Assignment 3 - Part A

Gary Geunbae Lee
CSED342 - Artificial Intelligence

Contact: TA Yejin Min (`yeajinmin@postech.ac.kr`)

Before moving onto the main part of the assignment (Part B), you will complete several simple review questions. Refer to the class slides before answering the questions. Write your answers to Word and save the file as "PartA_studentID.docx" (or pdf file).

A-1 CNF

Q1. Convert $\varphi := \neg(p \to q) \vee ((r \vee s) \to (q \vee t)) \vee (\neg p \to \neg v)$ into Conjunctive Normal Form. Show all the main logical equivalence steps. [5pts]

A-2 First Order Logic

Q2. For each choice, decide whether the FOL sentence correctly represents the English sentence(True or False). If it is incorrect, write a correct FOL formula. [6pts]

Choice a) There was a student at POSTECH who never did CSED342 assignment, but passed the class.

$$\exists x,\ IsStudent(x, POSTECH) \wedge \neg DoesAssign(x, 342) \Rightarrow Pass(x, 342)$$

Choice b) If a student dislikes Danny, they will fail the class.

$$\forall x,\ Student(x) \wedge Dislikes(x, Danny) \Rightarrow Fail(x, 342)$$

Choice c) All students at POSTECH who never did 342 assignment passed the class.

$$\forall x,\ IsStudent(x, POSTECH) \wedge \neg DoesAssign(x, 342) \wedge Pass(x, 342)$$

Q3. Which of the following is true w.r.t the English sentences 'Choice a' and 'Choice c' in the previous question? [3pts]

a $\models$ c	c $\models$ a	Both	Neither

Assignment 3 - Part B

Contact: TA Sujin Woo (`happysujin@postech.ac.kr`)

General Instructions

This (and every) assignment has a written part and a programming part.
You should write both types of answers in `submission.py` between

```
# BEGIN_YOUR_ANSWER
```
and
```
# END_YOUR_ANSWER
```

✎This icon means a written answer is expected. Some of these problems are multiple choice questions that impose negative scores if the answers are incorrect. So, don't write answers unless you are confident.

⌨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`.

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`.

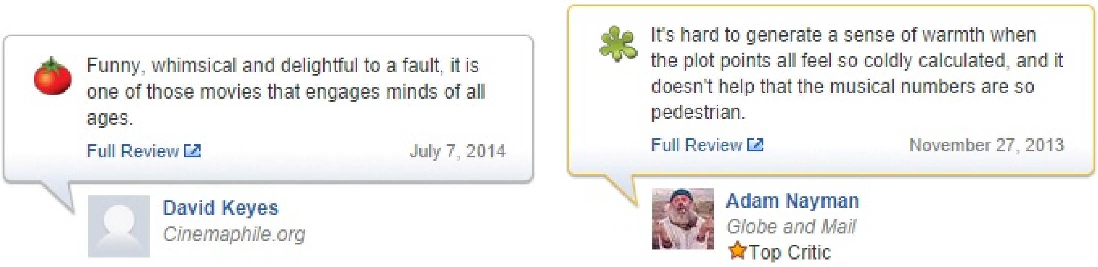
Advice for this homework:

- 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.

Problems

Problem 1. Hinge Loss

Here are two reviews of "Frozen," courtesy of Rotten Tomatoes (no spoilers!):



Funny, whimsical and delightful to a fault, it is one of those movies that engages minds of all ages.

Full Review July 7, 2014

David Keyes
Cinemaphile.org

It's hard to generate a sense of warmth when the plot points all feel so coldly calculated, and it doesn't help that the musical numbers are so pedestrian.

Full Review November 27, 2013

Adam Nayman
Globe and Mail
Top Critic

Rotten Tomatoes has classified these reviews as "positive" and "negative," respectively, as indicated by the in-tact tomato on the left and the splattered tomato on the right. In this assignment, you will create a simple text classification system that can perform this task automatically.

Problem 1a [4 points] ✎

We'll warm up with the following set of four mini-reviews, each labeled positive (+1) or negative (−1):

- (+1) pretty good
- (−1) bad plot
- (−1) not good
- (+1) pretty scenery

Each review x is mapped onto a feature vector $\phi(x)$, which maps each word to the number of occurrences of that word in the review. For example, the first review maps to the (sparse) feature vector $\phi(x) = \{\text{pretty} : 1, \text{good} : 1\}$. Recall the definition of the hinge loss:

$$\text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = \max\{0, 1 - \mathbf{w} \cdot \phi(x) y\},$$

where y is the correct label.

Suppose we run stochastic gradient descent, updating the weights according to

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w}),$$

once for each of the four examples in order. After the classifier is trained on the given four data points, what are the weights of the six words ('pretty', 'good', 'bad', 'plot', 'not', 'scenery') that appear in the above reviews? Use $\eta = 1$ as the step size and initialize $\mathbf{w} = [0, ..., 0]$. Assume that $\nabla_{\mathbf{w}} \text{Loss}_{\text{hinge}}(x, y, \mathbf{w}) = 0$ when the margin is exactly 1.

Problem 2: Sentiment Classification

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

Problem 2a [4 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 [12 points] ⌨

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

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

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:

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

You should first derive $\nabla_{\mathbf{w}} \mathrm{Loss}_{\mathrm{NLL}}(x, y, \mathbf{w})$, 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 [6 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 extract both unigram and bigram word features.