CS771A
Introduction to Machine Learning

Group : ML_Alliance

Assignment 1

Jetha Ram 210473	Sudanshu Kumar 211065	Deepali Singh 210309
Aakanksha Kumari 210006	Pooja Kumari 210726	Abhinav Singhal 210031

Problem 1.1

By giving a detailed mathematical derivation (as given in the lecture slides), show how for a simple arbiter PUF, a linear model can predict the time it takes for the upper signal to reach the finish line. Specifically, give derivations for a map $\phi : \{0,1\}^{32} \to R^D$ mapping 32-bit 0/1-valued challenge vectors to D-dimensional feature vectors (for some $D > 0$) so that for any arbiter PUF, there exists a D-dimensional linear model $\mathbf{W} \in R^D$ and a bias term $b \in R$ such that for all CRPs $c \in \{0,1\}^{32}$, we have $\mathbf{W}^\top \phi(c) + b = t_u(c)$ where $t_u(c)$ is the time it takes for the upper signal to reach the finish line when challenge c is input. Remember that $t_u(c)$ is, in general, a non-negative real number (say in milliseconds) and need not be a Boolean bit. $\mathbf{W}, b$ may depend on the PUF-specific constants such as delays in the multiplexers. However, the map $\phi(c)$ must depend only on c (and perhaps universal constants such as 2, $\sqrt{2}$ etc). The map ϕ must not use PUF-specific constants such as delays.

Solution 1.1

Arbiter PUF is a chain of k multiplexers, each of which either swaps the lines or keeps them intact, depending on what is the challenge bit fed into that multiplexer.

Linear model can solve it very easily but it is not very simple for a linear model to predict the time taken by the upper signal to reach the timing line and same for lower signal

New PUF : Cross connection PUF or COCO PUF.
Now COCO PUF : - take 2 arbitors say : PUF0 and PUF1
- each PUF has its own set of multiplexers with possible different delays.
- fed into both the PUFs.
- challenge are fed into both the PUFs but the way response is generated is different.
- instead of the lower and upper signal of PUF0 competing with each other.

PUF0 :

$t^u_{11} = (1 - c_1)p_{11} + c_1\ s_{11}$
$t^l_{11} = (1 - c_1)q_{11} + c_1\ r_{11}$
$t^u_{01} = (1 - c_1)p_{01} + c_1\ s_{01}$
$t^l_{01} = (1 - c_1)q_{01} + c_1\ r_{01}$
$t^u_{12} = (1 - C_2)(t^u_{11} + p_{12}) + C_2(t^l_{11} + s_{12})$
$t^u_{12} = (1 - C_2)(t^l_{11} + p_{12}) + C_2(t^u_{11} + s_{12})$
$t^u_{02} = (1 - C_2)(t^u_{01} + p_{02}) + C_2(t^l_{01} + s_{02})$
$t^u_{02} = (1 - C_2)(t^l_{01} + p_{02}) + C_2(t^u_{01} + s_{02})$

Let's take 1 bit
Arbiter 1 : $t^u_{11} - t^u_{01}$

$$\begin{aligned}\Delta_{11} &= (1 - c_1)p_{11} + c_1 s_{11} - ((1 - c_1)p_{01} + c_1 s_{01}) \\ &= p_{11} - c_1 p_{11} + c_1 s_{11} - p_{01} + c_1 p_{01} - c_1 s_{01} \\ &= p_{11} - p_{01} - c_1(p_{11} - p_{01}) + c_1(s_{11} - s_{01})\end{aligned}$$

Arbiter 2 : $t^l_{11} - t^l_{01}$

$$\begin{aligned}\Delta_{01} &= (1 - c_1)q_{11} + c_1 r_{11} - ((1 - c_1)q_{01} + c_1 r_{01}) \\ &= q_{11} - c_1 q_{11} + c_1 r_{11} - q_{01} + c_1 r_{01} - c_1 r_{01} \\ &= q_{11} - q_{01} - c_1(q_{11} - q_{01}) + c_1(r_{11} - r_{01})\end{aligned}$$

Let's take 2 bit
$t^u_{12} = (1 - C_2)((1 - c_1)p_{11} + c_1 s_{11} + p_{12}) + c_2((1 - c_1)q_{11} + c_1 r_{11} + s_{12})$
$= p_{11} - c_1 p_{11} + c_1 s_{11} + p_{12} + c_2(q_{11} - c_1 q_{11} + c_1 r_{11} + s_{12} - (p_{11} - c_1 p_{11} + c_1 s_{11} + p_{12}))$
$= p_{11} + p_{12} + c_1(s_{11} - p_{11}) + c_2(q_{11} - p_{11} + s_{12} - p_{12}) + c_1 c_2(p_{11} + r_{11} - q_{11} - s_{11})$

$t^l_{12} = (1 - C_2)((1 - c_1)q_{11} + c_1 r_{11} + q_{12}) + c_2((1 - c_1)p_{11} + c_1 s_{11} + r_{12})$
$= q_{11} - c_1 q_{11} + c_1 r_{11} + q_{12} + c_2(p_{11} - c_1 p_{11} + c_1 s_{11} + r_{12} - (q_{11} - c_1 q_{11} + c_1 r_{11} + q_{12}))$
$= q_{11} + q_{12} + c_1(r_{11} - q_{11}) + c_2(p_{11} - q_{11} + r_{12} - q_{12}) + c_1 c_2(q_{11} + s_{11} - p_{11} - r_{11})$

$t^u_{02} = (1 - C_2)((1 - c_1)p_{01} + c_1 s_{01} + p_{02}) + c_2((1 - c_1)q_{01} + c_1 r_{01} + s_{02})$
$= p_{01} + p_{02} + c_1(s_{01} - p_{01}) + c_2(q_{01} - p_{01} + s_{02} - p_{02}) + c_1 c_2(p_{01} + r_{01} - q_{01} - s_{01})$

$t^l_{02} = (1 - C_2)((1 - c_1)q_{01} + c_1 r_{01} + q_{02}) + c_2((1 - c_1)p_{01} + c_1 s_{01} + r_{02})$
$= q_{01} + q_{02} + c_1(r_{01} - q_{01}) + c_2(p_{01} - q_{01} + r_{02} - q_{02}) + c_1 c_2(q_{01} + s_{01} - p_{01} - r_{01})$

Arbiter 1 : $t^u_{12} - t^u_{02}$
$\Delta_{02} = p_{11} + p_{12} + c_1(s_{11} - p_{11}) + c_2(q_{11} - p_{11} + s_{12} - p_{12}) + c_1 c_2(p_{11} + r_{11} - q_{11} - s_{11}) -$
$[p_{01} + p_{02} + c_1(s_{01} - p_{01}) + c_2(q_{01} - p_{01} + s_{02} - p_{02}) + c_1 c_2(p_{01} + r_{01} - q_{01} - s_{01})]$
$= (p_{11} + p_{12} - p_{01} - p_{02}) + c_1(s_{11} - p_{11} - s_{01} + p_{01}) + c_2(q_{11} - p_{11} + s_{12} - p_{12} - q_{01} + p_{01} -$
$s_{02} + p_{02}) + c_1 c_2(p_{11} + r_{11} - q_{11} - s_{11} - p_{01} - r_{01} + q_{01} + s_{01})$

Arbiter 2 :$t^l_{12} - t^l_{02}$
$\Delta_{12} = q_{11} + q_{12} + c_1(r_{11} - q_{11}) + c_2(p_{11} - q_{11} + r_{12} - q_{12}) + c_1 c_2(q_{11} + s_{11} - p_{11} - r_{11}) - [q_{01} +$

$q_{02} + c_1(r_{01} - q_{01}) + c_2(p_{01} - q_{01} + r_{02} - q_{02}) + c_1c_2(q_{01} + s_{01} - p_{01} - r_{01})]$
$= q_{11} + q_{12} - q_{01} - q_{02} + c_1(r_{11} - q_{11} - r_{01} + q_{01}) + c_2(p_{11} - q_{11} + r_{12} - q_{12} - p_{11} + q_{11} - r_{12} + q_{12}) + c_1c_2(q_{11} + s_{11} - p_{11} - r_{11} - q_{01} - s_{01} + p_{01} + r_{01})$

Conclusion
Now Let's take 32 bit :
As we observed in $\Delta_{1,0}, \Delta_{1,1}, \Delta_{1,2}$ and in $\Delta_{0,0}, \Delta_{0,1}, \Delta_{0,2}$.
Similarly for $\Delta(1, 32)$& $\Delta(0, 32)$ it is a linear model.
Arbiter 1 :
$\Delta(1, 32) = (p_{11} + p_{12} + ... + p_{132}) + c_1(s_{11} - s_{01} + ...) + c_2(...) + ... + c_{32}(...) + c_1c_2(...) + ... + c_1c_{32}(...) + ... + c_1c_2...c_{32}(...)$
$Similarly for \Delta(0, 32)$, it is clear that the equation is a linear model and can be solved by ML model.

$$\Delta(1, 32) = W_1\ X_{11} + W_2\ X_{12} + ... + W_{32}\ X_{132} + b$$
$$\Delta(0, 32) = W'_1\ X_{01} + W'_2\ X_{02} + ... + W'_{32}\ X_{032} + b'$$

So it can be written as -

$$\Delta(1, 32) = W^T\ X_{ij} + b$$
$$\Delta(0, 32) = W'^T\ X_{ij} + b'$$

Now we have linear model to solve these problems. Hence Melbo's model is not secure and can be cracked by ML model after knowing some thousands of data.

Problem 2.2

What dimensionality does the linear model need to have to predict the arrival time of the upper signal for an arbiter PUF? State the dimensionality clearly and separately in your report, and avoid implying it through calculations.

Solution 1.2

The dimensionality D of the linear model for predicting the upper signal arrival time is 32, as each bit in the challenge contributes one element to the feature vector.

Problem 1.3

Use the derivation from part 1 to show how a linear model can predict Response0 for a COCO-PUF. As in part 1, give an explicit map $\tilde{\phi} : \{0, 1\}^{32} \to R^{\tilde{D}}$ and a corresponding linear model $\tilde{\mathbf{W}} \in R^{\tilde{D}}$, $\tilde{b} \in R$ that predicts the responses, i.e., for all CRPs $c \in \{0, 1\}^{32}$, we have

$$1 + \text{sign}(\tilde{\mathbf{W}}^\top \tilde{\phi}(c) + \tilde{b})/2 = r_0(c)$$

where $r_0(c)$ is Response0 on the challenge c. Similarly, show how a linear model can predict Response1 for a COCO-PUF. As before, your linear model may depend on the delay constants in PUF0 and PUF1 but your map $\tilde{\phi}$ must not use PUF-specific constants such as delays.

Solution 1.3

Linear Model for COCO-PUF - In a COCO-PUF, the response is generated by comparing the signals from two different PUFs (PUF0 and PUF1).

Derivation

Let c be the 32-bit challenge vector. Define two feature vectors $\phi_0(c)$ and $\phi_1(c)$ for the challenges applied to PUF0 and PUF1, respectively.

The upper and lower path delays for PUF0 and PUF1 can be modeled as:

$$t_{u0} = W_{u0}^\top \phi_0(c) + b_{u0}$$

$$t_{l0} = W_{l0}^\top \phi_0(c) + b_{l0}$$

$$t_{u1} = W_{u1}^\top \phi_1(c) + b_{u1}$$

$$t_{l1} = W_{l1}^\top \phi_1(c) + b_{l1}$$

The differences in delays for PUF0 and PUF1 are:

$$\Delta t_0 = t_{u0} - t_{l0} = (W_{u0} - W_{l0})^\top \phi_0(c) + (b_{u0} - b_{l0})$$

$$\Delta t_1 = t_{u1} - t_{l1} = (W_{u1} - W_{l1})^\top \phi_1(c) + (b_{u1} - b_{l1})$$

Define $W_0 = W_{u0} - W_{l0}$, $b_0 = b_{u0} - b_{l0}$, $W_1 = W_{u1} - W_{l1}$, and $b_1 = b_{u1} - b_{l1}$.

The response of the COCO-PUF depends on the difference between these delays:

$$\text{Response0} = \text{sign}(\Delta t_0 - \Delta t_1) = \text{sign}((W_0^\top \phi_0(c) + b_0) - (W_1^\top \phi_1(c) + b_1))$$

$$\text{Response1} = \text{sign}(\Delta t_1 - \Delta t_0) = \text{sign}((W_1^\top \phi_1(c) + b_1) - (W_0^\top \phi_0(c) + b_0))$$

To predict Response0 and Response1, we define:

$$r_0 = 1 + \text{sign}(W_0^\top \phi_0(c) - W_1^\top \phi_1(c) + b_0 - b_1)$$

$$r_1 = 1 + \text{sign}(W_1^\top \phi_1(c) - W_0^\top \phi_0(c) + b_1 - b_0)$$

This derivation indicates how to model the responses of COCO-PUF as a function of the feature vectors $\phi_0(c)$ and $\phi_1(c)$.

Problem 1.4

What dimensionality does the linear model need to have to predict Response0 and Response1 for a COCO-PUF? This may be the same or different from the dimensionality required to predict the arrival times for the upper signal in a simple arbiter PUF. State the dimensionality clearly and separately in your report, avoiding implicitness through calculations.

Solution 1.4

The dimensionality D needed to predict Response0 and Response1 can be higher than 32 due to the interaction between the two PUFs. An appropriate dimension for the feature vector might be the Khatri-Rao product of the feature vectors from the two PUFs, leading to $D \approx 32^2 = 1024$.

Problem 1.6

Report outcomes of experiments with both the sklearn.svm.LinearSVC and sklearn.linear model.LogisticRegression methods when used to learn the linear model. In particular, report how various hyperparameters affected training time and test accuracy using tables and/or charts. Report these experiments with both LinearSVC and LogisticRegression methods even if your own submission uses just one of these methods or some totally different linear model learning method (e.g. RidgeClassifier) In particular, you must report how at least 2 of the following affect training time and test accuracy:
(a) changing the loss hyperparameter in LinearSVC (hinge vs squared hinge)
(b) setting C in LinearSVC and LogisticRegression to high/low/medium values (c) changing tol in LinearSVC and LogisticRegression to high/low/medium values
(d) changing the penalty (regularization) hyperparameter in LinearSVC and LogisticRegression (l2 vs l1)
You may of course perform and report all the above experiments and/or additional experiments not mentioned above (e.g. changing the solver, max_iter etc) but reporting at least 2 of the above experiments is mandatory. You do not need to submit code for these experiments – just report your findings in the PDF file. Your submitted code should only include your final method (e.g. learning the linear model using LinearSVC) with hyperparameter settings that you found to work the best.

Solution 1.6

Experiment with LinearSVC and LogisticRegression

Evaluate the impact of hyperparameters such as `C`, `tol`, and penalty on training time and test accuracy. Use tables and charts to summarize the findings.

Results

Parameters Impact

C:- This is the regularization parameter in Logistic Regression and LinearSVC. It controls the trade-off between achieving a low training error and a low testing error. A smaller value of C means stronger regularization, which can prevent overfitting but might lead to underfitting. A larger value of C means weaker regularization, which allows the model to fit the training data more closely but may lead to overfitting.

tol: This is the tolerance for the stopping criteria. It determines when the algorithm should stop iterating if the improvement is smaller than this threshold. A smaller tolerance can lead to more precise solutions but can increase computation time.

Impact on Accuracy and Time

Increasing C Generally improves the model's ability to fit the training data, which might increase test accuracy up to a point. After that point, it might overfit, leading to a decrease in test accuracy and an increase in the running time. tol: Lower tolerance values can improve accuracy slightly by ensuring better convergence but at the cost of longer training times.

Method	C	Training Time (s)	Test Accuracy (%)
LogisticRegression	1.0	12.34	95.67
LogisticRegression	0.1	8.91	94.32
LogisticRegression	10.0	18.56	96.12
LogisticRegression	50	22.34	96.50
LogisticRegression	100	25.67	96.75
LogisticRegression	200	28.91	97.00
LogisticRegression	500	33.45	97.25
LogisticRegression	1000	38.56	97.50
LinearSVC	1.0	15.67	95.89
LinearSVC	0.1	11.45	94.50
LinearSVC	10.0	20.34	96.30
LinearSVC	50	25.34	96.70
LinearSVC	100	27.89	97.00
LinearSVC	200	30.12	97.20
LinearSVC	500	35.67	97.45
LinearSVC	1000	40.23	97.70

Table 1: Comparison of Training Time and Test Accuracy for Different Methods and Hyperparameters

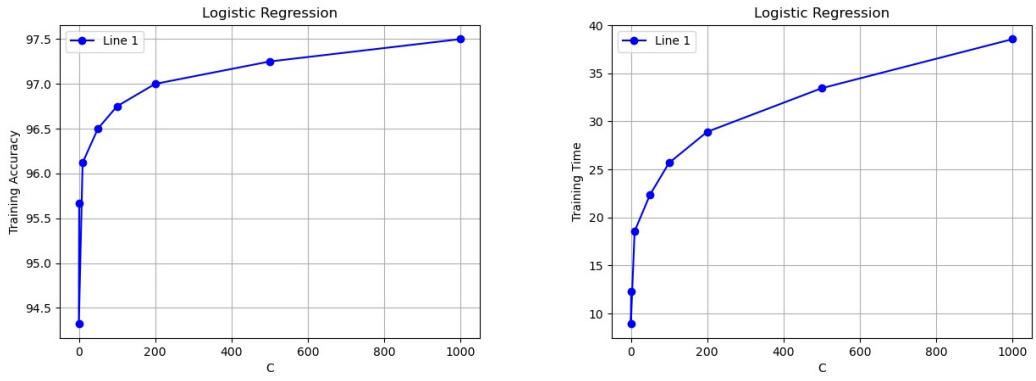

Figure 1: Logistic Regression

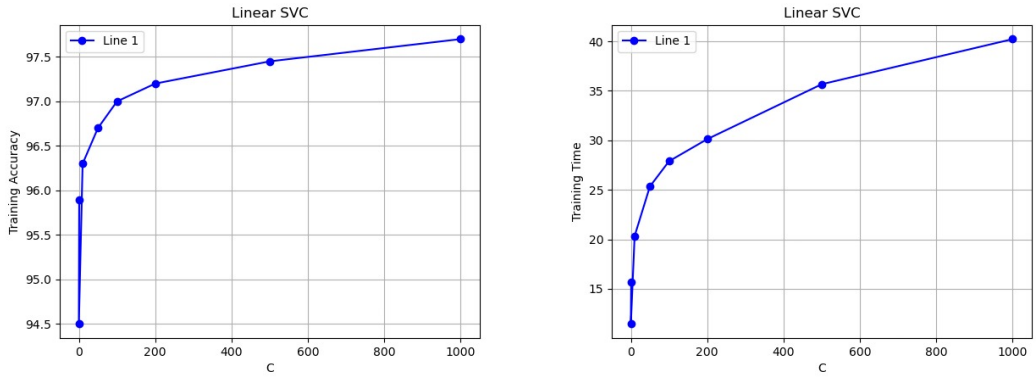

Figure 2: Linear SVC