CS 7650: Natural Language Processing
Spring 2023
Problem Set 1

Due: Tuesday, Feb 14, 11:59pm ET

1 Logistic vs Softmax (5 points)

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* $\mathbf{w}_1$.

2 Multiclass Naive Bayes

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	1	1	1	0	0	0	0	Unaffected
2	0	1	0	1	0	1	1	1	No Diagnosis
3	1	0	1	1	0	1	0	1	Affected
4	0	1	0	1	1	1	0	1	No Diagnosis
5	0	1	0	1	1	0	0	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	0	0	1	0	Unaffected

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

(b) (**3 pts**) The parameter $\phi_{y,j}$ is the probability of a symptom j appearing with label y. It is defined by the following equation, where V is the size of the vocabulary of symptoms:

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

The joint probability of a set of observed symptoms x and a label y can be modeled 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 vector of symptoms $x = (0, 1, 0, 1, 1, 0, 0, 1)$ 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 symptom 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 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.)

4 Maximum Likelihood Parameter Estimation (7 points)

In binary Naive Bayes, show that the maximum-likelihood estimate for the class prior parameter is

$$P(y=1) = \frac{c(y=1)}{m}$$

where $c(y=1)$ is the number of observations in the data containing the label $y=1$ and m is the total number of observations.

Recall that Naive Bayes models the likelihood of the training data as follows (see slide 63 in the lecture on binary classification):

$$L(\theta) = \prod_{j=1}^{m} P(y_j, x_j) = \prod_{j=1}^{m} P(y_j) \left[\prod_{i=1}^{n} P(x_{ji}|y_j) \right] \quad (1)$$

$$= \theta_y^{c(y=1)} (1-\theta_y)^{c(y=0)} \prod_{i=1}^{n} \prod_{y \in \{0,1\}} \theta_{x_i=1|y}^{c(x_i=1,y)} (1-\theta_{x_i=1|y})^{c(x_i=0,y)} \quad (2)$$

Where the model's parameters are: $\theta_y = P(y=1)$, and $\theta_{x|y} = P(x|y)$. $c(x_i=1, y=1)$ represents the number of training examples where x_i has the value 1 and y has the value 1.

Hint: Take the derivative of the Naive Bayes log-likelihood function with respect to θ_y, set equal to zero and solve to find the value of θ_y that maximizes L(θ).

5 Backpropagation (7 points)

In lecture, we discussed the backpropagation algorithm in the context of a simple 2-layer feedforward neural network (see slide 96): https://aritter.github.io/CS-7650-sp23/slides/lec6-nn.pdf

$$P(\mathbf{y}|\mathbf{x}) = \text{softmax}(Wg(Vf(x)))$$

with the following conditional log-likelihood objective:

$$\mathcal{L}(\mathbf{x}, i^*) = W\mathbf{z} \cdot e_{i^*} - \log \sum_j \exp(W\mathbf{z}) \cdot e_j$$

where e_{i^*} is a the one-hot vector representing the gold label, i^*, and activations at the hidden layer, $\mathbf{z}$ are defined as follows:

$$\mathbf{z} = g(Vf(\mathbf{x}))$$

We saw how gradients on the output weight matrix, W, are the same as in multi-class logistic regression using $\mathbf{z}$ as features. We then derived gradients on the input weight matrix V as follows:

$$\frac{\partial \mathcal{L}(\mathbf{x}, i^*)}{\partial V_{ij}} = \frac{\partial \mathcal{L}(\mathbf{x}, i^*)}{\partial \mathbf{z}} \frac{\partial \mathbf{z}}{\partial V_{ij}}$$

Your task in this question is to show that the "error at the hidden layer", $\frac{\partial \mathcal{L}(\mathbf{x}, i^*)}{\partial \mathbf{z}}$, can be computed from the "error at the network's output", $err(\text{root}) \stackrel{\text{def}}{=} e_{i^*} - P(\mathbf{y}|\mathbf{x})$, as follows:

$$err(\mathbf{z}) \stackrel{\text{def}}{=} \frac{\partial \mathcal{L}(\mathbf{x}, i^*)}{\partial \mathbf{z}} = W^T [e_{i^*} - P(\mathbf{y}|\mathbf{x})] \stackrel{\text{def}}{=} W^T err(\text{root})$$

Hint: Start with the log-likelihood objective defined above, and compute the derivative with respect to the vector $\mathbf{z}$.