CS771 Assignment 1

Challa Sai Yohitha (200289) **Nanneboina Hema** (200621) **Saloni Das** (200845) **Harshal Mehta** (200426)

Prashant Kumar Mahour (21103072)

Sparse Linear Model

A Sparse CDU(Conditional Delay Units) PUF with D CDUs out of which only S CDUs are actually participating in the response generation process.

Data:We have provided you with data from a Sparse CDU PUF with D = 2048 and S = 512 i.e. only 512 of the 2048 CDUs are actually active and the rest 1536 are disconnected and play no role in response generation. However, it is not known which 512 CDUs are active. A total of 1600 CRPs have been provided to you.Note that the number of training CRPs (N = 1600 in this case) was deliberately kept small (note that N $\leq$ D) to make the problem challenging.

Question 1

By giving a detailed mathematical derivation (as given in the lecture slides), show how a Sparse CDU PUF can be broken by a single sparse linear model. More specifically, give derivations for a map $\Phi : 0, 1^D \rightarrow R^D$ mapping D-bit 0/1-valued challenge vectors to D-dimensional feature vectors and show that for any S-sparse CDU PUF, there exists an S-sparse linear model i.e. $w \in R^D$ such that $||w||_0 \leq S$ i.e. w has at most S non-zero coordinates and that for all challenges $c \in 0, 1^D$ the following expression $w^T \Phi(c)$ gives the correct response. Note that no bias term (not even a hidden one) is allowed in the linear model.

Solution: These Sparse CDU PUFs can be broken down into a Sparse linear model.

Delays of PUFs are $p_1, p_2, \ldots, p_D$

and Select bits are $c_1, c_2, \ldots, c_D$

We know that all PUFs won't be part of total delay .Out of D PUFs at most S are non-zero,but we don't know which S.

Let consider total delay as, $y = p_1c_1 + p_2c_2 + p_3c_3 + \cdots + p_Dc_D$

Replace all p's by w's and c's by x's for convenience.

i.e $y = w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_Dx_D$

Let

$$w = [w_1, w_2, w_3, \ldots, w_D]^T$$

$$X_1 = [x_1, x_2, x_3, \ldots, x_D]^T$$

$$y_1 = w^T X_1, y_2 = w^T X_2, y_3 = w^T X_3, \ldots, y_N = w^T X_N$$

$$y_i = w^T X_i$$

where

$$w \in \mathbb{R}^D$$

$$X_1 : [0,1]^D$$

$$y_i \in R^D$$

Let

$$Y = [y_1, y_2, y_3, \ldots, y_N]^T$$

$$X = [X_1^T, X_2^T, X_3^T, \ldots, X_N^T]^T$$

$$\boxed{\textbf{Y=Xw}}$$

where

$$w \in \mathbb{R}^D such that ||w||_0 \leq S$$

$$X : [0,1]^{N \times D}$$

$$Y \in \mathbb{R}^N$$

The above expression represents **The Spare Linear Model**

Question 3

Below are described a few methods you can use to solve this problem. Give an account of which methods you tried (you can try methods other than those mentioned below as well) and why did you like one method over the other. Note that you only need to submit code for the method you found to work the best but in your report you must describe your experiences with at least one other method that you tried.

Solution: We solved this problem with two major methods, **LASSO technique** and **Projection gradient descent**,as theses two solve the problem along with sparsifying the solution unlike Naive technique.

THE LASSO TECHNIQUE

- The LASSO (least absolute shrinkage and selection operator) technique encourages learning of a sparse model by using L1 regularization by modifying the problem as follows:

$$v \overset{def}{=} arg \underset{w \in R^D}{min} \lambda \cdot ||w||_1 + \frac{1}{2n} \sum_{i=1}^{n} (w^T x_i - y_i)^2 \tag{1}$$

- To solve LASSO we used coordinate descent to solve as it has less run time when compared to sub-gradient techniques like ,stochastic and mini-batch stochastic variants. And coordinate descent updates each coefficient individually.

$$v_j \overset{def}{=} arg \underset{t \in R^D}{min} \lambda \cdot |t| + \frac{1}{2n} \sum_{i=1}^{n} (t \cdot x_{ij} + \sum_{k \neq j} v_k x_{ik} - y_i)^2 \tag{2}$$

- Optimized using coordinate descent and sparsified using soft thresholding

PROJECTION GRADIENT DESCENT

- This technique realizes that the hard thresholding operator is simply a projection onto the set of sparse vectors and applies gradient descent to the original problem, using hard thresholding to keep the iterates sparse

$$l(w) \overset{def}{=} \frac{1}{2n}\sum_{i=1}^{n}(w^T x_i - yi)^2$$

(3)

- Here we use the gradient of loss function ,update the coefficients using

$$z^t \leftarrow w^t - \eta \cdot \nabla l(w^t) \tag{4}$$

and applying the hard thresholding

- The results of this techniques are pretty much better than LASSO ,both mae and run time

Question 4

The method you submitted may have hyperparameters such as regularization constants, step lengths etc. Describe in detail how you chose the optimal value of these hyperparameters.

For example, if you used grid search, you must show which all values you tried for each hyperparameter and what criterion did you use to choose the best one

- For projection gradient descent we use a self learning rate called/step length = η =0.01 we took this we value for which it gives the minimum possible error.
- No hyper parameters are used

Table 1: MAE and Run Time for different values of step length for Projection gradient descent

η (step length) $\downarrow$	Run time	MAE
0.005	$0.9897sec$	$6.4e + 198$
0.004	$0.990sec$	$3.13e + 26$
0.001	$0.996sec$	42.87

Note: The values correspond to when the code was executed in one of the group member's VS code, not Google Colab.