Introduction to ML (CS771), Winter 2024
Indian Institute of Technology Kanpur
Assignment Number 1

Group Name: Model Mavericks
Instructor: Purushottam Kar
Date: April 2, 2024

Assignment
1

Team Members:

1. Himanshu Karnatak 231110017
2. Abhinandan Singh Baghel 231110002
3. Souvik Mukherjee 231110405
4. Hemanshu Pandey 231110409
5. Vishal Kumar 231110058

1 Companion Arbiter PUF (CAR-PUF)

1.1 Detailed mathematical derivation

Our major goal is to find ϕ(x), in terms of 'x' and constants. We then need to find 'W' and 'b'. We need the number of dimensions 'D', in our 'W' and 'ϕ(x)' to exceed the value of '32', which is the no bits in our challenge, to each A-PUF's, i.e. we need to do feature scaling

The two A-PUF's timing delay is found out using linear model, as we did for XOR-PUF, and is stored in some value, say Δ_w and Δ_r, now we have a τ value, for our CAR-PUF. The condition we need to satisfy is

1. If $|\Delta_w$-$\Delta_r| \leq \tau$, Then –> Response of CAR-PUF = '0'
2. If $|\Delta_w$-$\Delta_r| > \tau$, Then –> Response of CAR-PUF = '1'

We'll need to leverage the two linear model's output to fit in this equation, which is the response of our CAR-PUF.

$$\frac{1 + \text{sign}(W^T\ \phi(c) + b)}{2} = r$$

- Equation 1

We need to create a new linear model, with increased features, which takes in challenge for our CAR-PUFF, and predicts the response.

Let (u, p),(v, q) be the two linear models that can exactly predict the outputs of the two arbiter-PUFs sitting inside the CAR-PUF.

Let, the equations

1. $U^T . x + p$, predict Δ_w
2. $V^T . x + q$, predict Δ_r

Mapping the A-PUFF's linear models to CAR-PUFF response

1. " Sign ($|\Delta_w - \Delta_r| - \tau \leq 0$) " **-> Gives Response '0' <-** " Sign($W^T \phi(c) + b \leq 0$)"
2. " Sign ($|\Delta_w - \Delta_r| - \tau > 0$) " **-> Gives Response '1' <-** " Sign($W^T \phi(c) + b > 0$)"

So, we have " Sign ($|\Delta_w - \Delta_r| - \tau$) " $\approx$ " Sign ($W^T \phi(c) + b$)"

Also, if we observe carefully, we always have,

$$\text{Sign}(|\Delta_w - \Delta_r| - \tau) == \text{Sign}((\Delta_w - \Delta_r)^2 - \tau^2)$$

1.2 Expanding the term " $(\Delta_w - \Delta_r)^2 - \tau^2$ "

Let the vector 'x' denote the 32 bit challenge.

$(\Delta_w - \Delta_r)^2 - \tau^2$

$(U^T . x + p - V^T . x - q)^2 - \tau^2$

$(U^T . x + p - V^T . x - q).(U^T . x + p - V^T . x - q) - \tau^2$

$[(U^T - V^T) . x + (p - q)].[(U^T - V^T) . x + (p - q)] - \tau^2$

$[(U^T - V^T) . x . (U^T - V^T) . x] + [(U^T - V^T) . x . (p-q)] + [(p-q). (U^T - V^T) . x] + [(p - q)^2] - \tau^2$

- Equation 2

We need to map **- Equation 2** to **Equation 1**

Let us assume,

1. $(U^T - V^T)$ as 'l' - Column Vector (32X1)
2. (p - q) as 'm' - Constant (1X1)

- Equation 2 now becomes:

$(1^T . x . 1^T . x) + (1^T . x . m) + (m. 1^T . x) + (m)^2 - \tau^2$

- Equation 3

1.3 Visualizing, 'Equation 3' using a 2-D and 3-D Euclidean space

Let Col-Matrix l and, Let Col-Matrix x be:

$$l = \begin{bmatrix} l1 \\ l2 \end{bmatrix}$$

$$x = \begin{bmatrix} x1 \\ x2 \end{bmatrix}$$

Placing them in Eqn. 2, we have:

$(l1.x1 + l2.x2).(l1.x1 + l2.x2) + 2. m. (l1.x1 + l2.x2) + m^2 - \tau^2$

$(l1^2 . x1^2)+ (l2^2 . x2^2) + (2.l1.l2 .x1.x2) + (2.m.l1.x1) + (2.m.l2.x2) + m^2 - \tau^2$

Looking carefully, we can see, we have

- Constant terms (like, m^2, τ^2, etc), all will map to Bias of our CAR-PUF model.
- x_i^2 terms,where $i \in [n]$, where 'n' is n-bit challenge to the PUF
- x_{ij} terms,where $i,j \in [n]$, where 'n' is n-bit challenge to the PUF, where $i \neq j$
- x_i terms,where $i \in [n]$, where 'n' is n-bit challenge to the PUF
- All the coefficients of x_i^2, x_{ij} , and x_i will map to the Weight of our CAR-PUF model.
- Since, we also have $x_{ij} == x_{ji}$, we can consider them together, as same feature.
- In 2-D Euclidean space, We have, 2 x_i^2 terms, $\binom{2}{2}$ x_{ij} terms, 2 x_i terms, and summing up all constants, we'll have one constant term.

Let Col-Matrix l and, Let Col-Matrix x in the 3-D Euclidean space be:

$$l = \begin{bmatrix} l1 \\ l2 \\ l3 \end{bmatrix}$$

$$x = \begin{bmatrix} x1 \\ x2 \\ x3 \end{bmatrix}$$

The 3-D Euclidean space's equation 3 will look like this:

$(1^T . x . 1^T . x) + (1^T . x . m) + (m. 1^T . x) + (m)^2 - \tau^2$

$(l1.x1 + l2.x2 + l3.x3).(l1.x1 + l2.x2 + l3.x3) + 2. m. (l1.x1 + l2.x2 + l3.x3) + m^2 - \tau^2$

$(l1^2 . x1^2) + (l2^2 . x2^2) + (l3^2 . x3^2) + (2.l1.l2 .x1.x2) + (2.l1.l3 .x1.x3) + (2.l2.l3 .x2.x3) + (2.m.l1.x1) + (2.m.l2.x2) + (2.m.l3.x3) + m^2 - \tau^2$

- In 3-D Euclidean space, We have, 3 x_i^2 terms, $\binom{3}{2}$ x_{ij} terms, 3 x_i terms, and summing up all constants, we'll have one constant term.

- So, we can observe a pattern here, In a N-D Euclidean space, We have, N x_i^2 terms, $\binom{N}{2}$ x_{ij} terms, N x_i terms, and summing up all constants, we'll have one constant term.

- So, for 32-D Euclidean space We'll have
- In 32-D Euclidean space, We have, 32 x_i^2 terms, $\binom{32}{2}$ x_{ij} terms, 32 x_i terms, and summing up all constants, we'll have one constant term.
- Summing up 32 + $\binom{32}{2}$ + 32. We have 560 features, after feature expansion, and one constant term.
- On careful observation, we can rule out all the x_i^2, as we always, change our input features as, (0 –> -1) and (1 –> +1), i.e. our input to feature is either '+1' or '-1', placing them in the x_i^2, we'll always have a '+1'. So using the x_i^2, we're always left with a "+ N.1" term (where 'n' is n-bit challenge to the PUF) which can can be summed up and placed inside the constant, as the feature squaring terms are only adding up a constant term to our model.
- Now we have, 32 + $\binom{32}{2}$ + 32 - 32 features. We have 528 features, after feature expansion, and one constant term.
- Now, these expanded features can be mapped to our $\phi(x)$, their coefficients to Weight and the constants to the Bias term of our CAR-PUF model.

2 The Python Code

The Python code is uploaded in Zipped format.

3 Outcomes of experiments (how various hyper-parameters affected training time and test accuracy)

Feature Dimensions of CAR-PUF model, in all the cases is 528.

No.	Model	my_fit() time	y_fit() time	1-acc	Accuracy in %
1	LinearSVC(C=1.0,penalty='l2',loss="hinge")	13.9786	0.4178	0.0116	98.8400
2	LinearSVC(C=7.0,penalty='l2',loss="hinge")	18.0690	0.4638	0.0106	98.9400
3	LinearSVC(C=1.0,penalty='l2',loss="squared_hinge")	17.4795	0.4423	0.0108	98.9200
4	LinearSVC(C=7.0,penalty='l2',loss="squared_hinge")	17.7801	0.6294	0.0102	98.9800
5	LinearSVC(C=0.5)	18.5890	0.4310	0.0102	98.9800
6	LinearSVC(C=1.0)	17.4961	0.4436	0.0109	98.9100
7	LinearSVC(C=5.0)	17.9272	0.4529	0.0102	98.9800
8	LinearSVC(C=10.0)	17.2468	0.4625	0.0112	98.8800
9	LinearSVC(C=15.0)	18.1448	0.4302	0.0111	98.8900
10	LinearSVC(C=20.0)	17.7691	0.5788	0.0106	98.9400
11	LogisticRegression(C=0.5)	3.3370	0.5265	0.0081	99.1900
12	LogisticRegression(C=1.0)	2.7665	0.4815	0.0080	99.2000
13	LogisticRegression(C=5.0)	3.2273	0.6879	0.0072	99.2800
14	LogisticRegression(C=10.0)	3.0794	0.4633	0.0074	99.2600
15	LogisticRegression(C=15.0)	3.2256	0.4985	0.0070	99.3000
16	LogisticRegression(C=20.0)	4.3639	0.4784	0.0069	99.3100
17	LinearSVC(C=1.0,tol=0.00001)	9.3351	0.4599	0.0092	99.0800
18	LinearSVC(C=1.0,tol=0.0001)	16.8530	0.4329	0.0091	99.0900
19	LinearSVC(C=1.0,tol=0.01)	17.3059	0.4367	0.0099	99.0100
20	LinearSVC(C=1.0,tol=1.0)	19.0229	0.4853	0.0094	99.0600
21	LinearSVC(C=1.0,tol=2.0)	17.8967	0.4552	0.0105	98.9500
22	LogisticRegression(C=15,tol=0.00001)	3.4964	0.4758	0.0070	99.3000
23	LogisticRegression(C=15,tol=0.0001)	4.1404	0.6782	0.0070	99.3000
24	LogisticRegression(C=15,tol=0.01)	3.8896	0.5068	0.0070	99.3000
25	LogisticRegression(C=15,tol=1.0)	3.1359	0.7025	0.0069	99.3100
26	LogisticRegression(C=15,tol=2.0)	3.2396	0.6824	0.0069	99.3100

Table 1: Model performance on different settings; All time in sec

3.1 Effect of Changing the Loss Hyperparameter in LinearSVC

Serial No. 1, 2, 3, 4 represent this effect.

Inference:

By changing the loss hyperparameter, we noticed that it doesn't change accuracy much , however the time taken to fit the model is lower in hinge loss as compared to that in the squared hinge loss significantly.

3.2 Effect of Setting C in LinearSVC and LogisticRegression

Serial No. 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16 represent this effect.

Inference:

It can be noticed that when we are using Logistic Regression , it takes very less time to train the model compared to when we are using LinearSVC, also Logistic Regression gives better accuracy compared to LinearSVC for all the values of C that we have tried. However changing the values of C in a particular model (LinearSVC or LogisticRegression) doesn't change either the time or accuracy by a large factor.

3.3 Effect of Changing tol in LinearSVC and LogisicRegression

Serial No. 17, 18, 19, 20, 21, 22, 23, 24, 25, 26 represent this effect.

Inference:

Here also we can see that LogisticRegression is taking less time compared to the LinearSVC , however accuracies in this case is almost the same for both. In LinearSVC as we increase the tolerance then time taken to fit the model increases, on the other hand it doesn't increase so much in the LogisticRegression model.