Assignment 1

Group - 22 Gradient_geeks
CS 771 Submission report
IIT Kanpur
Kanpur, India

1 First Task - 1

By giving a detailed mathematical derivation (as given in the lecture slides), show how a CAR-PUF can be broken by a single linear model. Give derivations for a map ϕ:

$$\{0,1\}^{32} \rightarrow \mathbb{R}^D$$

mapping 32-bit 0/1-valued challenge vectors to D-dimensional feature vectors (for some $D > 0$) so that for any CAR-PUF, there exists a D-dimensional linear model $W \in \mathbb{R}^D$ and a bias term $b \in \mathbb{R}$ such that for all CRPs (c, r) with $c \in \{0,1\}^{32}, r \in \{0,1\}$, we have the two arbiter PUFs sitting inside the CAR-PUF. W, b will depend on u, v, p, q, τ. Note

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

Hint: Let $(u, p), (v, q)$ be the two linear models that can exactly predict the outputs of that $\phi(c)$ must depend only on c (and perhaps universal constants such as 2, map ϕ must not use PUF-specific constants such as u, v, p, q, τ.

1.1 Understanding CAR-PUF

- An arbiter PUF's response to a given challenge is determined by the difference in timing $(\Delta_u - \Delta_l)$ between two signals travelling through a series of multiplexers. The challenge bits determine the path of these signals, and the inherent delays in these paths are unpredictable but consistent.
- A CAR-PUF consists of two arbiter PUFs (a working PUF and a reference PUF) and a secret threshold τ. For a given challenge, the timing differences Δ_w (for the working PUF) and $-\Delta_r$ (and for the reference PUF) are compared against τ. The response is 0 if $|\Delta_w - \Delta_r| \leq \tau$ and 1 otherwise.

1.2 Task: Predicting CAR-PUF Responses

Predicting the CAR-PUF response using a linear model involves creating a feature vector $\phi(\mathbf{c})$ from the 32-bit challenge $\mathbf{c}$ that can be used in a linear model.

Mathematical Derivation for Linear Model:

- Linear Models for Arbiter PUFs: It's known that for each arbiter PUF, there exists a linear model that can predict its output. Let u and v be the weight vectors, and p and q be the bias terms for the linear models of the working and reference PUFs, respectively. These models predict the timing differences Δ_w and Δ_r.

$$\Delta_w = \mathbf{u}^T \phi(\mathbf{c}) + p = \tilde{\mathbf{u}}^T \phi\tilde{(\mathbf{c})} \tag{1}$$

$$\Delta_w = \mathbf{v}^T \phi(\mathbf{c}) + p = \tilde{\mathbf{v}}^T \phi\tilde{(\mathbf{c})} \tag{2}$$

$$\phi\tilde{(\mathbf{c})} = \begin{bmatrix} \prod_{i=1}^{32} 1 - 2c_i & \prod_{i=2}^{32} 1 - 2c_i & . & . & 1 - 2c_{32} & 1 \end{bmatrix} \tag{3}$$

- Combining Models for CAR-PUF: For a CAR-PUF, we are interested in the absolute difference in timing differences, $|\Delta_w - \Delta_r|$ and its comparison to τ.

$$\begin{aligned} \Delta = |\Delta_w - \Delta_r| &= |(\tilde{\mathbf{u}}^T \phi\widetilde{(\mathbf{c})}) - (\tilde{\mathbf{v}}^T \phi\widetilde{(\mathbf{c})})| && (4) \\ \Delta &= |(\tilde{\mathbf{u}} - \tilde{\mathbf{v}})^T \phi\widetilde{(\mathbf{c})}| && (5) \end{aligned}$$

Therefore, the model response would look like: $\frac{1+sign(|(\tilde{\mathbf{u}}-\tilde{\mathbf{v}})^T \phi\widetilde{(\mathbf{c})}|-\tau)}{2}$.

1.3 Simplifying the model

since $\tau \geq 0$ we get,

$$\begin{aligned} \text{sign}(|(\tilde{\mathbf{u}} - \tilde{\mathbf{v}})^T \phi\widetilde{(\mathbf{c})}| - \tau) &= \text{sign}(((\tilde{\mathbf{u}} - \tilde{\mathbf{v}})^T \phi\widetilde{(\mathbf{c})})^2 - \tau^2) \\ &= \text{sign}((\mathbf{y}^T \phi\widetilde{(\mathbf{c})})^2 - \tau^2) \end{aligned}$$

where, $\mathbf{y} = \tilde{\mathbf{u}} - \tilde{\mathbf{v}}$

On the expansion of squared terms:

$$(\mathbf{y}^T \tilde{\phi}(\mathbf{c}))^2 - (\tau)^2 = \sum_{i=1}^{33} y_i^2 + 2 \sum_{i<j} \sum_{j} y_i y_j \tilde{\phi}_i(\mathbf{c}) \tilde{\phi}_j(\mathbf{c}) - (\tau)^2 \tag{6}$$

$$\begin{aligned} \text{sign}((\mathbf{y}^T \tilde{\phi}(\mathbf{c}))^2 - (\tau)^2) &= \text{sign}(2 \times (\sum_{i<j} \sum_{j} y_i y_j \tilde{\phi}_i(\mathbf{c}) \tilde{\phi}_j(\mathbf{c}) + 0.5 \times (\sum_{i=1}^{33} y_i^2 - \tau^2))) \\ &= \text{sign}(\sum_{i<j} \sum_{j} y_i y_j \tilde{\phi}_i(\mathbf{c}) \tilde{\phi}_j(\mathbf{c}) + 0.5 \times (\sum_{i=1}^{33} y_i^2 - \tau^2))) \end{aligned}$$

Therefore the final model is:-

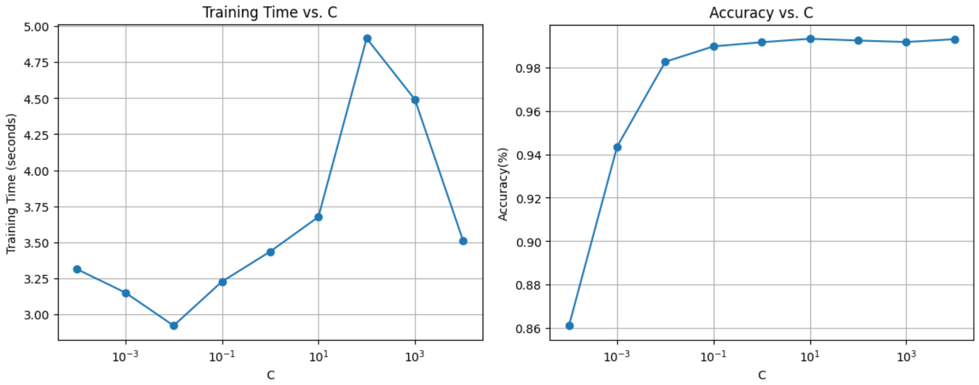
$$r = \frac{1 + \text{sign}(\mathbf{W}^T \phi(\mathbf{c}) + b)}{2} \tag{7}$$

$\mathbf{W} = \left[[y_i . y_j]_{i \neq j} \right]_{528 \times 1}$
$\phi(c) = \left[\left[\tilde{\phi}_i(\mathbf{c}) . \tilde{\phi}_j(\mathbf{c}) \right]_{i \neq j} \right]_{528 \times 1}$
$b = 0.5 \times (\sum_{i=1}^{33} y_i^2 - \tau^2)$
Where D = $\binom{33}{2} = 528$

2 Third Task - 3

Report outcomes of experiments with both the sklearn.svm.LinearSVC and sklearn.linear model.Logistic regression 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 Logistic Regression 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 the following affect training time and test accuracy:

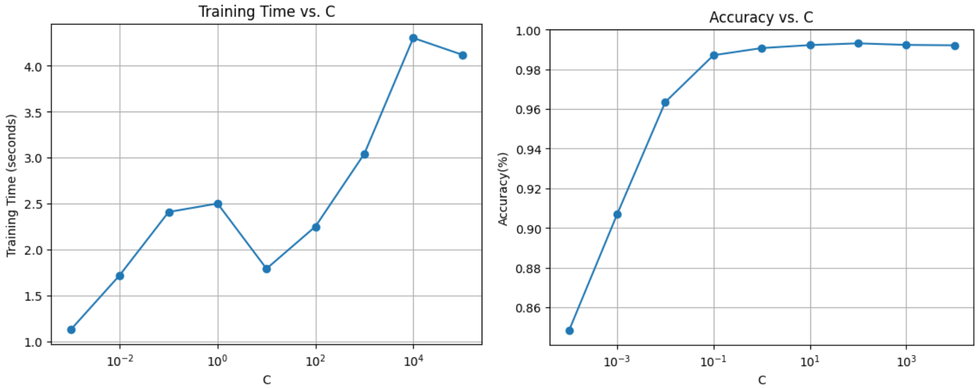
2.1 Graphs and tables showing varyiation of hyperparameters:



(a) Training Time

(b) Accuracy

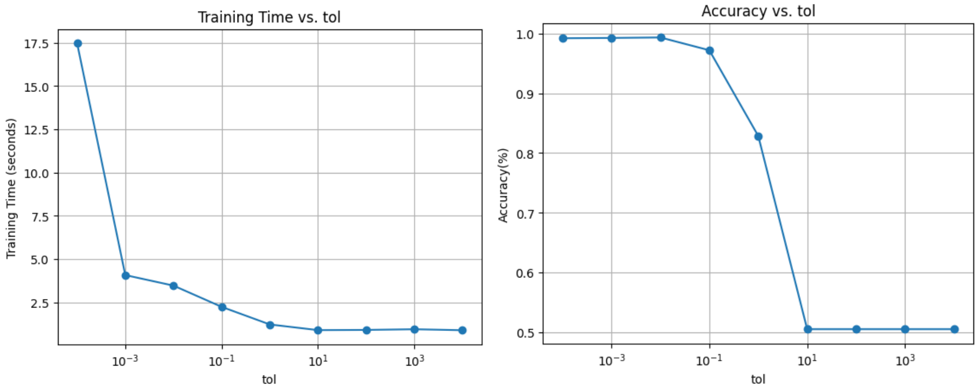
Figure 1: Effect of **C** for **LinearSVC**
, Other fixed hyper-parameters: intercept scaling = 0.5, tolerance = 0.001 and rest default values



(a) Training Time

(b) Accuracy

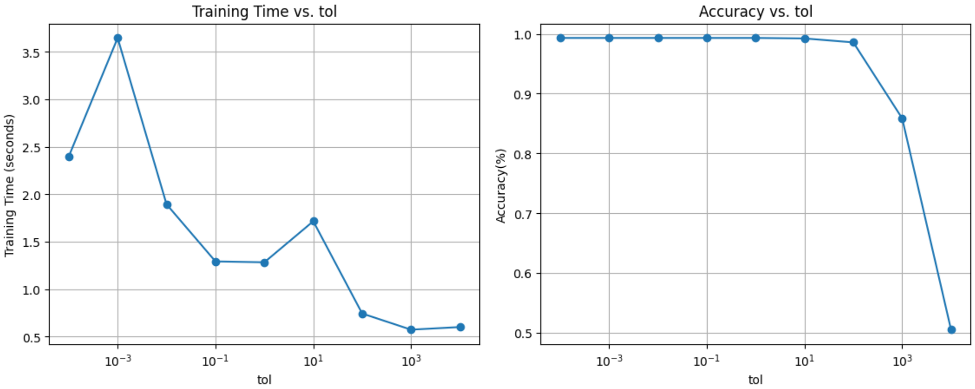
Figure 2: Effect of **C** for **Logistic Regression**
Other fixed hyperparameters: tolerance = 1e-4, intercept scaling = 1 and rest default values



(a) Training Time

(b) Accuracy

Figure 3: Effect of **Tolerance** for **LinearSVC**
Other fixed hyper-parameters : C = 100, intercept scaling = 0.5 and rest default values



(a) Training Time

(b) Accuracy

Figure 4: Effect of **Tolerance** for **Logistic Regression**
Other fixed hyperparameters: C = 100, intercept scaling = 1.0 and rest default values

Table 1: Effect of Loss on Accuracy and training time (tolerance = 0.001, C= 10 (for squared hinge loss)) was chosen for testing

Model	Training Time (s)	Mapping Time (s)	$1 - $ Accuracy
Linear SVC (Squared Hinge loss)	5.192	0.172	0.0066
Linear SVC (Hinge loss)	13.820	0.229	0.0118

Table 2: Effect of Penalty (l1 vs l2) on Accuracy and training time (tolerance = 0.001, C= 10 (for squared hinge loss)) was chosen for testing with LinearSVC

Model	Training Time (s)	Mapping Time (s)	$1 - $ Accuracy
Linear SVC (l2)	5.192	0.172	0.0066
Linear SVC (l1)	268.734	0.15	0.00696

3 Some critical observations

Effect of Tolerance - As shown in Fig. 4 we can observe that the accuracy of the model does not changes even for a very high tolerance because Logistic regression is a very simple model performing a Binary classification on a linearly separable data and it converges relatively quickly only after a relatively high value of tolerance it accuracy gets affected. But the same is not observed in LinearSVC because increasing the tolerance value allows the optimization process to stop earlier, potentially before it has converged to an optimal solution. As a result, the model may not find the best decision boundary, leading to lower accuracy. Increasing the tolerance value might weaken the regularization effect, leading to overfitting or poor generalization on unseen data, which can decrease accuracy.

Effect of C - In the evaluation of LinearSVC and Logistic Regression models, it was observed that increasing the regularization parameter, C, led to longer training times. Test accuracy initially increased with rising C values, reaching a peak at C = 10 for LinearSVC and C = 100 for Logistic Regression. Beyond these optimal points, further increases in C resulted in minimal changes in test accuracy. This suggests an optimal balance between training time and test accuracy, highlighting C = 10 for LinearSVC and C = 100 for Logistic Regression as representing the peak performance points.

Effect of Changing Loss - The discrepancy in accuracy and training time between hinge loss and squared hinge loss in LinearSVC can be attributed to their distinct characteristics and optimization behaviors. Hinge loss, being the default loss function in LinearSVC, prioritizes maximizing the margin between classes while penalizing misclassifications. This emphasis on maximizing the margin can result in a more conservative decision boundary, leading to potentially lower accuracy due to increased tolerance for margin violations. Additionally, hinge loss is non-differentiable at zero and relies on subgradient methods for optimization, which can be computationally expensive and require more iterations to converge. In contrast, squared hinge loss squares the margin violations, resulting in a smoother and differentiable loss function. This smoothness facilitates optimization using gradient-based methods, which often converge faster and more efficiently compared to subgradient methods. Consequently, squared hinge loss may achieve higher accuracy and faster training times by optimizing a more continuous and well-behaved loss function.