CS771 Assignment 1

ABHISHEK KUMAR PATHAK, VEDIT KUMAWAT, ANIKET KUMAR, SONAM, TANEYA SONI

TOTAL POINTS

57 / 60

QUESTION 1

1 Simple XORRO PUF 10 / 10

✓ **+ 10 pts** *A valid derivation showing how the simple XORRO PUF can be cracked using a single linear model*

- 3 pts Minor mistakes in derivation or else not enough details in the derivation.

+ 0 pts Completely wrong or else unanswered

QUESTION 2

2 Advanced XORRO PUF 10 / 10

✓ **+ 10 pts** *A valid derivation showing how the advanced XORRO PUF can be cracked using a collection of linear models*

- 3 pts Minor mistakes in derivation or else not enough details in the derivation.

+ 0 pts Completely wrong or else unanswered

QUESTION 3

3 Coding marks 32 / 35

+ 0 pts Enter marks directly using point adjustment

+ 32 *Point adjustment*

GROUP NO: 53

Grading scheme for code:

Training time tt (in sec): tt < 2 (7 marks), 2 <= tt < 4 (6 marks), 4 <= tt < 8 (5 marks), tt > 8 (4 marks)

Inference time ti (in sec): ti < 2 (10 marks), 2 <= ti < 4 (8 marks), 4 <= ti < 8 (6 marks), ti > 8 (4 marks)

Model size ms (in KB): ms < 50 (8 marks), 50 <= ms < 100 (7 marks), 100 <= ms < 200 (6 marks), ti > 200 (5 marks)

Accuracy ac: ac > 0.99 (10 marks), 0.95 <= ac < 0.99 (8 marks), 0.90 <= ac < 0.95 (6 marks), ac < 0.90 (4 marks)

tt = 0.514 sec : 7 marks

ti = 0.125 sec : 10 marks

ms = 86.55 KB : 7 marks

ac = 0.9502 : 8 marks

TOTAL: 32 marks

QUESTION 4

4 Hyperparameter Experiments 5 / 5

✓ **+ 5 pts** *Reporting effect of at least two hyperparameter changes on training time and test accuracy as asked in the assignment problem statement (see page 7 of assn1.pdf)*

+ 3 pts Effect of at only one hyperparameter change is reported

- 1 pts Minor issues with reporting the results e.g. graphs with unclear axis labels, ambiguous figures, etc

+ 0 pts Completely wrong or else unanswered

+ 3 pts only one of either change on training time or test accuracy in reported

Assignment-1

Sonam
221110057
sonamk22@iitk.ac.in

Aniket Kumar
190134
aniketkr@iitk.ac.in

Vedit Kumawat
190957
vedit@iitk.ac.in

Taneya Soni
221110062
taneyas22@iitk.ac.in

Abhishek Kumar Pathak
22111002
akpathak@iitk.ac.in

1 Question 1

By giving a detailed mathematical derivation (as given in the lecture slides), show how a simple XORRO PUF can be broken by a single linear model. Recall that the simple XORRO PUF has just two XORROs and has no select bits and no multiplexers (see above figure and discussion on Simple XORRO PUF). Thus, the challenge to a simple XORRO PUF has just R bits. More specifically, give derivations for a map $\phi : \{0,1\}^R \rightarrow R^D$ *mapping R-bit 0/1-valued challenge vectors to D-dimensional feature vectors (for some D >0) and show that for any simple XORRO PUF, there exists a linear model i.e.* $w \in R^D, b \in R$ *such that for all challenges* $c \in \{0,1\}^R$*, the following expression gives the correct response.*

$$\frac{1 + sign(w^T\phi(c) + b)}{2}$$

Obervation: To get time period ($frequency^{-1}$) of a RO, we just need to sum up time period of all individual gates. Similarly for XORRO also we need to do this.

Now, for the upper XORRO let time period be t_0 and δ^i_{pq} be the time taken by i^{th} gate present in upper XORRO when the input is p and q ($a_i = q$).

where a_i is the config bit corresponding to ith gate.

So, if the ith bit is set to 1 then time period of ith gate is δ^i_{01} + δ^i_{11}
If the ith bit is set to 0 then time period of ith gate is δ^i_{00} + δ^i_{10}

this is because we are guaranteed to have odd bits set which will flip the input of the first XOR gate which in turn flip the inputs of all the corresponding XOR gates.
Or, we can write time period of ith gate as:

$$t^i_0 = (1 - a_i)(\delta^i_{00} + \delta^i_{10}) + a_i(\delta^i_{01} + \delta^i_{11}) \tag{1}$$

$$
\begin{aligned}
t_0 &= \sum_{i=0}^{N} t_0^i \\
&= \textstyle\sum_{i=0}^{N-1} (1-a_i)(\delta_{10}^i + \delta_{00}^i) + a_i(\delta_{01}^i + \delta_{11}^i) \\
&= \sum_{i=0}^{N-1} (1-a_i)(\delta_{10}^i + \delta_{00}^i) + a_i(\delta_{01}^i + \delta_{11}^i) \\
&= \sum_{i=0}^{N-1} \delta_{10}^i + \delta_{00}^i + a_i(\delta_{01}^i - \delta_{10}^i + \delta_{11}^i - \delta_{00}^i) \\
&= \sum_{i=0}^{N-1} n_i + a_i m_i
\end{aligned}
\tag{2}
$$

$where$

$$
\begin{aligned}
n_i &= {\delta_{10}}^i + {\delta_{00}}^i \\
m_i &= \delta_{01}^i - \delta_{10}^i + \delta_{11}^i - {\delta_{00}}^i
\end{aligned}
\tag{3}
$$

similarly for lower gate we can write

$$t_1 = \sum_{i=0}^{N-1} j_i + a_i k_i \tag{4}$$

$$
\begin{aligned}
j_i &= {\lambda_{10}}^i + {\lambda_{00}}^i \\
k_i &= \lambda_{01}^i - \lambda_{10}^i + \lambda_{11}^i - {\lambda_{00}}^i
\end{aligned}
\tag{5}
$$

where λ_{pq}^i is the time taken by ith gate in lower XORRO when input is p and q.

Now, PUF will output 1 when upper XORRO has lower time period (higher frequency) and 0 if lower XORRO has lower time period .

Now,

$$
\begin{aligned}
t_0 - t_1 &= \sum (n_i - j_i) + a_i(m_i - k_i) \\
&= b + w^T a
\end{aligned}
\tag{6}
$$

where

$$a = [a_0, a_1, ...a_{n-1}]$$

$$w = [m_0 - k_0, m_1 - k_1...., m_{n-1} - k_{n-1}]^T$$

$$b = [n_0 - j_0, n_1 - j_1...., n_{n-1} - j_{n-1}]^T$$

If $(t_0 - t_1) < 0$ or $b + w^T a < 0$ implies

$$\frac{1 + sign(w^T a + b)}{2} = (1-1)/2 = 0$$

Basically,

$$(t_0 - t_1) < 0) -> 0$$
$$(t_0 - t_1) > 0) -> 1$$

Thus proved that XORRO PUF can be crackd by a linear model represented by

$$\frac{1 + sign(w^T \phi(c) + b)}{2}$$

where $\phi(c)$ is a map from $\{0,1\}^R$ to $\mathbb{R}^D$, where $D = R$ and

$$\phi(c_1, c_2, c_3 ...., c_n) = (c_1, c_2, c_3 ..., c_n)$$

2 Question 2

Show how to extend the above linear model to crack an Advanced XORRO PUF. Do this by treating an advanced XORRO PUF as a collection of multiple simple XORRO PUFs. For example, you may use M=$(2^{S-1})(2^S - 1)$ linear models, one for each pair of XORROs, to crack the advanced XORRO PUF.

In the advanced **XORRO PUFF** we have 2 multiplexers that chosses 2 **XORRO PUFFS** and this becomes a normal XORRO PUF and we can use solution of the previous question in order to solve this specific PUF.
Let assume first mux chooses XORRO m_1 and second mux chooses XORRO m_2.So we select all the data points corresponding to values (m_1, m_2) and then train a linear classsification model to estimate w_{m_1,m_2} ,b_{m_1,m_2} and we can predict out corresponding to given two muxes value by just using the above formula:-

$$y_{m_1,m2} = \frac{1 + sign\left(w^T_{m_1,m_2} x_{m_1,m_2} + b_{m_1,m_2} \right)}{2}$$

where x_{m_1,m_2} are the corresponding bits values to (m_1, m_2) XORROs.So this way we can solve advanced XORRO PUF and we get $(2^S) * (2^S - 1)$ model corresponding to each value of (m_1, m_2).

We can employ a clever trick to reduce the number of models by half and also increase the accuracy as we get much more data for each model.For mux value (m1,m2) and (m2,m1) instead of training two different models we can just train one corresponding to (m1,m2) .
During training if data is corresponding to mux value (m_2, m_1) we will flip it output and feed it to model (m_1, m_2) .
During prediction phase we can predict data corresponding to mux value (m_2, m_1) by predicting it through model (m_1, m_2) and flipping its output.
So it reduces our number of models from $(2^S)(2^S - 1)$ to $(2^{S-1})(2^S - 1)$.

3 Question-3

code is submitted in submit.py

1 Simple XORRO PUF **10 / 10**

✓ **+ 10 pts** *A valid derivation showing how the simple XORRO PUF can be cracked using a single linear model*

- 3 pts Minor mistakes in derivation or else not enough details in the derivation.

+ 0 pts Completely wrong or else unanswered

If $(t_0 - t_1) < 0$ or $b + w^T a < 0$ implies

$$\frac{1 + sign(w^T a + b)}{2} = (1 - 1)/2 = 0$$

Basically,

$$(t_0 - t_1) < 0) -> 0$$
$$(t_0 - t_1) > 0) -> 1$$

Thus proved that XORRO PUF can be crackd by a linear model represented by

$$\frac{1 + sign(w^T \phi(c) + b)}{2}$$

where $\phi(c)$ is a map from $\{0, 1\}^R$ to $\mathbb{R}^D$, where $D = R$ and

$$\phi(c_1, c_2, c_3...., c_n) = (c_1, c_2, c_3..., c_n)$$

2 Question 2

Show how to extend the above linear model to crack an Advanced XORRO PUF. Do this by treating an advanced XORRO PUF as a collection of multiple simple XORRO PUFs. For example, you may use M=$(2^{S-1})(2^S - 1)$ linear models, one for each pair of XORROs, to crack the advanced XORRO PUF.

In the advanced **XORRO PUFF** we have 2 multiplexers that chosses 2 **XORRO PUFFS** and this becomes a normal XORRO PUF and we can use solution of the previous question in order to solve this specific PUF.
Let assume first mux chooses XORRO m_1 and second mux chooses XORRO m_2.So we select all the data points corresponding to values (m_1, m_2) and then train a linear classsification model to estimate w_{m_1,m_2} ,b_{m_1,m_2} and we can predict out corresponding to given two muxes value by just using the above formula:-

$$y_{m_1,m2} = \frac{1 + sign\left(w^T_{m_1,m_2} x_{m_1,m_2} + b_{m_1,m_2}\right)}{2}$$

where x_{m_1,m_2} are the corresponding bits values to (m_1, m_2) XORROs.So this way we can solve advanced XORRO PUF and we get $(2^S) * (2^S - 1)$ model corresponding to each value of (m_1, m_2).

We can employ a clever trick to reduce the number of models by half and also increase the accuracy as we get much more data for each model.For mux value (m1,m2) and (m2,m1) instead of training two different models we can just train one corresponding to (m1,m2) .
During training if data is corresponding to mux value (m_2, m_1) we will flip it output and feed it to model (m_1, m_2) .
During prediction phase we can predict data corresponding to mux value (m_2, m_1) by predicting it through model (m_1, m_2) and flipping its output.
So it reduces our number of models from $(2^S)(2^S - 1)$ to $(2^{S-1})(2^S - 1)$.

3 Question-3

code is submitted in submit.py

2 Advanced XORRO PUF 10 / 10

✓ **+ 10 pts** *A valid derivation showing how the advanced XORRO PUF can be cracked using a collection of linear models*

- 3 pts Minor mistakes in derivation or else not enough details in the derivation.

+ 0 pts Completely wrong or else unanswered

If $(t_0 - t_1) < 0$ or $b + w^T a < 0$ implies

$$\frac{1 + sign(w^T a + b)}{2} = (1 - 1)/2 = 0$$

Basically,

$$(t_0 - t_1) < 0) - > 0$$
$$(t_0 - t_1) > 0) - > 1$$

Thus proved that XORRO PUF can be crackd by a linear model represented by

$$\frac{1 + sign(w^T \phi(c) + b)}{2}$$

where $\phi(c)$ is a map from $\{0, 1\}^R$ to $\mathbb{R}^D$, where $D = R$ and

$$\phi(c_1, c_2, c_3...., c_n) = (c_1, c_2, c_3..., c_n)$$

2 Question 2

Show how to extend the above linear model to crack an Advanced XORRO PUF. Do this by treating an advanced XORRO PUF as a collection of multiple simple XORRO PUFs. For example, you may use M=$(2^{S-1})(2^S - 1)$ linear models, one for each pair of XORROs, to crack the advanced XORRO PUF.

In the advanced **XORRO PUFF** we have 2 multiplexers that chosses 2 **XORRO PUFFS** and this becomes a normal XORRO PUF and we can use solution of the previous question in order to solve this specific PUF.
Let assume first mux chooses XORRO m_1 and second mux chooses XORRO m_2.So we select all the data points corresponding to values (m_1, m_2) and then train a linear classsification model to estimate w_{m_1,m_2} ,b_{m_1,m_2} and we can predict out corresponding to given two muxes value by just using the above formula:-

$$y_{m_1,m2} = \frac{1 + sign\left(w^T_{m_1,m_2} x_{m_1,m_2} + b_{m_1,m_2}\right)}{2}$$

where x_{m_1,m_2} are the corresponding bits values to (m_1, m_2) XORROs.So this way we can solve advanced XORRO PUF and we get $(2^S) * (2^S - 1)$ model corresponding to each value of (m_1, m_2).

We can employ a clever trick to reduce the number of models by half and also increase the accuracy as we get much more data for each model.For mux value (m1,m2) and (m2,m1) instead of training two different models we can just train one corresponding to (m1,m2) .
During training if data is corresponding to mux value (m_2, m_1) we will flip it output and feed it to model (m_1, m_2) .
During prediction phase we can predict data corresponding to mux value (m_2, m_1) by predicting it through model (m_1, m_2) and flipping its output.
So it reduces our number of models from $(2^S)(2^S - 1)$ to $(2^{S-1})(2^S - 1)$.

3 Question-3

code is submitted in submit.py

3 Coding marks 32 / 35

+ 0 pts Enter marks directly using point adjustment

+ 32 *Point adjustment*

GROUP NO: 53

Grading scheme for code:

Training time tt (in sec): tt < 2 (7 marks), 2 <= tt < 4 (6 marks), 4 <= tt < 8 (5 marks), tt > 8 (4 marks)

Inference time ti (in sec): ti < 2 (10 marks), 2 <= ti < 4 (8 marks), 4 <= ti < 8 (6 marks), ti > 8 (4 marks)

Model size ms (in KB): ms < 50 (8 marks), 50 <= ms < 100 (7 marks), 100 <= ms < 200 (6 marks), ti > 200 (5 marks)

Accuracy ac: ac > 0.99 (10 marks), 0.95 <= ac < 0.99 (8 marks), 0.90 <= ac < 0.95 (6 marks), ac < 0.90 (4 marks)

tt = 0.514 sec : 7 marks

ti = 0.125 sec : 10 marks

ms = 86.55 KB : 7 marks

ac = 0.9502 : 8 marks

TOTAL: 32 marks

4 Question-4

Report how various hyperparameters affected training time and test accuracy using charts. Report these experiments with both LinearSVC and LogisticRegression methods.

4.1 changing the loss hyperparameter in LinearSVC (hinge vs squared hinge)

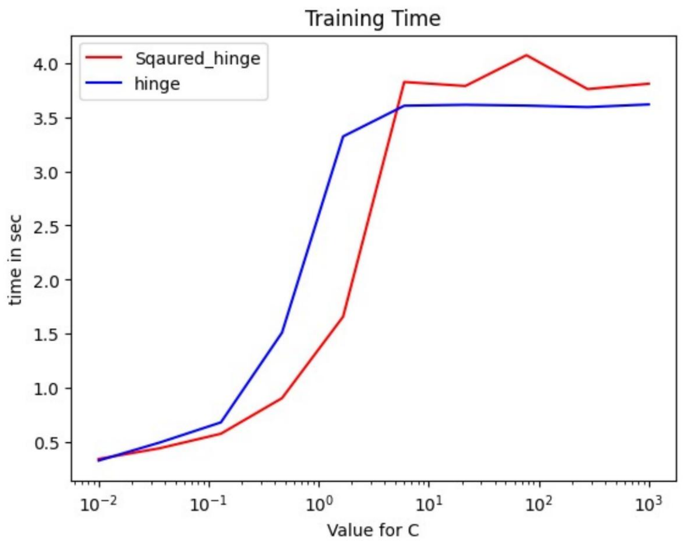

Figure 1:

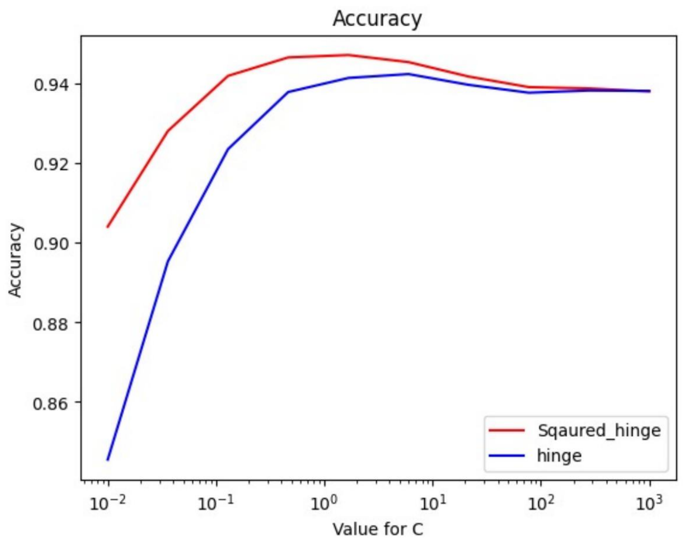

Figure 2:

As we can see from the graphs the squared hinge loss performs better than hinge both in training time and the accuracy at different values for C(hyper parameter).

4.2 setting C hyperparameter in LinearSVC and LogisticRegression to high/low/medium values

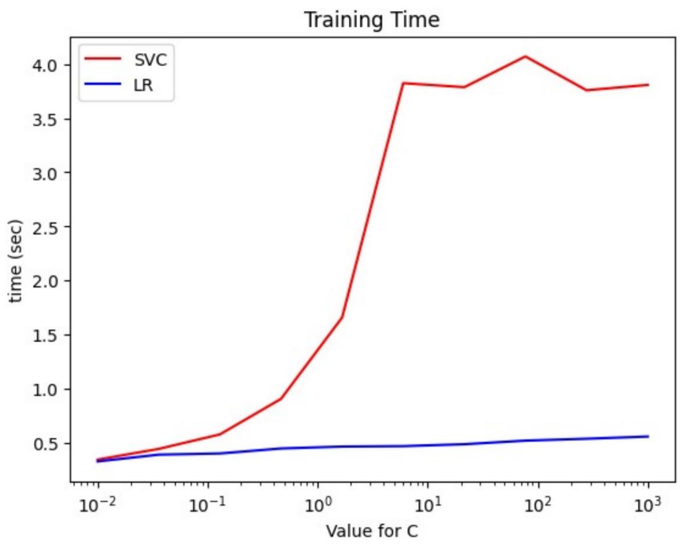

Figure 3:

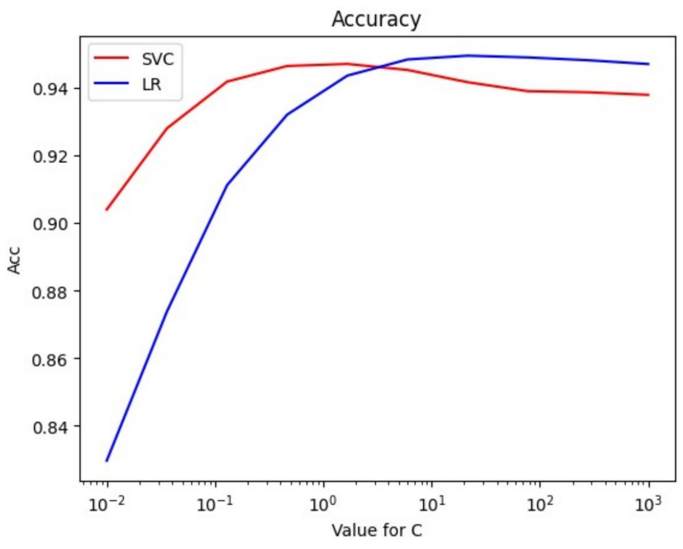

Figure 4:

The SVM performs much better at low values for C as compared to Logistic Regression but as C increases training time for SVM grows much faster as compared to logistic regression for which it increase just slightly as compared to SVC. Around C=5, SVC fails to converge with max iteration=10000. With increasing the value of C, we see the accuracy seems to go up on test data.

4.3 changing the tol hyperparameter in LinearSVC and LogisticRegression to high/low/medium values

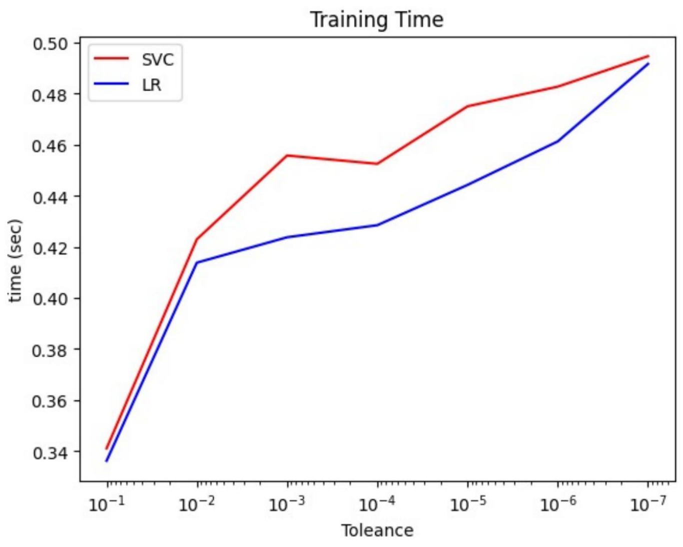

Figure 5:

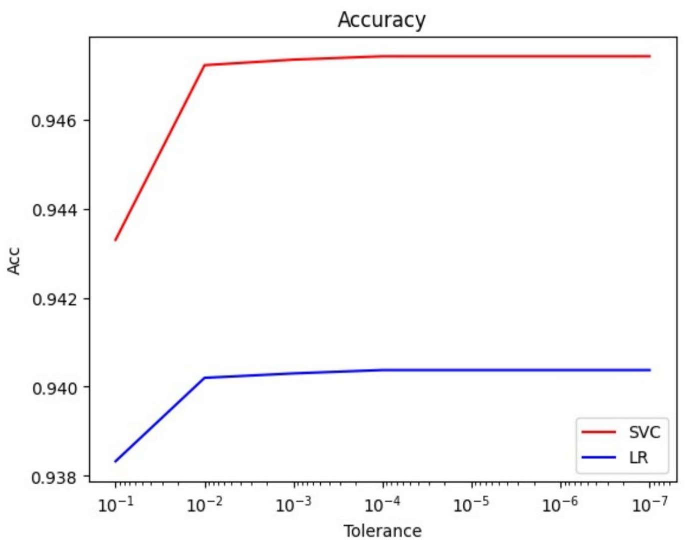

Figure 6:

On decreasing the value of tolerance, the accuracy seems to go up and training times also increases.

4.4 changing the penalty (regularization) hyperparameter in LinearSVC and LogisticRegression (l2 vs l1)

l2 penalty performs much better both in Linear SVC and Logistsic regression as compared to l1, both in training time as well as in accuracy.

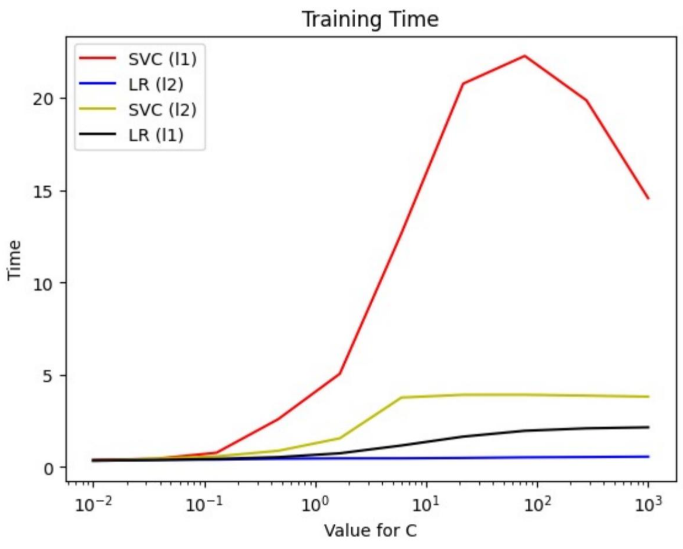

Figure 7:

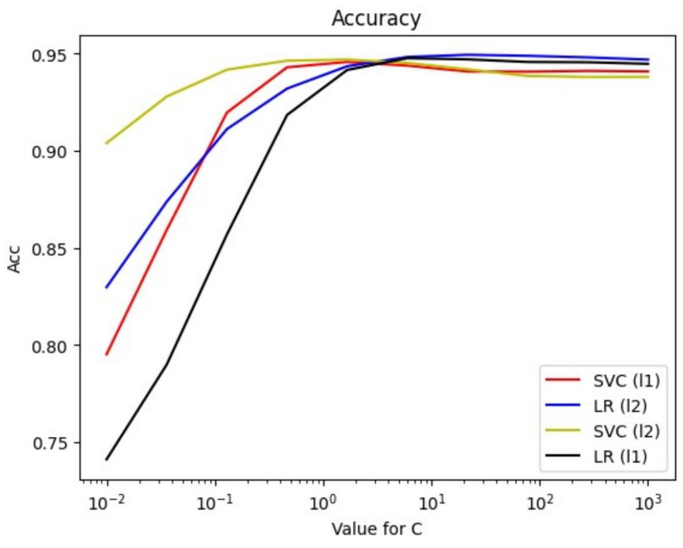

Figure 8:

4 Hyperparameter Experiments **5 / 5**

✓ **+ 5 pts** *Reporting effect of at least two hyperparameter changes on training time and test accuracy as asked in the assignment problem statement (see page 7 of assn1.pdf)*

+ 3 pts Effect of at only one hyperparameter change is reported

- 1 pts Minor issues with reporting the results e.g. graphs with unclear axis labels, ambiguous figures, etc

+ 0 pts Completely wrong or else unanswered

+ 3 pts only one of either change on training time or test accuracy in reported