Assignment 1. Foundations

Hwanjo Yu
CSED342 - Artificial Intelligence

Contact: TA Minsoo Kim (km19809@postech.ac.kr)
Deadline: 4 Mar 2024 at 2:00 pm (50% penalty for every 1 day)

General Instructions

Assignment 1 has four problems - problem 0, 1, 2, and 3. You only need to submit problem 1, 2, and 3. Problem 0 is not be graded, but must be solved by yourself before participating in this course.

This assignment has been developed in **Python 3.8**, so please use **Python 3.8** to implement your code. we recommend using **Conda environment**.[1]

You should modify the code in `submission.py` between

```
# BEGIN_YOUR_ANSWER
```

and

```
# END_YOUR_ANSWER
```

You can add other helper functions outside the answer block if you want, but do not import other libraries and do not make changes to files other than `submission.py`. You can solve all the problems without any libraries other than *collections* and *math*.

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, 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
```

[1] https://docs.conda.io/projects/conda/en/stable/user-guide/install/index.html

This will tell you only whether you passed the basic tests. The script will alert you if your code takes too long or crashes on the hidden tests, but does not say whether you got the correct output. You can also run a single test (e.g., `2a-0-basic`) by typing

```
python grader.py 2a-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`.

Problems

Problem 0. Optimization and probability

In this class, we will cast AI problems as optimization problems, that is, finding the best solution in a rigorous mathematical sense. At the same time, we must be adept at coping with uncertainty in the world, and for that, we appeal to tools from probability. The three problems below will not be scored, but please solve them because they are basic concepts.

Problem 0a [0 points]

Suppose you repeatedly roll a fair six-sided dice until you roll a 6 (and then you stop). Every time you roll a 3, you win 2 points, and every time you roll a 1 or a 2, you lose 1 points. You do not win or lose any points if you roll a 4 or a 5. What is the expected number of points you will have when you stop?

Problem 0b [0 points]

Suppose the probability of a coin turning up heads is $0 < p < 1$, and that we flip it 5 times and get $\{\text{H}, \text{T}, \text{H}, \text{H}, \text{T}\}$. We know the probability (likelihood) of obtaining this sequence is $L(p) = p(1-p)pp(1-p) = p^3(1-p)^2$. Now let's go backward and ask the question: what is the value of p that maximizes $L(p)$?

Hint: Consider taking the derivative of $\log L(p)$. Taking the derivative of $L(p)$ works too, but it is cleaner and more natural to differentiate $\log L(p)$. You can verify for yourself that the value of p which minimizes $\log L(p)$ must also minimize $L(p)$.

Problem 0c [0 points]

Let's practice taking gradients, which is a key operation for being able to optimize continuous functions. For $\mathbf{w} \in \mathbb{R}^d$ and constants $\mathbf{a}_i, \mathbf{b}_j \in \mathbb{R}^d$ and $\lambda \in \mathbb{R}$, define the scalar-valued function

$$f(\mathbf{w}) = \sum_{i=1}^{m} \sum_{j=1}^{n} (\mathbf{a}_i^\top \mathbf{w} - \mathbf{b}_j^\top \mathbf{w})^2 + \frac{\lambda}{2} \|\mathbf{w}\|_2^2,$$

where $\mathbf{w}$, $\mathbf{a}_i$ and $\mathbf{b}_j$ are column vectors (e.g. $\mathbf{w} = (w_1, ..., w_d)^\top$) and $\|\mathbf{w}\|_2 = \sqrt{\sum_{j=1}^{d} w_j^2}$ is known as the L_2 norm. Compute the gradient $\nabla_{\mathbf{w}} f(\mathbf{w})$.

Recall: the gradient is a d-dimensional vector of the partial derivatives with respect to each w_i:

$$\nabla_{\mathbf{w}} f(\mathbf{w}) = \left(\frac{\partial f(\mathbf{w})}{\partial w_1}, \ldots, \frac{\partial f(\mathbf{w})}{\partial w_d} \right)^\top.$$

If you're not comfortable with vector calculus, first warm up by working out this problem using scalars in place of vectors and derivatives in place of gradients. Not everything for scalars goes through for vectors, but the two should at least be consistent with each other (when $d = 1$). Do not write out summation over dimensions, because that gets tedious.

Problem 1. Implementing string functions

In this problem, you will implement short functions that handle Python strings. The main purpose of this exercise is to familiarize yourself with Python and strings.

If you're new to Python, the following articles provide pointers to various tutorials and examples for the language:

- Python for Programmers:
  `https://wiki.python.org/moin/BeginnersGuide/Programmers`
- Example programs of increasing complexity:
  `https://wiki.python.org/moin/SimplePrograms`

Hint: You can use the **.split()** method to split a string into a list.

Problem 1a [4 points]

Implement *find_alphabetically_first_word* in `submission.py`.

Problem 1b [4 points]

Implement *find_frequent_words* in `submission.py`.

Problem 1c [4 points]

Implement *find_non_singleton_words* in `submission.py`.

Problem 2. Implementing vector operations

In this problem, you will implement a bunch of short functions related to vector representations. The main purpose of this exercise is to understand vector representations and operations in programming.

Problem 2a [4 points]

Implement *dense_vector_dot_product* in `submission.py`.

Problem 2b [4 points]

Implement *increment_dense_vector* in `submission.py`.

Problem 2c [4 points]

Implement *dense_to_sparse_vector* in `submission.py`.

Problem 2d [4 points]

Implement *sparse_vector_dot_product* in `submission.py`.

Problem 2e [4 points]

Implement *increment_sparse_vector* in `submission.py`.

Problem 2f [4 points]

Implement *euclidean_distance* in `submission.py`.

Problem 3. Advanced Programming

In this problem, you have to be familiar with algorithms and basic data structures. This problem may be challenging and require programming skills. If you can solve this, you will be well-equipped to handle future assignments with ease.

Problem 3a [4 points]

Implement *mutate_sentence* in `submission.py`.

Hint: You can consider **Depth First Search (DFS)** to solve this problem.