X=3$	0.1	0.1	0.05

a) What is $P(X|Y=2)$?
$P(X|Y=2)$ = [0.5, 0.25, 0.25] for $X=1,2,3$

b) What is $P(Y|X=1)$?
$P(Y|X=1)$ = [0.2, 0.4, 0.4] for $Y=1,2,3$

c) Are X and Y independent? Justify your answer.
No. You can find many cases where $P(X=k) \neq P(X=k|Y=c)$. For example, $P(X=1)=0.5$ marginal but $P(X=1|Y=1)=0.4$.

Q4 Suppose you have a distribution $P(X,Y)$ where $X \in \{0,1\}$ and $Y \in \{0,1\}$. You know that the marginal distribution $P(X)=[0.5,0.5]$ and $P(Y)=[0.2,0.8]$.

a) If X and Y are independent, what do we know about the value of the joint probability $P(X=0,Y=0)$? Give upper and lower bounds as precisely as you can.
$P(X=0,Y=0)=P(X=0)P(Y=0)=0.1$, where the first equality is due to independence

b) If X and Y are not independent, what do we know about the value of the joint probability $P(X=0,Y=0)$? Give upper and lower bounds as precisely as you can.
$P(X=0,Y=0)=P(Y=0)P(X=0|Y=0)$ due to the chain rule of probability. We don't know what $P(X=0|Y=0)$ is. It could be as low as 0 if the joint probability table is:

$P(X,Y)$	$Y=0$	$Y=1$
$X=0$	0.0	0.5
$X=1$	0.2	0.3

It could be as high as 1 if the joint probability table is:

$P(X,Y)$	$Y=0$	$Y=1$
$X=0$	0.2	0.3
$X=1$	0.0	0.5

Therefore, $0 \leq P(X=0,Y=0) \leq 0.2$.

Q5 The binary entropy of a random variable X with discrete domain D is defined as:

$$H(X) = -\sum_{x \in D(X)} P(x) \log_2 P(x)$$

a) Compute the entropy of $P(X) = \text{Categorical}([\frac{1}{n}]_{i=1}^{n})$, the uniform distribution over n variables. Your answer should be written symbolically.

$\sum_{i=1}^{n} \frac{1}{n} \log n = \log n$

b) When you have a joint distribution over X and Y, entropy is defined as:

$$H(X, Y) = -\sum_{x \in D(X)} \sum_{y \in D(Y)} P(x, y) \log P(x, y)$$

How does this relate to the entropy of the marginal distributions $P(X)$ and $P(Y)$ when X and Y are independent?

Independent: $H(X, Y) = H(X) + H(Y)$. We can derive this by applying the definition of independence, then rearranging sums until we have terms like $\sum_{x \in D(X)} P(x)$ that sum to 1:

$$\begin{aligned}
&-\sum_{x \in D(X)} \sum_{y \in D(Y)} P(x, y) \log P(x, y) \\
&= -\sum_{x \in D(X)} \sum_{y \in D(Y)} P(x)P(y)[\log P(x) + \log P(y)] \\
&= -\sum_{x \in D(X)} \sum_{y \in D(Y)} P(x)P(y) \log P(x) - \sum_{x \in D(X)} \sum_{y \in D(Y)} P(x)P(y) \log P(y) \\
&= -\sum_{x \in D(X)} \left[P(x) \log P(x) \sum_{y \in D(Y)} P(y) \right] - \sum_{y \in D(Y)} \left[P(y) \log P(y) \sum_{x \in D(X)} P(x) \right] \\
&= -\sum_{x \in D(X)} [P(x) \log P(x)] - \sum_{y \in D(Y)} [P(y) \log P(y)] \\
&= H(X) + H(Y)
\end{aligned}$$

3 Language Basics / Coding Warmup

In this part of the assignment, you will read in and do some basic manipulation of a text corpus. Included with the assignment is a file `nyt.txt` containing 8860 sentences taken from New York Times articles, one sentence per line.

Q6 Here you will investigate tokenization schemes. Tokenization is the process of splitting raw text into words. In English, this involves splitting out punctuation and contractions (*shouldn't* becomes *should 'nt*) and is typically done with rules. In other languages like Chinese or Arabic, the process can be significantly more involved.

a) What are the 10 most frequent words in this dataset using whitespace tokenization? That is, split each sentence into words simply based on where the spaces are. List each word and its count and describe any patterns you see.

the, 26423; of 13072; to 11264; a 10447; and 9533; in 8677; that 5778; for 4089; is 3664; Mr. 3349. Largely English function words

b) What are the 10 most frequent words in this dataset using smarter tokenization? You can either use the tokenizer in `tokenizer.py` or invoke another tokenizer like NLTK (`nltk.word_tokenize(sentence)` after importing NLTK) or spaCy:[1]

```
from spacy.lang.en import English
nlp = English()
tokenizer = nlp.tokenizer
tok_sent = tokenizer(sentence) # see the spaCy docs about the object type here
# len(tok_sent) gives the length
# first token is in tok_sent[0]
```

List each word and its count and describe any patterns you see.

Depends on which tokenizer you use, but comma, period, and quote should now show up. Punctuation is the main difference here.

c) Explain in a few sentences how these differences in tokenization could affect a downstream text processing system. Discuss at least two ways.

Many possible answers: word counts get screwed up by merging in punctuation, therefore increasing sparsity; having punctuation appear separately makes punctuation tokens easier to recognize; mismatched tokenization with a downstream system will cause problems; good tokenization can be a bit slower (this isn't a major factor though). More answers later in the class.

Q7 In this part, we are going to confirm a phenomenon known as Zipf's Law. A word has *rank* n if it is the nth most common word. Zipf's Law states that the frequency of a word in a corpus is inversely proportional to its rank. Roughly speaking, this means that the fifth most common word should be five times less frequent than the most common word, and the tenth most common word should occur half as much as the fifth most common word.

[1] Instructions to install NLTK: `https://www.nltk.org/install.html` and spaCy: `https://spacy.io/usage`

a) Make a plot of inverse rank vs. word count for the smart tokenization scheme. Inverse rank is the reciprocal of the rank of the word: 1 for the most frequently occurring word, $\frac{1}{2}$ for the second most, $\frac{1}{3}$ for the third most, etc. Include your plot in your submission. Matplotlib is a good tool to use, but Excel/Matlab/Gnuplot/others are okay too.

A scatter plot is the best way to show this data. Here's what it looks like when using the spaCy tokenizer:

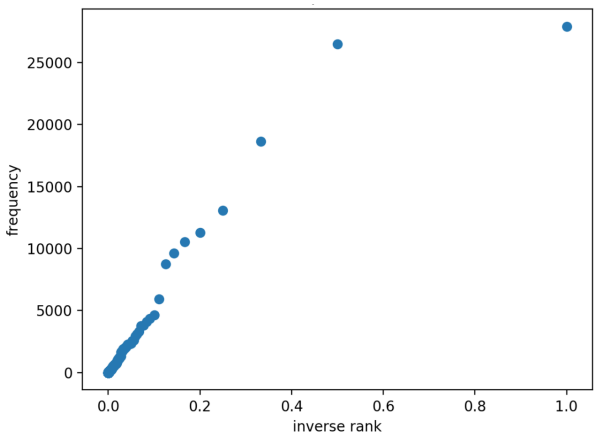

Figure 1: Zipf's law graph with inverse rank on the x-axis (1/rank) and word count on the y-axis.

Your solution should plot inverse rank as in the plot above or be a log-log plot or similar. If you show a plot with a 1/n style relationship, you should have a trend line for it. Otherwise, it's impossible to visually distinguish Zipf's law (polynomial decay) from exponential decay or other types of relationships.

b) Based on the plot, where does Zipf's law appear to hold? Are there any outliers?

It holds for higher ranks but not for the most frequent 10 types. Arguably there is a piecewise linear form of it. However, comma, the most common token, seems to be a significant outlier.

c) Look at your list of most frequent words. Identify **three words** out of the top 100 words in this dataset that you believe are unusually common in this data compared to written English text overall, and for each of these words, say why you think it is more common than expected here.

Mr., *Dole*, and *York* are good choices. This data has many articles about Bob Dole. These New York Times articles also talk about New York with high frequency, and old news reports tend to refer to men much more frequently than women, with *Mr. X* being the convention for referring to them.