CS771

ML Class of '14

Problem 1.1

`(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.

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.

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).

The following enumerates 6 parts to the question.

1. By giving a mathematical derivation, show there 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 $b_1, b_2, \ldots, b_n$ for any $n \in \mathbb{N}$, we have

$$\text{XOR}(b_1, \ldots, b_n) = f\left(\prod_{i=1}^{n} m(b_i)\right)$$

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

Solution: We show the result by choosing the mapping

$$m(0) = -1 \qquad m(1) = +1$$
$$f(-1) = 1 \qquad f(+1) = 0$$

Now, we prove by induction on n, that for our given choice of m and f, the equation in question holds. Call $p_n = \prod_{i=1}^{n} m(b_i)$

Base Case: For $n = 2$, we make the following cases:

* $b_1 = 0$ and $b_2 = 0$, then $m(b_1) = m(b_2) = -1 \Rightarrow p = +1 \Rightarrow f(p) = 0 =$ XOR$(0, 0)$
* $b_1 = 0$ and $b_2 = 1$, then $m(b_1) = -1$, $m(b_2) = +1 \Rightarrow p = -1 \Rightarrow f(p) = 1 =$ XOR$(0, 1)$
* $b_1 = 1$ and $b_2 = 0$ is same as the above case, since XOR is commutative
* $b_1 = 1$ and $b_2 = 1$, then $m(b_1) = +1 = m(b_2) \Rightarrow p = +1 \Rightarrow f(p) = 0 =$ XOR$(1, 1)$

Induction Hypothesis: If XOR$(b_1, \ldots, b_n) = p_n$ for all possible inputs $(b_1, \ldots, b_n)$, then XOR$(b_1, \ldots, b_{n+1}) = p_{n+1}$ for all possible inputs $(b_1, \ldots, b_{n+1})$

Induction Step: Let $x_n =$XOR$(b_1, \ldots, b_n)$. We use the associative property of XOR to show the above result, i.e.,

$$\text{XOR}(b_1, \ldots, b_n, b_{n+1}) = \text{XOR}(\text{XOR}(b_1, \ldots, b_n), b_{n+1}) = \text{XOR}(x_n, b_{n+1})$$

We know that the result $x_n \in \{0, 1\}$, and b_{n+1} ofcourse is either 0 or 1, so that XOR-ing them together is equivalent to the base case we had observed. Thus, we are done.

2. Let $(\mathbf{u}, a), (\mathbf{v}, b), (\mathbf{w}, c)$ 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 $\tilde{\mathbf{u}} = [\mathbf{u}, a], \tilde{\mathbf{v}} = [\mathbf{v}, b], \tilde{\mathbf{w}} = [\mathbf{w}, c], \tilde{\mathbf{x}} = [\mathbf{x}, 1] \in \mathbb{R}^9$. 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:

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

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

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

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)

Solution: Call $p_r = \prod_{i=1}^{n} r_i$ and $p_s = \prod_{i=1}^{n} \text{sign}(r_i)$

We first show the result when $r_i = 0$ for at least one i. In this case, sign(r_i)=0.
In the LHS, the product p_s becomes zero as sign$(r_i) = 0$, while in the RHS, the inner product p_r becomes zero as $r_i = 0$, leading to sign$(p_r) = 0$. Thus, both LHS and RHS are zero whenever at least one r_i is zero.

We now assume that $r_i \neq 0 \; \forall \; i$
In this case, we show the result by induction on n.

Base Case: $n = 1$. If $r_1 > 0$ then sign$(r_1) = 1 = p_s$, and $p_r = r_1 > 0 \Rightarrow$sign$(p_r) = 1$. Similarly, for $r_1 < 0$, sign$(r_1) = -1 = p_s$, and $p_r = r_1 < 0 \Rightarrow$sign$(p_r) = -1$. Therefore, base case holds.
Induction Hypothesis: If the result holds for all possible non-zero inputs of size n (call p_s, p_r), then it can be shown that the result holds for all possible non-zero inputs of size $(n+1)$ (call p'_s, p'_r) as well.
Induction Step: It is easy to see that $p'_s = p_s\cdot$sign(r_{n+1}), and $p'_r = p_r \cdot r_{n+1}$, and using the induction hypothesis, we have $p_s = sign(p_r)$
Making cases on p_r and r_{n+1}:

* $p_r > 0$ and $r_{n+1} > 0$: $p'_r = p_r \cdot r_{n+1} > 0 \Rightarrow$sign$(p'_r) = 1$. Also, $p'_s = p_s\cdot$sign$(r_{n+1}) =$sign$(p_r)\cdot$sign$(r_{n+1}) = 1 \cdot 1 = 1$. Thus $p'_s = 1 =$sign(p'_r) holds.

* $p_r > 0$ and $r_{n+1} < 0$: $p'_r = p_r \cdot r_{n+1} < 0 \Rightarrow \text{sign}(p'_r) = -1$. Also, $p'_s = p_s \cdot \text{sign}(r_{n+1}) = \text{sign}(p_r) \cdot \text{sign}(r_{n+1}) = 1 \cdot -1 = -1$. Thus $p'_s = -1 = \text{sign}(p'_r)$ holds.
* $p_r < 0$ and $r_{n+1} > 0$: $p'_r = p_r \cdot r_{n+1} < 0 \Rightarrow \text{sign}(p'_r) = -1$. Also, $p'_s = p_s \cdot \text{sign}(r_{n+1}) = \text{sign}(p_r) \cdot \text{sign}(r_{n+1}) = -1 \cdot 1 = -1$. Thus $p'_s = -1 = \text{sign}(p'_r)$ holds.
* $p_r < 0$ and $r_{n+1} < 0$: $p'_r = p_r \cdot r_{n+1} > 0 \Rightarrow \text{sign}(p'_r) = +1$. Also, $p'_s = p_s \cdot \text{sign}(r_{n+1}) = \text{sign}(p_r) \cdot \text{sign}(r_{n+1}) = -1 \cdot -1 = 1$. Thus $p'_s = -1 = \text{sign}(p'_r)$ holds.

Thus, by induction we conclude that the equation holds when all variables are non-zero. The result was already shown to hold for all inputs when at least one number in the input is 0.
Essentially, our cases have covered all possible inputs. The result follows.

3. The above calculation shows that all we need to get a hold of is the following quantity

$$(\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}})$$

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 $\phi : \mathbb{R}^9 \to \mathbb{R}^D$ such that for any triple $(\tilde{\mathbf{u}}, \tilde{\mathbf{v}}, \tilde{\mathbf{w}})$, there always exists a vector $\mathbf{W} \in \mathbb{R}^D$ such that for every $\tilde{\mathbf{x}} \in \mathbb{R}^9$, we have $(\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}) = \mathbf{W}^\top \phi(\tilde{\mathbf{x}})$ (10 marks)

Solution:

$$\tilde{\mathbf{u}} = \sum_{i=1}^{i=9} \mathbf{u_i}; \quad \tilde{\mathbf{v}} = \sum_{i=1}^{i=9} \mathbf{v_i}; \quad \tilde{\mathbf{w}} = \sum_{i=1}^{i=9} \mathbf{w_i}; \; \tilde{\mathbf{x}} = \sum_{i=1}^{i=9} \mathbf{x_i}$$

here $\mathbf{u_i}$ is the i^{th} vector component of the vector $\tilde{\mathbf{u}}$ and u_i is the magnitude of $\mathbf{u_i}$, similar notation is used for $\tilde{\mathbf{v}}, \tilde{\mathbf{w}}, \tilde{\mathbf{x}}$.

$$(\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) = \sum_{i=1}^{i=9} u_i \cdot x_i \;\; (\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) = \sum_{i=1}^{i=9} v_i \cdot x_i \;\; (\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}) = \sum_{i=1}^{i=9} w_i \cdot x_i$$

$$\implies (\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}) = \sum_{i=1}^{i=9} \sum_{j=1}^{j=9} \sum_{k=1}^{k=9} u_i x_i v_j x_j w_k x_k$$

Define $X_{ijk} = x_i x_j x_k; \quad P_{ijk} = u_i v_j w_k; \quad i, j, k \in \mathbf{N}, i, j, k \leq 9$

$$\implies (\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}) = \sum_{i=1}^{i=9} \sum_{j=1}^{j=9} \sum_{k=1}^{k=9} P_{ijk} X_{ijk}$$

We map every tuple $(i, j, k) \to \{0, 1, 2 ... 9^3 - 1\} := (i, j, k) \to 9^2 \cdot (i-1) + 9 \cdot (j-1) + k - 1$
Note that this is a bijective mapping as it converts the number $(i-1)(j-1)(k-1)$ written in base 9 (where $(i-1), (j-1), (k-1)$ are the base 9 digits) to base 10. Hence if we define $\phi(\tilde{\mathbf{x}})_{9^2 \cdot (i-1) + 9 \cdot (j-1) + k - 1} = X_{i,j,k} = x_i x_j x_k; \;\; W_{9^2 \cdot (i-1) + 9 \cdot (j-1) + k - 1} = P_{i,j,k};$, we get that

$$(\tilde{\mathbf{u}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{v}}^\top \tilde{\mathbf{x}}) \cdot (\tilde{\mathbf{w}}^\top \tilde{\mathbf{x}}) = \sum_{i=1}^{i=9} \sum_{j=1}^{j=9} \sum_{k=1}^{k=9} P_{ijk} X_{ijk} = \sum_{i=0}^{i=9^3-1} W_i \phi(\tilde{\mathbf{x}})_i = \mathbf{W}^\top \phi(\tilde{\mathbf{x}}); \forall \tilde{\mathbf{x}} \in \mathbb{R}^9$$

.

Note that the products $x_1 x_2 x_3$ and $x_2 x_1 x_3$ are the same. Hence effectively the number of dimensions can be further reduced to $\binom{9}{3} + 2 \cdot \binom{9}{2} + \binom{9}{1} = 165$.

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 $\eta/\sqrt{t}$ where we checked for $\eta = 0.1, 0.2, 0.5, 1, 2, 5$ using held out validation and found $\eta = 2$ 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)

Solution: **SDCM**:
We convert the problem into the dual problem and try to optimise the value of α(s) to get the optimal value of W.

Choosing the next coordinate of alpha to optimise: We tried choosing coordinates at random, choosing them in a cyclic manner $(1, 2, 3 \ldots 1, 2, 3 \ldots)$ and then choosing them through random permutations. After completing a permutation a new permutation is generated. The last yielded the best validation results.

Initialisation of α(s): We know that in a large given set of points very few points act as support vectors hence initialisation to small values was optimal. We initialise as

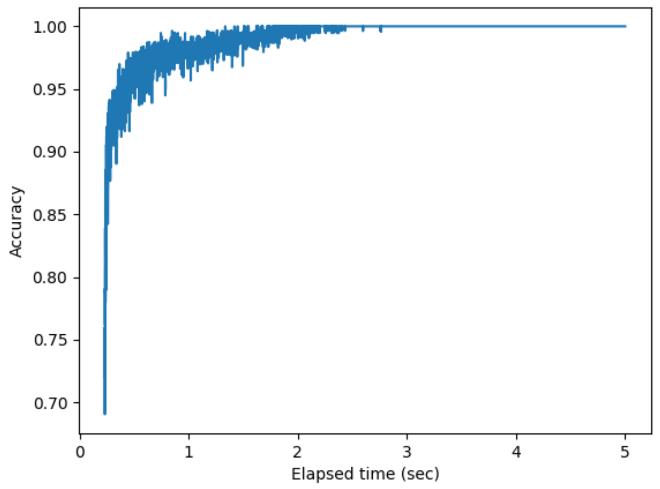
```
alphas = 0.0001*np.random.rand(n)
```

Since for a XOR PUF, an ideal linear classifier does exist then assuming, that the given training data doesn't contain noise, value of C doesn't affect the results as long as C is large. We use 10000 as the value of C.

Finally, we scale the W so obtained by 10, to account for the fact the we solved the dual problem for the normalised weights, in other words this accounts for the fact that the hinge loss is taken around the value 1 and not some other value like 0.1.

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)

Solution: We have shown below the convergence plot for the SDCM method:



The model was run for 5 seconds and the error was found to be 0 upto the 12^{th} decimal point.