

## Assignment-3

### Python

#### **Q1. (Base Conversion)**

You might be familiar with converting a given number in binary (base,  $b = 2$ ) into its decimal equivalent. This question requires you to generalise it to any base  $b$ , and the program should be written in Python.

Given a string of a fractional number  $N_b$  in base  $b$ , convert it into its decimal equivalent  $N_D$ . You need to check if the input is a valid number and get rid of leading zeros from the input.

$-999999999 < N_D < 999999999, N_D \in \mathbb{R}$   
 $1 \leq b \leq 37$

Example-

For $N_b = \text{HELLO.PY}$	&	$b = 35;$	output, $N_D = 26137359.742041$ .
For $N_b = 00101.101$	&	$b = 2;$	output, $N_D = 5.625$ .
For $N_b = \text{GJDGXR}$	&	$b = 36;$	output, $N_D = 999999999$ .
For $N_b = -4G$	&	$b = 17;$	output, $N_D = -84$ .
For $N_b = \text{HELLO.PY}$	&	$b = 10;$	output, "Invalid Input".

**Important Note:** You are **NOT** allowed to use built in functions like `int()` etc.

#### **Q2. (Intro to ML)**

In this question, we will code a 1-D linear regression problem. We will provide pseudo-code here, the students are required to transform it into a python code. Use numpy library for array/vector/matrices etc.

The data files are `train.csv` and `test.csv`.

File structure:

The structure of both files are similar. `Train.csv` has  $n_{\text{train}} = 10^4$  rows and `test.csv` has  $n_{\text{test}} = 10^3$  rows, each row corresponding to one data point. Each row has two values separated by comma. The first value is feature and second value is label of the data point.

Example-

If one row of `train.txt` is :

4,7

Then feature,  $x = 4$  and label,  $y = 7$ .

## Pseudo-code

### Step-1:

- Read files train.csv
- Create vector -  $X_{\text{train}}$  (dim -  $n_{\text{train}} \times 1$ ) and vector -  $y_{\text{train}}$  (dim -  $n_{\text{train}} \times 1$ )
- Add a column to  $X_{\text{train}}$  so that its dimension becomes  $n_{\text{train}} \times 2$ . First column of  $X_{\text{train}}$  should be all 1 and 2nd column is the same as before adding extra column.

### Example -

```
X_train = [  
    2  
    3  
    4  
    ]  
New X_train = [  
    1  2  
    1  3  
    1  4  
    ]
```

### Step-2:

Generate a 2-D vector  $w$  (dim:  $2 \times 1$ ) initialised randomly with floating point numbers.

### Step-3:

Plot  $y$  vs  $x$  using matplotlib where  $x$  is the feature and  $y$  is the label read from the file train.csv.

Consider  $x' = [1 \ x]$  (prepending 1 to  $x$  to generate 2-dimensional  $x$ -vector)

On the same figure plot the line  $w^T x'$  vs  $x$ .

Your figure should have a dot corresponding to each datapoint  $(x,y)$  and a straight line on the plot corresponding to  $w^T x'$ .

### Step-4:

Set  $w_{\text{direct}} = (X_{\text{train}}^T \cdot X_{\text{train}})^{-1} \cdot X_{\text{train}}^T \cdot y_{\text{train}}$

$X_{\text{train}}$  is the  $n_{\text{train}} \times 2$  matrix defined earlier and  $y_{\text{train}}$  is the corresponding label vector.

Plot  $y$  vs  $x$  using matplotlib where  $x$  is the feature and  $y$  is the label read from the file train.csv.

Consider  $x' = [1 \ x]$  (prepending 1 to  $x$  to generate 2-dimensional  $x$ -vector)

On the same figure plot the line  $w_{\text{direct}}^T x'$  vs  $x$ .

Your figure should have a dot corresponding to each datapoint (x,y) and a straight line on the plot corresponding to  $w\_direct^T x'$ .

Step-5:

w - 2-dim vector initialised earlier (step-2)

Loop: for nepoch = 1 to N (N is the number of pass through the data (~10), play with it to find

best fit)

Loop : for j = 1 to n\_train

$w \leftarrow w - \eta (y - w^T x') x'$  ( $\eta = 0.0001$  students can change this value)

If  $j \% 100 == 0$

Then plot y vs x as earlier and use current value of w to plot  $w^T x'$  vs x

Step-6:

Finally redraw the plot as earlier with latest value of w.

Step-7:

- Read files test.csv
- Create vector - X\_test (dim - n\_test x 1) and vector - y\_test (dim - n\_test x 1)
- Add a column to X\_test so that its dimension becomes n\_test x 2. First column of X\_test should be all 1 and 2nd column is the same as before adding extra column.
- Let  $y\_pred1 = X\_test * w$  (w is the final value after doing step 5)
- Calculate root mean squared error between  $y\_pred1$  and  $y\_test$ .
- Let  $y\_pred2 = X\_test * w\_direct$
- Calculate root mean squared error between  $y\_pred2$  and  $y\_test$ .

### Derivation of Updates (optional reading)

In the loop of step 5 we did the following

$$w \leftarrow w - \eta (y - w^T x') x'$$

The objective of the above step is to reduce the least squared error.

Let error =  $0.5(y - w^T x')^2$

So, derivative w.r.t. w gives:  $(y - w^T x') x'$

We need to descend down the gradient to reach the minima, so we subtract eta times the above derivative from  $w$  to gradually reach minima. We set eta to small value because otherwise,  $w$  may overshoot the minima.  $w_{\text{direct}}$  can also be computed on the same line by setting derivative to zero (Google it!).