Assignment 3. Logics, Sentiment Analysis

Gary Geunbae Lee
CSED342-01 Artificial Intelligence

Contact: TA Heejin Do (heejindo@postech.ac.kr)

General Instructions

This (and every) assignment has **written** and **programming** parts.

✎This icon means a written answer is expected. There are no partial credits. Submit `name_studentID.pdf` containing your answers to the written part.

⌨This icon means you should write code. You can add other helper functions outside the answer block if you want. DO NOT make changes to files other than `submission.py`. You should write your answers in `submission.py` between

```
# BEGIN_YOUR_ANSWER
```

and

```
# END_YOUR_ANSWER
```

Your code will be evaluated on two types of test cases, **basic** and **hidden**, which you can see in `grader.py`. Basic tests, which are fully provided to you, do not stress your code with large inputs or tricky corner cases. Hidden tests are more complex and do stress your code. The inputs of hidden tests are provided in `grader.py`, but the correct outputs are not. To run all the tests, type

```
python grader.py
```

This will tell you only whether you passed the basic tests. On the hidden tests, the script will alert you if your code takes too long or crashes, but does not say whether you got the correct output. You can also run a single test (e.g., `3a-0-basic`) by typing

```
python grader.py 3a-0-basic
```

We strongly encourage you to read and understand the test cases, create your own test cases, and not just blindly run `grader.py`.

Advice for this homework:

- Words are simply strings separated by whitespace. Don't normalize the capitalization of words (treat *great* and *Great* as different words).
- You might find some useful functions in `util.py`. Have a look around in there before you start coding.

Problem 1. Logics

Problem 1a [3 points]

Let x, y and z be integers. Determine the truth value of each of the following.

- $\forall x \exists y (x + y = 1)$
- $\forall x \forall y \exists z (xy < z)$
- $\exists z \forall x \forall y (xy < z)$

Problem 1b [3 points]

Youngsu, Hyeonsuk, Gwangsu, and Jungsuk are planning a winter vacation trip abroad. Using four literals, write the propositional logic formulas corresponding to the text below (Hint: Let Y, H, G, and J denote that Youngsu, Hyeonsuk, Gwangsu, and Jungsuk will go, respectively).

- (i) They go to the airline agency, but there are only two tickets left.
- (ii) Youngsu will only go if Hyeonsuk goes too.
- (iii) Jungsuk will only go if Gwangsu goes too.
- (iv) Hyeonsuk has found out that she has a schedule to record an interview broadcast, so she cannot go.

Problem 1c [1 points]

Using the answer to Problem 1b, find out who will go on a winter vacation trip.

Problem 2: Sentiment Classification

In this problem, we will build a binary linear classifier that reads the movie reviews and guesses whether they are ***positive*** or ***negative***.

Problem 2a [2 points] ⌨

Implement the function *extractWordFeatures*, which takes a review (string) as input and returns a feature vector $\phi(x)$ (you should represent the vector $\phi(x)$ as a *dict* in Python).

Problem 2b [8 points] ⌨

We're going to train a linear predictor, which can be represented by a logistic regression model. Here is the definition of linear prediction:

$$\begin{aligned} f_w(x) &= \begin{cases} +1 & \text{if } \mathbf{w} \cdot \phi(x) > 0 \\ -1 & \text{if } \mathbf{w} \cdot \phi(x) < 0 \end{cases} \\ &= \begin{cases} +1 & \text{if } \sigma(\mathbf{w} \cdot \phi(x)) > 0.5 \\ -1 & \text{if } \sigma(\mathbf{w} \cdot \phi(x)) < 0.5 \end{cases} \end{aligned}$$

where σ is a logistic (or sigmoid) function.

Your task is to implement the function *learnPredictor* using stochastic gradient descent, minimizing the negative log-likelihood loss (NLL) defined as:

$$\begin{aligned} \text{Loss}_{\text{NLL}}(x, y, \mathbf{w}) &= -\log(p_{\mathbf{w}}(y \mid x)) \\ p_{\mathbf{w}}(y \mid x) &= \begin{cases} \sigma(\mathbf{w} \cdot \phi(x)) & \text{if } y = 1 \\ 1 - \sigma(\mathbf{w} \cdot \phi(x)) & \text{if } y = -1 \end{cases} \end{aligned}$$

You should first derive $\nabla_{\mathbf{w}} \text{Loss}_{\text{NLL}}(x, y, \mathbf{w})$, then exploit the formula to update weights for each example. Also, you can print the training error and test error after each iteration through the data, so it's easy to see if your code is working.

Problem 2c [3 points] ⌨

The previous features include unigram(single) words only, which cannot consider the context of a word in an utterance. In this task, we'll incorporate bigram words into features. In other words, features include pairs of consecutive words. Implement *extractBigramFeatures*, which extracts both unigram and bigram word features.