CS 4650: Natural Language Processing
Spring 2022
Problem Set 1

Instructor: Dr. Wei Xu
TAs: Chao Jiang, Chase Perry, Rucha Sathe
Piazza: https://piazza.com/gatech/spring2022/cs4650a

Due: Thursday, Feb 3, 11:59pm ET

1 Logistic vs Softmax

(a) (**2 pts**) 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 $\mathbf{w}$ such that
$P_{Logistic}(y|\mathbf{x}) = P_{Softmax}(y|\mathbf{x}) \ \forall \ y \in \mathcal{Y}$? Show your work.
Hint: Expand the summation term and think about $\mathbf{w}$ *in terms of* $\mathbf{w}_0$ *and* $\mathbf{w}_1$*.*

(b) (**2 pts**) Recall that the Softmax function is a generalization of the logistic sigmoid for multiclass classification. In practice, machine learning software such as PyTorch uses a 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 Multiclass Naive Bayes with Bag of Words

Dr. Smith's clinic has recently begun to provide a preliminary analysis of whether or not a patient is affected by Virus X, based on the description of the patient's condition, uploaded online. The resulting variable can take 3 values, namely: Affected, Unaffected, No Diagnosis. The description of the patient's condition is filtered out on the basis of a select few symptoms, to determine whether or not the patient might be affected. Collected data is displayed in the table below. Dr. Smith's team wishes to put Naive Bayes algorithm to use to carry out the task at hand.

S.No.	body-ache	dehydrated	headache	cold	nauseous	fever	energetic	hungry	Y
1	0	0	1	1	0	0	0	0	Unaffected
2	1	0	0	1	0	1	1	1	No Diagnosis
3	0	0	1	1	0	1	1	1	Affected
4	0	1	1	1	1	1	0	0	No Diagnosis
5	0	1	0	1	1	0	1	0	Unaffected
6	0	1	1	1	0	0	1	1	Affected
7	0	0	0	1	0	0	0	1	Unaffected
8	1	0	0	0	1	0	0	0	Unaffected

(a) (**1 pt**) What is the probability θ_y of each label $y \in \{\text{Unaffected, No Diagnosis, Affected}\}$?

(b) (**3 pts**) 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) (**3 pts**) 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.

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

3 Perceptron: Linear Separability and Weight Scaling

(a) (**2 pts**) Suppose we have the following data representing the XOR function:

x_1	x_2	$f(x_1, x_2)$
0	0	-1
0	1	1
1	0	1
1	1	-1

Table 1: XOR function data

Evidently, the data is not linearly separable. Therefore the perceptron algorithm will not be able to learn a classifier for XOR, based on this data.

However, 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.

After the addition of the third dimension, can we say that the perceptron algorithm is actually capable of learning the XOR function? Why or why not?

(b) (**2 pts**) 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) (**2 pts**) 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 Feedforward Neural Network

(**2 pts**) In Question 3, we tried to design a perceptron architecture in order to learn the XOR function represented by Table 1. Now, we want you to design a feedforward neural network to compute the XOR function.

Use a single output node and **specify** the activation function you choose for it. Also use a single hidden layer with ReLU activation function. Describe all weights and offsets (bias terms).

(Hint: In class, we discussed a neural network design that solves the XOR problem using ***tanh*** *activation functions.)*

5 Dead Neurons

The ReLU activation function can lead to “dead neurons”, which can never be activated on any input. Consider a feedforward neural network with a single hidden layer and ReLU nonlinearity, assuming a binary input vector, $\mathbf{x} \in f\{0,1\}^D$ and scalar output y:

$$z_i = \text{ReLU}(\theta_i^{(x \to z)} \cdot \mathbf{x} + b_i)$$
$$\mathbf{y} = \theta^{(z \to y)} \cdot \mathbf{z}$$

Assume the above function is optimized to minimize a loss function (e.g., mean squared error) using stochastic gradient descent.

(a) (**2 pts**) Under what condition is node z_i “dead”? Your answer should be expressed in terms of the parameters $\theta_i^{(x \to z)}$ and b_i.

(b) (**2 pts**) Suppose that the gradient of the loss on a given instance is $\frac{\partial l}{\partial y} = 1$. Derive gradients $\frac{\partial l}{\partial b_i}$ and $\frac{\partial l}{\partial \theta_{j,i}^{(x \to z)}}$ for such an instance.

(c) (**2 pts**) Using your answers to the previous two parts, explain why a “dead” neuron can never be brought back to life during gradient-based learning.

(Hint: The notation used for this question is in line with that used in Eisenstein Chapter 3.)