Building Concise Logical Patterns by Constraining Tsetlin Machine Clause Size

Appendix

A Tsetlin Machine

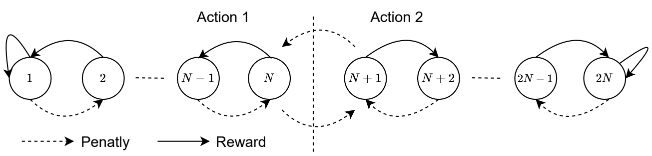


Figure 1: A two-action Tsetlin Automaton with $2N$ states.

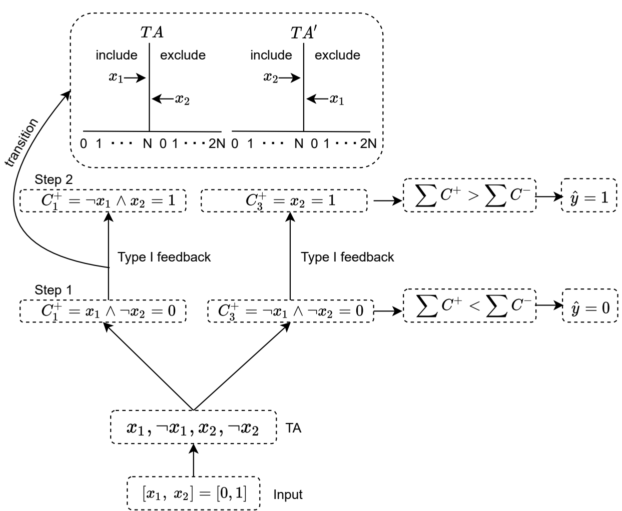


Figure 2: TM learning dynamics for an XOR-gate training sample, with input ($x_1 = 0, x_2 = 1$) and output target $y = 1$.

Structure. A TM in its simplest form takes a feature vector $\mathbf{x} = [x_1, x_2, \ldots, x_o] \in \{0,1\}^o$ of o propositional values as input and assigns the vector a class $\hat{y} \in \{0,1\}$. To minimize classification error, the TM produces n self-contained patterns. In brief, the input vector $\mathbf{x}$ provides the literal set $L = \{l_1, l_2, \ldots, l_{2o}\} = \{x_1, x_2, \ldots, x_o, \neg x_1, \neg x_2, \ldots, \neg x_o\}$, consisting of the input features and their negations. By selecting subsets $L_j \subseteq L$ of the literals, the TM can build arbitrarily complex patterns, ANDing the selected literals to form conjunctive clauses:

$$C_j(\mathbf{x}) = \bigwedge_{l_k \in L_j} l_k. \tag{1}$$

Above, $j \in \{1, 2, \ldots, n\}$ refers to a particular clause C_j and $k \in \{1, 2, \ldots, 2o\}$ refers to a particular literal l_k. As an example, the clause $C_j(\mathbf{x}) = x_1 \land \neg x_2$ consists of the literals $L_j = \{x_1, \neg x_2\}$ and evaluates to 1 when $x_1 = 1$ and $x_2 = 0$.

The TM assigns one TA per literal l_k per clause C_j to build the clauses. The TA assigned to literal l_k of clause C_j decides whether l_k is *Excluded* or *Included* in C_j. Figure 1 depicts a two-action TA with $2N$ states. For states 1 to N, the TA performs action *Exclude* (Action 1), while for states $N+1$ to $2N$ it performs action *Include* (Action 2). As feedback to the action performed, the environment responds with either a Reward or a Penalty. If the TA receives a Reward, it moves deeper into the side of the action. If it receives a Penalty, it moves towards the middle and eventually switches action.

With n clauses and $2o$ literals, we have $n \times 2o$ TAs. We organize the states of these in a $n \times 2o$ matrix $A = [a_k^j] \in \{1, 2, \ldots, 2N\}^{n \times 2o}$. We will use the function $g(\cdot)$ to map the automaton state a_k^j to Action 0 (*Exclude*) for states 1 to N and to Action 1 (*Include*) for states $N+1$ to $2N$: $g(a_k^j) = a_k^j > N$.

We can connect the states a_k^j of the TAs assigned to clause C_j with its composition as follows:

$$C_j(\mathbf{x}) = \bigwedge_{l_k \in L_j} l_k = \bigwedge_{k=1}^{2o} \left[g(a_k^j) \Rightarrow l_k \right]. \tag{2}$$

Here, l_k is one of the literals and a_k^j is the state of its TA in clause C_j. The logical *imply* operator $\Rightarrow$ implements the *Exclude*/*Include* action. That is, the *imply* operator is always 1 if $g(a_k^j) = 0$ (*Exclude*), while if $g(a_k^j) = 1$ (*Include*) the truth value is decided by the truth value of the literal.

Classification. Classification is performed as a majority vote. The clause outputs are combined into a classification decision through summation and thresholding using the unit step function $u(v) = 1$ **if** $v \geq 0$ **else** 0:

$$\hat{y} = u\left(\sum_{j=1}^{n/2} C_j^+(\mathbf{X}) - \sum_{j=1}^{n/2} C_j^-(\mathbf{X}) \right). \tag{3}$$

As an example, consider the input vector $\mathbf{x} = [0, 1]$ in the lower part of Figure 2. The figure depicts two clauses of positive polarity, $C_1(\mathbf{x}) = x_1 \land \neg x_2$ and $C_3(\mathbf{x}) = \neg x_1 \land \neg x_2$ (the negative polarity clauses are not shown). Both of the clauses evaluate to zero, leading to class prediction $\hat{y} = 0$.

	Value of the clause $C^i_j(\mathbf{X})$	1		0	
	Value of the literal $x_k/\neg x_k$	1	0	1	0
TA: **Include Literal**	$P(\text{Reward})$	$\frac{s-1}{s}$	NA	0	0
	$P(\text{Inaction})$	$\frac{1}{s}$	NA	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	$P(\text{Penalty})$	0	NA	$\frac{1}{s}$	$\frac{1}{s}$
TA: **Exclude Literal**	$P(\text{Reward})$	0	$\frac{1}{s}$	$\frac{1}{s}$	$\frac{1}{s}$
	$P(\text{Inaction})$	$\frac{1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$	$\frac{s-1}{s}$
	$P(\text{Penalty})$	$\frac{s-1}{s}$	0	0	0

Table 1: Type I Feedback for vanilla TM — Feedback upon receiving a sample with label $y = 1$, for a single TA to decide whether to Include or Exclude a given literal $x_k/\neg x_k$ into C^i_j. NA means not applicable.

	Value of the clause $C^i_j(\mathbf{X})$	1		0	
	Value of the literal $x_k/\neg x_k$	1	0	1	0
TA: **Include Literal**	$P(\text{Reward})$	0	NA	0	0
	$P(\text{Inaction})$	1.0	NA	1.0	1.0
	$P(\text{Penalty})$	0	NA	0	0
TA: **Exclude Literal**	$P(\text{Reward})$	0	0	0	0
	$P(\text{Inaction})$	1.0	0	1.0	1.0
	$P(\text{Penalty})$	0	1.0	0	0

Table 2: Type II Feedback — Feedback upon receiving a sample with label $y = 0$, for a single TA to decide whether to Include or Exclude a given literal $x_k/\neg x_k$ into C^i_j. NA means not applicable.

Learning. The upper part of Figure 2 illustrates learning. A TM learns online, processing one training example $(\mathbf{x}, y)$ at a time. Based on $(\mathbf{x}, y)$, the TM rewards and penalizes its TAs, which amounts to incrementing and decrementing their states. There are two kinds of feedback: Type I Feedback produces frequent patterns and Type II Feedback increases the discrimination power of the patterns.

Type I feedback is given stochastically to clauses with positive polarity when $y = 1$ and to clauses with negative polarity when $y = 0$. Conversely, Type II Feedback is given stochastically to clauses with positive polarity when $y = 0$ and to clauses with negative polarity when $y = 1$. The probability of a clause being updated is based on the vote sum v: $v = \sum_{j=1,3,\ldots}^{n-1} \bigwedge_{k=1}^{2o} \left[g(a_k^j) \Rightarrow l_k\right] - \sum_{j=2,4,\ldots}^{n} \bigwedge_{k=1}^{2o} \left[g(a_k^j) \Rightarrow l_k\right]$. The voting error is calculated as:

$$\epsilon = \begin{cases} T - v & y = 1, \\ T + v & y = 0. \end{cases} \tag{4}$$

Here, T is a user-configurable voting margin yielding an ensemble effect. The probability of updating each clause is $P(\text{Feedback}) = \frac{\epsilon}{2T}$.

After random sampling from $P(\text{Feedback})$ has decided which clauses to update, the following TA state updates can be formulated as matrix additions, subdividing Type I Feedback into feedback Type Ia and Type Ib:

$$A^*_{t+1} = A_t + F^{II} + F^{Ia} - F^{Ib}. \tag{5}$$

Here, $A_t = [a_k^j] \in \{1, 2, \ldots, 2N\}^{n \times 2o}$ contains the states of the TAs at time step t and A^*_{t+1} contains the updated state for time step $t + 1$ (before clipping). The matrices $F^{Ia} \in \{0, 1\}^{n \times 2o}$ and $F^{Ib} \in \{0, 1\}^{n \times 2o}$ contain Type I Feedback. A zero-element means no feedback and a one-element means feedback. As shown in Table 1, two rules govern Type I feedback:

- **Type Ia Feedback** is given with probability $\frac{s-1}{s}$ whenever both clause and literal are 1-valued.[1] It penalizes *Exclude* actions and rewards *Include* actions. The purpose is to remember and refine the patterns manifested in the current input $\mathbf{x}$. This is achieved by increasing selected TA states. The user-configurable parameter s controls pattern frequency, i.e., a higher s produces less frequent patterns.
- **Type Ib Feedback** is given with probability $\frac{1}{s}$ whenever either clause or literal is 0-valued. This feedback rewards *Exclude* actions and penalizes *Include* actions to coarsen patterns, combating overfitting. Thus, the selected TA states are decreased.

The matrix $F^{II} \in \{0, 1\}^{n \times 2o}$ contains Type II Feedback to the TAs, given per Table 2.

- **Type II Feedback** penalizes *Exclude* actions to make the clauses more discriminative, combating false positives. That is, if the literal is 0-valued and the clause is 1-valued, TA states below $N + 1$ are increased. Eventually the clause becomes 0-valued for that particular input, upon inclusion of the 0-valued literal.

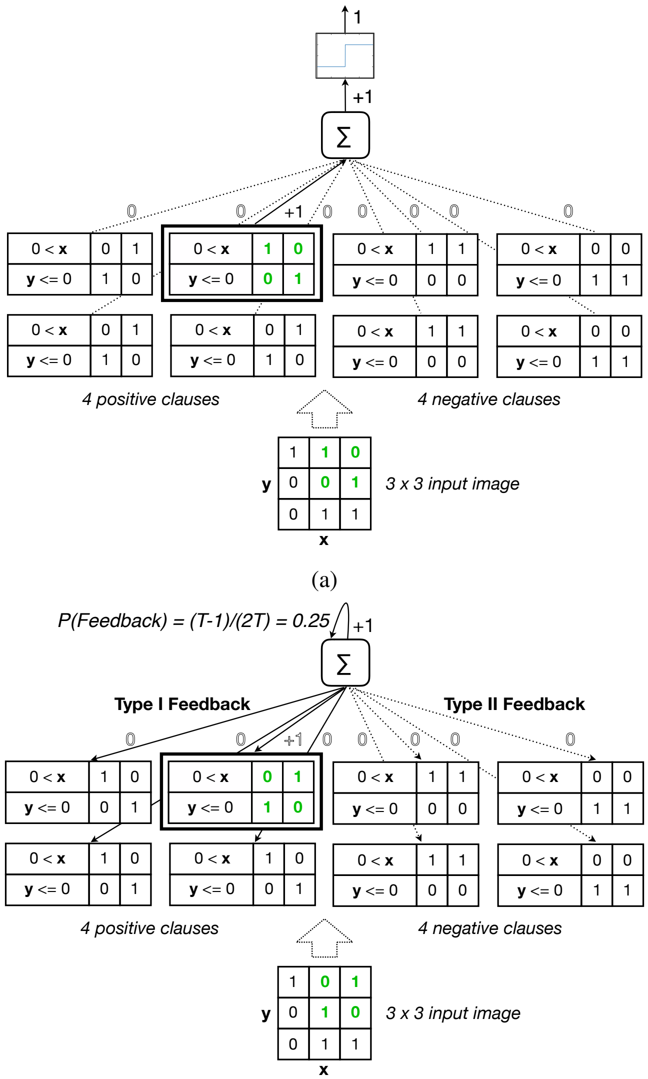
The final updating step for training example $(\mathbf{x}, y)$ is to clip the state values to make sure that they stay within value 1 and $2N$:

$$A_{t+1} = clip\left(A^*_{t+1}, 1, 2N\right). \tag{6}$$

For example, both of the clauses in Figure 2 receives Type I Feedback over several training examples, making them resemble the input associated with $y = 1$.

Let us consider a sample of XOR gate $(x_1 = 0, x_2 = 1) = 1$ to visualize the learning process as shown in Fig. 2. There are n clauses required to learn the XOR pattern and here let us consider $n = 4$ per class. Among 4 clauses, the clauses C_1 and C_3 votes for the presence $y = 1$ and C_0 and C_2 votes against it. For simplification, let us only consider how C_1 and C_3 learns the pattern for the given sample of XOR gate. At step 1, the clauses has not learnt the pattern for given sample, which leads to wrong prediction of class thereby triggering Type I feedback for corresponding literals. From Table 1 for literal x_1, if the clause score is 0 and literal is 0, it receives Inaction or Penalty for being included with the probability of $\frac{s-1}{s}$ and $\frac{1}{s}$ respectively. After several penalty, it changes

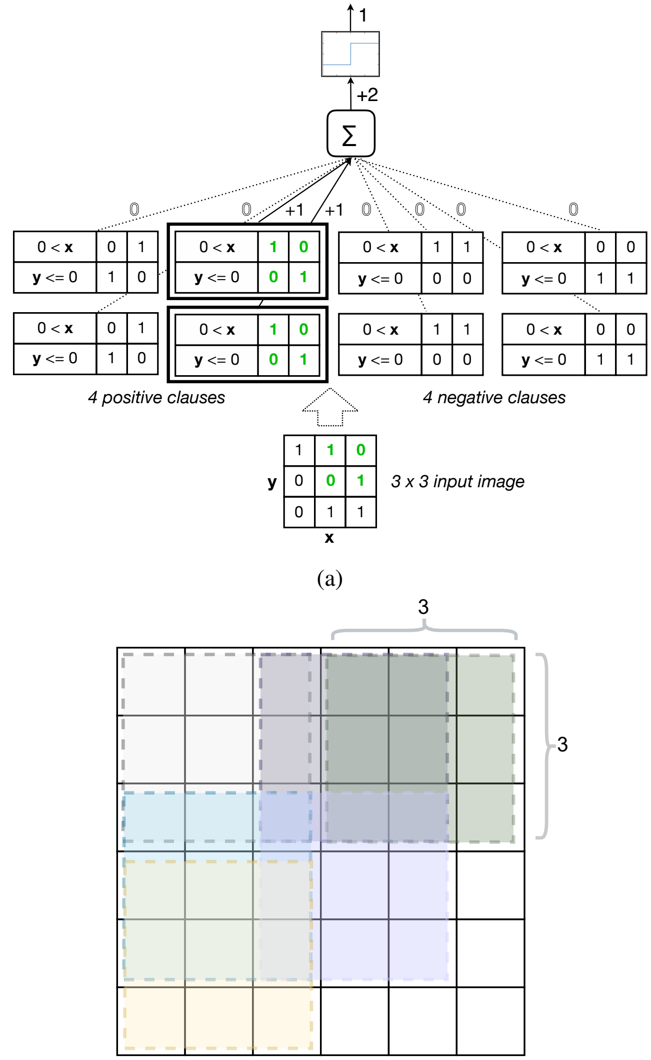
[1] Note that the probability $\frac{s-1}{s}$ is replaced by 1 when boosting true positives.



(a)

(b)

Figure 3: Example of inference (a) and learning (b) for the Noisy 2D XOR Problem.



(a)

(b)

Figure 4: (a) Goal state for the Noisy 2D XOR Problem. (b) Illustration of image, filter and patches.

its state to exclude action and gets removed from the clause C_1. On the other hand, the literal $\neg x_1$ gets penalty for being excluded and eventually jumps to include section as shown in C_1 at step 2. Similarly, when literal $\neg x_2 = 0$ and $C_1 = 0$, it receives Inaction or Penalty for being included with the probability of $\frac{s-1}{s}$ and $\frac{1}{s}$ respectively. After several penalties, $\neg x_2$ gets excluded and x_2 becomes included as shown in step 2. This indeed reaches intended pattern thereby making the clauses $C_1 = 1$ and $C_3 = 1$, and finally results in $\hat{y} = 1$.

Resource allocation ensures that clauses distribute themselves across the frequent patterns, rather than missing some and over-concentrating on others. That is, for any input X, the probability of reinforcing a clause gradually drops to zero as the clause output sum

$$v = \sum_{j=1}^{n/2} C_j^+(X) - \sum_{j=1}^{n/2} C_j^-(X) \tag{7}$$

approaches a user-configured target T for $y = 1$ (and $-T$ for $y = 0$). If a clause is not reinforced, it does not give feedback to its Tsetlin automata (TAs), and these are thus left unchanged. In the extreme, when the voting sum v equals or exceeds the target T (the TM has successfully recognized the input X), no clauses are reinforced. They are then free to learn new patterns, naturally balancing the pattern representation resources [Granmo, 2018].

Weighted Tsetlin Machine

The learning of weights is based on increasing the weights of clauses that receive Type Ia feedback (due to true positive output) and decreasing the weight of clauses that receive Type II feedback (due to false positive output). The overall rationale is to determine which clauses are inaccurate and thus must team up to obtain high accuracy as a team (low weight clauses), and which clauses are sufficiently accurate to operate more independently (high weight clauses). The weight updating procedure is summarized in Algorithm 1. Here, w_i is the weight of clause C_i at the n^{th} training round (ignoring polarity to simplify notation). The first step of a training round is to calculate the clause output. The weight of a clause is only updated if the clause output C_i is 1 and the clause has been selected for feedback (P_i = 1). Then the polarity of the clause and the class label y decide the type of feedback given. That is, like a regular TM, positive polarity clauses receive Type Ia feedback if the clause output is a true positive and Type II feedback if the clause output is a false positive. For clauses with negative polarity, the feedback types switch roles. When clauses receive Type Ia or Type II feedback, their weights are updated accordingly. We use the stochastic searching on the line (SSL) automaton to learn appropriate weights. SSL is an optimization scheme for unknown stochastic environments pioneered by Oommen [Oommen, 1997]. The goal is to find an unknown location λ^* within a search interval $[0, 1]$. In order to find λ^*, the only available information for the Learning Mechanism (LM) is the possibly faulty feedback from its attached environment E.

In SSL, the search space λ is discretized into N points, $\{0, 1/N, 2/N, ..., (N-1)/N, 1\}$ with N being the discretization resolution. During the search, the LM has a location $\lambda \in \{0, 1/N, 2/N, ..., (N-1)/N, 1\}$, and can freely move to the left or to the right from its current location. The environment E provides two types of feedback: $E = 1$ is the environment suggestion to increase the value of λ by one step, and $E = 0$ is the environment suggestion to decrease the value of λ by one step. The next location of λ, i.e., λ_{n+1}, can thus be expressed as follows:

$$\lambda_{n+1} = \begin{cases} \lambda_n + 1/N, & \text{if } E_n = 1, \\ \lambda_n - 1/N, & \text{if } E_n = 0. \end{cases} \tag{8}$$

$$\lambda_{n+1} = \begin{cases} \lambda_n, & \text{if } \lambda_n = 1 \text{ and } E_n = 1, \\ \lambda_n, & \text{if } \lambda_n = 0 \text{ and } E_n = 0. \end{cases} \tag{9}$$

Asymptotically, the learning mechanics is able to find a value arbitrarily close to λ^* when $N \to \infty$ and $n \to \infty$. In our case, the search space of clause weights is $[0, \infty]$, so we use resolution $N = 1$, with no upper bound for λ. Accordingly, we operate with integer weights. As in algorithm 1, if the clause output is a true positive, we simply increase the weight by 1. Conversely, if the clause output is a false positive, we decrease the weight by 1.

By following the above procedure, the goal is to make low precision clauses team up by giving them low weights, so that they together can reach the summation target T. By teaming up, precision increases due to the resulting ensemble effect. Clauses with high precision, however, gets a higher weight, allowing them to operate more independently.

Algorithm 1 Complete WTM learning process.

Input: Training data batch (B, x, y) ▷ $B \geq 1$
Initialize: Random initialization of TAs
Begin: n^{th} training round
for $i = 1, ..., m$ **do if** $p_i = 1$
  if ($y = 1$ **and** i is odd) **or** ($y = 0$ **and** i is even) **then**
    if $c_i = 1$ **then**
      $w_i \leftarrow w_i + 1$
      for feature $k = 1, ..., 2o$ **do**
        if $l_k = 1$ **then**
          Type Ia Feedback
        else:
          Type Ib Feedback
        end if
      end for
    else:
      $w_i \leftarrow w_i$ ▷ [No Change]
      Type Ib Feedback
    end if
  else: ($y = 1$ **and** i is even) **or** ($y = 0$ **and** i is odd)
    if $c_i = 1$ **then**
      if $w_i > 0$ **then**
        $w_i \leftarrow w_i - 1$
      end if
      for feature $k = 1, ..., 2o$ **do**
        if $l_k = 0$ **then**
          Type II Feedback
        else:
          Inaction
        end if
      end for
    else:
      $w_i \leftarrow w_i$ ▷ [No Change]
      Inaction
    end if
  end if
end for

The above weighting scheme has several advantages. First of all, increment and decrement operations on integers are computationally less costly than multiplication based updates of real-valued weights. Additionally, a clause with an integer weight can be seen as multiple copies of the same clause, making it more interpretable than real-valued weighting, as studied in the next section. Additionally, clauses can be turned completely off by setting their weights to 0 if they do not contribute positively to the classification task.

Convolutional Tsetlin Machine

Consider a set of images $\mathcal{X} = \{\mathbf{x}_e | 1 \leq e \leq E\}$, where e is the index of the images. Each image is of size $d_x \times d_y$ and consists of d_z binary layers, illustrated in Figure 4b. A vanilla TM models such an image with an input vector $\mathbf{x} = [x_k] \in \{0, 1\}^{d_x \times d_y \times d_z}$ that contains $d_x \times d_y \times d_z$ input features. Accordingly, each clause is composed from $d_x \times d_y \times d_z \times 2$ literals.

Structure. The CTM (CTM) [Granmo *et al.*, 2019] performs a convolution over the input image $\mathbf{x}$, dividing it into patches with spatial dimensions $d_w \times d_w$. That is, the input vector $\mathbf{x} = [x_k] \in \{0, 1\}^{d_x \times d_y \times d_z}$ produces $B = \left(\left\lceil \frac{d_x - d_w}{q} \right\rceil + 1\right) \times \left(\left\lceil \frac{d_y - d_w}{q} \right\rceil + 1\right)$ patches, with q being the step size of the convolution. For instance, Figure 4b illustrates $B = (6 - 3 + 1) \times (6 - 3 + 1) = 16$ patches of size 3×3, assuming step size $q = 1$.

Each patch $b \in \{1, 2, \ldots, B\}$, in turn, yields an input vector $\mathbf{x}^b = [x^b_k] \in \{0,1\}^{d_w \times d_w \times d_z}$ with a corresponding literal vector $\mathbf{l}^b = [l^b_k] \in \{0,1\}^{d_w \times d_w \times d_z \times 2}$. The CTM becomes location aware by augmenting each patch input vector $\mathbf{x}^b$ with the coordinates of $\mathbf{x}^b$ within $\mathbf{x}$, using threshold-based encoding.

Classification. The CTM is based on the classic TM procedure for classification. However, we now have B input vectors $\mathbf{x}^b$ per image rather than a single input vector $\mathbf{x}$. The convolution is performed by evaluating each clause C_j on each input vector $\mathbf{x}^b$, i.e., calculating $\bigwedge_{k=1}^{2o} \left[g(a^j_k) \Rightarrow l^b_k\right]$, and then ORing the evaluations per clause:

$$\hat{y} = 0 \leq \sum_{j=1,3,\ldots}^{n-1} \bigvee_{b=1}^{B} \left[\bigwedge_{k=1}^{2o} \left[g(a^j_k) \Rightarrow l^b_k\right]\right] - \sum_{j=2,4,\ldots}^{n} \bigvee_{b=1}^{B} \left[\bigwedge_{k=1}^{2o} \left[g(a^j_k) \Rightarrow l^b_k\right]\right]. \quad (10)$$

Figure 3a provides an example where a 3×3 input image produces four 2×2 patches. The CTM has four clauses of positive polarity and four clauses of negative polarity. Only one of the clauses of positive polarity matches. This clause matches the upper left corner of the input image, hence evaluating to 1. Accordingly, the net output sum is $+1$, yielding output $\hat{y} = 1$.

Learning. CTM learning leverages the TM learning procedure, per Eq. (5) and Eq. (6). However, when giving Type Ia or Type II Feedback to each clause C_j, the CTM does not use the original input vector $\mathbf{x}$. Instead, it randomly selects one of the patch input vectors $\mathbf{x}^b$ that made the clause evaluate to 1:

$$\mathbf{x}^b_j = RandomChoice(\{\mathbf{x}^b | \bigwedge_{k=1}^{2o} [g(a^j_k) \Rightarrow l^b_k] = 1, 1 \leq b \leq B.\}). \quad (11)$$

For Type Ib Feedback, on the other hand, CTM follows the standard updating scheme.

The reason for randomly selecting a patch input vector $\mathbf{x}^b$ is to have each clause extract a certain sub-pattern, and the randomness of the uniform distribution statistically spreads the clauses for different sub-patterns in the target image.

Figure 3b demonstrates a learning step. Only a single clause has recognized the input. Assuming a summation target (margin) of $T = 2$ and net clause output sum $+1$ the probability of giving each clause feedback becomes $P(Feedback) = \frac{(2-1)}{2 \cdot 2} = 0.25$. Since the training example is $y = 1$, the positive polarity clauses receives Type I Feedback with probability 0.25, while the negative polarity clauses receive Type II feedback again with probability 0.25. After several such updates, we have a more balanced representation of the input patterns in Figure 4a, with two clauses now recognizing the input.

x_1	x_2	Output
0	0	0
1	1	0
0	1	1

Table 3: A sub-pattern in "XOR" case.

B Detailed transition of a XOR sub-pattern given the clause size constraint

Here we detail the convergence of the XOR operator when only one literal as budget is given, i.e., $\|C^i_j(\mathbf{X})\| = \mathbf{1}$. Specifically, we study the transitions of TAs for the sub-pattern shown in Table 3. Compared with the analysis in [Jiao *et al.*, 2023], the changes due to the new constraint are highlighted in red.

For simplicity, we ignore the class index i in C^i_j because we study only one class, i.e., the XOR operator. Without loss of generality, we look at clause C_3. C_3 has in total 4 TA, i.e., TA^3_1 with actions "Include x_1" or "Exclude x_1", TA^3_2 with actions "Include $\neg x_1$" or "Exclude $\neg x_1$", TA^3_3 with actions "Include x_2" or "Exclude x_2", and TA^3_4 with actions "Include $\neg x_2$" or "Exclude $\neg x_2$". To analyze the convergence of those four TAs, we perform a quasi-stationary analysis, where we freeze the behavior of three of them, and then study the transitions of the remaining one. More specifically, the analysis is organized as follows:

1. We freeze TA^3_1 and TA^3_2 respectively at "Exclude" and "Include". In this case, the first bit becomes $\neg x_1$. There are four sub-cases for TA^3_3 and TA^3_4:
   (a) We study the transition of TA^3_3 when it has the action "Include" as its current action, given different training samples shown in Table 3 and different actions of TA^3_4 (i.e., when the action of TA^3_4 is frozen at "Include" or "Exclude").
   (b) We study the transition of TA^3_3 when it has "Exclude" as its current action, given different training samples shown in Table 3 and different actions of TA^3_4 (i.e., when the action of TA^3_4 is frozen at "Include" or "Exclude").
   (c) We study the transition of TA^3_4 when it has "Include" as its current action, given different training samples shown in Table 3 and different actions of TA^3_3 (i.e., when the action of TA^3_3 is frozen at "Include" or "Exclude").
   (d) We study the transition of TA^3_4 when it has "Exclude" as its current action, given different training samples shown in Table 3 and different actions of TA^3_3 (i.e., when the action of TA^3_3 is frozen as "Include" or "Exclude").
2. We freeze TA^3_1 and TA^3_2 respectively at "Include" and "Exclude". In this case, the first bit becomes x_1. The sub-cases for TA^3_3 and TA^3_4 are identical to the sub-cases in the previous case.
3. We freeze TA^3_1 and TA^3_2 at "Exclude" and "Exclude". In this case, the first bit is excluded and will not influence

the final output. The sub-cases for TA_3^3 and TA_4^3 are identical to the sub-cases in the previous case.

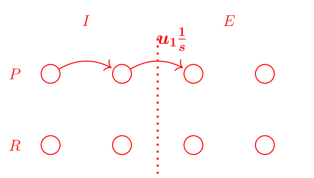
4. We freeze TA_1^3 and TA_2^3 at "Include" and "Include". In this case, we always have $C_3 = 0$ because the clause contains the contradiction $x_1 \wedge \neg x_1$. The sub-cases for TA_3^3 and TA_4^3 are identical to the sub-cases in the previous case.

In the analysis below, we will study each of the four cases, one by one.

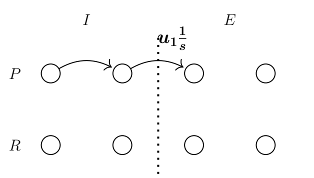
Case 1

In this case, the first bit is in the form of $\neg x_1$ always. We now analyze the first sub-case, i.e., Sub-case 1 (a). We here study the transition of TA_3^3 when its current action is "Include". Depending on different training samples and actions of TA_4^3, we have the following possible transitions. Below, "I" and "E" mean "Include" and "Exclude", respectively. For sake of conciseness, we remove the instances where no transition happens.

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=E. Therefore, we have Type I feedback for literal $x_2 = 1$, $C_3 = \neg x_1 \wedge x_2 \wedge 0 = 0$.

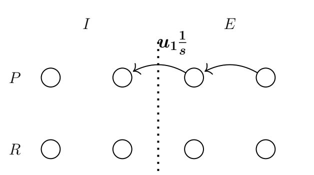


Condition: $x_1 = 0$, $x_2 = 1, y = 1$, TA_4^3=I. Therefore, we have Type I feedback for literal $x_2 = 1, C_3 = 0$.

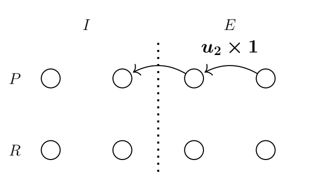


We now consider Sub-case 1 (b). The literal $\neg x_1$ is still included, and we study the transition of TA_3^3 when its current action is "Exclude". The possible transitions are listed below.

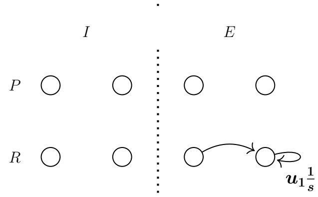
Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=E. Therefore, Type I, $x_2 = 1$, $C_3 = \neg x_1 = 1$.



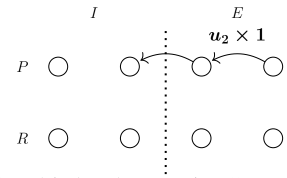
Condition: $x_1 = 0$, $x_2 = 0$, $y = 0$, TA_4^3=E. Therefore, Type II, $x_2 = 0$, $C_3 = \neg x_1 = 1$.



Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=I. Therefore, Type I, $x_2 = 1$, $C_3 = \neg x_1 \wedge \neg x_2 = 0$.

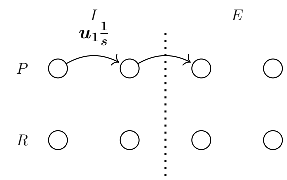


Condition: $x_1 = 0$, $x_2 = 0$, $y = 0$, TA_4^3=I. Therefore, Type II, $x_2 = 0$, $C_3 = \neg x_1 \wedge \neg x_2 = 1$.

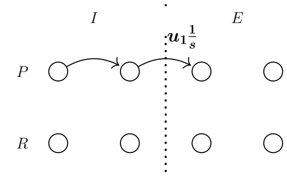


Now let us move onto the third sub-case in Case 1, i.e., Sub-case 1 (c). The literal $\neg x_1$ is still included, and we study the transition of TA_4^3 when its current action is "Include". Note that we are now studying TA_4^3 that corresponds to $\neg x_2$ rather than x_2. Therefore, the literal in Tables 1 and 2 becomes $\neg x_2$.

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=E. Therefore, Type I, $\neg x_2 = 0$, $C_3 = \neg x_1 \wedge \neg x_2 = 0$.

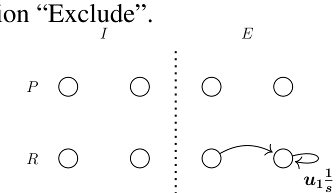


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=I. Therefore, Type I, $\neg x_2 = 0, C_3 = 0$.

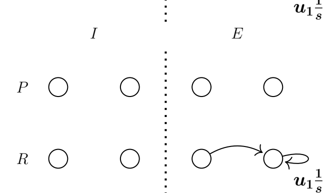


For the Sub-case 1 (d), we study the transition of TA_4^3 when it has the current action "Exclude".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=E. Therefore, Type I, $\neg x_2 = 0$, $C_3 = \neg x_1 = 1$.



Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=I. Therefore, Type I, $\neg x_2 = 0$, $C_3 = \neg x_1 \wedge x_2 \wedge 0 = 0$.



So far, we have gone through all sub-cases in Case 1.

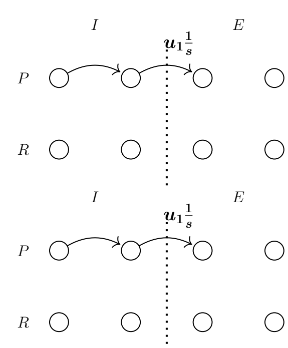
Case 2

Case 2 studies the behavior of TA_3^3 and TA_4^3 when TA_1^3 and TA_2^3 select "Include" and "Exclude", respectively. In this case, the first bit is in the form of x_1 always. There are here also four sub-cases and we will detail them presently.

We first study TA_3^3 with action "Include", providing the below transitions.

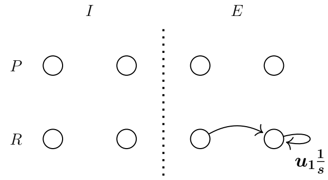
Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=E. Therefore, Type I, $x_2 = 1, C_3 = 0$.

Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=I. Therefore, Type I, $x_2 = 1, C_3 = 0$.

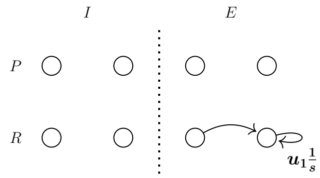


We then study TA_3^3 with action "Exclude", and transitions are shown below.

Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=E.
Therefore, Type I, $x_2 = 1, C_3 = 0$.

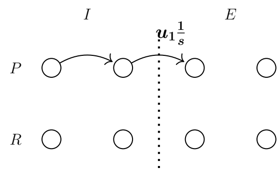


Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=I.
Therefore, Type I, $x_2 = 1, C_3 = 0$.

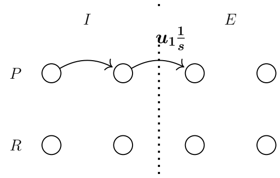


We now study TA_4^3 with action "Include" and the transitions are presented below.

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=E.
Therefore, Type I, $\neg x_2 = 0$, $C_3 = x_1 \wedge \neg x_2 = 0$.

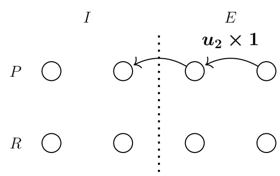


Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=I.
Therefore, Type I, $\neg x_2 = 0$, $C_3 = 0$.

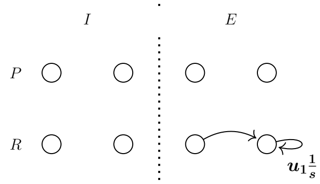


We study lastly TA_4^3 with action "Exclude", leading to the following transitions.

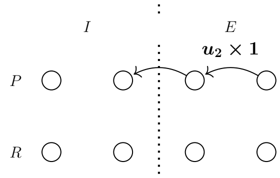
Conditions: $x_1 = 1$, $x_2 = 1$, $y = 0$, TA_3^3=E.
Therefore, Type II, $\neg x_2 = 0$, $C_3 = x_1 = 1$.



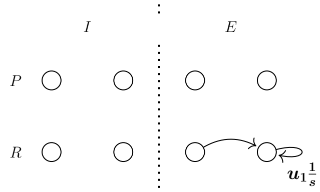
Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=E.
Therefore, Type I, $\neg x_2 = 0, C_3 = 0$.



Conditions: $x_1 = 1$, $x_2 = 1$, $y = 0$, TA_3^3=I.
Therefore, Type II, $\neg x_2 = 0$, $C_3 = x_1 \wedge x_2 = 1$.



Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=I.
Therefore, Type I, $\neg x_2 = 0$, $C_3 = x_1 \wedge x_2 = 0$.

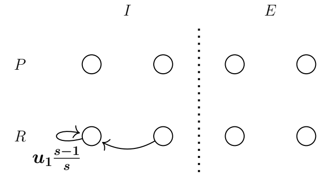


Case 3

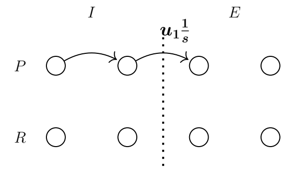
Now we move onto Case 3, where TA_1^3 and TA_2^3 both select "Exclude". We study the behavior of TA_3^3 and TA_4^3 for different sub-cases. In this case, the first bit x_1 does not play any role for the output.

We first examine TA_3^3 with action "Include", providing the transitions below.

Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=E.
Therefore, Type I, $x_2 = 1, C_3 = x_2 = 1$.

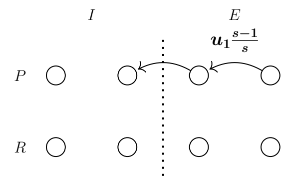


Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=I.
Therefore, Type I, $x_2 = 1, C_3 = 0$.

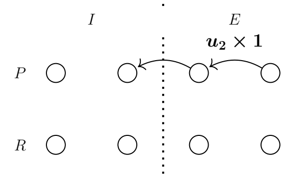


We then study TA_3^3 with action "Exclude", transitions shown below. In this situation, if TA_4^3 is also excluded, C_3 is "empty" since all its associated TA select action "Exclude". To make the training proceed, according to the training rule of TM, we assign $C_3 = 1$ in this situation.

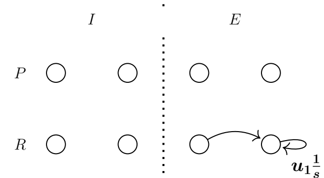
Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=E.
Therefore, Type I, $x_2 = 1, C_3 = 1$.



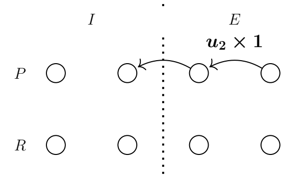
Condition: $x_1 = 0$, $x_2 = 0$, $y = 0$, TA_4^3=E.
Therefore, Type II, $x_2 = 0, C_3 = 1$.



Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=I.
Therefore, Type I, $x_2 = 1$, $C_3 = \neg x_2 = 0$.

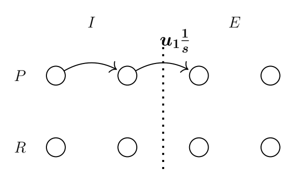


Condition: $x_1 = 0$, $x_2 = 0$, $y = 0$, TA_4^3=I.
Therefore, Type II, $x_2 = 0$, $C_3 = \neg x_2 = 1$.

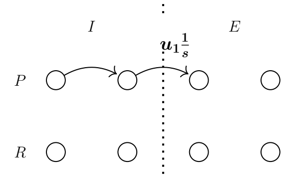


We thirdly study TA_4^3 with action "Include", covering the transitions shown below.

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$ TA_3^3=E.
Therefore, Type I, $\neg x_2 = 0, C_3 = 0$.



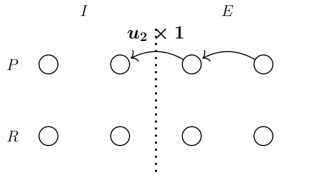
Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=I.
Therefore, Type I, $\neg x_2 = 0, C_3 = 0$.



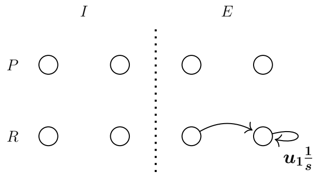
Lastly, we study TA_4^3 with action "Exclude", transitions shown below. Similarly, in this situation, when TA_3^3 is also excluded, C_3 becomes "empty" again, as all its associated TAs select action "Exclude". Following the training rule of

TM, we assign $C_3 = 1$.

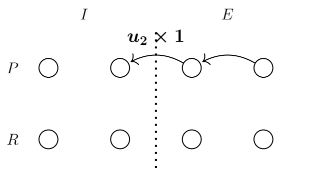
Conditions: $x_1 = 1$, $x_2 = 1$, $y = 0$, TA_3^3=E.
Therefore, Type II, $\neg x_2 = 0, C_3 = 1$.



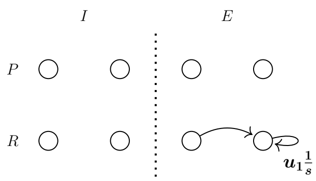
Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=E.
Therefore, Type I, $\neg x_2 = 0, C_3 = 1$.



Conditions: $x_1 = 1$, $x_2 = 1$, $y = 0$, TA_3^3=I.
Therefore, Type II, $\neg x_2 = 0, C_3 = 1$.



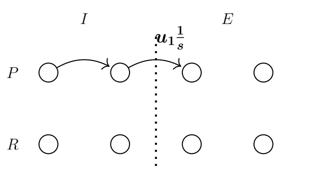
Conditions: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=I.
Therefore, Type I, $\neg x_2 = 0, C_3 = 1$.



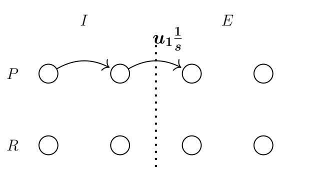
Case 4

Now, we study Case 4, where $\neg x_1$ and x_1 both select "Include". For this reason, in this case, we always have $C_3 = 0$. We study firstly TA_3^3 with action "Include" and the transitions are shown below.

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=E.
Therefore, Type I, $x_2 = 1, C_3 = 0$.

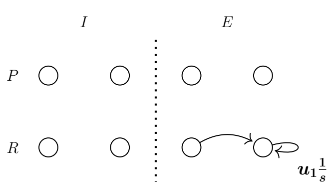


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=I.
Therefore, Type I, $x_2 = 1, C_3 = 0$.

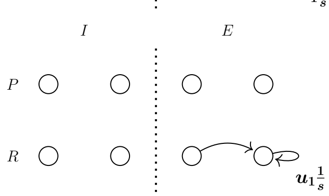


We secondly study TA_3^3 with action "Exclude".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=E.
Therefore, Type I, $x_2 = 1, C_3 = 0$.

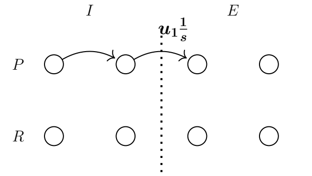


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_4^3=I.
Therefore, Type I, $x_2 = 1, C_3 = 0$.

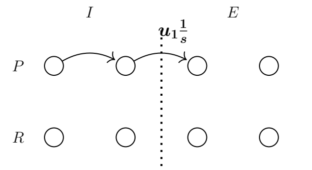


Now, we study TA_4^3 with action "Include".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=E.
Therefore, Type I, $\neg x_2 = 0, C_3 = 0$.

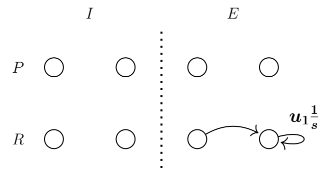


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=I.
Therefore, Type I, $\neg x_2 = 0, C_3 = 0$.

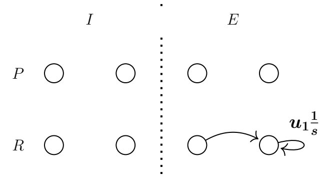


We lastly study TA_4^3 with action "Exclude".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=E.
Therefore, Type I, $\neg x_2 = 0, C_3 = 0$.



Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_3^3=I.
Therefore, Type I, $\neg x_2 = 0, C_3 = 0$.



Based on the above analyses, we can now summarize the transitions of TA_3^3 and TA_4^3, given different configurations of TA_1^3 and TA_2^3 in Case 1 – Case 4 (i.e., given four different combinations of x_1 and $\neg x_1$). The arrow shown below means the direction of transitions.

Scenario 1: Study TA_3^3 = I and TA_4^3 = I.

Case 1: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ E

Case 2: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ E

Case 3: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ E

Case 4: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ E

From the facts presented above, it is confirmed that regardless of the state of TA_1^3 and TA_2^3, if TA_3^3=I and TA_4^3=I, they (TA_3^3 and TA_4^3) will move towards the opposite half of the state space (i.e., towards "Exclude"), away from the current state. So, the state with TA_3^3=I and TA_4^3=I is not absorbing.

Scenario 2: Study TA_3^3 = I and TA_4^3= E.

Case 1: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ E

Case 2: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ I, E

Case 3: we can see that
$\mathrm{TA}_3^3 \rightarrow$ I
$\mathrm{TA}_4^3 \rightarrow$ I, E

Case 4: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ E

In this scenario, the starting point of TA_3^3 is "Include" and that of TA_4^3 is "Exclude". Clearly, actions "Include" and "Exclude" for TA_3^3 and TA_4^3 are not absorbing because none of the cases will make TA_3^3 and TA_4^3 only move towards "Include" and "Exclude".

Scenario 3: Study TA_3^3 = E and TA_4^3 = I.

Case 1: we can see that
$\mathrm{TA}_3^3 \rightarrow$ I, E
$\mathrm{TA}_4^3 \rightarrow$ E

Case 2: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ E

Case 3: we can see that
$\mathrm{TA}_3^3 \rightarrow$ I, E
$\mathrm{TA}_4^3 \rightarrow$ E

Case 4: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ E

From the transitions of TA_3^3 and TA_4^3 in Scenario 3, we can conclude that the state with TA_3^3 = E and TA_4^3 = I is not absorbing.

Scenario 4: Study TA_3^3 = E and TA_4^3 = E.

Case 1: we can see that
$\mathrm{TA}_3^3 \rightarrow$ I
$\mathrm{TA}_4^3 \rightarrow$ E

Case 2: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ I, E

Case 3: we can see that
$\mathrm{TA}_3^3 \rightarrow$I
$\mathrm{TA}_4^3 \rightarrow$I, E

Case 4: we can see that
$\mathrm{TA}_3^3 \rightarrow$ E
$\mathrm{TA}_4^3 \rightarrow$ E

From the transitions of TA_3^3 and TA_4^3 in Scenario 4, we can see that the state with TA_3^3 = E and TA_4^3 = E seems absorbing in Case 4, i.e., when TA_1^3 and TA_2^3 have both actions as Include. However, the condition in Case 4, i.e., TA_1^3=I and TA_2^3=I, is transient. For this reason, state TA_3^3 = E and TA_4^3 = E becomes not absorbing.

From the above analysis, we can conclude that when we freeze TA_1^3 and TA_2^3 with certain actions, there is no absorbing case.

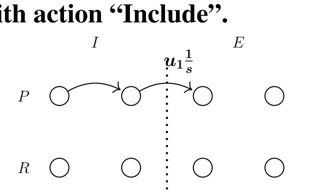
So far, we have studied the behavior of TA_3^3 and TA_4^3 when the transitions of TA_1^3 and TA_2^3 are frozen. In what follows, following the same principle above, we freeze the actions of TA_3^3 and TA_4^3 and study the transitions of TA_1^3 and TA_2^3.

Case 1

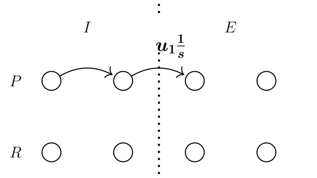
Here TA_3^3 is frozen as "Exclude" and TA_4^3 is "Include". In this situation, the outputs of TA_3^3 and TA_4^3 give $\neg x_2$.

We firstly study TA_1^3 with action "Include".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=E.
Therefore, Type I, $x_1 = 0$, $C_3 = x_1 \wedge \neg x_2 = 0$.

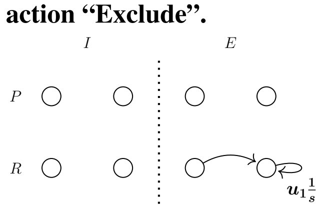


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=I.
Therefore, Type I, $x_1 = 0$, $C_3 = 0$.

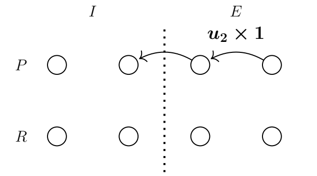


We now study TA_1^3 with action "Exclude".

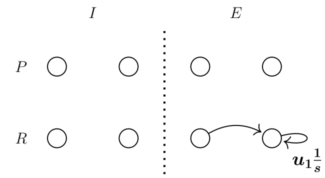
Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=E.
Therefore, Type I, $x_1 = 0$, $C_3 = \neg x_2 = 0$.



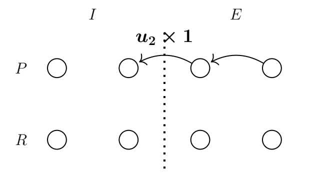
Condition: $x_1 = 0$, $x_2 = 0$, $y = 0$, TA_2^3=E.
Therefore, Type II, $x_1 = 0$, $C_3 = \neg x_2 = 1$.



Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=I.
Therefore, Type I, $x_1 = 0$, $C_3 = \neg x_1 \wedge \neg x_2 = 0$.

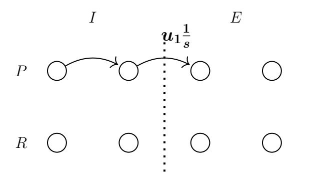


Condition: $x_1 = 0$, $x_2 = 0$, $y = 0$, TA_2^3=I.
Therefore, Type II, $x_1 = 0$, $C_3 = \neg x_1 \wedge \neg x_2 = 1$.

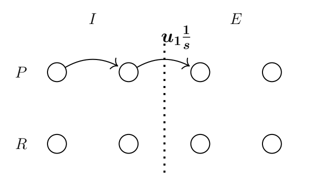


We thirdly study TA_2^3 with action "Include".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=E.
Therefore, Type I, $\neg x_1 = 1$, $C_3 = \neg x_1 \wedge \neg x_2 = 0$

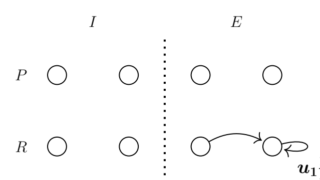


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=I.
Therefore, Type I, $\neg x_1 = 1$, $C_3 = 0$

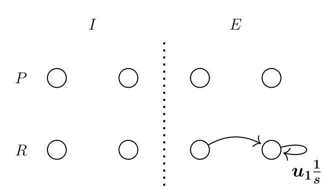


We finally study TA_2^3 with action "Exclude".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=E.
Therefore, Type I, $\neg x_1 = 1$, $C_3 = \neg x_2 = 0$



Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=I.
Therefore, Type I, $\neg x_1 = 1$, $C_3 = x_1 \wedge \neg x_2 = 0$.

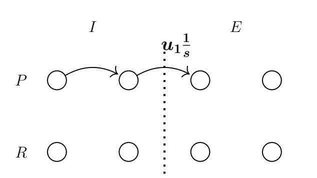


Case 2

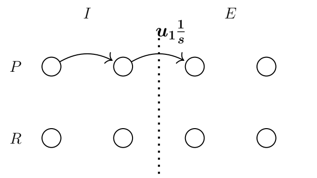
Here TA_3^3 is frozen as "Include" and TA_4^3 is as "Exclude". In this situation, the outputs of TA_3^3 and TA_4^3 give x_2.

We now study TA_1^3 with action "Include".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=E.
Therefore, Type I, $x_1 = 0$, $C_3 = x_1 \wedge x_2 = 0$.

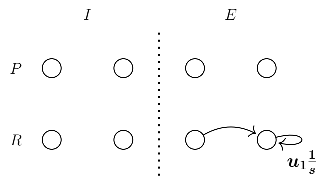


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=I.
Therefore, Type I, $x_1 = 0$,
$C_3 = \neg x_1 \wedge x_1 \wedge x_2 = 0$.

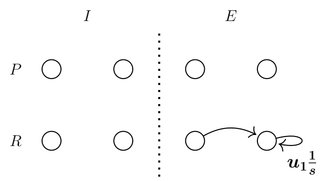


We now study TA_1^3 with action "Exclude".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=E.
Therefore, Type I, $x_1 = 0$,
$C_3 = x_1 = 1$.

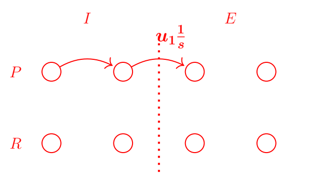


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=I.
Therefore, Type I, $x_1 = 0$,
$C_3 = \neg x_1 \wedge x_2 \wedge 0 = 0$.

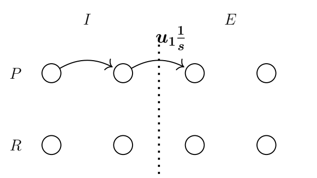


We now study TA_2^3 with action "Include".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=E.
Therefore, Type I, $\neg x_1 = 1$,
$C_3 = \neg x_1 \wedge x_2 \wedge 0 = 0$.

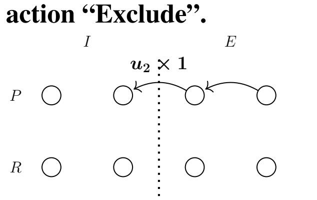


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=I.
Therefore, Type I, $\neg x_1 = 1$,
$C_3 = 0$.

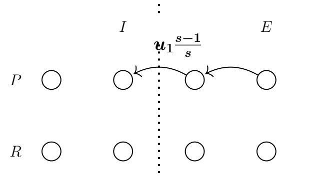


We now study TA_2^3 with action "Exclude".

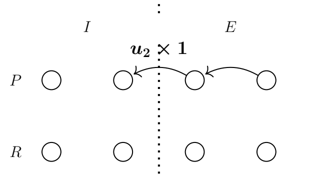
Condition: $x_1 = 1$, $x_2 = 1$, $y = 0$, TA_1^3=E.
Therefore, Type II, $\neg x_1 = 0$,
$C_3 = x_2 = 1$.



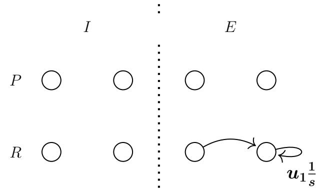
Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=E
Therefore, Type I, $\neg x_1 = 1$,
$C_3 = x_2 = 1$.



Condition: $x_1 = 1$, $x_2 = 1$, $y = 0$, TA_1^3=I.
Therefore, Type II, $\neg x_1 = 0$,
$C_3 = x_1 \wedge x_2 = 1$.



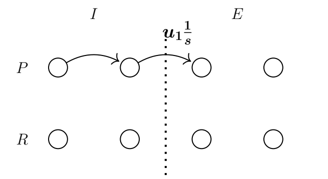
Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=I.
Therefore, Type I, $\neg x_1 = 1$,
$C_3 = x_1 \wedge x_2 = 0$.



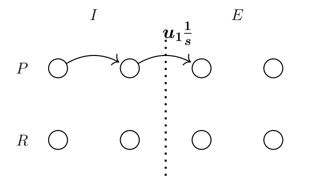
Case 3
Here TA_3^3 is frozen as "Exclude" and TA_4^3 is as "Exclude". In this case, the second bit x_2 does not play any role for the output.

We now study TA_1^3 with action "Include".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=E.
Therefore, Type I, $x_1 = 0$,
$C_3 = x_1 = 0$.

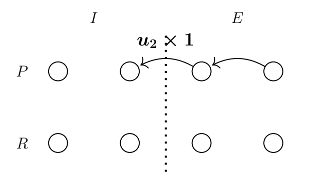


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=I.
Therefore, Type I, $x_1 = 0$,
$C_3 = x_1 \wedge \neg x_1 = 0$.

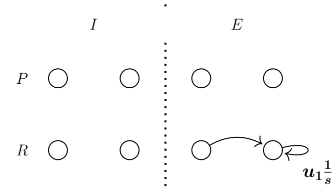


We now study TA_1^3 with action "Exclude".

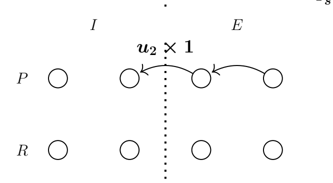
Condition: $x_1 = 0$, $x_2 = 0$, $y = 0$, TA_2^3=E.
Therefore, Type II, $x_1 = 0$,
$C_3 = 1$.



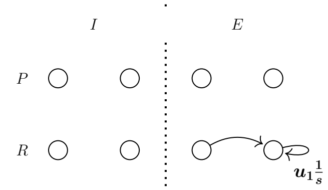
Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=E.
Therefore, Type I, $x_1 = 0$,
$C_3 = 1$.



Condition: $x_1 = 0$, $x_2 = 0$, $y = 0$, TA_2^3=I.
Therefore, Type II, $x_1 = 0$,
$C_3 = \neg x_1 = 1$.

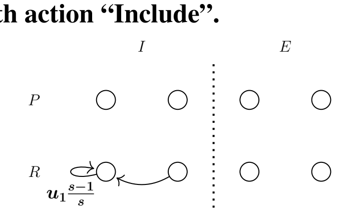


Condition; $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=I.
Therefore, Type I, $x_1 = 0$,
$C_3 = \neg x_1 = 1$.

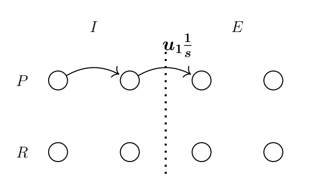


We now study TA_2^3 with action "Include".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=E.
Therefore, Type I, $\neg x_1 = 1$,
$C_3 = \neg x_1 = 1$.

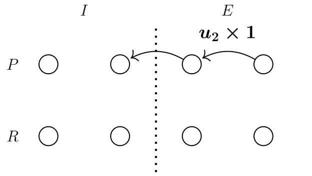


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=I.
Therefore, Type I, $\neg x_1 = 1$,
$C_3 = 0$.

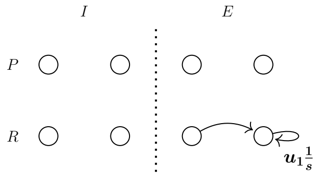


We now study TA_2^3 with action "Exclude".

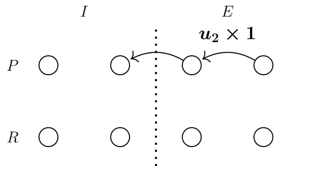
Condition: $x_1 = 1$, $x_2 = 1$, $y = 0$, TA_1^3=E.
Therefore, Type II, $\neg x_1 = 0$, $C_3 = x_1 = 1$.



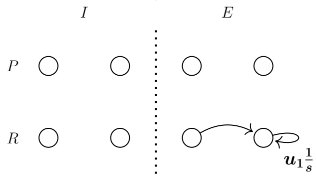
Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=E.
Therefore, Type I, $\neg x_1 = 0$, $C_3 = x_1 = 1$.



Condition: $x_1 = 1$, $x_2 = 1$, $y = 0$, TA_1^3=I.
Therefore, Type II, $\neg x_1 = 0$, $C_3 = x_1 = 1$.



Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=I.
Therefore, Type I, $\neg x_1 = 0$, $C_3 = x_1 = 1$.

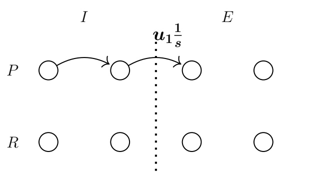


Case 4

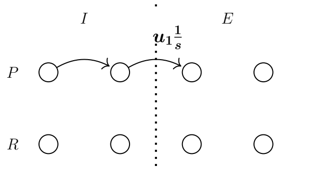
Here both TA_3^3 and TA_4^3 are frozen as "Include". In this situation, the output of the clause is always 0.

We now study TA_1^3 with action "Include".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=E.
Therefore, Type I, $x_1 = 0$, $C_3 = 0$.

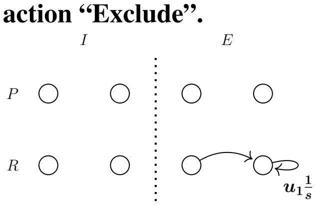


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=I.
Therefore, Type I, $x_1 = 0$, $C_3 = 0$

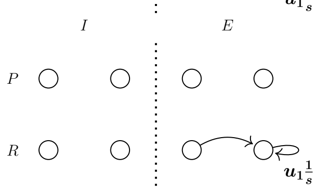


We now study TA_1^3 with action "Exclude".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_2^3=E.
Therefore, Type I, $x_1 = 0$, $C_3 = 0$.

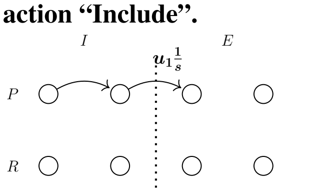


Condition: $x_1 = 0$, $x_2 = 1$, $y = 0$, TA_2^3=I.
Therefore, Type II, $x_1 = 0$, $C_3 = 0$.

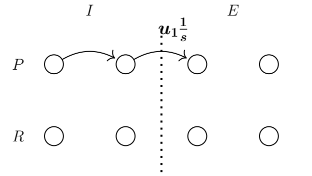


We now study TA_2^3 with action "Include".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=E.
Therefore, Type I, $\neg x_1 = 1$, $C_3 = 0$.

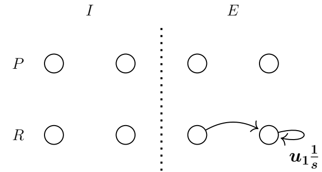


Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=I.
Therefore, Type I, $\neg x_1 = 1$, $C_3 = 0$.

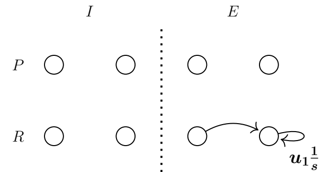


We now study TA_2^3 with action "Exclude".

Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=E.
Therefore, Type I, $\neg x_1 = 1$, $C_3 = 0$.



Condition: $x_1 = 0$, $x_2 = 1$, $y = 1$, TA_1^3=I.
Therefore, Type I, $\neg x_1 = 1$, $C_3 = 0$.



Based on the analysis performed above, we can show the directions of transitions for TA_1^3 and TA_2^3 given different configurations of TA_3^3 and TA_4^3.

Scenario 1: Study TA_1^3 = I and TA_2^3 = E.

Case 1: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ E

Case 2: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ I, E

Case 3: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ I, E

Case 4: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ E

Scenario 2: Study TA_1^3 = I and TA_2^3 = I.

Case 1: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ E

Case 2: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ E

Case 3: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ E

Case 4: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ E

Scenario 3: Study TA_1^3 = E and TA_2^3 = I.

Case 1: we can see that
$TA_1^3 \rightarrow$ I, E
$TA_2^3 \rightarrow$ E

Case 2: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ E

Case 3: we can see that
$TA_1^3 \rightarrow$ I
$TA_2^3 \rightarrow$ I

Case 4: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ E

Scenario 4: Study TA_1^3 = E and TA_2^3 = E.

Case 1: we can see that
$TA_1^3 \rightarrow$ I, E
$TA_2^3 \rightarrow$ E

Case 2: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ I

Case 3: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ I, E

Case 4: we can see that
$TA_1^3 \rightarrow$ E
$TA_2^3 \rightarrow$ E

According to the above transitions, we can conclude that there is no absorbing state.

Based on the above analysis, for the sub-pattern described in Table 3, given $\|C_j^i(\mathbf{X})\| = \mathbf{1}$, there is no absorbing state, indicating that the CSC-TM cannot converge to the intended sub-pattern. The same applies to the other sub-pattern in the XOR operator, i.e., $[x_1 = 1,\ x_2 = 0]$. Therefore, we can conclude that given $\|C_j^i(\mathbf{X})\| = \mathbf{1}$, the XOR operator cannot be learnt by CSC-TM.

References

[Granmo *et al.*, 2019] Ole-Christoffer Granmo, Sondre Glimsdal, Lei Jiao, Morten Goodwin, Christian W. Omlin, and Geir Thore Berge. The Convolutional Tsetlin Machine. *arXiv preprint arXiv:1905.09688*, 2019.

[Granmo, 2018] Ole-Christoffer Granmo. The Tsetlin Machine - A Game Theoretic Bandit Driven Approach to Optimal Pattern Recognition with Propositional Logic. *arXiv preprint arXiv:1804.01508*, 2018.

[Jiao *et al.*, 2023] Lei Jiao, Xuan Zhang, Ole-Christoffer Granmo, and K. Darshana Abeyrathna. On the Convergence of Tsetlin Machines for the XOR operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(5):6072–6085, 2023.

[Oommen, 1997] B.J. Oommen. Stochastic searching on the line and its applications to parameter learning in nonlinear optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 27(4):733–739, 1997.