Programming for Artificial Intelligence (Python)
Homework 4
Due: April 12 before class

We have three required questions in this homework currently. Expect more questions after the next few classes.

Question 1 (code, standard deviation)

You must use the wrapper function structure to answer this question.

In class, we learned how to use the `wrapper` function to make the `mean` function work with `None` values in a list. If we want to compute the standard deviation, we can use a function from the `numpy` package.

In the Anaconda console (terminal), you can download and install `numpy`

```
conda install -y numpy
```

Then you can `import`:

```python
import numpy
```

When you want to compute the standard deviation of a list of numbers, you run

```python
import numpy
x = [1,2,3]
numpy.std(x)
```

However, the same problem is that if you run

```
import numpy
x = [1,2,None]
numpy.std(x)
```

there will be a `TypeError`. Please write a wrapper function to help compute the standard deviation even when there are `None` values. Name your function `mysd`. For example,

```
x = [1,2,None]
mysd(x)
```

should return `0.5`.

Hint: You can remove the `None` values in the wrapper as we did in class.

Question 2 (code, log information)

We have defined the standard deviation function in the Question 1. Now suppose we want to print the computation time. Name this function `mysdlog`.

For example, when you call

```
x = [range(10000000)]
mysdlog(x)
```

the output should look like:

```
# start time: 1711516581.5497375
# s=2886751.3459480824
# end time: 1711516582.29468
```

Question 3 (words, code, Logistic regression)

In this question, we explore the logistic regression problem. Whenever you want to use the log() function, you can use `numpy.log()`.

Recall that, in class, we introduced logistic regression with x being the time studying and y being the pass or fail (if one passes, $y = 1$; if one fails, $y = 0$).

Here are the data:

```
x=[0.1,0.2,0.25,0.26,0.3,0.65,0.8,0.83,0.9,0.93]
y=[0,  1,  0,   0,   0,  1,   1,  0,   1,  1]
```

Suppose we want to fit the folowwing model to get $\hat{a}$ and $\hat{b}$

$$\Pr(y = 1) = f(x) = \frac{e^{a+bx}}{1 + e^{a+bx}} \tag{1}$$

Note that the **response** variable is y but the model is about the probability that $y = 1$.

3.1 Likelihood (in words)

Please write done the probability of the data based on our model. That is, $L(a, b) = \Pr(y^{[1]} = 0, y^{[2]} = 1, y^{[3]} = 0, \dots, y^{[9]} = 1, y^{[10]} = 1)$.

Attention: this L is not loss function. Here it means the likelihood function.

3.2 Likelihood (code)

In Python, write a function to help you compute the likelihood in 3.1. Here is a prototype:

```python
def lik(x, y, a, b):
  pass
```

Hint1: You might need the `exp` function from the `numpy` package: `numpy.exp()`.

Hint2: To test if your code is correct. If you call `lik(x,y,1,0)`, you should have `0.0002937983603191943`

3.3 Log-likelihood (in words)

Now consider the likelihood function. Working with the original form is tedious. But we know that

$$\max L(a, b) \Leftrightarrow \max \log L(a, b).$$

What is the log of the likelihood (which is called the **log-likelihood**):

$$l(a, b) = \log L(a, b) =?$$

3.4 Derivatives (in words)

Let's compute the derivatives of that log likelihood. Please write them down:

$$\frac{\partial l(a,b)}{\partial a} = ?$$
$$\frac{\partial l(a,b)}{\partial b} = ?$$

3.5 Derivatives (code)

Please write two Python functions `la(x,y,a,b)` and `lb(x,y,a,b)` to compute the derivatives in Question 3.4.

3.6 Gradient descent (code)

We introduced gradient descent in class to find the **minimum** of a function. But our task now is to **maximize** the log-likelihood $l(a, b)$. In order to use gradient descent, let's convert our log-likelihood into a **negative log-likelihood**:

$$n(a,b) = -l(a,b)$$
$$\frac{\partial n(a,b)}{\partial a} = -\frac{\partial l(a,b)}{\partial a}$$
$$\frac{\partial n(a,b)}{\partial b} = -\frac{\partial l(a,b)}{\partial b}.$$

Set the initial values `a0=1` and `b0=0`. Use the learning rate `gamma=0.001` and implement gradient descent in the following loop:

```python
a0 = 1
b0 = 0
for _ in range(100):
  pass

print(ahat, bhat)
```

Note: You should get `ahat=0.792` and `bhat=-0.021`.

Attention! We are not setting the convergence rule. Just run the gradient descent algorithm 100 times.

3.7 Gradient descent with a stopping criterion (code)

This time, let's slightly change the code in Question 3.6. Instead of running the code 100000 times, let's use a `while` loop. Stop the loop if the function value satisfies $|l(a_{n+1}, b_{n+1}) - l(a_n, b_n)| < 0.01$. What are $\hat{a}$ and $\hat{b}$?