CS771 Assignment 1

RATHOD PREET, PRIYANKA MEENA, SHUBHAM KUMAR, SUBODH KUMAR, SHREYASI MANDAL

TOTAL POINTS

33 / 65

QUESTION 1

1 XOR derivation 5 / 5

✓ **+ 3 pts** *A correct example of a pair of functions m, f that works*

✓ **+ 2 pts** *Derivation of the result*

- 1 pts Derivation does not take into account edge cases e.g. all 0s or all 1s inputs.

+ 0 pts Completely wrong or else unanswered

QUESTION 2

2 Sign derivation 5 / 5

✓ **+ 5 pts** *A proof of the result with sufficient calculations*

- 2 pts Minor mistakes in proof e.g. not taking into account edge cases e.g. all 0s

- 1 pts Insufficient calculations

+ 0 pts Completely wrong or else unanswered

QUESTION 3

3 Map derivation 0 / 10

+ 10 pts A proof of the result with sufficient calculations

+ 5 pts BONUS: if the derived dimensionality is less than 100

✓ **- 2 pts** *Insufficient calculations*

- 2 pts Minor mistakes e.g. not taking into account the bias term or wrong dimensionality or sign error

+ 0 pts Completely wrong or else unanswered

QUESTION 4

4 Code evaluation 13 / 35

- 10 pts Illegal use of libraries or other non-permitted actions

+ 0 pts Completely wrong or else unanswered

+ 13 *Point adjustment*

GROUP NO: 62

Grading scheme for code:
Feature map dimensionality d: d < 200 (5 bonus), 200 < d <= 600 (2 bonus), d > 600 (0 bonus)
Hinge loss h: h < 1 (3 marks), 1 <= h < 10 (2 marks), h > 10 (1 mark) -- for all timeouts
Error rate e: e < 0.01 (3 marks), 0.01 <= e < 0.1 (2 marks), e > 0.1 (1 mark) -- for all timeouts

d = 512 : 2 bonus marks
For timeout t = 0.2 sec, h = 8.54345897 : 2 marks
For timeout t = 0.2 sec, e = 0.3185 : 1 mark
For timeout t = 0.5 sec, h = 12.2374774 : 1 mark

For timeout t = 0.5 sec, e = 0.2805 : 1 mark

For timeout t = 1.0 sec, h = 18.67850125 : 1 mark

For timeout t = 1.0 sec, e = 0.27585 : 1 mark

For timeout t = 2.0 sec, h = 30.98932262 : 1 mark

For timeout t = 2.0 sec, e = 0.2478 : 1 mark

For timeout t = 5.0 sec, h = 64.43440984 : 1 mark

For timeout t = 5.0 sec, e = 0.23062 : 1 mark

TOTAL: 13 marks

QUESTION 5

5 Hyperparameter description 5 / 5

✓ **+ 3 pts** *Enumeration of all hyperparameters and values considered for those hyperparameters*

✓ **+ 2 pts** *Description of how the best value was obtained for each hyperparameter*

+ 0 pts Completely wrong or else unanswered

1

2

QUESTION 6

6 Convergence plot 5 / 5

✓ **+ 5 pts** *A digitally generated plot is provided*

- 1 pts Minor mistakes like unlabeled x axis or y axis

+ 0 pts Completely wrong or else unanswered

+ 3 pts Not the classification accuracy graaph

CS771 (Introduction to Machine Learning) : Assignment 1

Shreyasi Mandal
200956
Department of Computer Science
Indian Institute of Technology, Kanpur
shreyansi20@iitk.ac.in

Rathod Preet
200775
Department of Computer Science
Indian Institute of Technology, Kanpur
preetr20@iitk.ac.in

Priyanka Meena
200731
Department of Computer Science
Indian Institute of Technology, Kanpur
priyankam20@iitk.ac.in

Subodh Kumar
201007
Department of Computer Science
Indian Institute of Technology, Kanpur
subodhkum20@iitk.ac.in

Subham Kumar
200966
Department of Computer Science
Indian Institute of Technology, Kanpur
shuhamk20@iitk.ac.in

1

Let $(b_1, b_2, ..., b_n)$ contains x 0's and y 1's. If y is odd, then we should get 1 as answer of $XOR(b_1, b_2, ..., b_n)$ and 0 otherwise (Note that the result of XOR does not depend on x). By using this fact, we can logically give a negative mapping to 1 and positive mapping to 0.

b_i	$m(b_i)$
0	+1
1	-1

To find mapping $f : \{-1, +1\} \longrightarrow \{0, 1\}$, we'll take special case of 2 bits and then generalize the result.

b_1	b_2	$XOR(b_1, b_2)$	$f(m(b_1) \cdot m(b_2))$
0	0	0	$f(1)$
0	1	1	$f(-1)$
1	0	1	$f(-1)$
1	1	0	$f(1)$

By looking at the table, we can clearly figure out the mapping f as follows:

x	$f(x)$
+1	0
-1	1

Preprint. Under review.

1 XOR derivation **5 / 5**

✓ **+ 3 pts** *A correct example of a pair of functions m, f that works*

✓ **+ 2 pts** *Derivation of the result*

- 1 pts Derivation does not take into account edge cases e.g. all 0s or all 1s inputs.

+ 0 pts Completely wrong or else unanswered

To verify the obtained mapping, in general, let us assume we have x 0's and y 1's, then

$$\prod_{i=1}^{n} m(b_i) = \left(\prod_{i=1}^{x} m(0)\right) \cdot \left(\prod_{i=1}^{y} m(1)\right)$$

$$\prod_{i=1}^{n} m(b_i) = \left(\prod_{i=1}^{x} 1\right) \cdot \left(\prod_{i=1}^{x} (-1)\right)$$

$$\prod_{i=1}^{n} m(b_i) = (-1)^y$$

So, we get the following result:

y	$t = \prod_{i=1}^{n} m(b_i)$	$f(t)$	$XOR(b_1, b_2, ..., b_n)$
odd	-1	1	1
even	+1	0	0

In above table, last two columns are exactly same. So our mapping fits for any general case also.

2

Let us assume that we have n non-zero real numbers $(r_1, r_2, ..., r_n)$, out of which x are positive $(r_x = [r_{x_i}])$ and y are negative $(r_y = [r_{y_i}])$.
We can write any real number x as $x = sign(x) \cdot |x|$. This implies, $sign(x) = \frac{x}{|x|}$.

$$LHS = \prod_{i=1}^{n} sign(r_i)$$

$$LHS = \prod_{i=1}^{n} \frac{r_i}{|r_i|}$$

We can separately multiply positive and negative terms and get:

$$LHS = \left(\prod_{i \in r_x} \frac{r_i}{|r_i|}\right) \cdot \left(\prod_{j \in r_y} \frac{r_j}{|r_j|}\right)$$

$$LHS = \left(\prod_{i \in r_x} (+1)\right) \cdot \left(\prod_{j \in r_y} (-1)\right)$$

$$LHS = (+1)^x \cdot (-1)^y$$

$$LHS = (-1)^y$$

Now we will simplify RHS and see what the result comes out to be:

$$RHS = sign\left(\prod_{i=1}^{n} r_i\right)$$

$$RHS = \frac{\prod_{i=1}^{n} r_i}{|\prod_{i=1}^{n} r_i|}$$

$$RHS = \frac{\prod_{i=1}^{n} r_i}{\prod_{i=1}^{n} |r_i|}$$

$$RHS = \prod_{i=1}^{n} \frac{r_i}{|r_i|}$$

By applying similar steps as applied in LHS, we will get

$$RHS = (-1)^y$$

2 Sign derivation 5 / 5

✓ **+ 5 pts** *A proof of the result with sufficient calculations*

- 2 pts Minor mistakes in proof e.g. not taking into account edge cases e.g. all 0s

- 1 pts Insufficient calculations

+ 0 pts Completely wrong or else unanswered

Hence,

$$LHS = RHS$$

When we have one or more r_i equal to 0, we will proceed as follows:

$$LHS = \prod_{i=1}^{n} sign(r_i)$$

but if at least on of the r_i is 0 then the whole product will be 0.

$$LHS = 0$$

For RHS,

$$RHS = sign\left(\prod_{i=1}^{n} r_i\right)$$

Here also, if at least one of r_i is 0, then whole product will become 0. So,

$$RHS = sign(0)$$

$$RHS = 0$$

Hence,

$$LHS = RHS$$

3

$$(\tilde{\mathbf{u}}^T\tilde{x}) \cdot (\tilde{\mathbf{v}}^T\tilde{x}) \cdot (\tilde{\mathbf{w}}^T\tilde{x}) = \left(\sum_{i=1}^{9} \tilde{u_i}\tilde{x_i}\right) \cdot \left(\sum_{i=1}^{9} \tilde{v_i}\tilde{x_i}\right) \cdot \left(\sum_{i=1}^{9} \tilde{w_i}\tilde{x_i}\right)$$

$$(\tilde{\mathbf{u}}^T\tilde{x}) \cdot (\tilde{\mathbf{v}}^T\tilde{x}) \cdot (\tilde{\mathbf{w}}^T\tilde{x}) = \sum_{i=1}^{9}\sum_{j=1}^{9}\sum_{k=1}^{9} \tilde{u_i}\tilde{v_j}\tilde{w_k}\tilde{x_i}\tilde{x_j}\tilde{x_k}$$

We can figure out from above equation that,

$$\mathbf{W} = (\tilde{u_1}\tilde{v_1}\tilde{w_1}, \tilde{u_1}\tilde{v_1}\tilde{w_2}, ..., \tilde{u_1}\tilde{v_1}\tilde{w_9}, \tilde{u_1}\tilde{v_2}\tilde{w_1}, ..., \tilde{u_1}\tilde{v_2}\tilde{w_9}, ..., \tilde{u_9}\tilde{v_9}\tilde{w_9})$$

And,

$$\phi(\tilde{x}) = (\tilde{x_1}\tilde{x_1}\tilde{x_1}, \tilde{x_1}\tilde{x_1}\tilde{x_2}, ..., \tilde{x_9}\tilde{x_9}\tilde{x_9})$$

4

The given problem was solved by learning the linear model **W**. At first, we implemented the *solver* method using **Stochastic Dual Coordinate Maximization (SDCM)** method. We tried **random**, **randperm**, and **cyclic** coordinate selection choices, each of which led to an accuracy of **61.18**%

After this, we began implementing gradient descent variants namely, **Vanilla Gradient Descent (VGD)** and **Stochastic Gradient Descent (SGD)**.
For step lengths, we decided to proceed with two schemes - **linear** scheme

$$\eta_t = \frac{\eta}{t}$$

and **quadratic** scheme

$$\eta_t = \frac{\eta}{\sqrt{t}}$$

After numerous experiments, the combinations of - **SGD + quadratic** and **VGD + linear** were selected.
The bias term and learning rate (η) were set as **-2** and **0.0001** respectively, after much experimentation as seen from the table 1 below -

Looking at the above results, we decided to use the **Vanilla Gradient Descent** method, with **linear** step length scheme, a learning rate (η) of **0.0001**, a bias term of **-2** getting an accuracy of **77.87%**.

3 Map derivation 0 / 10

+ 10 pts A proof of the result with sufficient calculations

+ 5 pts BONUS: if the derived dimensionality is less than 100

✓ **- 2 pts** *Insufficient calculations*

- 2 pts Minor mistakes e.g. not taking into account the bias term or wrong dimensionality or sign error

+ 0 pts Completely wrong or else unanswered

Hence,

$$LHS = RHS$$

When we have one or more r_i equal to 0, we will proceed as follows:

$$LHS = \prod_{i=1}^{n} sign(r_i)$$

but if at least on of the r_i is 0 then the whole product will be 0.

$$LHS = 0$$

For RHS,

$$RHS = sign\left(\prod_{i=1}^{n} r_i\right)$$

Here also, if at least one of r_i is 0, then whole product will become 0. So,

$$RHS = sign(0)$$

$$RHS = 0$$

Hence,

$$LHS = RHS$$

3

$$(\tilde{\mathbf{u}}^T\tilde{x}) \cdot (\tilde{\mathbf{v}}^T\tilde{x}) \cdot (\tilde{\mathbf{w}}^T\tilde{x}) = \left(\sum_{i=1}^{9} \tilde{u_i}\tilde{x_i}\right) \cdot \left(\sum_{i=1}^{9} \tilde{v_i}\tilde{x_i}\right) \cdot \left(\sum_{i=1}^{9} \tilde{w_i}\tilde{x_i}\right)$$

$$(\tilde{\mathbf{u}}^T\tilde{x}) \cdot (\tilde{\mathbf{v}}^T\tilde{x}) \cdot (\tilde{\mathbf{w}}^T\tilde{x}) = \sum_{i=1}^{9}\sum_{j=1}^{9}\sum_{k=1}^{9} \tilde{u_i}\tilde{v_j}\tilde{w_k}\tilde{x_i}\tilde{x_j}\tilde{x_k}$$

We can figure out from above equation that,

$$\mathbf{W} = (\tilde{u_1}\tilde{v_1}\tilde{w_1}, \tilde{u_1}\tilde{v_1}\tilde{w_2}, ..., \tilde{u_1}\tilde{v_1}\tilde{w_9}, \tilde{u_1}\tilde{v_2}\tilde{w_1}, ..., \tilde{u_1}\tilde{v_2}\tilde{w_9}, ..., \tilde{u_9}\tilde{v_9}\tilde{w_9})$$

And,

$$\phi(\tilde{x}) = (\tilde{x_1}\tilde{x_1}\tilde{x_1}, \tilde{x_1}\tilde{x_1}\tilde{x_2}, ..., \tilde{x_9}\tilde{x_9}\tilde{x_9})$$

4

The given problem was solved by learning the linear model **W**. At first, we implemented the *solver* method using **Stochastic Dual Coordinate Maximization (SDCM)** method. We tried **random**, **randperm**, and **cyclic** coordinate selection choices, each of which led to an accuracy of **61.18**%

After this, we began implementing gradient descent variants namely, **Vanilla Gradient Descent (VGD)** and **Stochastic Gradient Descent (SGD)**.
For step lengths, we decided to proceed with two schemes - **linear** scheme

$$\eta_t = \frac{\eta}{t}$$

and **quadratic** scheme

$$\eta_t = \frac{\eta}{\sqrt{t}}$$

After numerous experiments, the combinations of - **SGD + quadratic** and **VGD + linear** were selected.
The bias term and learning rate (η) were set as **-2** and **0.0001** respectively, after much experimentation as seen from the table 1 below -

Looking at the above results, we decided to use the **Vanilla Gradient Descent** method, with **linear** step length scheme, a learning rate (η) of **0.0001**, a bias term of **-2** getting an accuracy of **77.87%**.

4 Code evaluation 13 / 35

- 10 pts Illegal use of libraries or other non-permitted actions

+ 0 pts Completely wrong or else unanswered

+ 13 *Point adjustment*

GROUP NO: 62

Grading scheme for code:

Feature map dimensionality d: d < 200 (5 bonus), 200 < d <= 600 (2 bonus), d > 600 (0 bonus)

Hinge loss h: h < 1 (3 marks), 1 <= h < 10 (2 marks), h > 10 (1 mark) -- for all timeouts

Error rate e: e < 0.01 (3 marks), 0.01 <= e < 0.1 (2 marks), e > 0.1 (1 mark) -- for all timeouts

d = 512 : 2 bonus marks

For timeout t = 0.2 sec, h = 8.54345897 : 2 marks

For timeout t = 0.2 sec, e = 0.3185 : 1 mark

For timeout t = 0.5 sec, h = 12.2374774 : 1 mark

For timeout t = 0.5 sec, e = 0.2805 : 1 mark

For timeout t = 1.0 sec, h = 18.67850125 : 1 mark

For timeout t = 1.0 sec, e = 0.27585 : 1 mark

For timeout t = 2.0 sec, h = 30.98932262 : 1 mark

For timeout t = 2.0 sec, e = 0.2478 : 1 mark

For timeout t = 5.0 sec, h = 64.43440984 : 1 mark

For timeout t = 5.0 sec, e = 0.23062 : 1 mark

TOTAL: 13 marks

Hence,

$$LHS = RHS$$

When we have one or more r_i equal to 0, we will proceed as follows:

$$LHS = \prod_{i=1}^{n} sign(r_i)$$

but if at least on of the r_i is 0 then the whole product will be 0.

$$LHS = 0$$

For RHS,

$$RHS = sign\left(\prod_{i=1}^{n} r_i\right)$$

Here also, if at least one of r_i is 0, then whole product will become 0. So,

$$RHS = sign(0)$$

$$RHS = 0$$

Hence,

$$LHS = RHS$$

3

$$(\tilde{\mathbf{u}}^T\tilde{x}) \cdot (\tilde{\mathbf{v}}^T\tilde{x}) \cdot (\tilde{\mathbf{w}}^T\tilde{x}) = \left(\sum_{i=1}^{9} \tilde{u_i}\tilde{x_i}\right) \cdot \left(\sum_{i=1}^{9} \tilde{v_i}\tilde{x_i}\right) \cdot \left(\sum_{i=1}^{9} \tilde{w_i}\tilde{x_i}\right)$$

$$(\tilde{\mathbf{u}}^T\tilde{x}) \cdot (\tilde{\mathbf{v}}^T\tilde{x}) \cdot (\tilde{\mathbf{w}}^T\tilde{x}) = \sum_{i=1}^{9}\sum_{j=1}^{9}\sum_{k=1}^{9} \tilde{u_i}\tilde{v_j}\tilde{w_k}\tilde{x_i}\tilde{x_j}\tilde{x_k}$$

We can figure out from above equation that,

$$\mathbf{W} = (\tilde{u_1}\tilde{v_1}\tilde{w_1}, \tilde{u_1}\tilde{v_1}\tilde{w_2}, ..., \tilde{u_1}\tilde{v_1}\tilde{w_9}, \tilde{u_1}\tilde{v_2}\tilde{w_1}, ..., \tilde{u_1}\tilde{v_2}\tilde{w_9}, ..., \tilde{u_9}\tilde{v_9}\tilde{w_9})$$

And,

$$\phi(\tilde{x}) = (\tilde{x_1}\tilde{x_1}\tilde{x_1}, \tilde{x_1}\tilde{x_1}\tilde{x_2}, ..., \tilde{x_9}\tilde{x_9}\tilde{x_9})$$

4

The given problem was solved by learning the linear model **W**. At first, we implemented the *solver* method using **Stochastic Dual Coordinate Maximization (SDCM)** method. We tried **random**, **randperm**, and **cyclic** coordinate selection choices, each of which led to an accuracy of **61.18**%

After this, we began implementing gradient descent variants namely, **Vanilla Gradient Descent (VGD)** and **Stochastic Gradient Descent (SGD)**.
For step lengths, we decided to proceed with two schemes - **linear** scheme

$$\eta_t = \frac{\eta}{t}$$

and **quadratic** scheme

$$\eta_t = \frac{\eta}{\sqrt{t}}$$

After numerous experiments, the combinations of - **SGD + quadratic** and **VGD + linear** were selected.
The bias term and learning rate (η) were set as **-2** and **0.0001** respectively, after much experimentation as seen from the table 1 below -

Looking at the above results, we decided to use the **Vanilla Gradient Descent** method, with **linear** step length scheme, a learning rate (η) of **0.0001**, a bias term of **-2** getting an accuracy of **77.87%**.

Table 1: Experimentation with hyperparameters

Hyperparameters				
GD Variant	Step Length	η	Bias	Accuracy
VGD	linear	0.009	10	67.07%
VGD	quadratic	0.001	0	73.56%
VGD	linear	0.001	0	77.39%
SGD	quadratic	0.0006	-20	68.23%
SGD	quadratic	0.0001	10	63.06%
VGD	linear	0.0001	-2	77.87%
VGD	quadratic	0.001	10	76.17%

5

Taking the x-axis as "time taken" and y-axis as "test classification accuracy", we get the following convergence curve (figure 1)-

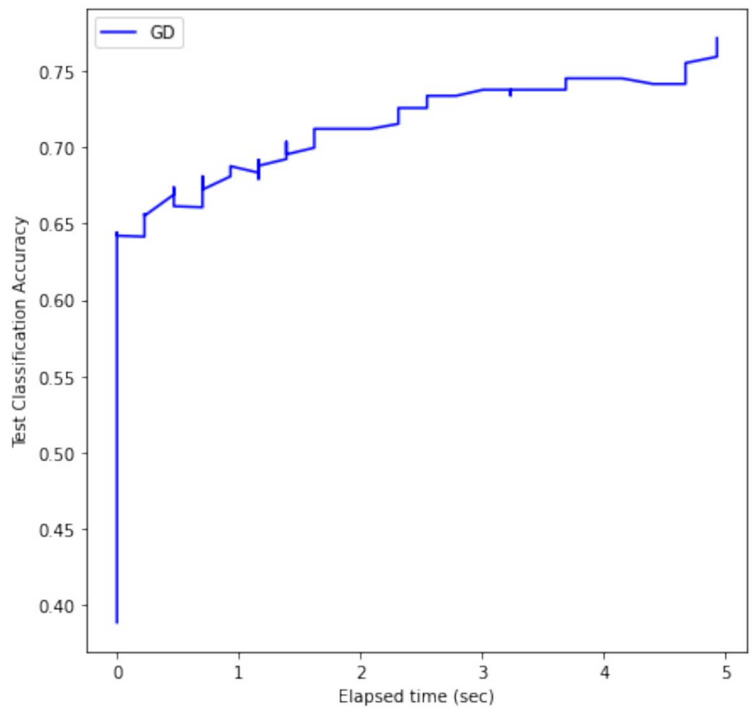

Figure 1: Convergence Curve

5 Hyperparameter description 5 / 5

✓ **+ 3 pts** *Enumeration of all hyperparameters and values considered for those hyperparameters*

✓ **+ 2 pts** *Description of how the best value was obtained for each hyperparameter*

+ 0 pts Completely wrong or else unanswered

1

2

Table 1: Experimentation with hyperparameters

Hyperparameters				
GD Variant	Step Length	η	Bias	Accuracy
VGD	linear	0.009	10	67.07%
VGD	quadratic	0.001	0	73.56%
VGD	linear	0.001	0	77.39%
SGD	quadratic	0.0006	-20	68.23%
SGD	quadratic	0.0001	10	63.06%
VGD	linear	0.0001	-2	77.87%
VGD	quadratic	0.001	10	76.17%

5

Taking the x-axis as "time taken" and y-axis as "test classification accuracy", we get the following convergence curve (figure 1)-

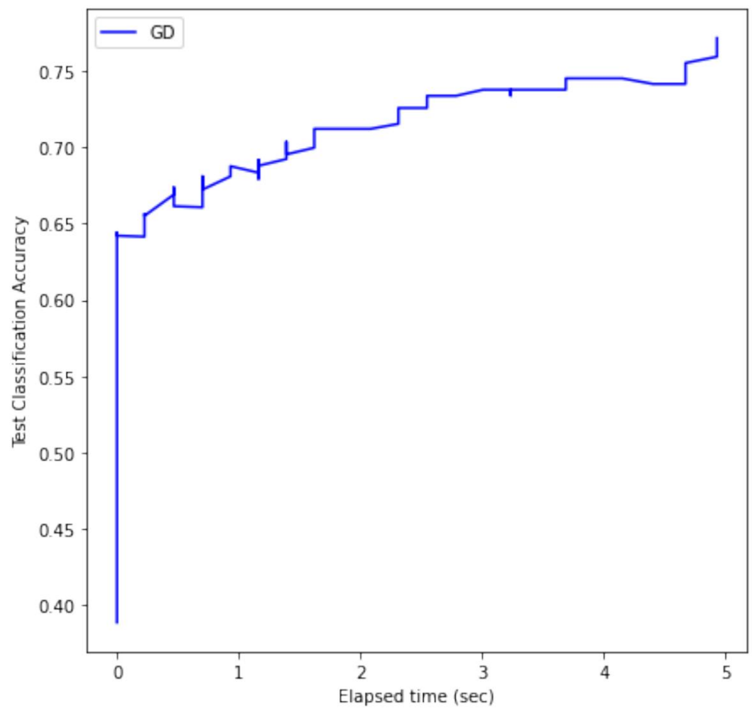

Figure 1: Convergence Curve

6 Convergence plot **5 / 5**

✓ **+ 5 pts** *A digitally generated plot is provided*

- 1 pts Minor mistakes like unlabeled x axis or y axis

+ 0 pts Completely wrong or else unanswered

+ 3 pts Not the classification accuracy graaph