CS771
Introduction to Machine Learning
Indian Institute of Technology, Kanpur

Assignment 1

Submission Deadline:
Feb 18, 2023, 23:55hrs

Adi Pratap Singh (200036), Naman Singla (200619),
UJJAWAL GOYAL (201058), Ablokit (200030),
Ritik(200805)

Question 1

Solution

Let t_i denotes the total delay occurred while reaching from i=0 to i=i. (t_0=0)
t_i denotes the input to i^{th} XOR from the ring
a_i denotes the input to i^{th} XOR (i^{th} bit in challenge vector)

Let's use the following transformations:-

$$d_i = 1 - 2a_i \qquad \dots \text{(i)}$$
$$c_i = 1 - 2b_i \qquad \dots \text{(ii)}$$

NOTE:- b_i depends on the number set bits in a_0,...,a_{i-1}. If there are odd number of set bits then the value is reversed otherwise it remains the same. Hence, we can write:

$$c_i = c_0 \cdot \prod_{i=0}^{i=1} d_i \qquad \dots (iv)$$

Let $\delta^i_{b_i a_i}$ denotes the delay added by the i^{th} XOR.

We can state that:-

$$\delta^i_{b_i a_i} = \frac{(1+d_i)}{2} \cdot \left(\frac{(1+c_i)}{2}\delta^i_{00} + \frac{(1-c_i)}{2}\delta^i_{10} \right) + \frac{(1-d_i)}{2} \cdot \left(\frac{(1+c_i)}{2}\delta^i_{01} + \frac{(1-c_i)}{2}\delta^i_{11} \right)$$

$$= \frac{1}{4}\left(d_i(\delta^i_{00} + \delta^i_{10} - \delta^i_{01} - \delta^i_{11}) + c_i(\delta^i_{00} - \delta^i_{10} + \delta^i_{01} - \delta^i_{11}) + d_i c_i(\delta^i_{00} - \delta^i_{10} - \delta^i_{01} + \delta^i_{11}) + (\delta^i_{00} + \delta^i_{10} + \delta^i_{01} + \delta^i_{11}) \right)$$

$$\delta^i_{b_i a_i} = d_i\alpha_i + c_i\beta_i + d_i c_i \gamma_i + \zeta_i \qquad \dots (v)$$

Where,

$$\alpha_i = \frac{1}{4} d_i(\delta^i_{00} + \delta^i_{10} - \delta^i_{01} - \delta^i_{11})$$

$$\beta_i = \frac{1}{4} d_i(\delta^i_{00} - \delta^i_{10} + \delta^i_{01} - \delta^i_{11})$$

$$\gamma_i = \frac{1}{4} d_i(\delta^i_{00} - \delta^i_{10} - \delta^i_{01} + \delta^i_{11})$$

$$\zeta_i = \frac{1}{4} d_i(\delta^i_{00} + \delta^i_{10} + \delta^i_{01} + \delta^i_{11})$$

All defined on only i_{th} XOR.

Noe we can write , $t_{i+1} = t_{i+1} + \delta^i_{b_i a_i}$.

$$\Rightarrow t_r = \sum_{i=0}^{R-1} \delta^i_{b_i a_i} = \sum_{i=0}^{R-1} (d_i\alpha_i + c_i\beta_i + d_i c_i \gamma_i + \zeta_i) \qquad (t_0 = 0)$$

$$t_R|_{b_0=0} + t_R|_{b_0=1} = t_R|_{c_0=1} + t_R|_{c_0=-1}$$

$$= \sum_{i=0}^{R-1} (d_i\alpha_i + (\prod_{i=0}^{i=1} d_i)\beta_i + d_i(\prod_{i=0}^{i=1} d_i)\gamma_i + \zeta_i) + \sum_{i=0}^{R-1} (d_i\alpha_i - (\prod_{i=0}^{i=1} d_i)\beta_i - d_i(\prod_{i=0}^{i=1} d_i)\gamma_i + \zeta_i) \qquad \dots (iv)$$

$$\Rightarrow t_R|_{b_0=0} + t_R|_{b_0=1} = 2 \cdot \sum_{i=0}^{R-1} (d_i\alpha_i + \zeta_i) = W^T X + b \qquad \dots (vi)$$

Let X = $[d_i]$, W = $[2\alpha_i]$, b = $2\zeta_i$

frequency =

$$\frac{1}{t_R|_{b_0=0} + t_R|_{b_0=1}}$$

response =

$$\frac{1 + sign(freq_u - freq_l)}{2}$$

=

$$\frac{1 + sign((t_R|_{b_0=0} + t_R|_{b_0=1})_l - (t_R|_{b_0=0} + t_R|_{b_0=1})_u)}{2}$$

=

$$\frac{1 + sign((W^T X + b)_l - (W^T X + b)_u)}{2}$$

=

$$\frac{1 + sign((W^T X + b))}{2}$$

where, $freq_u$ = frequency of upper XOR and $freq_l$ = frequency of lower XOR.

Question 2

Solution

In order to extend the above linear model to crack Advanced XORRO PUF, we will create a separate model for each possible pair of PUF which might get selected by the multiplexers. Here, we must note that the same PUF can't be selected to form the set. Since the number of select bits is denoted by s, the number of the unique pair sets of PUFs is:

$$\begin{aligned} M &= \frac{2^s * 2^s}{2} - 2^s \\ &= 2^{s-1}(2^s - 1) \end{aligned} \tag{2.1}$$

A typical item of the training dataset contains the R and the select bits S_1, S_2. Hence using S_1, S_2 we can find the corresponding model to that pair of PUFs and train it with the input.
While encountering the testing dataset we will firstly figure out the model corresponding to pair of PUFs using S_1, S_2. Then we will use this model to predict the output for corresponding R.
To implement this we will construct a 2D array of models model at [i,j] will take S1=i and S2=j. We will also assume that i¿j, if i¡j then we will invert the response for that challenge. Doing so will decrease the model size as well as increase the accuracy.

Question 3

Solution

Submitted on google form

Question 4

Solution

Let us discuss the details of LinearSVM and Linear Classifier:

1. **LinearSVM**

   - For "Square Hinge Loss" as Loss function:

     (a) For C = 1, tol = 0.001, penalty=l1, and number of iterations = 3000
     Training time: 2.0876035690307617
     Accuracy : 0.9449

     (b) For C = 2, tol = 0.001, penalty=l1 and number of iterations = 5000
     Training time: 3.779726982116699
     Accuracy : 0.94415

     (c) For C = 1, tol = 0.1, penalty=l2 and number of iterations = 2000
     Training time: 0.6685817241668701
     Accuracy : 0.944675

     (d) For C = 2, tol = 0.1, penalty=l1 and number of iterations = 5000
     Training time: 0.6685817241668701
     Accuracy : 0.944675

   - For "Hinge Loss" as Loss function:

     (a) For C = 1, tol = 0.1 and number of iterations = 1000
     Training time: 0.6603553295135498
     Accuracy : 0.940825

     (b) For C = 2, tol = 0.1 and number of iterations = 2000
     Training time: 0.9010498523712158
     Accuracy : 0.940825

2. **Logistic Regression**

   - For solver="lbfgs":

     (a) For C = 1, tol = 0.001, penalty=l2, and number of iterations = 3000
     Training time: 0.7795505523681641
     Accuracy : 0.9479

(b) For C = 2, tol = 0.001, penalty=l2 and number of iterations = 5000
Training time: 0.9277830123901367
Accuracy : 0.94945

(c) For C = 5, tol = 0.001, penalty=l2 and number of iterations = 5000
Training time: 0.9344983100891113
Accuracy : 0.949825

(d) For C = 10, tol = 0.001, penalty=l2 and number of iterations = 5000
Training time: 1.2722845077514648
Accuracy : 0.94985

(e) For C = 20, tol = 0.001, penalty=l2 and number of iterations = 5000
Training time: 1.448258876800537
Accuracy : 0.949625

- For ”liblinear” as Loss function:

(a) For C = 20, penalty=l2, tol = 0.1 and number of iterations = 1000
Training time: 0.5792207717895508
Accuracy : 0.94775

(b) For C = 10, penalty=l2, tol = 0.1 and number of iterations = 1000
Training time: 0.524174690246582
Accuracy : 0.948525

(c) For C = 5, penalty=l2, tol = 0.1 and number of iterations = 1000
Training time: 0.47158122062683105
Accuracy : 0.948525

(d) For C = 2, penalty=l2, tol = 0.1 and number of iterations = 1000
Training time: 0.43340182304382324
Accuracy : 0.9477

- For solver=”newton-cg”:

(a) For C = 1, tol = 0.001, penalty=l2, and number of iterations = 1000
Training time: 1.0327095985412598
Accuracy : 0.94795

(b) For C = 2, tol = 0.001, penalty=l2 and number of iterations = 1000
Training time: 1.123051643371582

Accuracy : 0.94945

(c) For C = 5, tol = 0.001, penalty=l2 and number of iterations = 1000
Training time: 1.2377519607543945
Accuracy : 0.949825

(d) For C = 10, tol = 0.001, penalty=l2 and number of iterations = 1000
Training time: 1.23878812789917
Accuracy : 0.949825

(e) For C = 20, tol = 0.001, penalty=l2 and number of iterations = 1000
Training time: 1.2839670181274414
Accuracy : 0.949575