CS 7650 Problem Set 2

Instructor: Dr. Wei Xu
TAs: Mounica Maddela, Nathan Vaska, and Sarah Wiegreffe

1 Logistic vs Softmax

a. Recall the Logistic and Softmax functions

$$P_{Logistic}(y = 1|\mathbf{x}) = \frac{e^{\mathbf{w}^T\mathbf{x}}}{1 + e^{\mathbf{w}^T\mathbf{x}}}$$

$$P_{Softmax}(y|\mathbf{x}) = \frac{e^{\mathbf{w}_y^T\mathbf{x}}}{\sum_{y' \in \mathcal{Y}} e^{\mathbf{w}_{y'}^T\mathbf{x}}}$$

Given $\mathcal{Y} = \{0, 1\}$, what should be the value of $\mathbf{w}$ in the logistic function such that $P_{Logistic}(y|\mathbf{x}) = P_{Softmax}(y|\mathbf{x}) \ \forall \ y \in \mathcal{Y}$? Show your work.
Hint: Think about $\mathbf{w}$ in terms of $\mathbf{w}_0$ and $\mathbf{w}_1$.

b. Recall that the Softmax function is a generalization of the sigmoid in logistic regression for multiclass classification. In practice, machine learning software such as PyTorch uses one Softmax implementation for both binary and multiclass classification. Recall that the Softmax function produces a vector output $\mathbf{z} \in \mathbb{R}^{|\mathcal{Y}|}$ and the Logistic function a single scalar value z, representing class probabilities. Write the equation for a decision rule to produce $\hat{y}$ from the Softmax function in the binary case (when $\mathcal{Y} = \{0, 1\}$; you can break ties arbitrarily). Write the decision rule to produce $\hat{y}$ from the Logistic function. Compare the two rules. How are they similar and/or different? (1-2 sentences).

2 Classification Objectives

In maximum-likelihood optimization, for a dataset of size N, the optimal parameters can be found by maximizing the probability of the dataset:

$$\hat{W} = \text{argmax}_W P(y^{(1:N)} = y_*^{(1:N)}|\mathbf{x}^{(1:N)}) \tag{1}$$

$$= \text{argmax}_W \prod_{i=1}^{N} P(y^{(i)} = y_*^{(i)}|\mathbf{x}^{(i)}) \tag{2}$$

$$= \text{argmax}_W \sum_{i=1}^{N} logP(y^{(i)} = y_*^{(i)}|\mathbf{x}^{(i)}) \tag{3}$$

a. The product of probabilities from equation 2 is not equal to the sum of log probabilities from equation 3. What property of the *log* function ensures that equation 2 equals equation 3 (i.e., that the same W will maximize both a probability and its logarithm)?

Hint: plot the $log(x)$ function (with any base) and study it on the domain of x-values that a probability can take.

b. This formulation is often preferred for optimization because optimizing in *log*-space (equation 3) is more numerically stable than optimizing the product of probabilities (equation 2) directly. Here, we will show why.

(a) Assume you have been given a dataset of size $10,000$. We will assume that the initial probabilities for each instance are fixed at 0.1 (a simplification for demonstration purposes). Run the following code in Python3 to compute the likelihood of the dataset using the product-of-probabilities approach (equation 2):

```
>> from functools import reduce
>> f = 10000 * [0.1]
>> prob_of_dataset = reduce(lambda x, y: x*y, f)
```

What is the returned value for the likelihood of the dataset? Is this logically correct?

(b) Now, compute the same value using the sum-of-log-probabilities approach (equation 3):

```
>> from functools import reduce
>> import numpy as np
>> f = 10000 * [np.log(0.1)]
>> prob_of_dataset = reduce(lambda x, y: x+y, f)
```

What is the returned value?

(c) Now, assume you have been given an oracle weight-update that increased the probability of the dataset. Probabilities for each instance have now increased to `0.2` (again, a simplification for demonstration purposes). Re-run the above code and report the new values for parts (a) and (b). Which formulation (equation 2 or 3) adequately represents the increase in the data likelihood?

(d) In 1-2 sentences, explain what went wrong with the product-of-probabilities method.

3 Perceptron: Linear Separability and Weight Scaling

a. Suppose we have the following data:

Features (x_1, x_2)	Label (y)
(0, 0)	-1
(0, 1)	1
(1, 0)	1
(1, 1)	-1

Notice that the data above is not linearly separable. Therefore the perceptron algorithm will not be able to learn a classifier that gives the correct prediction for all four above data points.

We can add a 3^{rd} dimension/feature to each input such that the data becomes linearly separable. If we add $(1, 0, 0, 1)$ to the 3^{rd} dimension (x_3) of the four data points in order, will the new data be linearly separable? Assume 0 is the threshold for classification. Justify your answer.

b. Suppose we have a trained Perceptron with parameters (W, b). If we scale W by a positive constant factor c, will the new set of weights produce the exact same prediction for all the test data? Assume the threshold for classification is 0. Justify your answer.

c. With the same setting as 2, this time we translate W by a positive constant factor c (add c to each element of W), will the new set of weights produce the exact same prediction for all the test data? Justify your answer.

4 Multiclass Naive Bayes with Bag of Words

A candy company has been recording data on what consumers think about different product descriptions. Consumer sentiment is categorized as positive, neutral, or negative, while product descriptions are characterized by the number of times certain words are mentioned in each description. The table below shows the collected data. The candy company wants to use the Naive Bayes algorithm to help predict whether new product descriptions will be favorably received before committing to expensive user studies.

Description	scrumptious	delicious	sour	sweet	healthy	organic	unbelievable	sugary	Y
1	0	0	1	1	0	0	0	0	Positive
2	1	0	0	1	0	1	1	1	Neutral
3	0	0	1	1	1	1	0	1	Negative
4	0	1	1	1	1	1	0	0	Negative
5	0	1	0	1	1	0	1	0	Positive
6	0	1	1	1	0	0	1	1	Neutral
7	0	0	0	1	0	0	0	1	Positive
8	1	0	0	0	0	0	1	0	Positive

a. What is the probability θ_y of each label $y \in \{\text{Positive, Neutral, Negative}\}$?

b. The parameter $\phi_{y,j}$ is the probability of a token j appearing with label y. It is defined by the following equation, where V is the size of the vocabulary set:

$$\phi_{y,j} = \frac{\text{count}(y, j)}{\sum_{j'=1}^{V} \text{count}(y, j')}$$

The probability of a count of words x and a label y is defined as follows:

$$\text{p}(x, y; \theta, \phi) = \text{p}(y; \theta) \cdot \text{p}(x|y; \phi) = \text{p}(y; \theta) \prod_{j=1}^{V} \phi_{y,j}^{x_j}$$

Find the most likely label $\hat{y}$ for the following word counts vector $x = (0, 1, 0, 1, 1, 0, 0, 1)$ using $\hat{y} = \text{argmax}_y \log \text{p}(x, y; \theta; \phi)$. Show final log (base-10) probabilities for each label rounded to 3 decimals. Treat $\log(0)$ as $-\infty$.

c. When calculating argmax_y, if $\phi_{y,j} = 0$ for a label-word pair, the label y is no longer considered. This is an issue, especially for smaller datasets where a feature may not be present in all documents for a certain label. One approach to mitigating this high variance is to smooth the probabilities. Using add-1 smoothing, which redefines $\phi_{y,j}$, again find the most likely label $\hat{y}$ for the following word counts vector $x = (0, 1, 0, 1, 1, 0, 0, 1)$ using $\hat{y} = \text{argmax}_y \log \text{p}(x, y; \theta; \phi)$. Make sure to show final log probabilities.

$$\text{add-1 smoothing: } \phi_{y,j} = \frac{1 + \text{count}(y, j)}{V + \sum_{j'=1}^{V} \text{count}(y, j')}$$

5 Nonlinear classifiers

a. Which of the following statements are true? Select all that are true.

(a) A neural network with a lot of hidden layers might lead to gradients being too small in the initial layers.

(b) While learning neural networks using backpropagation, we can clip the magnitude of the gradients to prevent the successive updates from being too large.

(c) Feedforward neural networks can handle input features of variable length.

(d) Residual connection in a neural network make it easier to learn the parameters of the lower levels of the network.

b. The chain rule is the backbone of neural net backpropagation. The following question will walk you through the application of the chain rule to a function representing a dummy neural network loss.

Consider the equation below, where L is the loss and x is the input to the function:

$$L = -(sin(-x^{-1}) + cos(e^{-x^{-1}}))^2$$

The function can be decomposed into a sequence of smaller functions, as shown in the table below. Using the chain rule and calculus, fill in the remaining derivative and component cells.

Term	Equation	Derivative (Input 1)	Derivative (Input 2)	Component 1	Component 2
L	$L = -Z^2$	$\frac{dL}{dZ} = \frac{dL}{dZ}$	N/A	$\frac{dL}{dZ} = -2Z$	N/A
Z	$Z = sin(W) + cos(Y)$	$\frac{dL}{dW} = \frac{dL}{dZ}\frac{dZ}{dW}$	$\frac{dL}{dY} =$	$\frac{dZ}{dW} = cos(W)$	
W	$W = -S$	$\frac{dL}{dS} =$	N/A		N/A
Y	$Y = e^S$	$\frac{dL}{dY} =$	N/A		N/A
S	$S = -x^{-1}$	$\frac{dL}{dx} =$	N/A		N/A

Given that x is given as the input and S, Y, W, Z and L are calculated during the forward pass, what quantities do we need to calculate during the backwards pass to fully define the derivative for each term? (Do not actually perform the calculation, just state the quantities).

c. **(Extra Credit)** In a neural network, it is likely that the objective function will not maintain convexity with respect to the network's weights. With additional critical points, finding the global optimum is no longer as easy. Although local optima pose a threat to convergence, saddle points can cause more problems in practice. Which of the following statements are true? Select all that are true.

(a) The gradient at saddle points approaches 0, slowing or halting learning.

(b) The gradient at saddle points approaches $-/+ \infty$, slowing or halting learning.

(c) In higher dimensions/large networks, the likelihood of saddle points becomes much lower than local optima.

(d) Dropout can be used to address saddle points by adding feature noising, allowing gradient descent to escape local optima.

(e) Dropout can be used to address saddle points by adding extra features, allowing gradient descent to escape local optima.

(f) Stochastic gradient descent (SGD) makes it more difficult to escape/avoid saddle points because it does not use all of the training data to make parameter updates.

(g) Stochastic gradient descent (SGD) helps to escape/avoid saddle points because making an update based on a random sample of the training data may provide enough noise to escape the saddle point.

(h) The 2nd derivative of the objective function with respect to the input can be used to distinguish saddle points from local optima.