CS771
Assignment 1

Group Name: ML EXPRESS

Name	**Roll Number**
Kumar Sambhav	210545
Mali Ashish Ramniwas	210574
Yogesh	211205
Sunny Munda	211077
Jahnvi Singh	210460

Step-by-Step Derivation

1. Understanding the Given Equations

From the image:

$$t_1^u = (1 - c_2) \cdot (t_1^u + p_2) + c_2 \cdot (t_1^l + s_2)$$
$$t_2^l = (1 - c_2) \cdot (t_1^l + q_2) + c_2 \cdot (t_1^u + r_2)$$

For the initial conditions:

$$t_1^u = (1 - c_1)p_1 + c_1 s_1$$
$$t_1^l = (1 - c_1)q_1 + c_1 r_1$$

2. Express t_3^u and t_4^u

To express t_3^u and t_4^u, we recursively apply the equations provided:

$$t_3^u = (1 - c_3) \cdot (t_2^u + p_3) + c_3 \cdot (t_2^l + s_3)$$
$$t_3^l = (1 - c_3) \cdot (t_2^l + q_3) + c_3 \cdot (t_2^u + r_3)$$

$$t_4^u = (1 - c_4) \cdot (t_3^u + p_4) + c_4 \cdot (t_3^l + s_4)$$
$$t_4^l = (1 - c_4) \cdot (t_3^l + q_4) + c_4 \cdot (t_3^u + r_4)$$

3. General Formulation

The recursive formulation allows us to generalize for t_{64}^u as:

$$t_i^u = (1 - c_i) \cdot (t_{i-1}^u + p_i) + c_i \cdot (t_{i-1}^l + s_i)$$
$$t_i^l = (1 - c_i) \cdot (t_{i-1}^l + q_i) + c_i \cdot (t_{i-1}^u + r_i)$$

Derivation of $\phi(c)$ and W

Step-by-Step Mathematical Derivation

1. Challenge Vector Transformation:

- Each bit c_i in the challenge vector c is transformed using $d_i = 1 - 2c_i$. This results in $d_i = 1$ if $c_i = 0$ and $d_i = -1$ if $c_i = 1$.

2. Cumulative Product:

- For a given challenge vector, a cumulative product x_i is computed such that:

$$x_i = \prod_{j=1}^{i} d_j$$

- This creates a sequence of products up to each bit in the challenge vector.

3. Outer Product and Feature Vector Construction:

- The feature vector $\phi(c)$ is created by taking the outer product of the cumulative product vector with itself and then extracting the upper triangle (excluding the diagonal). This includes all pairwise interactions of the cumulative products.
- Finally, $\phi(c)$ is constructed by concatenating the cumulative product vector x with the upper triangle elements.

Detailed Mathematical Expression for $\phi(c)$

Given a 32-bit challenge vector $c = [c_1, c_2, \ldots, c_{32}]$:

1. Transform the challenge bits:

$$d_i = 1 - 2c_i$$

For all $i \in \{1, 2, \ldots, 32\}$.

2. Compute the cumulative product vector x:

$$x_i = \prod_{j=1}^{i} d_j$$

For all $i \in \{1, 2, \ldots, 32\}$.

3. Construct the feature vector $\phi(c)$:

$$\phi(c) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{32} \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_{31} x_{32} \end{bmatrix}$$

The total number of terms in $\phi(c)$ is 32 (cumulative products) + 496 (pairwise interactions) = 528.

Weight Vector W and Bias b

1. Weight Vector W:

$$W = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_{32} \\ w_{112} \\ w_{113} \\ \vdots \\ w_{31,32} \end{bmatrix}$$

W is a 528-dimensional vector.

2. Bias Term b:

The bias term b is a scalar value.

Linear Model Expression

Given $\phi(c)$ and W, the linear model to predict t_u is:

$$t_u = W^T \phi(c) + b$$

Answering the Questions

Question 1

- **Mathematical Derivation of $\phi(c)$:**

$$\phi(c) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{32} \\ x_1 x_2 \\ x_1 x_3 \\ \vdots \\ x_{31} x_{32} \end{bmatrix}$$

  Where $x_i = \prod_{j=1}^{i}(1 - 2c_j)$.

- **Existence of W and b:**

  For any arbiter PUF, there exists a weight vector $W \in \mathbb{R}^{528}$ and a bias term $b \in \mathbb{R}$ such that:

$$t_u = W^T \phi(c) + b$$

Question 2

- **Dimensionality of the Linear Model:**

  The dimensionality of the linear model needed to predict the arrival time of the upper signal for an arbiter PUF is **528**.

Question 3

Linear Model for Response0

To model Response, we need to compare the lower signals from PUF0 and PUF1:

$$r_0(c) = \frac{1 + sign(t_l^{PUF0}(c) - t_l^{PUF1}(c))}{2}$$

From part 1, we have the linear model for the lower signal time t_l:

$$t_l^{PUF0}(c) = W_{0l}^T \phi(c) + b_{0l}$$

$$t_l^{PUF1}(c) = W_{1l}^T \phi(c) + b_{1l}$$

Substituting these into the response equations:

$$r_0(c) = \frac{1 + sign((W_{0l}^T \phi(c) + b_{0l}) - (W_{1l}^T \phi(c) + b_{1l}))}{2}$$

Simplifying:

$$r_0(c) = \frac{1 + sign((W_{0l} - W_{1l})^T \phi(c) + (b_{0l} - b_{1l}))}{2}$$

Define:

$$\tilde{W} = W_{0l} - W_{1l}$$

$$\tilde{b} = b_{0l} - b_{1l}$$

Thus, the linear model for Response0 is:

$$r_0(c) = \frac{1 + sign(\tilde{W}^T \tilde{\phi}(c) + \tilde{b})}{2}$$

Linear Model for Response1

To model Response1, we need to compare the upper signals from PUF0 and PUF1:

$$r_1(c) = \frac{1 + sign(t_u^{PUF0}(c) - t_u^{PUF1}(c))}{2}$$

From part 1, we have the linear model for the upper signal time t_u:

$$t_u^{PUF0}(c) = W_{0u}^T \phi(c) + b_{0u}$$

$$t_u^{PUF1}(c) = W_{1u}^T \phi(c) + b_{1u}$$

Substituting these into the response equation:

$$r_1(c) = \frac{1 + sign((W_{0u}^T \phi(c) + b_{0u}) - (W_{1u}^T \phi(c) + b_{1u}))}{2}$$

Simplifying:

$$r_1(c) = \frac{1 + sign((W_{0u} - W_{1u})^T \phi(c) + (b_{0u} - b_{1u}))}{2}$$

Define:

$$\tilde{W}_1 = W_{0u} - W_{1u}$$

$$\tilde{b}_1 = b_{0u} - b_{1u}$$

Thus, the linear model for Response1 is:

$$r_1(c) = \frac{1 + sign(\tilde{W}_1^T \tilde{\phi}(c) + \tilde{b}_1)}{2}$$

Question 4

To predict Response0 and Response1 for a COCO-PUF, we need to derive the dimensionality of the feature vector $\tilde{\phi}(c)$ used in the linear models.

Deriving the Dimensionality

1. Initial Feature Vector:

The challenge c is a 32-bit binary vector. Each bit c_i can be either 0 or 1.

2. First-Order Terms:

The first-order terms in the feature vector are derived from the challenge bits themselves and the products of differences:

$$\{1, (1 - 2c_1), (1 - 2c_2), \ldots, (1 - 2c_{32})\}$$

This gives us $1 + 32 = 33$ terms.

3. Second-Order Terms:

The second-order terms are derived from the pairwise products of the first-order terms:

$$\{(1 - 2c_i)(1 - 2c_j) \mid \text{for all } 1 \leq i < j \leq 32\}$$

The number of unique pairwise products is given by the combination formula $\binom{32}{2}$:

$$\binom{32}{2} = \frac{32 \cdot 31}{2} = 496$$

4. Total Dimensionality:

The total dimensionality $\tilde{\phi}(c)$ of the feature vector $\phi(c)$ is the sum of the first-order and second-order terms:

$$\tilde{D} = 33 + 496 = 529$$

Summary

To predict Response0 and Response1 for a COCO-PUF, the linear model requires a feature vector $\phi(c)$ with a dimensionality of 529. This dimensionality includes:

- 1 constant term
- 32 first-order terms
- 496 second-order terms

Therefore, the linear models for predicting r_0 and r_1 need to have a dimensionality of 529.

Question 5

```python
import numpy as np
import sklearn
from scipy.linalg import khatri_rao
from sklearn.svm import LinearSVC

# You are allowed to import any submodules of sklearn that learn linear models e.g. sklearn.svm etc
# You are not allowed to use other libraries such as keras, tensorflow etc
# You are not allowed to use any scipy routine other than khatri_rao

# SUBMIT YOUR CODE AS A SINGLE PYTHON (.PY) FILE INSIDE A ZIP ARCHIVE
# THE NAME OF THE PYTHON FILE MUST BE submit.py

# DO NOT CHANGE THE NAME OF THE METHODS my_fit, my_map etc BELOW
# THESE WILL BE INVOKED BY THE EVALUATION SCRIPT. CHANGING THESE NAMES WILL CAUSE EVALUATION FAILURE

# You may define any new functions, variables, classes here
# For example, functions to calculate next coordinate or step length
def get_x_vector(row):
    row_d = 1 - 2 * row
    x_vector = np.cumprod(row_d)
    return x_vector

def get_final_vector(x_vector):
    outer_product = np.outer(x_vector, x_vector)
    # Get the indices of the upper triangle without including the diagonal
    upper_triangle_indices = np.triu_indices(len(x_vector), k=1)
    # Extract the upper triangle excluding the diagonal from the outer product
    final_vector = outer_product[upper_triangle_indices]
    return np.concatenate([x_vector, final_vector])



################################
# Non Editable Region Starting #
################################
def my_fit( X_train, y0_train, y1_train ):
################################
#  Non Editable Region Ending  #
################################

    # Map challenges to feature vectors
    X_train_mapped = my_map(X_train)

    # Train LinearSVC model for Response0
    model_0 = LinearSVC(dual=False, C=3, tol=0.00015)
    model_0.fit(X_train_mapped, y0_train)

    # Train LinearSVC model for Response1
    model_1 = LinearSVC(dual=False, C=3, tol=0.00015)
    model_1.fit(X_train_mapped, y1_train)

    # Extract weights and bias terms
    w0 = model_0.coef_[0]
    b0 = model_0.intercept_[0]
    w1 = model_1.coef_[0]
    b1 = model_1.intercept_[0]

```

```python

    # Use this method to train your models using training CRPs
    # X_train has 32 columns containing the challenge bits
    # y0_train contains the values for Response0
    # y1_train contains the values for Response1

    # THE RETURNED MODELS SHOULD BE TWO VECTORS AND TWO BIAS TERMS
    # If you do not wish to use a bias term, set it to 0
    return w0, b0, w1, b1


################################
# Non Editable Region Starting #
################################
def my_map( X ):
################################
#  Non Editable Region Ending  #
################################
# Apply get_x_vector to each row of X
    feat = np.apply_along_axis(get_x_vector, axis=1, arr=X)
    # Apply get_final_vector to each element of x_vector
    feat = np.apply_along_axis(get_final_vector, axis=1, arr=feat)

    # Use this method to create features.
    # It is likely that my_fit will internally call my_map to create features for train points

    return feat

```

Question 6

Part 1

Accuracy for LinearSVC with hinge loss: 97.84%

Accuracy for LinearSVC with squared hinge loss: 97.39%

Part 2

Changing the loss hyperparameter in LinearSVC (hinge vs squared hinge)

	L1	**L2**
LinearSVC	85.6%	82.6%
LogisticRegression	88.2%	87.5%