CS 771-A Assignment 1 Submission

Team Yo
Sarthak Kohli (200886)
Dishay Mehta (200341)
Shubhan Ravi (200971)
Mohit Gupta (200597)
Shrey Mehta (200580)

Abstract

This is the report of the first assignment of Introduction to Machine Learning (CS771A) 2022-23-I offering.

Question 1

By giving a mathematical derivation, show the exists a way to map the binary digits 0, 1 to signs -1, +1 as say, $m : \{0, 1\} \to \{-1, 1\}$ and another way $f : \{-, 1, 1\} \to \{0, 1\}$ to map signs to bits (note that m and f need not be inverses of each other) so that for any set of binary digits $b_1, b_2, b_3.....b_n$ for any $n \in N$, we have

$$XOR(b_1, b_2, ...b_n) = f(\prod_{i=1}^{n} m(b_i))$$

Thus, the XOR function is not that scary – it is essentially a product.

Solution

Our Mappings m and f are:

$$m(x) = 1 - 2 * x \text{ where } x \in \{0, 1\}$$
$$\Rightarrow m(0) = 1 \text{ and } m(1) = -1$$
$$f(x) = \frac{(1-x)}{2} \text{ where } x \in \{-1, 1\}$$
$$\Rightarrow f(-1) = 1 \text{ and } f(1) = 0$$

To Prove:

$$XOR(b_1, b_2, ...b_n) = f(\prod_{i=1}^{n} m(b_i))$$

Proof:

Approach 1:

The XOR function for two inputs b_1 and b_2 looks like:

b_1	b_2	$XOR(b_1, b_2)$
0	0	0
0	1	1
1	0	1
1	1	0

Preprint. Under review.

XOR function acting on n binary inputs behaves as per the following rules:

- XOR is commutative. Hence the order in which we compute XOR does not matter. For example $XOR(b_1, b_2, b_3) = XOR(XOR(b_1, b_2), b_3) = XOR(XOR(b_1, b_3), b_2) = XOR(XOR(b_3, b_2), b_1)$
- If the number of ones in $(b_1, b_2, b_3.....b_n)$ is odd, then $XOR(b_1, b_2, b_3.....b_n) = 1$
  Consider number of ones as $2 * p + 1$ where p is any non-negative integer. Now, We can make exactly p pairs of ones and each pair will give XOR as 0. Now , We have p zeroes and one 1, XOR of p zeros is 0. Hence , we reduce the problem to some non negative number of zeroes(1+number of zeroes present initially) + one 1 which is nothing but $XOR(1, 0)$ which is 1
- If the number of ones in $(b_1, b_2, b_3....b_n)$ is even , then $XOR(b_1, b_2, b_3.....b_n) = 0$
  Consider number of ones as $2 * p$ where p is any non-negative integer. Now, we can make exactly p pairs of ones and each pair will give XOR as 0. The problem has been reduced to taking XOR of some non-negative number of zeroes which is 0 evidently

Property Used:

$(-1)^x = 1$ if x is even ; $(-1)^x = -1$ if x is odd.

Proof :

- **Case 1:** $\prod_{i=1}^{n} m(b_i) = 1$

  $\Rightarrow$there are even number of $b_i's$ such that $m(b_i) = -1$
  $\Rightarrow$ even number of $b_i's$ such that $b_i = 1$
  $\Rightarrow f(\prod_{i=1}^{n} m(b_i)) = f(1) = 0$ (since $f(1) = 0$)
  $\Rightarrow XOR(b_1, b_2, b_3, b_4.....b_n) = 0$ (By using property of XOR)
  $\Rightarrow XOR(b_1, b_2, b_3, b_4.....b_n) = f(\prod_{i=1}^{n} m(b_i))$
- **Case 2:** $\prod_{i=1}^{n} m(b_i) = -1$

  $\Rightarrow$there are odd number of $b_i's$ such that $m(b_i) = -1$
  $\Rightarrow$ odd number of $b_i's$ such that $b_i = 1$
  $\Rightarrow f(\prod_{i=1}^{n} m(b_i)) = f(-1) = 1$ (since $f(-1) = 1$)
  $\Rightarrow XOR(b_1, b_2, b_3, b_4.....b_n) = 1$ (By using property of XOR)
  $\Rightarrow XOR(b_1, b_2, b_3, b_4.....b_n) = f(\prod_{i=1}^{n} m(b_i))$

$$\Rightarrow XOR(b_1, b_2, b_3, ...b_n) = f(\prod_{i=1}^{n} m(b_i))$$

Hence Proved

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 $\tilde{u}$ = [u, a], $\tilde{v}$ = [v, b], $\tilde{w}$ = [w, c], $\tilde{x}$ = [x, 1] $\in \mathrm{R}^9$. 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:

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

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

$$\prod_{i=1}^{n} \text{sign}\,(r_i) = \text{sign}\,(\prod_{i=1}^{n} \mathrm{r}_i) \qquad \forall\ \mathrm{n} \geq 1$$

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.

Solution

To Prove:

$$\prod_{i=1}^{n} \text{sign}\,(r_i) = \text{sign}\,(\prod_{i=1}^{n} \mathrm{r}_i) \qquad \forall\ \mathrm{n} \geq 1$$

Let,

$$\text{LHS} = \prod_{i=1}^{n} \text{sign}\,(r_i) \qquad \& \qquad \text{RHS} = \text{sign}\,(\prod_{i=1}^{n} \mathrm{r}_i)$$

Case 1: At least one r_i = 0, say for i=k. Thus r_k = 0

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

$$\text{LHS} = (\prod_{i=1}^{k-1} \text{sign}\,(r_i))\ (\text{sign}\,(r_k))\ (\prod_{i=k+1}^{n} \text{sign}\,(r_i))$$

$$\text{LHS} = (\prod_{i=1}^{k-1} \text{sign}\,(r_i))\ (\text{sign}\,(0))\ (\prod_{i=k+1}^{n} \text{sign}\,(r_i))$$

$$\text{LHS} = (\prod_{i=1}^{k-1} \text{sign}\,(r_i))\ (0)\ (\prod_{i=k+1}^{n} \text{sign}\,(r_i))$$

$$\textbf{LHS = 0}$$

$$\text{RHS} = \text{sign}\,(\prod_{i=1}^{n} \mathrm{r}_i)$$

$$\text{RHS} = \text{sign}\,(\ (\prod_{i=1}^{k-1} \mathrm{r}_i)\ (\mathrm{r}_k)\ (\prod_{i=k+1}^{n} \mathrm{r}_i)\ )$$

$$\text{RHS} = \text{sign}(\ (\prod_{i=1}^{k-1} \mathrm{r}_i)\ (0)\ (\prod_{i=k+1}^{n} \mathrm{r}_i)\ )$$

RHS = sign (0)

RHS = 0

Thus , LHS = RHS

Case 2: $\mathbf{r}_i \neq \mathbf{0}$ $\forall$ i $\in$ {1, 2, 3, ... , n}

Proof by Induction:

Base Case:

For n = 1:

LHS = sign (r_1)

RHS = sign (r_1)

LHS = RHS => True for n=1

For n = 2:
If $r_1 < 0$ and $r_2 < 0$,

sign (r_1) = -1 sign (r_2) = -1

$(r_1)(r_2) > 0$

sign $((r_1)(r_2)) = 1$

LHS = (sign (r_1)) (sign (r_2)) = (-1)(-1) = 1

RHS = sign $((r_1)(r_2)) = 1$

LHS = RHS

If $r_1 < 0$ and $r_2 > 0$,

sign (r_1) = -1 sign (r_2) = 1

$(r_1)(r_2) < 0$

sign $((r_1)(r_2)) = -1$

LHS = (sign (r_1)) (sign (r_2)) = (-1)(1) = -1

RHS = sign $((r_1)(r_2)) = -1$

LHS = RHS

If $r_1 > 0$ and $r_2 < 0$,

sign (r_1) = 1 sign (r_2) = -1

$(r_1)(r_2) < 0$

sign $((r_1)(r_2)) = -1$

LHS = (sign (r_1)) (sign (r_2)) = (1)(-1) = -1

$$\text{RHS} = \text{sign}((r_1)(r_2)) = -1$$

$$\text{LHS} = \text{RHS}$$

If $r_1 > 0$ and $r_2 > 0$,

$$\text{sign}(r_1) = 1 \qquad \text{sign}(r_2) = 1$$

$$(r_1)(r_2) > 0$$

$$\text{sign}((r_1)(r_2)) = 1$$

$$\text{LHS} = (\text{sign}(r_1))(\text{sign}(r_2)) = (1)(1) = 1$$

$$\text{RHS} = \text{sign}((r_1)(r_2)) = 1$$

$$\text{LHS} = \text{RHS}$$

As in every case, **LHS = RHS** => **True for n=2**

For n > 2:

Induction Hypothesis:

Let the equation LHS = RHS be true for n = k
Thus,

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

Induction Step:

Then for n = k + 1,

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

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

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

Using result for n = 2,

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

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

$$\text{LHS} = \text{RHS}$$

Thus, {LHS = RHS for n=k} => {LHS = RHS for n=k+1}, and LHS = RHS for n=1 and n=2.
Therefore by induction, **LHS = RHS for all n**.

Combining result of **Case 1** and **Case 2**, we get

$$\prod_{i=1}^{n} \text{sign}\,(r_i) = \text{sign}\,(\prod_{i=1}^{n} \text{r}_i) \qquad \forall\ \text{n} \geq 1$$

Hence Proved

Question 3

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

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

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 $\tilde{x}$ (in this case 8 + 1 = 9) and there exists a way to map 9 dimensional vectors to D dimensional vectors as $\phi : R^9 -> R^D$ such that for any triple $(\tilde{u}, \tilde{v}, \tilde{w})$, there always exists a vector W $\in \mathrm{R}^D$ such that for every $\tilde{x} \in R^9$, we have $(\tilde{u}^T\tilde{x})$ $(\tilde{v}^T\tilde{x})$ $(\tilde{w}^T\tilde{x})$ = $W^T\phi(\tilde{x})$.
Hint: First try solving this for the simpler where there are only 2 PUFs. If we expand the terms of $(\tilde{u}^T\tilde{x})(\tilde{v}^T\tilde{x})$ = $(\sum_{j=1}^{9} \tilde{u_j}\tilde{x_j})(\sum_{j=1}^{9} \tilde{v_j}\tilde{x_j})$, 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 $\tilde{X}$ = $(\tilde{x_1}...\tilde{x_9})$ to $\phi(\tilde{x}) = (\tilde{x_1}\tilde{x_1}, \tilde{x_1}\tilde{x_2}...\tilde{x_9}\tilde{x_9})$, then we are done since we can now get $(\tilde{u}^T\tilde{x})(\tilde{v}^T\tilde{x})$ = $W^T\phi(\tilde{x})$ by taking

$$\text{W} = (\tilde{u_1}\tilde{v_1}, \tilde{u_1}\tilde{v_2}, ...\tilde{u_1}\tilde{v_9}, \tilde{u_2}\tilde{v1}....\tilde{u_9}\tilde{v_9})$$

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.

Solution

To Prove:

There exist a function $\phi : \mathbb{R}^9 \to \mathbb{R}^D$ such that for any triple ($\tilde{\mathbf{u}}$, $\tilde{\mathbf{v}}$, $\tilde{\mathbf{w}}$), there always exist a vector $\mathbf{W} \in \mathbb{R}^D$ such that for every $\tilde{\mathbf{x}} \in \mathbb{R}^9$, we have

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

Proof:

If we expand the Left hand side (LHS) of equality, we get

$$\text{LHS} = (\tilde{\mathbf{u}}^T\tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^T\tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{w}}^T\tilde{\mathbf{x}})$$

$$\text{LHS} = (\sum_{i=1}^{9} \tilde{u_i}\tilde{x_i})\ (\sum_{j=1}^{9} \tilde{v_j}\tilde{x_j})\ (\sum_{k=1}^{9} \tilde{w_k}\tilde{x_k})$$

$$\text{LHS} = \sum_{i=1}^{9}\sum_{j=1}^{9}\sum_{k=1}^{9} \tilde{u_i}\tilde{v_j}\tilde{w_k}\tilde{x_i}\tilde{x_j}\tilde{x_k}$$

We were inspired from how numbers were represented in base 9 for the following statements. Thus if we take D = 9^3 = 729 , and thus create a function $\phi : \mathbb{R}^9 \to \mathbb{R}^{729}$ which maps

$$\tilde{\mathbf{x}} = (\tilde{x_1}, \tilde{x_2}, ...\tilde{x_9}) \text{ to } \phi(\tilde{\mathbf{x}}) = (\phi(\tilde{\mathbf{x}})_1, \phi(\tilde{\mathbf{x}})_2, ..., \phi(\tilde{\mathbf{x}})_{729})$$

$$\text{where } \phi(\tilde{\mathbf{x}})_i = \tilde{x}_{\alpha(i)}\tilde{x}_{\beta(i)}\tilde{x}_{\gamma(i)} \text{ for all i} \in \{1, 2, ... , 729\}$$

$$\text{with } \alpha(i) = \lceil \tfrac{i}{81} \rceil \ ; \beta(i) = \left\lceil \tfrac{(i-1)\%81+1}{9} \right\rceil \ ; \gamma(i) = (i-1)\%9 + 1$$

we are done, since we can now get

$$(\tilde{\mathbf{u}}^T\tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^T\tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{w}}^T\tilde{\mathbf{x}}) = \mathbf{W}^T\phi(\tilde{\mathbf{x}}) \text{ by taking}$$

$$\mathbf{W} = (W_1, W_2, ..., W_{729})$$

where W_i = $\tilde{u}_{\alpha(i)}\tilde{v}_{\beta(i)}\tilde{w}_{\gamma(i)}$ for all i $\in$ {1, 2, ... , 729}

with $\alpha(i) = \lceil \frac{i}{81} \rceil$; $\beta(i) = \left\lceil \frac{(i-1)\%81+1}{9} \right\rceil$; $\gamma(i) = (i-1)\%9 + 1$

Similarly, it can be proved for any greater number of PUFs also.

Hence Proved

Question 4

The code is submitted as per given instructions.

Question 5

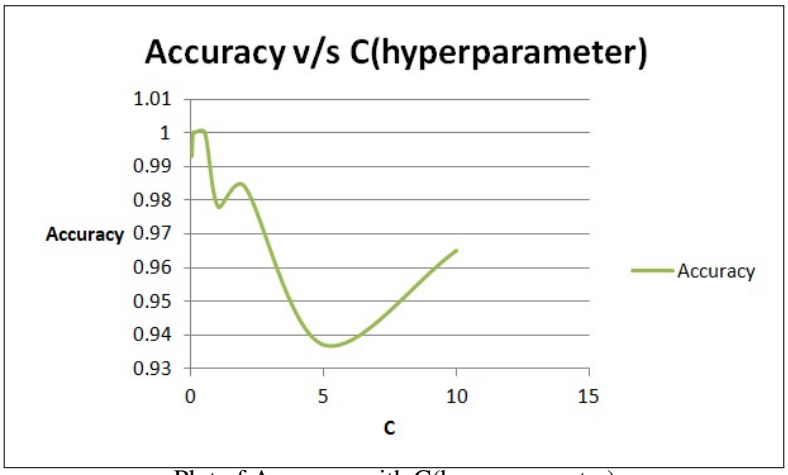
For the method you implemented, describe in your PDF report what were the hyperparameters e.g. step length, policy on choosing the next coordinate if doing SDCA, mini-batch size if doing MBSGD etc and how did you arrive at the best values for the hyperparameters, e.g. you might say "We used step length at time t to be $\frac{\eta}{t}$ where we checked for = 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.

Solution

We have used SDCM Model method for implementing the solver function, which has only one hyperparameter C.
On trail and error with the hyperparameter C on various values, we were able to attain an accuracy of 1.000 on the test and the secret-test dataset for the value of C = 0.1.

For the various values of C, the accuracy on training the model with the secret-train data and testing it on the secret-test data, the accuracy obtained is in the following table:

C (hyperparameter)	Accuracy
0.01	0.993
0.05	1.000
0.1	1.000
0.5	1.000
1	0.978
2	0.984
5	0.937
10	0.965
100	0.955
1000	0.950



Plot of Accuracy with C(hyperparameter)

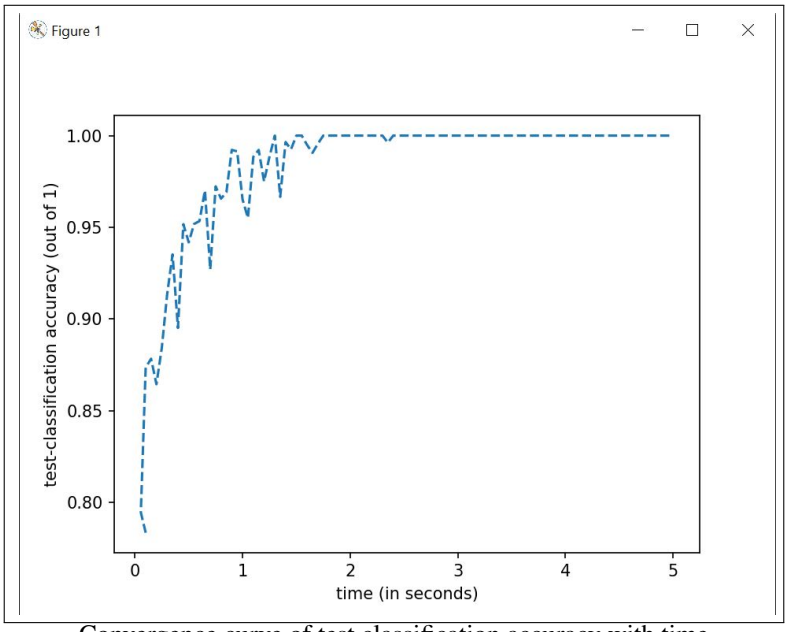
From these observations, we chose C = 0.1 to be a good value for the SDCM method.

Along with SDCM, we also had tried to implement the solver using Primal Gradient Descent, SGD and MBSGD methods and were able to obtain a maximum accuracy of 0.88 using them. So, we did go ahead with submitting those models.

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.

Solution



Convergence curve of test classification accuracy with time