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) $\begin{bmatrix} 6 \\ 2 \\ 4 \end{bmatrix}^\top \begin{bmatrix} 5 \\ 2 \\ 1 \end{bmatrix}$

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
```

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 $P(X|Y=2)$?

b) What is $P(Y|X=1)$?

c) Are X and Y independent? Justify your answer.

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.

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.

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

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?

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.

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.

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

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.

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

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.