HW4-Week 6-Emotion Classification Using Logistic Regression

(100 points + 10 bonus) **Discussion Week 6**
Due Sunday May 14th, 11:59 pm PST

Ellen Yim

Objective: In this assignment, you will investigate the core principles of the **binary** logistic regression algorithm.

Instructions: The assignment is divided into five sections. Complete all sections and submit the following:

1. A PDF or word doc answering the below questions.
2. Source code files **(e.g., Colab)** containing your implementation of the logistic regression classifier and data preprocessing steps.

Filter your data to contain only information from the 2 categories **`{Joy,Sadness}.Treat Joy as your positive class, and Sadness as your negative class for coding and computation.`** Again, double count sentences that are in both categories.

Binary logistic regression: summary

Given:

- a set of classes: (+ sentiment,- sentiment)
- a vector **x** of features `[x1, x2, …, xn]`
  - x1= count("awesome")
  - x2 = log(number of words in review)
- A vector **w** of weights `[w1, w2, …, wn]`
  - w_i for each feature f_i

$$P(y=1) = \sigma(w \cdot x + b) = \frac{1}{1+e^{-(w \cdot x+b)}}$$

Section 1: Understanding Logistic Regression (10 points)

1.1: Research the logistic regression algorithm and write a brief summary of the algorithm's principles, assumptions, and applications.

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. It delivers a binary outcome limited to 2 possible outcomes i.e. true/false or 0/1. It is used to analyze relationships between one or more independent variables, used in predictive modeling where the model estimates the probability of whether an instance belongs to a specific category or not. It uses a sigmoid function to map predictions and their probabilities. Some applications include the possibility of enrolling to a university or identifying spam emails. [1]

- **Upload dataset (as a csv file)**
- **Colab File:** **https://colab.research.google.com/drive/1PKy34Hh7XlQBIUu74YnPkTXXAZ8vCH7A?usp=sharing**

Section 2: Preprocessing the Data (10 points)

2.1: Download the dataset you created last week **[https://docs.google.com/spreadsheets/d/1FO779z232nz8pVEk2-lfA7pRX0BW1Nqlk1x-oPoxRmc/edit?usp=sharing]** . Please use the first 30 rows as training, the next 10 rows (30-40) as validation, and the next 10 rows (40-50) as testing set.

```
import pandas as pd
df = pd.read_csv('hw4LR-CS173-published-sheet.csv')
training_set = df[0:30]
validation_set = df[30:40]
testing_set = df[40:]
```

2.2: Construct features and gold-reference labels; try stemming and use lexicon word columns as you want to match as many words as possible:

- x1: Counts of joy lexicon in the document (sentences)
- x2: Counts of Sadness lexicon in the document
- x3: total number of tokens in the document

```
Joy lexicons count in joy sentences
{'sunny': 1, 'laughter': 1, 'youth': 1, 'zeal': 1, 'adorable': 1, 'rekindle': 1,
'friendship': 2, 'engaged': 1, 'child': 1, 'happy': 2, 'achieve': 1, 'delighted':
```

[1] https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-logistic-regression/

2, 'unconstrained': 1, 'serenity': 1, 'love': 4, 'amusing': 2, 'abundance': 1, 'aspire': 1, 'perfection': 2, 'festive': 1, 'chuckle': 1, 'joy': 4, 'excitement': 2, 'hero': 1, 'daughter': 1, 'glee': 2, 'beach': 2, 'treasure': 1, 'beautiful': 3, 'Gratifying': 1, 'happiness': 1, 'healing': 1, 'achievement': 1, 'utopian': 1, 'smile': 1, 'zealous': 2, 'fulfillment': 2, 'winner': 1, 'accompaniment': 1, 'alive': 1, 'sweet': 1, 'perfect': 1, 'exuberance': 1, 'delicious': 1, 'award': 1, 'amour': 1, 'art': 1, 'favorite': 1, 'birthday': 1, 'twinkle': 1, 'wonderful': 1, 'happily': 1}
Sad lexicons count in sad sentences
{'devastating': 2, 'abduction': 1, 'wretched': 1, 'wrongly': 1, 'lonely': 2, 'abandoned': 4, 'death': 2, 'agony': 3, 'crushed': 1, 'tragedy': 1, 'longing': 3, 'suffering': 2, 'remorse': 1, 'homesick': 1, 'lowest': 1, 'demise': 1, 'savage': 1, 'betrayed': 1, 'ache': 1, 'abysmal': 1, 'deceased': 1, 'weeping': 1, 'grief': 1, 'abuse': 2, 'abandon': 1, 'anguish': 2, 'endless': 1, 'abyss': 1, 'unrequited': 1, 'alcoholism': 1, 'accident': 1, 'anxiety': 1, 'degrading': 1, 'deformity': 1, 'lone': 1, 'isolation': 1, 'inability': 1}
feature vector for each sentence of joy lexicons in document
[1, 0, 25] # 1st val=count of joy lexicons, 2nd val=count of sad lexicons, 3rd val=total # of tokens in sentence
[1, 0, 7]
[0, 0, 10]
[1, 0, 10]
[1, 0, 22]
[1, 0, 18]
[1, 0, 22]
[1, 0, 21]
[0, 0, 21]
[1, 0, 18]
[1, 0, 27]
[0, 0, 41]
[2, 0, 23]
[1, 0, 20]
[1, 0, 11]
[2, 0, 16]
[1, 0, 31]
[1, 0, 13]
[1, 0, 11]
[1, 0, 37]
[1, 0, 9]
[1, 0, 23]
[1, 0, 11]
[1, 0, 27]
[4, 0, 31]
[2, 0, 27]
[1, 0, 27]
[0, 0, 20]
[1, 0, 24]
[1, 0, 15]
feature vectors for each sentence of sad lexicons in document

```
[2, 1, 20]
[0, 1, 7]
[0, 0, 12]
[1, 0, 20]
[0, 1, 21]
[3, 1, 14]
[0, 1, 16]
[0, 1, 16]
[2, 1, 22]
[0, 0, 28]
[0, 1, 10]
[0, 1, 24]
[0, 1, 23]
[0, 3, 14]
[0, 1, 8]
[0, 0, 18]
[0, 1, 13]
[0, 0, 8]
[0, 1, 21]
[0, 2, 45]
[0, 2, 18]
[0, 1, 26]
[0, 1, 7]
[0, 1, 21]
[0, 4, 29]
[6, 1, 36]
[0, 2, 24]
[0, 0, 32]
[0, 1, 14]
[0, 1, 17]
```

- Have the dataset file as a csv file (**'hw4LR-CS173-published-sheet.csv'**)
- Code for section 2 is labeled in colab file
- Run code from Section 2.1 first, then run cells in section 2.2

Section 3: Implementing Logistic Regression Classifier (30 points)

3.1: Write a python function that computes **p(y=1)** where y is the document and 1 means Joy class. (15 points)

```
def prediction(row, coeff):
 yhat = coeff[0]
 for i in range(len(row) - 1):
   yhat += coeff[i-1] * row[i]

 return 1.0 / (1.0 + exp(-yhat))
```

- Made a list for the expected class (first 30 rows are in joy sentences (represented by 1), last 30 rows are sad sentences (represented by 0) [denoted as classes_joy_sad_dataset] and appended each index value to features row list
- Code sectioned under Section 3.1, make sure to run code from Section 2
- Features is the list of each feature input for each sentence (feature input row: [count of joy lexicons in sentence, count of sad lexicons in sent., count of total # of tokens of sentence]
- coeff initialized (weights) to [1,1,1]
- Example: first sentence of whole document:
  [1, 0, 25, 1]
    - 1 joy lexicon, 0 sad lexicon, 25 tokens in sentence, 1 -actual class prediction (joy)
    - Predicted=0.731 (using prediction function on first sentence)

3.2: Initialize all the weights to be zero. Write Python code to compute the loss for the first example (row 1) (15 points):

$$L_{\text{CE}}(\hat{y}, y) = -[y \log \sigma(w \cdot x + b) + (1 - y) \log (1 - \sigma(w \cdot x + b))]$$

```
from nltk.text import log
weights = [0,0,0]
# loss function on first sentence
# Lce (yhat, y) = -[y log sigmoid (w*x + b) +
(1-y)log(1-sigmoid(w*x+b))]
def Lce(y):
  return -log(y)

row = features[0]
print(row)
pred = prediction(row, weights)
print(pred)
loss = Lce(pred)
print(loss)

[1, 0, 25, 1] # first row values
0.5 # prediction on first sentence
0.6931471805599453 # loss
```

- Using prediction function from previous question, took features[0] (first row/sentence) and computed predicted value if weights were 0, then with loss function above to compute the loss

- Code is under Section 3.2 https://colab.research.google.com/drive/1PKy34Hh7XlQBIUu74YnPkTXXAZ8vCH7A#scrollTo=SywiNXiD1Qbc
- Run code from section 2 and 3.1

Section 4: Learning & Optimization (30 points)

4.1: Implement the SGD for logistic regression. Use the validation set to decide your best **learning rate= [0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5]**. What is your best learning rate and what is the lowest **validation loss**? List learning rate and validation loss for other learning rates listed above.

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ

where: L is the loss function
f is a function parameterized by θ
x is the set of training inputs $x^{(1)}$, $x^{(2)}$,..., $x^{(m)}$
y is the set of training outputs (labels) $y^{(1)}$, $y^{(2)}$,..., $y^{(m)}$

$\theta \leftarrow 0$
repeat til done
For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)
1. Optional (for reporting): # How are we doing on this tuple?
   Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output $\hat{y}$?
   Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$) from the true output $y^{(i)}$?
2. $g \leftarrow \nabla_\theta L(f(x^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss?
3. $\theta \leftarrow \theta - \eta\, g$ # Go the other way instead

return θ

```
0.005 # best learning rate
Lowest validation loss:  0.6420246729486955
```

Other learning rate and validation loss output
(for joy class)

Section 4: Evaluating the Classifier on Test Set (20 points + 10 bonus points)

4.1: Generate a confusion matrix (2x2) and analyze the results on the **test dataset**. You can use the existing Python package. Copy and paste your confusion matrix here. (10 points)

```
[[9 0]
 [9 0]]
```

- TP (predicted joy, expected joy): 9
- FP (predicted joy, expected sad): 9
- FN (predicted sad, expected joy): 0
- TN (predicted sad, expected sad): 0
- Code under label "Section 5" in Colab: https://colab.research.google.com/drive/1PKy34Hh7XlQBIUu74YnPkTXXAZ8vCH7A#scrollTo=uZZmQrW79EFG

4.2: Calculate the accuracy, precision, recall, and F1-score for the Logistic regression classifier of the **Joy** category on the test dataset. Copy and paste your results here and indicate sections of code in colab for this computation. (10 points)

```
Accuracy:  0.5
Precision (micro):  0.5
Precision (macro):  0.16666666666666666
Precision (weighted):  0.25
Recall (micro):  0.5
Recall (macro):  0.3333333333333333
Recall (weighted):  0.5
F1-score (micro):  0.5
F1-score (macro):  0.2222222222222222
F1-score (weighted):  0.3333333333333333
```

- Code is under Section 5.2 in colab file

4.3: Can you come up with better input features that give you better performance with your existing implementation? Write it in Colab and share your features here. Do a detailed analysis of how these new features impact the validation loss and test results. (10 points)