CS771 Assignment 1

ARCHIT GUPTA, ALLAN ROBEY, DIVYESH DEVANGKUMAR TRIPATHI, SUBHAJIT PANDAY, AVNISH TRIPATHI

TOTAL POINTS

59 / 65

QUESTION 1

1 XOR derivation 5 / 5

✓ **+ 3 pts** *A correct example of a pair of functions m, f that works*

✓ **+ 2 pts** *Derivation of the result*

- 1 pts Derivation does not take into account edge cases e.g. all 0s or all 1s inputs.

+ 0 pts Completely wrong or else unanswered

QUESTION 2

2 Sign derivation 5 / 5

✓ **+ 5 pts** *A proof of the result with sufficient calculations*

- 2 pts Minor mistakes in proof e.g. not taking into account edge cases e.g. all 0s

- 1 pts Insufficient calculations

+ 0 pts Completely wrong or else unanswered

QUESTION 3

3 Map derivation 10 / 10

✓ **+ 10 pts** *A proof of the result with sufficient calculations*

+ 5 pts BONUS: if the derived dimensionality is less than 100

- 2 pts Insufficient calculations

- 2 pts Minor mistakes e.g. not taking into account the bias term or wrong dimensionality or sign error

+ 0 pts Completely wrong or else unanswered

not generalized

QUESTION 4

4 Code evaluation 29 / 35

- 10 pts Illegal use of libraries or other non-permitted actions

+ 0 pts Completely wrong or else unanswered

+ 29 *Point adjustment*

GROUP NO: 43

Grading scheme for code:
Feature map dimensionality d: d < 200 (5 bonus), 200 < d <= 600 (2 bonus), d > 600 (0 bonus)
Hinge loss h: h < 1 (3 marks), 1 <= h < 10 (2 marks), h > 10 (1 mark) -- for all timeouts
Error rate e: e < 0.01 (3 marks), 0.01 <= e < 0.1 (2 marks), e > 0.1 (1 mark) -- for all timeouts

d = 729 : 0 bonus marks
For timeout t = 0.2 sec, h = 0.0309 : 3 marks
For timeout t = 0.2 sec, e = 0.0131 : 2 marks
For timeout t = 0.5 sec, h = 0.00521 : 3

marks

For timeout t = 0.5 sec, e = 0.0 : 3 marks

For timeout t = 1.0 sec, h = 0.000896 : 3 marks

For timeout t = 1.0 sec, e = 0.0 : 3 marks

For timeout t = 2.0 sec, h = 0.00676 : 3 marks

For timeout t = 2.0 sec, e = 0.00141 : 3 marks

For timeout t = 5.0 sec, h = 0.00715 : 3 marks

For timeout t = 5.0 sec, e = 0.003 : 3 marks

TOTAL: 29 marks

QUESTION 5

5 Hyperparameter description 5 / 5

✓ **+ 3 pts** *Enumeration of all hyperparameters and values considered for those hyperparameters*

✓ **+ 2 pts** *Description of how the best value was obtained for each hyperparameter*

+ 0 pts Completely wrong or else unanswered

QUESTION 6

6 Convergence plot 5 / 5

✓ **+ 5 pts** *A digitally generated plot is provided*

- 1 pts Minor mistakes like unlabeled x axis or y axis

+ 0 pts Completely wrong or else unanswered

+ 3 pts Not the classification accuracy graaph

CS 771 Introduction to Machine Learning
Assignment 1

Group Name: Beginners2

Archit Gupta (21111015) **Avnish Tripathi (22111014)** **Allan Robey (22111007)**

Devesh Tripathi (22111020) **Subhajit Panday (22111058)**

Problem (Odd-even rules don't seem to add anything new). Melbo was a bit perturbed that the arbiter PUF model for verification was broken so easily using a simple linear model. In an effort to make an unbreakable PUF, Melbo came up with a simple but promising idea. Recall that a PUF is a chain of say k multiplexers, each of which either swaps the lines or keeps them intact, depending on what is the challenge bit fed into that multiplexer. The multiplexers each have delays which are hard to replicate but consistent. If the top signal reaches the finish line first (whether it is blue or red does not matter), the response is 0 else if the bottom signal reaches first (again, color does not matter), the response is 1.

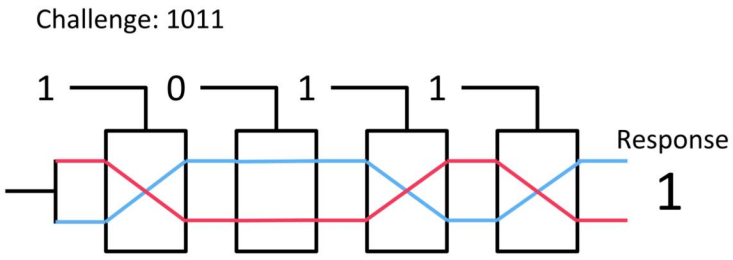


Figure 1: A simple arbiter PUF with 4 multiplexers

Melbo's puffier PUF takes 3 PUFs and feed the same challenge into all of them. However, since the 3 PUFs have 3 different set of multiplexers with different delays, the 3 PUFs need not give the same response even if they are given the same challenge. If an odd number of PUFs (i.e. any one PUF or all three PUFs) have the response 1, Melbo's puffier PUF would give a response 1 and if an even number of PUFs (i.e. no PUF or any two PUFs) have the response 1, Melbo's puffier PUF would give a response 0. This is often called the XOR operation (which stands for eXclusive OR) and Melbo's puffier PUF is also known as a XOR-PUF.

Preprint. Under review.

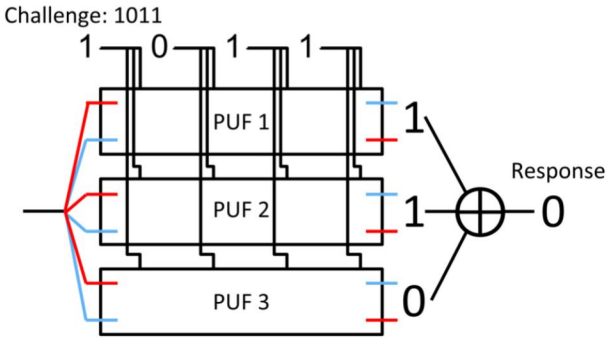


Figure 2: A XOR PUF with 3 PUFs, each with 4 multiplexers

Melbo thinks that with multiple PUFs and such a complicated odd-even rule in place, there is no way any machine learning model, let alone a linear one, can predict the responses if given a few thousand challenge-response pairs. Your job is to prove Melbo wrong! You will do this by showing that even in this case, there does exist a linear model that can perfectly predict the responses of a XOR-PUF and this linear model can be learnt if given enough challenge-response pairs (CRPs).

1 Question 1

Give a mathematical derivation to show the exists a way to map the binary digits 0, 1 to signs -1, +1 as say, m : $\{0, 1\} \rightarrow \{-1, +1\}$ and another way f : $\{-1, +1\} \rightarrow \{0, 1\}$ to map signs to bits (not that m and f need not be inverses of each other) so that for any set of binary digits $b_1, b_2, ..., b_n$ for any $n \in N$, we have

$$XOR(b_1, ..., b_n) = f(\prod_{i=1}^{n} m(b_i)).$$

Thus, the XOR function is not that scary – it is essentially a product. (5 marks)

Answer Function m: $\{0, 1\} \rightarrow \{-1, +1\}$
Let m(x) = 1 - 2x ; such that it maps the binary digits 0, 1 to +1, -1 respectively.

Function f: $\{-1, +1\} \rightarrow \{0, 1\}$
Let f(x) = (1 - x) / 2 ; such that it maps +1, -1 to binary digits 0, 1 respectively.

L.H.S:

$$XOR(b_1, ..., b_n)$$
$$= b_1 \oplus b_2 \oplus b_3 \oplus .... \oplus b_n$$

R.H.S:

$$f(\prod_{i=1}^{n} m(b_i))$$
$$= f(m(b_1) \cdot m(b_2) \cdot m(b_3) \cdot ... \cdot m(b_n))$$
$$= f((1 - 2b_1) \cdot (1 - 2b_2) \cdot (1 - 2b_3) \cdot ... \cdot (1 - 2b_n))$$

Case 1: Even number of 1's in the binary input.
L.H.S = 0 (∵ XOR of even number of 1's will be equal to 0)
Now, R.H.S:
If b = 1, then (1-2b) = -1
If b = 0, then (1-2b) = 1
Therefore, the expression becomes
f(-1 * -1 * -1 * ... (2k times) * 1 * 1 * 1 * .. (n-2k times))
= f(1) = (1-1)/2 = 0

$\therefore$ L.H.S = R.H.S

Case 2: Odd number of 1's in the binary input.
L.H.S = 1 ($\because$ XOR of odd number of 1's will be equal to 1)
Now, R.H.S:
f(-1 * -1 * -1 * ... (2k+1 times) * 1 * 1 * 1 * .. (n-2k-1 times)) = f(-1 * 1) = f(-1) = (1-(-1))/2 = 1
$\therefore$ L.H.S = R.H.S

Thus from case1 and case2 we've shown that there exists a way to map binary digits 0, 1 to signs +1, -1 using function m : $\{0, 1\} \rightarrow \{-1, +1\}$ and function f : $\{-1, +1\} \rightarrow \{0, 1\}$ to map signs +1, -1 to 0,1 so that for any set of binary digits $b_1, b_2, ..., b_n$ for any $n \in N$, we have

$$XOR(b_1, ..., b_n) = f(\prod_{i=1}^{n} m(b_i)).$$

2 Question 2

Let (**u**,a), (**v**,b), (**w**,c) be the three linear models that can exactly predict the outputs of the three individual PUFs sitting inside the XOR-PUF. For sake of simplicity, let us hide the bias term inside the model vector by adding a unit dimension to the original feature vector so that we have $\tilde{\mathbf{u}} = [\mathbf{u}, a]$, $\tilde{\mathbf{v}} = [\mathbf{v}, b]$, $\tilde{\mathbf{w}} = [\mathbf{w}, c]$, $\tilde{\mathbf{x}} = [\mathbf{x}, 1] \in R^9$. The above calculation shows that the response of the XOR-PUF can be easily obtained (by applying f) if we are able to get hold of the following quantity:

$$\text{sign}(\tilde{\mathbf{u}}^T\tilde{\mathbf{x}}) \text{ . sign}(\tilde{\mathbf{v}}^T\tilde{\mathbf{x}}) \text{ . sign}(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})$$

To exploit the above result, first give a mathematical proof that for any real numbers (that could be positive, negative, zero) $r_1, r_2, ..., r_n$ for any $n \in N$, we always have

$$\prod_{i=1}^{n} sign(r_i) = sign(\prod_{i=1}^{n} r_i)$$

Assume that sign(0) = 0. Make sure you address all edge cases in your calculations e.g. if one or more of the numbers is 0. (5 marks)

Answer To proof:

$$\prod_{i=1}^{n} sign(r_i) = sign(\prod_{i=1}^{n} r_i)$$

where,$r_1, r_2, ..., r_n$are real numbers and $n \in N$

Proof using Mathematical Induction -

Basis: Checking the validity of the statement for n=1
L.H.S:

$$\prod_{i=1}^{1} \text{sign } (r_i) = \text{sign } (r_i)$$

R.H.S:

$$\text{sign } (\prod_{i=1}^{1} r_i) = \text{sign } (r_i)$$

hence, L.H.S = R.H.S
$\therefore$ statement is true for n=1.

Hypothesis: Let statement is true for n=k. i.e.

$$\prod_{i=1}^{k} \text{sign } (r_i) = \text{sign } (\prod_{i=1}^{k} r_i)$$

1 XOR derivation **5 / 5**

✓ **+ 3 pts** *A correct example of a pair of functions m, f that works*

✓ **+ 2 pts** *Derivation of the result*

- 1 pts Derivation does not take into account edge cases e.g. all 0s or all 1s inputs.

+ 0 pts Completely wrong or else unanswered

$\therefore$ L.H.S = R.H.S

Case 2: Odd number of 1's in the binary input.
L.H.S = 1 ($\because$ XOR of odd number of 1's will be equal to 1)
Now, R.H.S:
f(-1 * -1 * -1 * ... (2k+1 times) * 1 * 1 * 1 * .. (n-2k-1 times)) = f(-1 * 1) = f(-1) = (1-(-1))/2 = 1
$\therefore$ L.H.S = R.H.S

Thus from case1 and case2 we've shown that there exists a way to map binary digits 0, 1 to signs +1, -1 using function m : $\{0, 1\} \rightarrow \{-1, +1\}$ and function f : $\{-1, +1\} \rightarrow \{0, 1\}$ to map signs +1, -1 to 0,1 so that for any set of binary digits $b_1, b_2, ..., b_n$ for any $n \in N$, we have

$$XOR(b_1, ..., b_n) = f(\prod_{i=1}^{n} m(b_i)).$$

2 Question 2

Let (**u**,a), (**v**,b), (**w**,c) be the three linear models that can exactly predict the outputs of the three individual PUFs sitting inside the XOR-PUF. For sake of simplicity, let us hide the bias term inside the model vector by adding a unit dimension to the original feature vector so that we have $\tilde{\mathbf{u}} = [\mathbf{u}, a]$, $\tilde{\mathbf{v}} = [\mathbf{v}, b]$, $\tilde{\mathbf{w}} = [\mathbf{w}, c]$, $\tilde{\mathbf{x}} = [\mathbf{x}, 1] \in R^9$. The above calculation shows that the response of the XOR-PUF can be easily obtained (by applying f) if we are able to get hold of the following quantity:

$$\text{sign}(\tilde{\mathbf{u}}^T\tilde{\mathbf{x}}) \text{ . sign}(\tilde{\mathbf{v}}^T\tilde{\mathbf{x}}) \text{ . sign}(\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})$$

To exploit the above result, first give a mathematical proof that for any real numbers (that could be positive, negative, zero) $r_1, r_2, ..., r_n$ for any $n \in N$, we always have

$$\prod_{i=1}^{n} sign(r_i) = sign(\prod_{i=1}^{n} r_i)$$

Assume that sign(0) = 0. Make sure you address all edge cases in your calculations e.g. if one or more of the numbers is 0. (5 marks)

Answer To proof:

$$\prod_{i=1}^{n} sign(r_i) = sign(\prod_{i=1}^{n} r_i)$$

where,$r_1, r_2, ..., r_n$are real numbers and $n \in N$

Proof using Mathematical Induction -

Basis: Checking the validity of the statement for n=1
L.H.S:

$$\prod_{i=1}^{1} \text{sign } (r_i) = \text{sign } (r_i)$$

R.H.S:

$$\text{sign } (\prod_{i=1}^{1} r_i) = \text{sign } (r_i)$$

hence, L.H.S = R.H.S
$\therefore$ statement is true for n=1.

Hypothesis: Let statement is true for n=k. i.e.

$$\prod_{i=1}^{k} \text{sign } (r_i) = \text{sign } (\prod_{i=1}^{k} r_i)$$

$$\text{i.e. sign } (r_1) \cdot \text{sign } (r_2) \cdot ... \cdot \text{sign } (r_k) = \text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) \qquad (i)$$

Induction Step: We will now prove the statement for n = k+1 i.e.

$$\prod_{i=1}^{k+1} \text{sign } (r_i) = \text{sign } (\prod_{i=1}^{k+1} r_i)$$

L.H.S:

$$\prod_{i=1}^{k+1} \text{sign } (r_i)$$
$$= \text{sign } (r_1) \cdot \text{sign } (r_2) \cdot ... \cdot \text{sign } (r_k) \cdot \text{sign } (r_{k+1})$$
$$= \text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) \cdot \text{sign } (r_{k+1}) \qquad \text{(From (i))}$$

R.H.S:

$$\text{sign } (\prod_{i=1}^{k+1} r_i)$$
$$= \text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k \cdot r_{k+1})$$

Now, let us consider the following cases:

Case 1: Atleast one r_i is 0 where i = 0 to k+1
L.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) \cdot \text{sign } (r_{k+1})$$
$$\text{Either, } r_1 \cdot r_2 \cdot ... \cdot r_k = 0 \text{ or } r_{k+1} = 0$$
$$\therefore \text{Either, sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) = 0 \text{ or sign } (r_{k+1}) = 0$$
$$\therefore \text{L.H.S = 0}$$

R.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k \cdot r_{k+1})$$
$$\text{Now, } r_1 \cdot r_2 \cdot ... \cdot r_k \cdot r_{k+1} = 0$$
$$\therefore \text{R.H.S = 0}$$

Case 2: $r_1 \cdot r_2 \cdot ... \cdot r_k$ is negative ; r_{k+1} is negative
L.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) \cdot \text{sign } (r_{k+1})$$
$$= -1 * -1$$
$$= 1$$

R.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k \cdot r_{k+1})$$
$$= \text{sign (negative * negative)}$$
$$= \text{sign (positive)}$$
$$= 1$$

Case 3: $r_1 \cdot r_2 \cdot ... \cdot r_k$ is negative ; r_{k+1} is positive
L.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) \cdot \text{sign } (r_{k+1})$$
$$= -1 * +1$$
$$= -1$$

R.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k \cdot r_{k+1})$$
$$= \text{sign (negative * positive)}$$

$$= \text{sign (negative)}$$

$$= -1$$

Case 4: $r_1 \cdot r_2 \cdot ... \cdot r_k$ is positive ; r_{k+1} is negative
L.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) \cdot \text{sign } (r_{k+1})$$

$$= +1 * -1$$

$$= -1$$

R.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k \cdot r_{k+1})$$

$$= \text{sign (positive * negative)}$$

$$= \text{sign (negative)}$$

$$= -1$$

Case 5: $r_1 \cdot r_2 \cdot ... \cdot r_k$ is positive ; r_{k+1} is positive
L.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) \cdot \text{sign } (r_{k+1})$$

$$= +1 * +1$$

$$= +1$$

R.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k \cdot r_{k+1})$$

$$= \text{sign (positive * positive)}$$

$$= \text{sign (positive)}$$

$$= +1$$

In all the 5 cases, we got L.H.S = R.H.S i.e. $\prod_{i=1}^{k+1} \text{sign } (r_i) = \text{sign } (\prod_{i=1}^{k+1} r_i)$. Thus, statement is true for n=k+1.
Hence, by the principal of mathematical induction we proved that

$$\prod_{i=1}^{n} sign(r_i) = sign(\prod_{i=1}^{n} r_i)$$

where,$r_1, r_2, ..., r_n$are real numbers and $n \in N$

3 Question 3

The above calculation tells us that all we need to get hold of is the following quantity

$$(\tilde{\mathbf{u}}^T \tilde{\mathbf{x}}) \,.\, (\tilde{\mathbf{v}}^T \tilde{\mathbf{x}}) \,.\, (\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})$$

Now show that the above can be expressed as a linear model but possibly in a different dimensional space. Show that there exists a dimensionality D such that D depends only on the number of PUFs (in this case 3) and the dimensionality of $\tilde{\mathbf{x}}$ (in this case 8 + 1 = 9) and there exists a way to map 9 dimensional vectors to D dimensional vectors as $\phi : R^9 \rightarrow R^D$ such that for any triple $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{w}})$, there always exists a vector $W \in R^D$ such that for every $\tilde{\mathbf{x}} \in R^9$, we have $(\tilde{\mathbf{u}}^T \tilde{\mathbf{x}}) \,.\, (\tilde{\mathbf{v}}^T \tilde{\mathbf{x}}) \,.\, (\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) = \mathbf{W}^T \phi(\tilde{\mathbf{x}})$ (10 marks)

2 Sign derivation **5 / 5**

✓ **+ 5 pts** *A proof of the result with sufficient calculations*

- 2 pts Minor mistakes in proof e.g. not taking into account edge cases e.g. all 0s

- 1 pts Insufficient calculations

+ 0 pts Completely wrong or else unanswered

$$= \text{sign (negative)}$$

$$= -1$$

Case 4: $r_1 \cdot r_2 \cdot ... \cdot r_k$ is positive ; r_{k+1} is negative
L.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) \cdot \text{sign } (r_{k+1})$$

$$= +1 * -1$$

$$= -1$$

R.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k \cdot r_{k+1})$$

$$= \text{sign (positive * negative)}$$

$$= \text{sign (negative)}$$

$$= -1$$

Case 5: $r_1 \cdot r_2 \cdot ... \cdot r_k$ is positive ; r_{k+1} is positive
L.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k) \cdot \text{sign } (r_{k+1})$$

$$= +1 * +1$$

$$= +1$$

R.H.S:

$$\text{sign } (r_1 \cdot r_2 \cdot ... \cdot r_k \cdot r_{k+1})$$

$$= \text{sign (positive * positive)}$$

$$= \text{sign (positive)}$$

$$= +1$$

In all the 5 cases, we got L.H.S = R.H.S i.e. $\prod_{i=1}^{k+1} \text{sign } (r_i) = \text{sign } (\prod_{i=1}^{k+1} r_i)$. Thus, statement is true for n=k+1.
Hence, by the principal of mathematical induction we proved that

$$\prod_{i=1}^{n} sign(r_i) = sign(\prod_{i=1}^{n} r_i)$$

where,$r_1, r_2, ..., r_n$are real numbers and $n \in N$

3 Question 3

The above calculation tells us that all we need to get hold of is the following quantity

$$(\tilde{\mathbf{u}}^T \tilde{\mathbf{x}}) \,.\, (\tilde{\mathbf{v}}^T \tilde{\mathbf{x}}) \,.\, (\tilde{\mathbf{w}}^T \tilde{\mathbf{x}})$$

Now show that the above can be expressed as a linear model but possibly in a different dimensional space. Show that there exists a dimensionality D such that D depends only on the number of PUFs (in this case 3) and the dimensionality of $\tilde{\mathbf{x}}$ (in this case 8 + 1 = 9) and there exists a way to map 9 dimensional vectors to D dimensional vectors as $\phi : R^9 \rightarrow R^D$ such that for any triple $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{w}})$, there always exists a vector $W \in R^D$ such that for every $\tilde{\mathbf{x}} \in R^9$, we have $(\tilde{\mathbf{u}}^T \tilde{\mathbf{x}}) \,.\, (\tilde{\mathbf{v}}^T \tilde{\mathbf{x}}) \,.\, (\tilde{\mathbf{w}}^T \tilde{\mathbf{x}}) = \mathbf{W}^T \phi(\tilde{\mathbf{x}})$ (10 marks)

Hint: First try solving this for the simpler case where there are only 2 PUFs. If we expand the terms of $(\tilde{\mathbf{u}}^T\tilde{\mathbf{x}})\ (\tilde{\mathbf{v}}^T\tilde{\mathbf{x}}) = (\sum_{j=1}^{9} \tilde{u_j}\tilde{x_j})\ (\sum_{j=1}^{9} \tilde{v_j}\tilde{x_j}))$, we get an expression of the form $\sum_{j=1}^{9}\sum_{k=1}^{9} \tilde{u_j}\tilde{v_k}\tilde{x_j}\tilde{x_k}$. Thus, if we create a $9^2 = 81$ -dimensional function that maps

$$\tilde{\mathbf{x}} = (\tilde{x_1}, ..., \tilde{x_9}) \text{ to } \phi(\tilde{\mathbf{x}}) = (\tilde{x_1}\tilde{x_1}, \tilde{x_1}\tilde{x_2}, ..., \tilde{x_1}\tilde{x_9}, \tilde{x_2}\tilde{x_1}, ..., \tilde{x_9}\tilde{x_9}) \ ,$$

then we are done since we can now get $(\tilde{\mathbf{u}}^T\tilde{\mathbf{x}})\ .\ (\tilde{\mathbf{v}}^T\tilde{\mathbf{x}}) = \mathbf{W}^T\phi(\tilde{\mathbf{x}})$ by taking

$$\mathbf{W} = (\tilde{u_1}\tilde{v_1}, \tilde{u_1}\tilde{v_2}, ..., \tilde{u_1}\tilde{v_9}, \tilde{u_2}\tilde{v_1}, ..., \tilde{u_9}\tilde{v_9})$$

Closely understand the trick in this simpler case and then extend it to the case of 3 PUFs to solve this part of the problem. Give detailed calculations for your solution.

Answer

$$\begin{aligned}
&(\tilde{\mathbf{u}}^T\tilde{\mathbf{x}})\ .\ (\tilde{\mathbf{v}}^T\tilde{\mathbf{x}})\ .\ (\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}) \\
&= (\sum_{i=1}^{9} \tilde{u_i}\tilde{x_i})\ (\sum_{i=1}^{9} \tilde{v_i}\tilde{x_i})\ (\sum_{i=1}^{9} \tilde{w_i}\tilde{x_i}) \\
&= \sum_{i=1}^{9}\sum_{j=1}^{9}\sum_{k=1}^{9} \tilde{u_i}\tilde{v_j}\tilde{w_k}\tilde{x_i}\tilde{x_j}\tilde{x_k}
\end{aligned}$$

Here, we've created a $9^3 = 729$ -dimensional function that maps

$$\begin{gathered}
\tilde{\mathbf{x}} = (\tilde{x_1}, ..., \tilde{x_9}) \text{ to } \phi(\tilde{\mathbf{x}}) \text{ where, } \phi(\tilde{\mathbf{x}}) = \\
(\tilde{x_1}\tilde{x_1}\tilde{x_1}, \tilde{x_1}\tilde{x_1}\tilde{x_2}, \tilde{x_1}\tilde{x_1}\tilde{x_3}, ..., \tilde{x_1}\tilde{x_1}\tilde{x_9}, \\
\tilde{x_1}\tilde{x_2}\tilde{x_1}, \tilde{x_1}\tilde{x_2}\tilde{x_2}, \tilde{x_1}\tilde{x_2}\tilde{x_3}, ..., \tilde{x_1}\tilde{x_2}\tilde{x_9}, \\
. \\
\tilde{x_1}\tilde{x_9}\tilde{x_1}, \tilde{x_1}\tilde{x_9}\tilde{x_2}, \tilde{x_1}\tilde{x_9}\tilde{x_3}, ..., \tilde{x_1}\tilde{x_9}\tilde{x_9}, \\
\tilde{x_2}\tilde{x_1}\tilde{x_1}, \tilde{x_2}\tilde{x_1}\tilde{x_2}, \tilde{x_2}\tilde{x_1}\tilde{x_3}, ..., \tilde{x_2}\tilde{x_1}\tilde{x_9}, \\
\tilde{x_2}\tilde{x_2}\tilde{x_1}, \tilde{x_2}\tilde{x_2}\tilde{x_2}, \tilde{x_2}\tilde{x_2}\tilde{x_3}, ..., \tilde{x_2}\tilde{x_2}\tilde{x_9}, \\
. \\
. \\
. \\
\tilde{x_9}\tilde{x_9}\tilde{x_1}, \tilde{x_9}\tilde{x_9}\tilde{x_2}, \tilde{x_9}\tilde{x_9}\tilde{x_3}, ..., \tilde{x_9}\tilde{x_9}\tilde{x_9})
\end{gathered}$$

If we take $\mathbf{W} =$

$$\begin{gathered}
(\tilde{u_1}\tilde{v_1}\tilde{w_1}, \tilde{u_1}\tilde{v_1}\tilde{w_2}, \tilde{u_1}\tilde{v_1}\tilde{w_3}, ..., \tilde{u_1}\tilde{v_1}\tilde{w_9}, \\
\tilde{u_1}\tilde{v_2}\tilde{w_1}, \tilde{u_1}\tilde{v_2}\tilde{w_2}, \tilde{u_1}\tilde{v_2}\tilde{w_3}, ..., \tilde{u_1}\tilde{v_2}\tilde{w_9}, \\
. \\
\tilde{u_1}\tilde{v_9}\tilde{w_1}, \tilde{u_1}\tilde{v_9}\tilde{w_2}, \tilde{u_1}\tilde{v_9}\tilde{w_3}, ..., \tilde{u_1}\tilde{v_9}\tilde{w_9}, \\
\\
\tilde{u_2}\tilde{v_1}\tilde{w_1}, \tilde{u_2}\tilde{v_1}\tilde{w_2}, \tilde{u_2}\tilde{v_1}\tilde{w_3}, ..., \tilde{u_2}\tilde{v_1}\tilde{w_9}, \\
\tilde{u_2}\tilde{v_2}\tilde{w_1}, \tilde{u_2}\tilde{v_2}\tilde{w_2}, \tilde{u_2}\tilde{v_2}\tilde{w_3}, ..., \tilde{u_2}\tilde{v_2}\tilde{w_9}, \\
. \\
. \\
. \\
\tilde{u_9}\tilde{v_9}\tilde{w_1}, \tilde{u_9}\tilde{v_9}\tilde{w_2}, \tilde{u_9}\tilde{v_9}\tilde{w_3}, ..., \tilde{u_9}\tilde{v_9}\tilde{w_9})
\end{gathered}$$

Then, we get

$$(\tilde{\mathbf{u}}^T\tilde{\mathbf{x}})\ .\ (\tilde{\mathbf{v}}^T\tilde{\mathbf{x}})\ .\ (\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}) = \mathbf{W}^T\phi(\tilde{\mathbf{x}})$$

Thus, the given expression can be expressed as a linear model in a D = 729 = 9^3 dimensional space where 3 is the number of PUFs and 9 is the dimensionality of $\tilde{\mathbf{x}}$. Thus, it is shown that the dimensionality D of a linear model for the expression $(\tilde{\mathbf{u}}^T\tilde{\mathbf{x}})\ .\ (\tilde{\mathbf{v}}^T\tilde{\mathbf{x}})\ .\ (\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})$ depends only upon the number of PUFs and dimensionality of $\tilde{\mathbf{x}}$ and a linear model can predict the responses of Melbo's Puffier PUF.

3 Map derivation **10 / 10**

✓ **+ 10 pts** *A proof of the result with sufficient calculations*

+ 5 pts BONUS: if the derived dimensionality is less than 100

- 2 pts Insufficient calculations

- 2 pts Minor mistakes e.g. not taking into account the bias term or wrong dimensionality or sign error

+ 0 pts Completely wrong or else unanswered

not generalized

4 Question 4

Answer Code in Python file

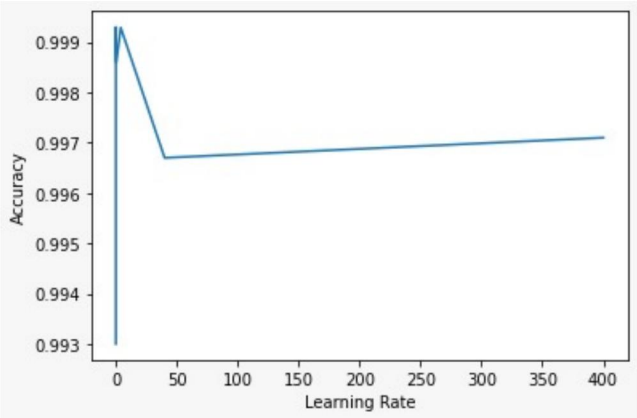
5 Question 5

For the method you implemented, describe in your PDF report what were the hyper-parameters e.g. step length, policy on choosing the next coordinate if doing SDCA, mini-batch, size if doing MBSGD etc and how did you arrive at the best values for the hyper-parameters, e.g. you might say "We used step length at time t to be η/ t where we checked for η = 0.1,0.2,0.5,1,2,5 using held out validation and found η = 2 to work the best". For another example, you might say, "We tried random and cyclic coordinate selection choices and found cyclic to work best using 5-fold cross validation". Thus, you must tell us among which hyperparameter choices did you search for the best and how. (5 marks)

Answer We have used SPGD to find the linear model.
Hyperparameters include initial weight matrix, learning rate and Lambda.

For determining initial weight vector:
Normal or Naïve Initialization- In normal distribution weights can be a part of normal or gaussian distribution with mean as zero and a unit standard deviation.Random initialization is done so that convergence is not to a false minima.
Reference: https://www.numpyninja.com/post/weight-initialization-techniques

For determining learning rate:
For determining the learning rate we first, made use of Armijo Goldstein rule wherein we started with some large learning rate and tried few values in decreasing order until the objective value has a sufficient reduction. and then used random search to find the optimal value of the parameter. Given below is a graph for determining the learning rate using Armijo Goldstein rule.



Now, we narrowed down our search for the optimal value using random search in the range [0.001, 0.002, 0.003, 0.004, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

4 Code evaluation 29 / 35

- 10 pts Illegal use of libraries or other non-permitted actions

+ 0 pts Completely wrong or else unanswered

+ 29 *Point adjustment*

GROUP NO: 43

Grading scheme for code:

Feature map dimensionality d: d < 200 (5 bonus), 200 < d <= 600 (2 bonus), d > 600 (0 bonus)

Hinge loss h: h < 1 (3 marks), 1 <= h < 10 (2 marks), h > 10 (1 mark) -- for all timeouts

Error rate e: e < 0.01 (3 marks), 0.01 <= e < 0.1 (2 marks), e > 0.1 (1 mark) -- for all timeouts

d = 729 : 0 bonus marks

For timeout t = 0.2 sec, h = 0.0309 : 3 marks

For timeout t = 0.2 sec, e = 0.0131 : 2 marks

For timeout t = 0.5 sec, h = 0.00521 : 3 marks

For timeout t = 0.5 sec, e = 0.0 : 3 marks

For timeout t = 1.0 sec, h = 0.000896 : 3 marks

For timeout t = 1.0 sec, e = 0.0 : 3 marks

For timeout t = 2.0 sec, h = 0.00676 : 3 marks

For timeout t = 2.0 sec, e = 0.00141 : 3 marks

For timeout t = 5.0 sec, h = 0.00715 : 3 marks

For timeout t = 5.0 sec, e = 0.003 : 3 marks

TOTAL: 29 marks

4 Question 4

Answer Code in Python file

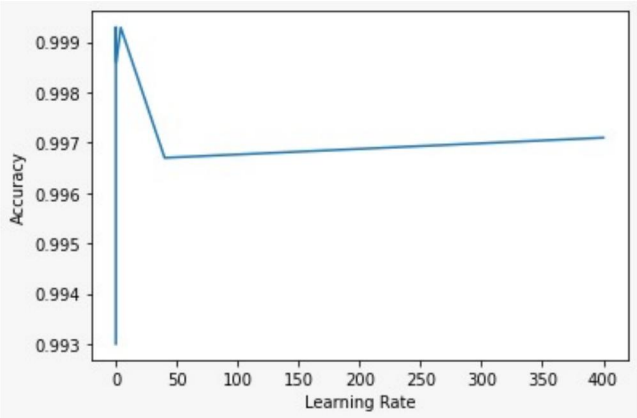
5 Question 5

For the method you implemented, describe in your PDF report what were the hyper-parameters e.g. step length, policy on choosing the next coordinate if doing SDCA, mini-batch, size if doing MBSGD etc and how did you arrive at the best values for the hyper-parameters, e.g. you might say "We used step length at time t to be η/ t where we checked for η = 0.1,0.2,0.5,1,2,5 using held out validation and found η = 2 to work the best". For another example, you might say, "We tried random and cyclic coordinate selection choices and found cyclic to work best using 5-fold cross validation". Thus, you must tell us among which hyperparameter choices did you search for the best and how. (5 marks)

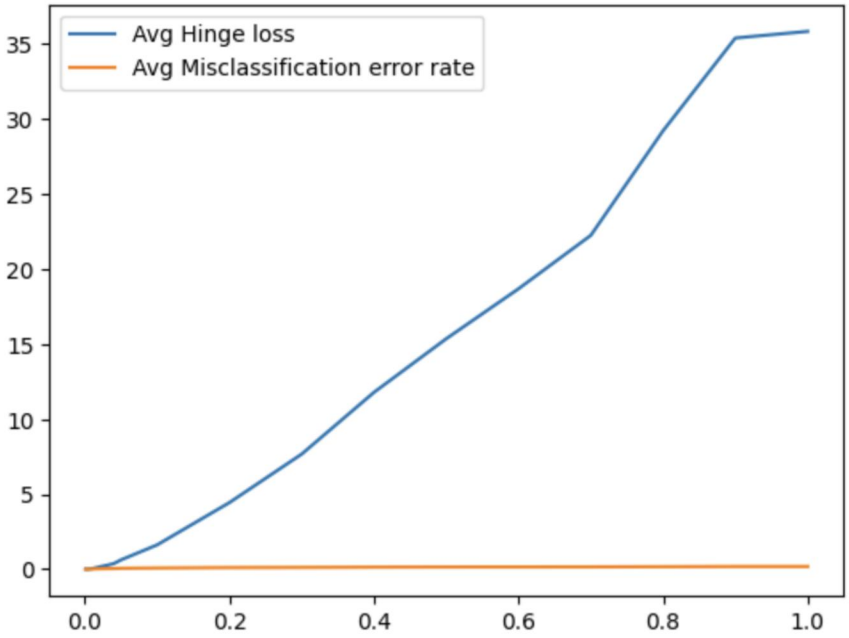
Answer We have used SPGD to find the linear model.
Hyperparameters include initial weight matrix, learning rate and Lambda.

For determining initial weight vector:
Normal or Naïve Initialization- In normal distribution weights can be a part of normal or gaussian distribution with mean as zero and a unit standard deviation.Random initialization is done so that convergence is not to a false minima.
Reference: https://www.numpyninja.com/post/weight-initialization-techniques

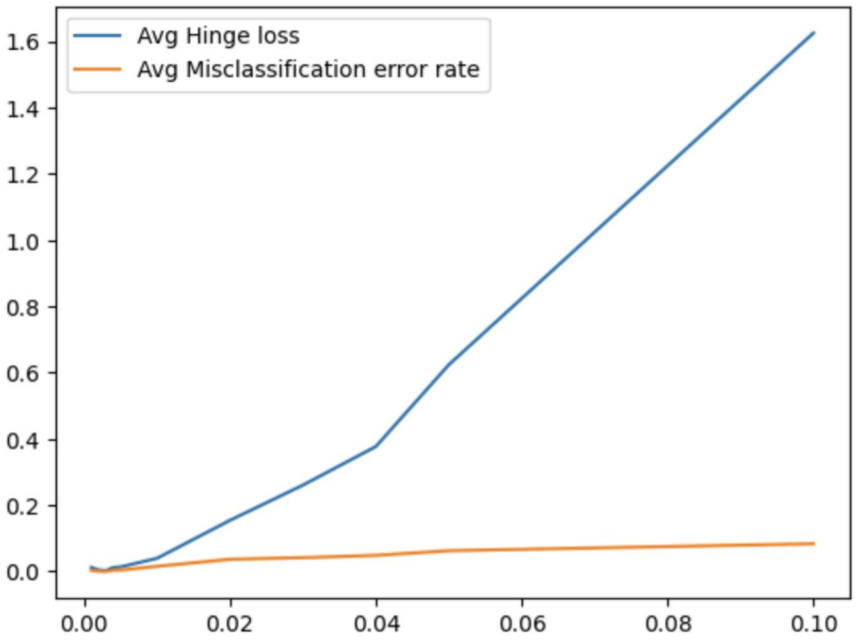
For determining learning rate:
For determining the learning rate we first, made use of Armijo Goldstein rule wherein we started with some large learning rate and tried few values in decreasing order until the objective value has a sufficient reduction. and then used random search to find the optimal value of the parameter. Given below is a graph for determining the learning rate using Armijo Goldstein rule.



Now, we narrowed down our search for the optimal value using random search in the range [0.001, 0.002, 0.003, 0.004, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]



Again narrowing down: subset range [0.001, 0.002, 0.003, 0.004, 0.005, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1]

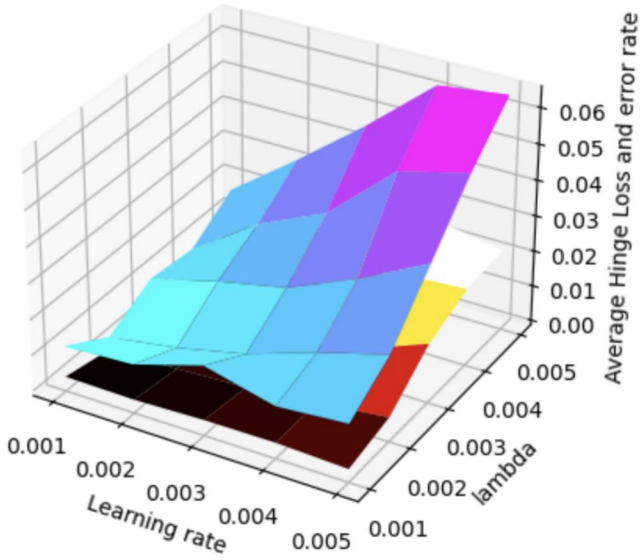


Again narrowing down: subset range 0.001, 0.002, 0.003, 0.004, 0.005]

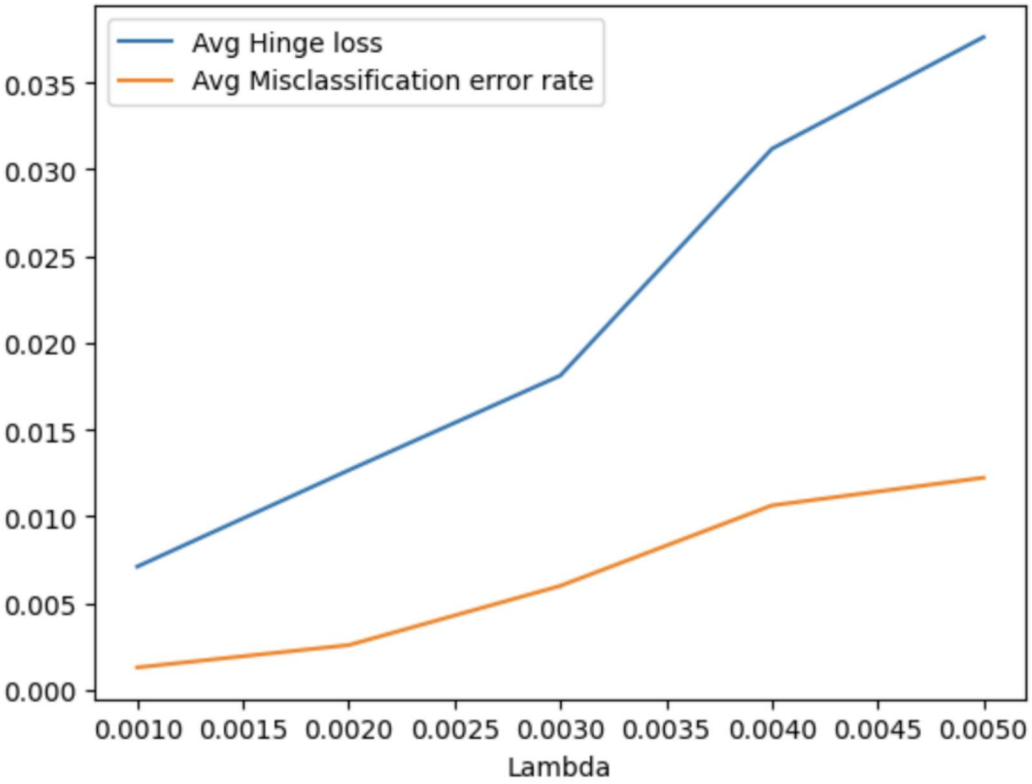
For finding value of Hyperparameter lambda:
For finding value of Hyperparameter lambda there are two famous techniques random search and grid search. We have used both these techniques. First, grid search is used to get an idea of the required range and then random search is used to get a good value of the hyperparameter within that range.
In grid search,we pass predefined values for hyperparameters to the GridSearchCV function. We do this by defining a dictionary in which we mention a particular hyperparameter along with the values it can take. Here is an example of it
'lambda': [0.0001,0.001,0.1, 1, 10, 100, 1000]

GridSearchCV tries all the combinations of the values passed in the dictionary and evaluates the model for each combination using the Cross-Validation method. Hence after using this function we get accuracy/loss for every combination of hyperparameters and we can choose the one with the best performance.
Reference: https://www.mygreatlearning.com/blog/gridsearchcv/

We then, plotted a gradient where x-axis: Learning rate ; y-axis:lambda ; z-axis: average hinge loss and avg misclassification rate taking values of lambda from the feasable range



Taking learning rate value = 0.003; we now found the best value for lambda



Final values of Hyperparameters used: lambda = 0.001 ; learning rate=0.003

5 Hyperparameter description 5 / 5

✓ **+ 3 pts** *Enumeration of all hyperparameters and values considered for those hyperparameters*

✓ **+ 2 pts** *Description of how the best value was obtained for each hyperparameter*

+ 0 pts Completely wrong or else unanswered

6 Question 6

Plot the convergence curves in your PDF report offered by your chosen method as we do in lecture notebooks. The x axis in the graph should be time taken and the y axis should be the test classification accuracy (i.e. higher is better). Include this graph in your PDF file submission as an image. (5 marks)

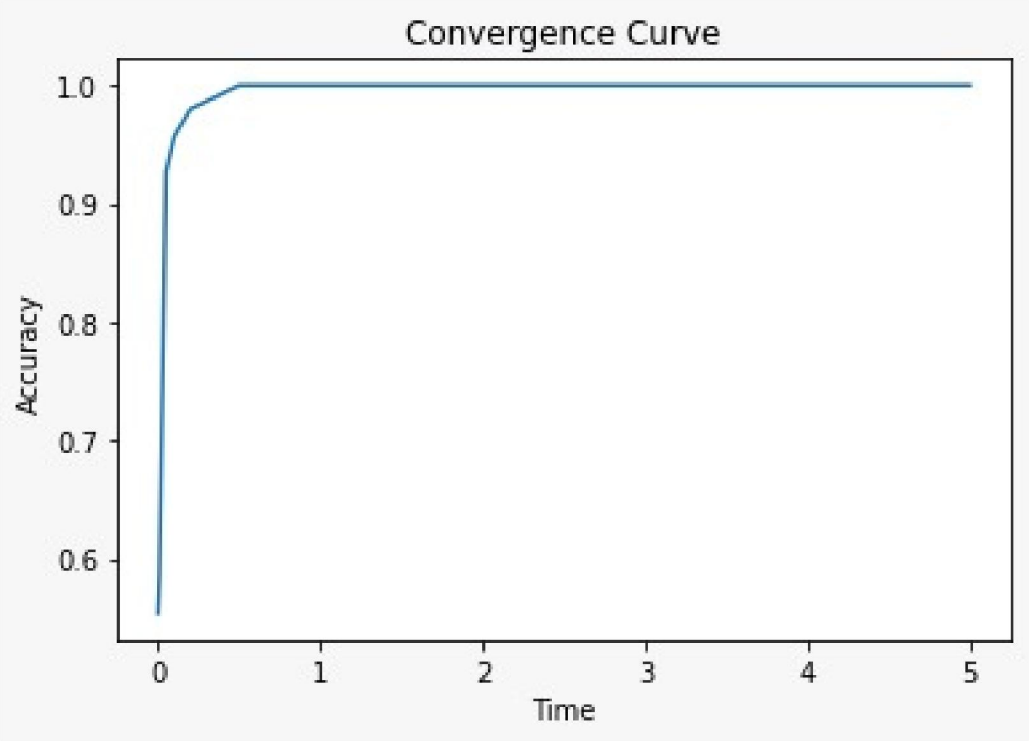


Figure 3: Convergence Curve

answer

6 Convergence plot **5 / 5**

✓ **+ 5 pts** *A digitally generated plot is provided*

- 1 pts Minor mistakes like unlabeled x axis or y axis

+ 0 pts Completely wrong or else unanswered

+ 3 pts Not the classification accuracy graaph