Goals. The main purpose of this assignment is to help review prerequisite concepts in this course, including probability and the basics of machine learning.

Part 1: Review of Probability

Probability is key to machine learning, including NLP. Language models are just machines that take prior context as input and produce probability distributions over continuations.

Q1. Take the following joint probability distribution over random variables A and B:

	$B=1$	$B=2$
$A=1$	0.20	0.30
$A=2$	0.12	0.18
$A=3$	0.08	0.12

a. Compute the two marginal distributions $p(A)$ and $p(B)$. $p(A=1)=0.50,\ p(A=2)=0.30,\ p(A=3)=0.20.\ \ p(B=1)=0.40,\ \ p(B=2)=0.60.$

b. Are A and B independent? Why or why not? Yes, because every cell in the table can be computed as $p(A,B)=p(A)p(B)$.

c. What is $p(B=1 \mid A=3)$? Compare your answer to $p(B=1)$ and explain what the comparison tells you. $p(B{=}1 \mid A{=}3)=\frac{p(A{=}3,B{=}1)}{p(A{=}3)}=\frac{0.08}{0.20}=0.40$. This is the same as $p(B=1)$, meaning that the value of B does not depend on the value of A here.

Q2. A spam filter sees an incoming email. The probability that a randomly chosen email is spam is $p(S=1)=0.3$. The probability that an email contains the string "`$10,000`" given that it is spam is $p(M=1 \mid S=1)=0.8$; the probability that it contains "`$10,000`" given that it is *not* spam is $p(M=1 \mid S=0)=0.1$.

a. What is the probability that a randomly chosen email contains the string "`$10,000`"? In other words, what is $p(M=1)$? **Hint:** Recall the law of total probability. 0.31

b. You observe an email containing the string "`$10,000`". What is the probability that it is spam, i.e., $p(S=1 \mid M=1)$? **Hint:** Recall Bayes' rule. $p(S{=}1 \mid M{=}1)=\frac{p(M{=}1|S{=}1)p(S{=}1)}{p(M{=}1)}=\frac{0.24}{0.31}\approx 0.774$

Q3. Assume we have a random variable X that can take one of four integer values. The probability distribution over these values is as follows: $p(2)=0.1, p(4)=0.1, p(6)=0.3, p(8)=0.5$.

a. What is the expectation over this distribution? 6.4

b. What is the entropy of this distribution? Use base-2 logarithms. $H(X)=-\sum_x p(x)\log_2 p(x)\approx 1.69$

c. Suppose you permute the four probabilities among the four values (keeping the same four values and the same four probabilities, just paired up differently). Does the entropy change? In one sentence, why? The entropy does not change. It would not affect the value of the terms in the summation from (b), just their ordering; because of the commutativity of addition, the final answer therefore does not change.

Q4. Consider a unigram language model that draws tokens independently and uniformly at random from the vocabulary $V = \{\texttt{the}, \texttt{dog}, \texttt{ran}, \texttt{far}, \texttt{away}\}$, so $|V| = 5$.

a. What is the probability of generating `dog` in a single sample? $\frac{1}{5}$

b. What is the probability that a single sample produces either `the` *or* `away`? $\frac{2}{5}$

c. In three samples, what is the probability of generating `the`, `dog`, and `ran` in *any* order? For example, `ran the dog` would fit this criterion, as would `dog ran the`. $\frac{6}{125}$

Part 2: Review of Linear Algebra

How is linear algebra related to NLP? Thanks to neural networks, they're very closely intertwined these days. We usually represent words as vectors of real numbers (**embeddings**, or embedding vectors). These vectors are refined over many layers of a neural network into increasingly abstract representation vectors.[1] We also learn many **weight matrices** that will be multiplied with a representation vector as part of the process of computing the hidden representation vector for the next layer. Thus, a neural network is composed largely of matrix–vector and matrix–matrix multiplications, followed by some calculus to update the values of the weight matrices.

We also often use linear algebraic concepts like L_2 norms to compute the magnitude of a representation, and dot products or cosine similarities to compare the similarities of two representation vectors.

Q5. What is the rank of this matrix? $\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 1 & 0 & 1 \end{bmatrix}$

2

Q6. Perform the following matrix multiplications. Write "undefined" if the matrix multiplication is not possible.

(a) $\begin{bmatrix} 2 & 0 & 1 \\ -1 & 3 & 4 \end{bmatrix} \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

(a) Undefined

(b) $\begin{bmatrix} 2 & 0 & 1 \\ -1 & 3 & 4 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ -2 \end{bmatrix}$

(b) $\begin{bmatrix} 4 \\ -8 \end{bmatrix}$

(c) $\begin{bmatrix} 2 & 0 & 1 \\ -1 & 3 & 4 \end{bmatrix}^\top \begin{bmatrix} 1 & 0 & 2 \\ 2 & 1 & -1 \end{bmatrix}$

(c) $\begin{bmatrix} 0 & -1 & 5 \\ 6 & 3 & -3 \\ 9 & 4 & -2 \end{bmatrix}$

Q7. Write the inner product (dot product) of the following two vectors: $\begin{bmatrix} 4 & -2 & 1 \end{bmatrix} \begin{bmatrix} -3 & 5 & 2 \end{bmatrix}$

-20

Q8. Compute the Frobenius (L_2) norm of this matrix: $\begin{bmatrix} 2 & 2 & 4 \\ 4 & 5 & 4 \end{bmatrix}$ 9

[1] Some use "embedding" to refer to representation vectors only before the first layer of a neural network, while others use it to refer to representation vectors in *any* layer. In this course, I will use "embedding" primarily to refer to representations before the first layer of a neural network.

Part 3: Review of Machine Learning Basics

Almost every model we study this semester is going to be a **neural network** that takes some vector(s) as input(s) and outputs probabilities. We don't expect you to know what a neural language model is or how it works yet, but we do expect some prior knowledge of the basics of machine learning and neural networks.

Q9. Suppose we want to classify movie reviews as positive ($y = 1$) or negative ($y = 0$). We represent each review as a vector $\mathbf{x}$ whose entries are the counts of the words below, and we classify using logistic regression, $\hat{y} = \sigma(\mathbf{w}^\top \mathbf{x} + b)$, where σ is the sigmoid function. Assume the following learned weights $\mathbf{w}$, with bias $b = 0.5$:

feature	`great`	`terrible`	`not`	`movie`	`this`	`was`
weight	2.0	−2.5	−1.0	0.0	0.0	0.0

a. Write the feature vector $\mathbf{x}$ for the review *"this movie was not great"*. Then, compute $\hat{y}$ using the feature vector and the weights above. Which label does the model predict, using a decision threshold of 0.5? $\mathbf{x} = \begin{bmatrix}1 & 0 & 1 & 1 & 1 & 1\end{bmatrix}$
$\hat{y} = \sigma(\begin{bmatrix}2.0 & -2.5 & -1.0 & 0.0 & 0.0 & 0.0\end{bmatrix}^\top \begin{bmatrix}1 & 0 & 1 & 1 & 1 & 1\end{bmatrix} + 0.5) = \sigma(1 + 0.5) \approx 0.818$
Because $\hat{y} > 0.5$, the model's decision is **positive**.

b. Suppose that after training, this model achieves 99% accuracy on the training set but 61% accuracy on the development set. What is this phenomenon called? Name two things you could do to address it. This is called "overfitting". To address it, we could (i) train for fewer steps, or (ii) collect more data.

c. This review is in fact negative, so the gold label is $y = 0$. Give the binary cross-entropy loss $\mathcal{L} = -[y \log \hat{y} + (1 - y) \log(1 - \hat{y})]$ for this example. You can leave it in terms of log; you do not need to evaluate it numerically. $-[0 \log 0.818 + 1 \log 0.182] = -\log 0.182$

d. For logistic regression with a cross-entropy loss, the gradients take the following form:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = (\hat{y} - y)\,\mathbf{x}, \qquad \frac{\partial \mathcal{L}}{\partial b} = \hat{y} - y.$$

Compute $\frac{\partial \mathcal{L}}{\partial \mathbf{w}}$ and $\frac{\partial \mathcal{L}}{\partial b}$ for this example. Then write the gradient descent update rule and apply one step with learning rate $\alpha = 0.5$, reporting the updated $\mathbf{w}$ and b.
$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} \approx 0.818 \times \begin{bmatrix}1 & 0 & 1 & 1 & 1 & 1\end{bmatrix} = \begin{bmatrix}0.818 & 0 & 0.818 & 0.818 & 0.818 & 0.818\end{bmatrix}$
$\frac{\partial \mathcal{L}}{\partial b} = 0.818$
$\mathbf{w}' = \mathbf{w} - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \begin{bmatrix}1.591 & -2.5 & -1.409 & -0.409 & -0.409 & -0.409\end{bmatrix}$
$b' = b - \alpha \frac{\partial \mathcal{L}}{\partial b} = 0.091$

Q10. Consider a neural network with one hidden layer that maps an input vector $\mathbf{x} \in \mathbb{R}^2$ to a distribution over two classes:

$$\mathbf{h} = \mathrm{ReLU}(W^{(1)}\mathbf{x} + b^{(1)})$$
$$\hat{\mathbf{y}} = \mathrm{softmax}(W^{(2)}\mathbf{h} + b^{(2)})$$

where $\text{ReLU}(z) = \max(0, z)$ is applied elementwise, and

$$W^{(1)} = \begin{bmatrix} 1 & 2 \\ 0 & -3 \\ 1 & 1 \end{bmatrix}, \quad b^{(1)} = \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix}, \quad W^{(2)} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & 1 \end{bmatrix}, \quad b^{(2)} = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}.$$

a. Compute $\mathbf{h}$. $\mathbf{h} = \text{ReLU}(\begin{bmatrix} 1 & 2 \\ 0 & -3 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ -2 \end{bmatrix}) = \text{ReLU}(\begin{bmatrix} 1 \\ 3 \\ -1 \end{bmatrix}) = \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix}$.

b. Compute the predicted distribution $\hat{\mathbf{y}}$. You can leave this in terms of a softmax over a vector.

$\hat{\mathbf{y}} = \text{softmax}(\begin{bmatrix} 1 & 0 & -1 \\ 2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ -1 \end{bmatrix}) = \text{softmax}(\begin{bmatrix} 1 \\ 4 \end{bmatrix})$

Q11. The following are more general conceptual questions.

a. In general, why should you use a development set and *not* the test set to tune hyperparameters? If we tune hyperparameters on the test set, we might obtain unrealistically good results that won't generalize well to new examples.

b. Explain in one sentence each what generally tends to go wrong if (i) the learning rate is too large, or (ii) the learning rate is too small. When the learning rate is too large, a model may never converge to a good solution, or have very unstable results. When the learning rate is too small, a model will take far more training steps than necessary to converge to a good solution.

c. Let's say we removed the non-linearity from the neural network in Q10. How would this change the class of functions the network can learn? This would reduce the class of functions the model can learn to just *linear* or *affine* functions. No matter how many linear layers we stack on top of each other, such a network would always be equivalent to a single linear (or affine) transformation of the inputs.

Part 4: Review of Differential Calculus

To update the weight matrices of a neural network, we use derivatives. We'll go more in-depth on how this works in class and in the book, but for now, it will be helpful to refresh your knowledge of the basics.

To update the weight matrices of a neural network, we use derivatives. We'll go more in-depth on how this works in class and in the book, but for now, it will be helpful to refresh your knowledge of the basics.

Q12. What is $\frac{\partial}{\partial y}(x^4y^2 + 2xy)$? $2x^4y + 2x$

Q13. We will often denote exponential functions like e^x as $\exp(x)$. What is $\frac{\partial}{\partial x}\log\left(1 + \exp(x^2y)\right)$? $\frac{2xy \cdot \exp(x^2y)}{1+\exp(x^2y)}$