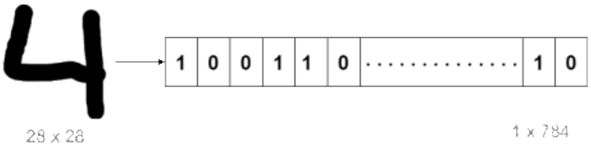
sMulticlass Tsetlin machine

19 February 2025 20:35

Part A : Structure of Tsetlin Machine
Part B : Training of Tsetlin Machine

For Multiclass TM using MNIST Dataset

10 Classes, images consist of hand written digits from 0-9, size = 28X28, train = 60K, testing = 10K.



Structure

Classes : 10	states = 10
Clauses : 1000	s = 10
Features : 1 X 784	T = 50

1) TA_STATE

Clauses	X0	X1	X2	X3	X4	X5					X782	X783
0	10 11	11 11	11 10	10 10	..	...	.	..	.	.	..	11 11
1												
2												
3												
4												
5												
.												
.												
998												
999												

2) CLAUSE_COUNT

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0

3) CLAUSE_SIGN

CLASSES |CLAUSES	0	1	2					98	99
0	INDEX, 0 SIGN, 1	1 -1	2 1						99 -1
1									
2									
3									
4									
5									
6									
7									
8									
9									

CLAUSE_COUNT

0	1	2	3	4	5	6	7	8	9
100	100	100	100	100	100	100	100	100	100

4) CLAUSE_OUTPUT

INDEX	0	1	2												998	999
OUTPUT	0	0	0												0	0

5) CLASS_SUM

CLASS	0	1	2	3	4	5	6	7	8	9
CLASS_SUM	0	0	0	0	0	0	0	0	0	0

6) FEEDBACK_TO_CLAUSES

CLAUSE	0	1	2	3				998	999
FEEDBACK	0	0	0	0				0	0

This is structure, till code line number 36

Functions in MCTM class

1) ACTION(state)

TA

Exclude space = 0	**Include space = 1**

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

2) GET_STATE(clause, feature, automaton_type)

Clauses	X0	X1	X2	X3	X4	X5		X782	X783
0	10 11	11 11	11 10	10 10	..	…	.	..	11 11
1									
2									
3									
4									
5									
.									
998									
999									

Clause 0 , ~X2 = 10

3) CALCULATE_CLAUSE_OUTPUT(X)

STEP 1) MAKE OUTPUT 1

INDEX	0	1	2												998	999
OUTPUT	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Feature Vector

X0	X1	X2	.	.				X784	X785
X0 ~X0									

ACTION INCLUDE

ACTION INCLUDE NEGATED

IF ACTION_INCLUDED == 1 & X[K] == 0 | ACTION_INCLUDED_NEGATED == 1 & X[K] == 1
CLAUSE_OUTPUT = 0

EXAMPLE 1 : CLAUSE 732 : X4 & X34 & X76 AND IN THE INPUT IF X4 = 1, X34 = 1, X76=0 THEN OUTPUT = 0

EXAMPLE 2 : CLAUSE 970 : ~X2 & ~X480 & X702 AND IN THE INPUT IF X2 = 1, X480 = 0, X702=1 THEN OUTPUT = 0

4) SUM_UP_CLASS_VOTES()

CLASS	0	1	2	3	4	5	6	7	8	9
CLASS_SUM	0	0	0	0	0	0	0	0	0	0

LET'S SAY THIS IS TARGET CLASS

CLAUSE_COUNT

0	1	2	3	4	5	6	7	8	9
100	100	100	100	100	100	100	100	100	100

CLASSES |CLAUSES	0	1	2					98	99
0	INDEX, 0 SIGN, 1	1 -1	2 1						99 -1
1									
2									
3	INDEX SIGN								
4									
5									
6									
7									
8									
9									

CLAUSE_OUTPUT

INDEX	0	1	2												998	999
OUTPUT	0	0	0						0	1					0	0

CLASS_SUM += CLAUSE_OUTPUT * CLAUSE_SIGN

IF > THRESHOLD, THEN CLASS_SUM = THRESHOLD
IF < -THRESHOLD, THEN CLASS_SUM = -THRESHOLD

UPDATE

4) PREDICT(X)

CLASS	0	1	2	3	4	5	6	7	8	9
CLASS_SUM	25	32	44	11	31	9	04	20	10	19

CLASS_SUM	25	32	44	11	31	9	04	20	10	19

INITIALIZE AS MAX CLASS SUM THEN ITERATE OVER ARRAY TO LOOK FOR MAXIMUM.

5) EVALUATE(X, Y, number_of_features)

Xi/EXAMPLE	X0	X1	X2	X3							X783
0											
1											
2											
3											
.											
.											
.											
.											
.											
.											
.											
.											
.											
.											
.											
NUMBER_OF_EXAMPLE --1											

Y/LABEL

Xi =	0	0	0	0	0	0	0	.	.	.	.	.	0	0

FILL THIS FROM INPUT VECTOR, GET PREDICTION, IF WRONG ERROR += 1

6) UPDATE(X, target_class):

! ATTENTION TO FEEDBACKS

TARGET_CLASS AND NEGATIVE/ANTI_TARGET_CLASS != TARGET CLASS

CALCULATE_CLAUSE_OUTPUT(X)

SUM_UP_CLASS_VOTES()

CLAUSE	0	1	2	3				998	999
FEEDBACK	0	0	0	0				0	0

IF RANDOM > $\frac{T - CLASS\_SUM[TARGET]}{2T}$

0 < < 1

IF RANDOM > $\frac{T + CLASS\_SUM[TARGET]}{2T}$

0 < < 1

CLAUSE	0	1	2	3	+	-	+	-	+			+	-	+	-	+		998	999
FEEDBACK	0	0	0	0	1	-1	1	-1	1			-1	1	-1	1	-1		0	0

TARGET CLASS

ANTI TARGET CLASS

FOR ALL CLAUSES WITH POSITIVE FEEDBACK IF CLAUSE_OUTPUT = 0 , --> ERASE IT

Clauses	X0	X1	X2	X3	X4	X5		X782	X783
0	10 11	11 11	11 10	10 10	..	...	.	..	11 11
1									
2									
3									
4									
5									
.									
998									
999									

CHOOSE LITERAL IF

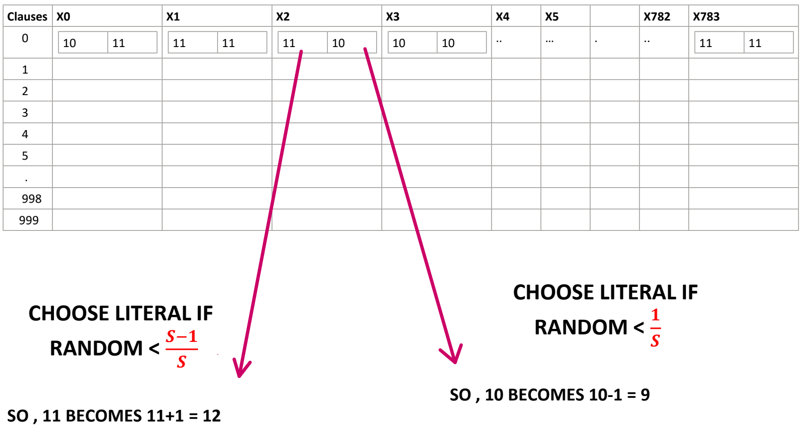
RANDOM < $\frac{1}{s}$

SO , 11 BECOMES 11-1 = 10

EFFECT OF S

FOR ALL CLAUSES WITH POSITIVE FEEDBACK IF CLAUSE_OUTPUT = 1 , -->RECOGNIZE IT

IF X[K] = 1 MEANING, INPUT VECTOR CONTAINS LITERAL IN TRUE FORM SO



Clauses	X0	X1	X2	X3	X4	X5		X782	X783
0	10 11	11 11	11 10	10 10	..	...	.	..	11 11
1									
2									
3									
4									
5									
.									
998									
999									

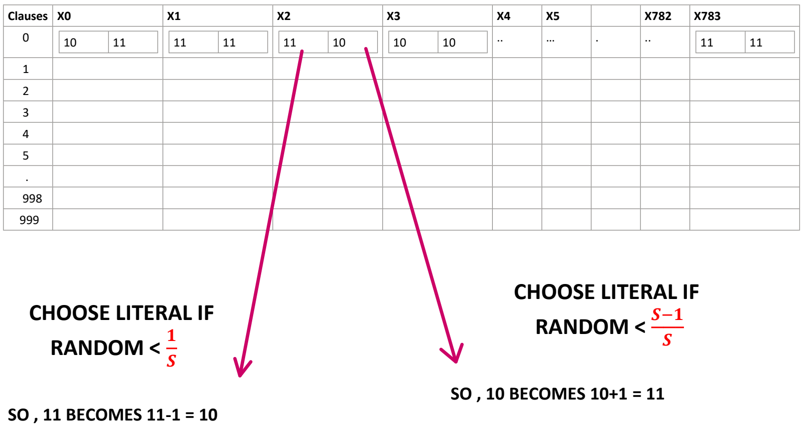
CHOOSE LITERAL IF RANDOM < $\frac{S-1}{S}$

SO , 11 BECOMES 11+1 = 12

CHOOSE LITERAL IF RANDOM < $\frac{1}{S}$

SO , 10 BECOMES 10-1 = 9

IF X[K] = 0 MEANING, INPUT VECTOR CONTAINS LITERAL IN COMPLIMENTERY FORM SO



Clauses	X0	X1	X2	X3	X4	X5		X782	X783
0	10 11	11 11	11 10	10 10	..	...	.	..	11 11
1									
2									
3									
4									
5									
.									
998									
999									

CHOOSE LITERAL IF RANDOM < $\frac{1}{S}$

SO , 11 BECOMES 11-1 = 10

CHOOSE LITERAL IF RANDOM < $\frac{S-1}{S}$

SO , 10 BECOMES 10+1 = 11

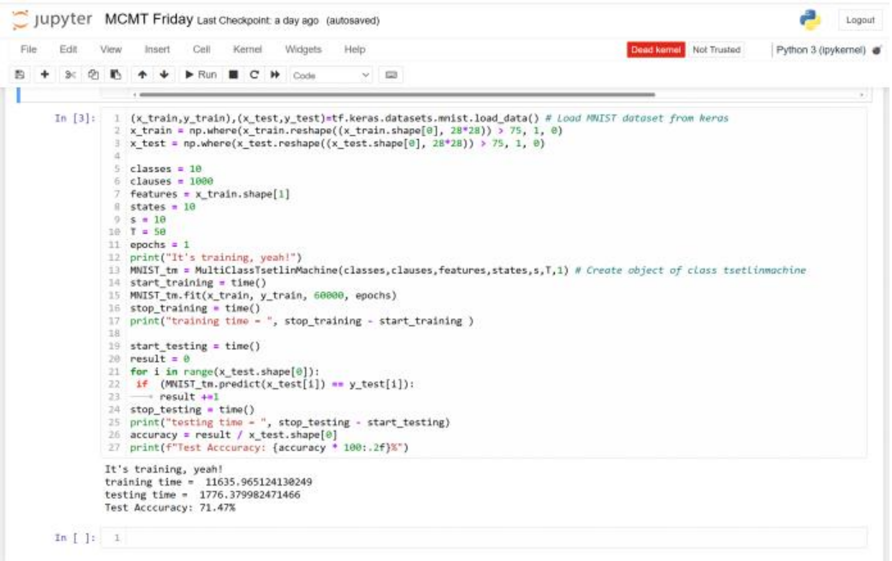
FOR ALL CLAUSES WITH NEGATIVE FEEDBACK IF CLAUSE_OUTPUT = 1 , --> MAKE IT 0

IF X[K] = 0 AND IT IS IN EXCLUDE SPACE THEN TRY TO BRING X_K TO INCLUDE SPACE SO --> CLAUSE_OUTPUT = 0

IF X[K] = 1 AND IT IS IN EXCLUDE SPACE THEN TRY TO BRING ~X_K TO INCLUDE SPACE SO --> CLAUSE_OUTPUT = 0

7) FIT(X, Y, number_of_examples, epoch)

Finally use this function to train by calling update function.



In [3]:

```
(x_train,y_train),(x_test,y_test)=tf.keras.datasets.mnist.load_data() # Load MNIST dataset from keras
x_train = np.where(x_train.reshape((x_train.shape[0], 28*28)) > 75, 1, 0)
x_test = np.where(x_test.reshape((x_test.shape[0], 28*28)) > 75, 1, 0)

classes = 10
clauses = 1000
features = x_train.shape[1]
states = 10
s = 10
T = 50
epochs = 1
print("It's training, yeah!")
MNIST_tm = MultiClassTsetlinMachine(classes,clauses,features,states,s,T,1) # Create object of class tsetlinmachine
start_training = time()
MNIST_tm.fit(x_train, y_train, 60000, epochs)
stop_training = time()
print("training time = ", stop_training - start_training )

start_testing = time()
result = 0
for i in range(x_test.shape[0]):
 if  (MNIST_tm.predict(x_test[i]) == y_test[i]):
	result +=1
stop_testing = time()
print("testing time = ", stop_testing - start_testing)
accuracy = result / x_test.shape[0]
print(f"Test Acccuracy: {accuracy * 100:.2f}%")
```

```
It's training, yeah!
training time =  11635.965124130249
testing time =  1776.379982471466
Test Acccuracy: 71.47%
```

In [ ]: